Coded Caching for Cache-Aided Communication and Computing with Nonuniform Demands

by

Yong Deng

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

Faculty of Engineering and Applied Science
University of Ontario Institute of Technology (Ontario Tech University)
Oshawa, Ontario, Canada

October 2021

THESIS EXAMINATION INFORMATION

Submitted by: **Yong Deng**

Doctor of Philosophy in **Electrical and Computer Engineering**

Thesis Title: Coded Caching for Cache-Aided Communication and Computing with Nonuniform Demands

An oral defense of this thesis took place on October 26th, 2021 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee: Prof. Amirkianoosh Kiani

Research Supervisor: Prof. Min Dong

Examining Committee Member: Prof. Shahram ShahbazPanahi

Examining Committee Member: Prof. Ying Wang

University Examiner: Prof. Shahryar Rahnamayan

External Examiner: Prof. Jun Chen, McMaster University

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

In this dissertation, we consider the caching system of multiple cache-enabled users with nonuniform demands. We thoroughly characterize the structure of the optimal cache placement of the coded caching scheme (CCS). We show there are at most three file groups in the optimal placement and obtain the closed-form solution in each file grouping case. A simple algorithm is developed to obtain the final optimal cache placement, which only computes a set of closed-form solutions in parallel. We propose a tighter lower bound for caching and characterize the file subpacketization in the optimal CCS.

Upon obtaining the optimal placement of the CCS, we characterize the memory-rate tradeoff for caching with uncoded placement. Focusing on the modified coded caching scheme (MCCS), we formulate a cache placement optimization problem to obtain the optimal placement. We present two lower bounds for caching with uncoded placement and compare them with the MCCS to characterize the memory-rate tradeoff and provide insights. We extend our study to accommodate both nonuniform file popularity and sizes and characterize the exact memory-rate tradeoff for two users.

We then study the memory-rate tradeoff for decentralized caching. We formulate the cache placement optimization problem for the decentralized modified coded caching scheme (D-MCCS). To solve this non-convex problem, we develop a successive Geometric Programming (GP) approximation algorithm and a two-file-group-based low-complexity approach. We propose a lower bound for decentralized caching and compare it with the optimized D-MCCS to characterize the memory-rate tradeoff in special cases.

Beyond caching problems, we consider the coded distributed computing (CDC) with arbitrary number of files of nonuniform popularity. We formulate a mixed-integer linear programming (MILP) problem that jointly optimizes the mapping and shuffling strategies to minimize the shuffling load. To solve this generally NP-hard problem, we propose a two-file-group-based low-complexity approach that achieves close shuffling load as conventional branch-and-cut method which has high computational complexity.

Keywords: *Coded Caching; Nonuniform Demands; Memory-Rate Tradeoff; Heterogeneous Code Distributed Computing; Optimization*

AUTHOR'S DECLARATION

I, Yong Deng, hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

_____ Yong Deng

STATEMENT OF CONTRIBUTIONS

Part of this dissertation described in Chapters 3, 4 and 5 have been published as:

1. Yong Deng and Min Dong, "Memory-Rate Tradeoff for Caching with Uncoded Placement under Nonuniform Random Demands," submitted to *IEEE Transactions on Information Theory*, March 2021. Available at arXiv preprint arXiv:2103.09925.

2. Yong Deng and Min Dong, "Fundamental Structure of Optimal Cache Placement for Coded Caching with Nonuniform Demands," submitted to *IEEE Transactions on Information Theory*, April 2020. Revised, available at arXiv preprint, arXiv:1912.01082.

3. Yong Deng and Min Dong, "Memory-Rate Tradeoff for Decentralized Caching under Nonuniform File Popularity," in *Proc. of the 19th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, Oct. 18-21, 2021.

4. Yong Deng and Min Dong, "Memory-Rate Tradeoff for Caching with Uncoded Placement under Nonuniform File Popularity," in *Proc. of the 54th Asilomar Conference on Signals, Systems, and Computers*, Nov. 3-6, 2020.

5. Yong Deng and Min Dong, "Optimal Uncoded Placement and File Grouping Structure for Improved Coded Caching under Nonuniform Popularity," in *Proc. of the 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, Jun. 15-19, 2020.

6. Yong Deng and Min Dong, "Subpacketization Level in Optimal Placement for Coded Caching with Nonuniform File Popularity," in *Proc. of the 53rd Asilomar Conference on Signals, Systems, and Computers*, Nov. 3-6, 2019.

7. Yong Deng and Min Dong, "Optimal Cache Placement for Modified Coded Caching with Arbi trary Cache Size," in *Proc. of the 20th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Jul. 2-5, 2019.

I hereby certify that I am the primary contributor of this dissertation. I performed the majority of the experiments and writing of the manuscript.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Prof. Min Dong, who is such a great and respectful researcher. None of my achievements can be done without her guidance in the past four years. Dr. Dong not only taught me how to do good research, but also showed me how to do everything rigorously in my daily life. Through her, I became to know what is and how to be a respectful researcher. I would like to thank her great patience while discussing with me and for keep showing me the right way to approach and solve problems. She has also taught me how to do the presentation professionally. I am so proud of the experience of working with and being advised by her. Although I still have a long way to go, what I learned from her throughout this journey is incomparable.

I would like to thank Prof. Jun Chen for serving as my external examiner. I would also like to express my sincere gratitude to all the committee members of my defence, Prof. Shahram ShahbazPanahi, Prof. Ying Wang and Prof. Shahryar Rahnamayan, who have been given me valuable feedback and suggestions to help me improve my dissertation. I would like to thank Prof. Amirkianoosh Kiani who serves as the chair of my examining committee.

After all, I would like to thank my wife Hongli Wang, my parents Congyou Deng and Xingfeng Liang and my sister Yunfang Deng who have been so supportive to me with their unconditional love.

Contents

List of Acronyms

CCS	Coded Caching Scheme
MCCS	Modified Coded Caching Scheme
D-CCS	Decentralized Coded Caching Scheme
D-MCCS	Decentralized Modified Coded Caching Scheme
D2D	Device to Device
LP	Linear Programming
GP	Geometric Programming
CGP	Complementary Geometric Programming
CDC	Coded Distributed Computing
IV	Intermediate Value
MILP	Mixed Integer Linear Programming
NP	Nondeterministic Polynomial-Time

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The last decades have witnessed a surge in wireless traffic due to the proliferation of mobile devices [1]. The rapidly growing content distribution services including on-demand video streaming, Virtual Reality (VR), video-sharing-focused social networking are the major contributor to the traffic exploding in today's wireless networks. Facing the drastic increase of data-intensive wireless applications and services, future wireless communication networks need to effectively manage the data traffic congestion and meet the requirements of timely content delivery. The availability of local caches at the network edge, either at base stations or users, creates new network resources and opportunities to increase the user service capacity. Caching technologies are anticipated to become key technological drivers for content delivery in future wireless networks [2–4].

Caching was originally introduced in the computer system to improve the system performance by storing popularly used data at memory with fast access [5]. Later, the idea of caching popular data at memories distributed among the network was studied in different wireline contexts such as web caching system and content delivery network (CDN) to reduce the overall service delay [6, 7]. In wireless networks, caching has attracted extensive attention as a key technique to alleviate the traffic load to meet the timely service requirement. In a caching system, the cache placement strategy specifies what data to be stored at the users' local memories and how to store them ahead of time, without the

knowledge of their actual demands during the file request and delivery stage. Many recent works have studied the cache placement design for caching to understand the effect of caching on reducing the network load [8–10]. The cache placement problem was first introduced in [8] to find optimal cache placement that minimizes the total access cost in an arbitrary network. The problem was shown to be NP-hard and tractable approximate solutions have been proposed in terms of linear programming (LP) relaxation [9] or greedy algorithm [10]. Moreover, there are also extensive works address the cache placement problem in the different networks, such as D2D network [11], multi-tier heterogeneous network [12], femto-caching network [13, 14], fog radio access network [15], etc.

Conventional uncoded caching schemes [8–15] can improve the hit rate to explore the local caching gain, but are not efficient when there are multiple caches [16]. Coded caching has been recently introduced in the seminal work [17] for a shared link cache-aided system that is formed by a single service provider (*e.g.,* a server or base station) and multiple users, as shown in Fig. 1.1. Specifically, a *coded caching scheme* (CCS) is proposed in [17] that combines a carefully designed cache placement of uncoded contents and a coded multicast delivery strategy to explore the caching gain. Through analysis, it is shown that by exploring both global and local caching gain, the CCS can serve an infinite number of users simultaneously with finite resources [17]. Since then, coded caching has drawn considerable attention, with extensions of the CCS to the decentralized scenario [18], transmitter caching in mobile edge networks [19, 20], user caching in device-to-device networks [21, 22] and for both transmitter and receiver caching in wireless interference networks [23].

Recently, the idea of exploiting coded multicasting opportunities to reduce the communication load has also been extended to the distributed computing networks for large scale data-intensive applications to provide low-latency services [24]. In popular distributed computing frameworks including MapReduce and Spark, a job is split into a number of target functions that are executed through three independent phases, named as *Map*, *Shuffle* and *Reduce* [24]. In the Map phase, each worker computes the map tasks using its local

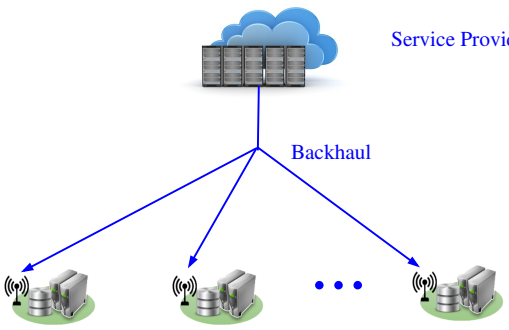

Fig. 1.1: An example of cache-aided systems, where users are connected to the central service provider through a backhaul link. Each user has a local cache to alleviate the burden of the backhaul.

data (files) to generate Intermediate Values (IVs). In the Shuffle phase, workers exchange their local IVs so that each worker obtains its required IVs. In the Reduce phase, using the IVs collected in the Shuffle phase, each worker accomplishes its assigned target functions.

The major performance bottleneck of distributed computing comes from the communication overhead during the Shuffle phase. For example, more than 90% of the execution time is committed for data shuffling when computing a Tarasort job on Amazon EC2 platform [25]. To mitigate the communication bottleneck, the seminar work in [26] proposed a coded distributed computing (CDC) scheme that greatly slashes the data shuffling time through the utilizing of coding. Recently, the design of the CDC was well studied assuming nonuniform mapping and reducing loads among the workers, which are referred as the heterogeneous CDCs [27–30]. Further, the study of CDC has also been extended into various systems, including wireless CDC [31], heterogeneous IV sizes [32], etc.

1.2 Coded Caching with Nonuniform Demands

A key design issue in coded caching is the cache placement that specifies what portion of each file to cache and how to store it. An effective cache placement scheme maximizes caching gain and minimizes the transmission load in the network in the content delivery phase (*i.e.,* the delivery rate). The works mentioned earlier all assume uniform file popularity under homogenous demands, for which a symmetric cache placement strategy (*i.e.,*

identical cache placement for all files) is optimal [33]. In the more general scenario of files with heterogeneous demands leading to nonuniform file popularity, the cache placement may be different among files, complicating both caching design and analysis. There is a fundamental question on whether to distinguish files of different popularities and to what extent. On the one hand, different cache placements for files with distinct popularities may help capture the difference in demands to improve caching efficiency. On the other hand, depending on the degree of difference, ignoring this difference in file popularity and simply using the symmetric cache placement may be a good tradeoff between performance gain and implementation complexity.

Several recent works have considered the cache placement design for the CCS under nonuniform file popularity [16, 34–36], where a typical method is to construct a cache placement scheme and bound its performance. For complexity reduction, file grouping is commonly used as a tractable method for the cache placement design. It was first proposed in [16], which divides files into groups based on their popularities and allocates chunks of cache to different groups. Files within each file group are treated the same with identical cache placement. Following this, different methods to partition files into file groups have been proposed [34–36]. These existing studies show that file grouping is an effective method to handle nonuniform file popularity for cache placement. However, the file grouping methods used in these existing schemes are all somewhat heuristic, with two file groups typically considered to separate the most popular files from the remaining ones. The optimal cache placement and its relation to file grouping remain unknown. Different from the method of construction, the optimization approach was adopted in [33, 37] to formulate the cache placement into an optimization problem to find the solution. Both works have focused on developing numerical methods to solve the optimization problem. However, the numerical results cannot provide insights into the optimal cache placement.

Indeed, characterizing the optimal cache placement structure may bring us a deeper understanding of the effect of nonuniform file popularity on the caching gain offered by the CCS. Furthermore, in the cache placement for the CCS, each file is partitioned into subfiles

to be stored at different sets of users. The number of required subfiles may potentially grow exponentially with the number of users. This could prevent the practical use of the CCS for files with finite sizes and limit the caching gain that can be achieved. There have been studies on the tradeoff between the subpacketization level and the coded caching gain by the CCS under uniform file popularity [38–43]. The analysis of subpacketization is more challenging for nonuniform file popularity and, therefore, scarce in the literature, as different files may be partitioned in different ways. Obtaining the optimal cache placement structure will help characterize the file subpacketization in the CCS to understand the practical limits and make an appropriate tradeoff between the subpacketization level and coded caching gain for the CCS.

1.3 Memory-Rate Tradeoff for Caching under Uncoded Placement

For understanding the fundamental limit of coded caching, many research efforts were devoted to characterizing the memory-rate tradeoff for caching with uncoded cache placement. For files of uniform popularity and sizes, this tradeoff has been studied extensively, typically by developing an achievable scheme and compare it to an information-theoretic lower bound [17, 44–46]. When the system has fewer users than files, it has been shown that the CCS with optimized cache placement achieves the minimum peak delivery rate for caching with uncoded placement, *i.e.,* the exact memory-rate tradeoff [45, 46]. Recently, a *modified coded caching scheme* (MCCS) was proposed [44] with an improved delivery strategy to remove redundancy among coded messages, resulting in further delivery rate reduction from that of the CCS. For general scenarios of arbitrary users and files with random requests, the MCCS with optimized cache placement has been shown to characterize the exact memory-rate tradeoff that minimizes both average and peak delivery rate [44] under uniform file popularity and sizes.

For files with nonuniform popularity or sizes, different cache placement strategies were proposed for the CCS to handle nonuniformity [16, 33–37, 47–51]. In particular,

several CCS-based schemes, either for nonuniform file popularity [35, 36] or for nonuniform file sizes [50,51], were shown to achieve an average rate that is a constant factor away from the lower bound for caching with any cache placement. However, these gaps are still large for practical concerns. Only recently, for two files of nonuniform popularity, a coded caching scheme was proposed [52], which achieves the lower bound for caching with uncoded placement. For the MCCS, existing studies are scarce, and only [53] studied the cache placement optimization under nonuniform file popularity. In general, for files with nonuniform popularity and sizes, characterizing the memory-rate tradeoff for caching with uncoded placement is challenging. How well the CCS and the MCCS perform in terms of memory-rate tradeoff remains unknown.

1.4 Decentralized Caching

The above studies on caching generally rely on a carefully designed centrally coordinated cache placement strategy to store a portion of each file content in a subset of users. However, a coordinated cache placement may not always be possible in practice, which may limit the practical use of coded caching. For this issue, decentralized caching has been considered [18], where no coordination among users is required, and each user caches uncoded contents independently from each other. Specifically, for a system with a central server connecting to multiple cache-equipped users, a *decentralized coded caching scheme* (D-CCS) was proposed in [18], which consists of a decentralized (uncoded) placement scheme and a coded delivery strategy. Interestingly, under uniform file popularity, it is shown that the performance of the D-CCS is close to the centralized coded caching scheme [18]. For nonuniform file popularity, with the number of users requesting files (*i.e.,* active users) known at the server, it has been shown that the achievable rate of the D-CCS is within a factor away from the tightest lower bound developed in [36]. However, since the lower bound is for any caching, the gap is still large for practical consideration.

Recently, for files with uniform popularity, a *decentralized modified coded caching scheme* (D-MCCS) was proposed in [44] that removes the redundancy in the coded mes-

sages used in the D-CCS to further reduce the delivery rate. It has been further shown that the D-MCCS attains the lower bounds on both average and peak rates for decentralized caching and thus characterizes the exact memory-rate tradeoff [44]. For files with nonuniform popularity, there is no study on the cache placement optimization for the D-MCCS or how optimal the D-MCCS is for decentralized caching. In general, the memory-rate tradeoff for decentralized caching remains unknown.

1.5 Heterogeneous Coded Distributed Computing

MapReduce is a popular distributed computing framework for large scale data-intensive applications [24, 54]. To mitigate the communication bottleneck during the Shuffle phase, the seminar work in [26] proposed a coded distributed computing (CDC) that greatly reduces the data shuffling load through the use of coding. Noticeably, the general idea of the CDC is to exploit the coded multicast opportunity during the data shuffling phase, which is essentially an extension of the CCS to the distributed computing. Focusing on the homogeneous system, the CDC scheme [26] characterizes the exact computation-communication tradeoff for distributed computing. Existing studies on the CDC do not take into consideration the data popularity distribution for the design of the CDC scheme [27–30, 32, 55–59]. Specifically, they typically focus on the scenario of one job being executed and all the data are accessed by the job. However, in practice, the skewed data popularity is a typical characteristic existing in the MapReduce clusters [55, 56, 60]. Empirical analysis on the industrial MapReduce clusters (including Facebook, Cloudera and Microsoft Bing's data center) have shown that the file access pattern in the MapReduce cluster demonstrates a Zipf distribution where a small number of files account for most of the accesses by the jobs [60]. Extensive existing studies on the replica of the files with nonuniform popularity in the traditional MapReduce clusters with uncoded shuffling have shown that the understanding of data popularity is essential to mitigate the data shuffling bottleneck [55, 56, 60]. To our best knowledge, there is no existing work focuses the CDC under nonuniform file popularity. Moreover, existing CDC schemes are only achievable for some specific number

of files, which typically needs to be sufficiently large [26–30, 32].

1.6 Objectives and Contributions

The objectives and the contributions of this dissertation are summarized below.

1.6.1 Objectives

The main objectives of this dissertation are summarized as follows:

- Study the cache placement for coded caching under nonuniform file popularity. The main goals are to obtain the optimal cache placement solution for the CCS and develop a tighter lower bound for caching under nonuniform file popularity.

- Characterize the memory-rate tradeoff for caching with uncoded placement under nonuniform file popularity. The main goals are to obtain the optimal cache placement for the MCCS using the optimization framework and develop a lower bound for caching with uncoded placement. Ultimately, we aim to characterize the memory-rate tradeoff by comparing the optimized MCCS with the proposed lower bound.

- Characterize the memory-rate tradeoff for decentralized caching under nonuniform popularity. The main goals are to obtain the optimal cache placement for the D-MCCS and develop a lower bound for decentralized caching. Lastly, we compare the optimized D-MCCS with the proposed lower bound for the memory-rate tradeoff characterization.

- Develop a heterogeneous CDC scheme for MapReduce based distributed computing with nonuniform mapping and reducing loads among the workers and an arbitrary number of files with nonuniform popularity.

1.6.2 Contributions

The main contributions of this dissertation are summarized as follows:

- **Coded Caching with Nonuniform Demands**. We thoroughly characterize the structure of the optimal uncoded cache placement for the coded caching scheme (CCS) under nonuniform file popularity. Formulating the cache placement as an optimization problem to minimize the average delivery rate, we identify the file grouping structure under the optimal solution. We show that, regardless of file popularity, there are at most three file groups under the optimal cache placement. We further characterize the complete structure of the optimal cache placement and obtain the closed-form solution in each possible file grouping case. A simple algorithm is developed to obtain the final optimal cache placement, which only computes a set of candidate closed-form solutions in parallel. We provide insights into the file groups formed by the optimal cache placement. The optimal placement solution also indicates that coding between file groups may be explored during delivery, in contrast to the existing heuristic file grouping schemes. Using the file grouping in the optimal cache placement, we propose a new information-theoretic converse bound for coded caching that is tighter than existing ones. Moreover, using the optimal cache placement solution, we characterize the file subpacketization in the optimal CCS and show that the maximum subpacketization level in the worst case scales as $\mathcal{O}(2^K/\sqrt{K})$ for K users.

- **Memory-Rate Tradeoff for Caching with Uncoded Placement under Nonuniform Demands**. We aim to characterize the memory-rate tradeoff for caching with uncoded cache placement, under nonuniform file popularity. Focusing on the modified coded caching scheme (MCCS) recently proposed by Yu, *etal.*, we formulate the cache placement optimization problem for the MCCS to minimize the average delivery rate under nonuniform file popularity, restricting to a class of popularity-first placements. We then present two information-theoretic lower bounds on the average rate for caching with uncoded placement, one for general cache placements and the other restricted to the popularity-first placements. By comparing the average rate of the optimized MCCS with the lower bounds, we prove that the optimized MCCS at-

tains the general lower bound for the two-user case, providing the exact memory-rate tradeoff. Furthermore, it attains the popularity-first-based lower bound for the case of general K users with distinct file requests. In these two cases, our results also reveal that the popularity-first placement is optimal for the MCCS, and zero-padding used in coded delivery incurs no loss of optimality. For the case of K users with redundant file requests, our analysis shows that there may exist a gap between the optimized MCCS and the lower bounds due to zero-padding. We next fully characterize the optimal popularity-first cache placement for the MCCS, which is shown to possess a simple file-grouping structure and can be computed via an efficient algorithm using closed-form expressions. Finally, we extend our study to accommodate both nonuniform file popularity and sizes, where we show that the optimized MCCS attains the lower bound for the two-user case, providing the exact memory-rate tradeoff. Numerical results show that, for general settings, the gap between the optimized MCCS and the lower bound only exists in limited cases and is very small.

- **Memory-Rate Tradeoff for Decentralized Caching with Nonuniform Demands**. We study the memory-rate tradeoff for decentralized caching under nonuniform file popularity. We formulate the cache placement optimization problem for a recently proposed decentralized modified coded caching scheme (D-MCCS) to minimize the average rate. To solve this non-convex optimization problem, we develop two algorithms: a successive Geometric Programming (GP) approximation algorithm, which guarantees convergence to a stationary point but has a high computational complexity, and a low-complexity approach based on a two-file-group-based placement strategy. We further propose a lower bound on the average rate for decentralized caching under nonuniform file popularity. The lower bound is given as a non-convex optimization problem, for which we propose a similar successive GP approximation algorithm to compute a stationary point. We show that the optimized MCCS attains the lower bound for the special case of no more than two active users requesting files, or for the general case but satisfying a special condition. Thus, the optimized MCCS

characterizes the exact memory-rate tradeoff for decentralized caching in these cases. In general, our numerical result shows that the optimized D-MCCS performs close to the lower bound.

- **Heterogeneous Coded Distributed Computing**. We study the heterogeneous CDC with arbitrary number files of nonuniform file popularity. To handle this more general system setup, we propose a file placement strategy that can accommodate arbitrary number of files in the Map phase. In the Shuffle phase, we adopt a nested coded shuffling strategy that exploits the coded multicasting opportunities for all the IVs. We then formulate an optimization problem that jointly optimizes the proposed file placement and shuffling strategies to minimize the expected shuffling load. The problem is a mixed integer linear programming (MILP) problem that is generally NP-hard. We develop an approximate approach through the proposition of a two-file-groups-based file placement strategy. Specifically, we convert the original problem into a linear programming (LP) problem that optimizes the shuffling strategy for a given two-file-group-based file placement strategy. Following this, we obtain the optimal two-file-group-based solution through a search over all possible two-file-group-based placements. Numerical studies show that the expected shuffling load of the optimal two-file-group-based solution is close to that of the traditional branch-and-cut method which has high computational complexity.

1.7 List of Publications

The research in this dissertation has resulted in the following list of publications.

Journal papers

Submitted:

1. Yong Deng and Min Dong, "Memory-Rate Tradeoff for Caching with Uncoded Placement under Nonuniform Random Demands," submitted to *IEEE Transactions on Information Theory*, March 2021. Available at arXiv preprint arXiv:2103.09925.

2. Yong Deng and Min Dong, "Fundamental Structure of Optimal Cache Placement for Coded Caching with Nonuniform Demands," submitted to *IEEE Transactions on Information Theory*, April 2020. Revised, available at arXiv preprint, arXiv:1912.01082.

In Preparation:

1. Yong Deng and Min Dong, "Memory-Rate Tradeoff for Decentralized Caching under Nonuniform File Popularity and Sizes," in preparation.

Conference papers

Published:

1. Yong Deng and Min Dong, "Memory-Rate Tradeoff for Decentralized Caching under Nonuniform File Popularity," in *Proc. of the 19th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, Oct. 18-21, 2021.

2. Yong Deng and Min Dong, "Memory-Rate Tradeoff for Caching with Uncoded Placement under Nonuniform File Popularity," in *Proc. of the 54th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 3-6, 2020.

3. Yong Deng and Min Dong, "Optimal Uncoded Placement and File Grouping Structure for Improved Coded Caching under Nonuniform Popularity," in *Proc. of the 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, Jun. 15-19, 2020.

4. Yong Deng and Min Dong, "Subpacketization Level in Optimal Placement for Coded Caching with Nonuniform File Popularities," in *Proc. of the 53rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 3-6, 2019.

5. Yong Deng and Min Dong, "Optimal Cache Placement for Modified Coded Caching with Arbitrary Cache Size," in *Proc. of the 20th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Jul. 2-5, 2019.

In Preparation:

1. Yong Deng and Min Dong, "Heterogeneous Coded Distributed Computing for Arbitrary Number of Files with Nonuniform Popularity," to be submitted in Oct. 2021.

1.8 Notations

In this dissertation, the cardinality of set $\mathcal{S}$ is denoted by $|\mathcal{S}|$, and the size of file W is denoted by $|W|$. The bitwise "XOR" operation between two subfiles is denoted by $\oplus$. Notations $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling functions, respectively. Notation $\mathbf{a} \succcurlyeq \mathbf{0}$ means element-wise non-negative in vector $\mathbf{a}$. We extend the definition of $\binom{K}{l}$ and define $\binom{K}{l} = 0$, for $l < 0$ or $l > K$. The index set for $\mathcal{S}$ is defined by $\mathcal{I}_{|\mathcal{S}|} = \{1, \ldots, |\mathcal{S}|\}$.

Chapter 2

Literature Review

2.1 Coded Caching

The CCS has been studied in many works for various system scenarios to understand the fundamental limit of coded caching [17–21, 23, 33]. In these works, the cache placement for the CCS has been studied for the peak delivery rate under uniform file popularity[1], where the optimal cache placement in this case is the same for all files [17, 33]. The cache placement under nonuniform file popularity has been investigated in [16, 34–36]. It was first studied in [16], where a file grouping strategy independent of the number of users K was proposed to reduce the design complexity by treating files in each group to be the same and using the symmetric decentralized CCS for each group. Following this, by incorporating the knowledge of K in the file grouping design, several suboptimal file grouping schemes have been proposed to lower the average delivery rate [34–36]. In [34], a specific multi-level file popularity model is considered, where the number of files at each level and the number of users requesting the files at each level are fixed. Under this model, a caching scheme using two file groups was proposed and shown to be order-optimal depending on the number of levels. With a more general file popularly distribution, a simple RLFU-GCC scheme was proposed in [35], which splits files into two file groups, with one containing the most popular files and allocated the entire cache. The performance of this scheme was shown to be order-optimal for the Zipf distribution. As an extension to an arbitrary

[1]For uniform file popularity, it can be shown that the peak rate and average rate are identical for the CCS.

	Approach	File grouping strategy	Cache placement strategy
[16]	Proposed a suboptimal scheme	Multiple file groups	Decentralized
[17],[18], [19]	Proposed a suboptimal scheme	One or two file groups	Decentralized
[61]	Via optimization, (suboptimal) numerical methods	N/A	Decentralized
[15], [20]	Via optimization, numerical method	N/A	Centralized
Our work	Via optimization, closed-form optimal solution	Optimal file groups	Centralized

Table 2.1: Comparison with existing cache placement schemes for the CCS

popularity distribution, a mixed caching strategy was proposed [36] by adding a choice of an uncoded caching scheme to the above two-file-group caching scheme. This added scheme has three file groups for cache placement and uses uncoded delivery. All the above works [16, 34–36] use decentralized CCS for each file group, and there is no coding opportunity between the file groups in these schemes. Different from the above approaches, the optimization framework is considered to find the optimal cache placement for the CCS under nonuniform file popularity in the centralized scenario [33, 37] and the decentralized setting [61].[2] Numerical methods are resorted to solve these problems, which cannot be used to characterize the optimal cache placement. The optimal cache placement for the CCS under arbitrary file popularity distribution and its relationship with file grouping remains unknown. We summarize the differences between our work and the above mentioned existing works for the CCS in Table 2.1.

For understanding the fundamental limit of coded caching, information-theoretic converse bounds are developed in the literature. The lower bounds on the peak and average rates for files with uniform popularity have been developed and improved by several works [17, 18, 62, 63]. For nonuniform file popularity, different lower bounds on the average rate have been developed to demonstrate the performance of the proposed file grouping based caching schemes [16, 34–36]. A lower bound was first developed in [16], where a genie-based method was used to compute the sum peak delivery rates of multiple file groups that are heuristically partitioned. The number of file groups depends on popularity distribution, and the bound is generally loose. The genie-based method is commonly used to obtain the

[2]From the optimization perspective, the decentralized cache placement problem is a subproblem of the cache placement optimization problem in the centralized scenario. In other words, any decentralized cache placement is a feasible point of the centralized cache placement optimization problem.

lower bounds [34–36]. It constructs a virtual system where only a group of most popular files need to be delivered to the users via the shared link, and these popular files are treated equally. The group of most popular files is formed either heuristically or through a suboptimal method, resulting in different tightness of the lower bound. In [35], focusing on the Zipf distribution of file popularity, the authors proposed a method to determine the group of most popular files for different Zipf parameters, and a lower bound on the average rate is developed using the peak delivery rate in this file group. A lower bound for an arbitrary file popularity distribution was obtained in [36] by categorizing the most popular files via a different strategy. Furthermore, a file merging process was proposed to tighten the bound further by including some moderately popular files into the group of most popular files. From these existing studies, the proposed file grouping strategies appear to have a strong influence on the tightness of the lower bound. In this work, we show that the file group structure in the optimal cache placement would lead to a tighter lower bound.

File subpacketization in the cache placement has been studied in [38–41] for uniform file popularity, where different methods were proposed to reduce the subpacketization level in the cache placement with a higher delivery rate as a tradeoff. The Pareto-optimal coded caching schemes that characterize the tradeoff between the high subpacketization level and the rate were provided in [40]. The cache placement of the CCS given in [17] for specific cache sizes is proved to be both optimal [39] and Pareto-optimal [40] in achieving the highest cache gain with the minimum subpacketization level. In [42], the existence of coded caching schemes with the linear growth of the subpacketization level for a large number of users is shown. In [43], using multiple antennas is suggested to reduce the subpacketization level. The problem under nonuniform file popularity is much more complicated, and the study is scarce. In [53], a cache placement optimization problem is considered that uses the subpacketization level as a constraint. The influence of the subpacketization level on the average rate was explored through numerical simulations. Unfortunately, the simulation approach is not able to provide insights into the subpacketization feature in the optimal cache placement solution.

Besides nonuniform file popularity, other types of nonuniformity have also been considered in the coded caching design, including file sizes [33, 64], cache sizes [65, 66], and link qualities [67–69].

2.2 Memory-Rate Tradeoff for Caching

With a surge of interest in caching, there are many recent works study caching with uncoded placement. For uniform file popularity and sizes, the exact memory-rate tradeoff has been fully characterized for both peak rate [44–46] and average rate [44], which is achieved by the CCS and the MCCS, respectively. Beyond uncoded placement, the average rate of the optimized MCCS was shown to be at most a factor of two away from the optimal caching with any placement considered [70].

When heterogeneity exists in the system, the characterization of the memory-rate tradeoff is generally an open problem. For nonuniform file popularity, the cache placement problem was studied for the CCS [16, 33–37, 47, 48] and the MCCS [53] to minimize the achievable delivery rate. For the CCS, to simplify the placement problem amid nonuniformity, suboptimal file-grouping-based cache placement strategies were proposed in [16, 34–36]. They are shown to achieve an average rate that is a constant factor away from the lower bound for caching with any placement. Nonetheless, the gap is generally still large for practical consideration. Several works used the optimization approach to study the cache placement problem [33, 37, 47], either obtaining certain properties or devising numerical methods to solve the problem. The optimal placement structure has been completely characterized in [48], which shows inherit file grouping structure with at most three groups. For the MCCS, the complication in the improved delivery strategy adds challenges to the analysis, and the cache placement problem was studied only in [53]. However, the problem was numerically solved in that work, which cannot provide insight into the optimal cache placement structure.

Note that none of the above works [16, 33–37, 47, 48, 53] provided any lower bound for caching with uncoded placement to characterize the memory-rate tradeoff. The gap

between the achievable rate of either the CCS or the MCCS and the optimal caching with uncoded placement remains unknown. Most recently, the exact memory-rate tradeoff under uncoded placement for the case of two files was characterized [52]. However, the caching scheme proposed in [52] is only designed for two files, which is not extendable to general scenarios.

When files only have nonuniform sizes, the CCS has again been shown to achieve a peak rate a constant factor away from the lower bound for caching with any placement [50, 51], where the gap may be large for practical concerns. A limited number of recent works also considered joint nonuniformity in cache sizes, file popularity and sizes [33,71]. The cache placement optimization for the CCS was considered in [33], where simplification methods were developed for the optimization problem with well-performed numerical solutions. However, [33] focused on the optimization framework for the CCS, but did not address the optimality of the optimized CCS as compared to any information-theoretic lower bound. In [71], the memory-rate tradeoff has been characterized under general placement, in the case of full nonuniformity in cache size, file popularity and sizes, but only for a system of two users and two files, where a caching scheme was proposed to achieve the lower bound. Except for these recent studies, the MCCS has never been explored for files with both nonuniform popularity and sizes.

Besides the above-mentioned works, coded caching schemes and the memory-rate tradeoff for caching have also been investigated in various systems or network configurations, including heterogeneous user profiles [72–75], nonuniform cache sizes [66, 76], correlated files [77], decentralized placement for nonuniform file popularity and sizes, and cache sizes [61], heterogeneous distortion [78, 79], multi-antenna transmission and shared caches [80].

2.3 Decentralized Caching

For a system with a central server connecting to multiple cache-equipped users, a *decentralized coded caching scheme* (D-CCS) was first proposed in [18], which consists of a

decentralized (uncoded) placement scheme and a coded delivery strategy. The D-CCS has since attracted many interests, with extensions to nonuniform cache sizes [81, 82], and nonuniform file popularity [16, 35, 36] or sizes [61]. For nonuniform file popularity, existing works mainly focus on the cache placement strategies for the D-CCS [16, 35, 36, 61] to reduce the achievable rates. To quantify the performances of these proposed schemes for D-CCS, [16, 35, 36, 61] proposed different lower bounds on the average rate for caching with any placement. With the number of users requesting files (*i.e.,* active users) known at the server, it has been shown that the achievable rate of the D-CCS is within a factor away from the tightest lower bound developed in [36]. However, since the lower bound is for any caching, the gap is still large for practical consideration. These existing results [16, 35, 36, 61] are both not sufficient to characterize the memory-rate tradeoff for decentralized caching, especially for the case when the users who request files are unknown to the server.

2.4 Coded Distributed Computing

The Coded Distributed Computing (CDC) was first proposed in [83]. It enables an inverse-linear tradeoff between computation and communication load that greatly improves the conventional distributed computing. Furthermore, a coded MapReduce framework was established in [84] which exploits a particular form of coding to significantly reduce the inter-server communication load of MapReduce. Since then, the CDC has been extended to a variety of different models, including combinatorial design based CDC [85], CDC with storage constraints [86], and the wireless CDC [87, 88].

All the above works of CDC consider the homogeneous systems, where the workers have uniform mapping and reducing load. Existing works on CDC for heterogeneous systems concentrate on the heterogeneity of end workers, e.g., [89–92]. In particular, consider end users with heterogeneous computation capabilities, [89, 90] propose different file assignment scheme for a system of three end users. In [91, 92], it is shown that the communication load can be further reduced by assigning more output functions to nodes with

more input files. In [93], a heterogeneous CDC scheme is proposed that jointly considers the file allocation and function assignment. To the best of our knowledge, there is currently no CDC scheme proposed for the the case where the files have different popularities of being accessed by the jobs.

Chapter 3

Fundamental Structure of Optimal Cache Placement for Coded Caching with Nonuniform Demands

In this chapter, we obtain the optimal cache placement for the CCS under nonuniform demands and thoroughly characterize the solution structure. We identify the inherent file group structure under the optimal placement. We show that there are at most three file groups under the optimal solution, and obtain the cache placement solution in closed-form for each possible file group structure. Following this, we develop a simple and efficient algorithm to obtain the optimal cache placement solution. Using our optimal cache placement, we provide a new converse bound on the average delivery rate of caching with any placement and quantify the subpacketization level under the optimal cache placement.

3.1 System Model and Problem Setup

3.1.1 System Model

Consider a cache-aided transmission system with a server connecting to K users, each with a local cache, over a shared error-free link, as shown in Fig. 3.1. The server has a database consisting of N files, $\{W_1, \ldots, W_N\}$. Each file W_n is of size F bits and is requested with probability p_n. Let $\mathbf{p} = [p_1, \ldots, p_N]^T$ denote the popularity distribution of all N files, where $\sum_{n=1}^{N} p_n = 1$. Without loss of the generality, we label files according to

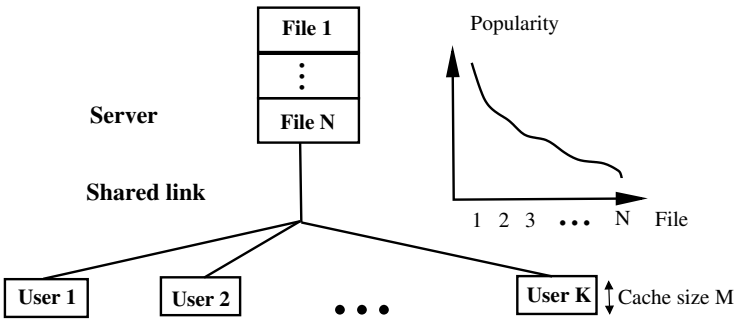

Figure 3.1: An example of cache-aided systems, where end users are connected to the central service provider through a shared link. Each user has a local cache to alleviate the burden of the shared link. The files in the server have nonuniform popularities.

the decreasing order of their popularities: $p_1 \geq p_2 \geq \cdots \geq p_N$. Each user k has a local cache of capacity MF bits, which is referred to as cache size M (normalized by the file size), where M is a real number and $M \in [0, N]$. Denote the file and user index sets by $\mathcal{N} \triangleq \{1, \ldots, N\}$ and $\mathcal{K} \triangleq \{1, \ldots, K\}$, respectively.

The coded caching operates in two phases: the cache placement phase and the content delivery phase. In the cache placement phase, a portion of uncoded file contents from $\{W_1, \ldots, W_N\}$ are placed in each user k's local cache, according to a cache placement scheme. The cached content at user k is described by a caching function $\phi_k(\cdot)$ of N files as $Z_k \triangleq \phi_k(W_1, \ldots, W_N)$. During data transmission, each user k independently requests a file with index d_k from the server. Let $\mathbf{d} \triangleq [d_1, \ldots, d_K]^T$ denote the demand vector of all K users. In the content delivery phase, based on the demand vector $\mathbf{d}$ and the cached contents at users, the server generates coded messages of uncached portions of requested files and sends them to the users. The generated codeword can be described by an encoding function $\psi_{\mathbf{d}}(\cdot)$ of the N files for demand $\mathbf{d}$ as $X_{\mathbf{d}} = \psi_{\mathbf{d}}(W_1, \ldots, W_N)$. Upon receiving the codeword, each user k applies a decoding function $\varphi_k(\cdot)$ to reconstruct its (estimated) requested file $\hat{W}_{\mathbf{d},k}$ from the received codeword and its cached content as $\hat{W}_{\mathbf{d},k} \triangleq \varphi_k(X_{\mathbf{d}}, Z_k)$. A valid coded caching scheme requires that each user k is able to reconstruct its requested file, $\hat{W}_{\mathbf{d},k} = W_{d_k}$, $k \in \mathcal{K}$, for any demand $\mathbf{d}$, over an error-free link.

3.1.2 Cache Placement Problem Construction

The cache placement is a crucial design issue in coded caching. Among existing studies for the CCS, a common approach is to propose a cache placement scheme, construct a lower bound on the minimum data rate, and evaluate the proposed scheme by comparing its performance with the lower bound. In this work, we use an optimization approach for the cache placement design for the CCS. Through construction, we formulate the cache placement problem into a design optimization problem.

Cache placement

For K users, there are 2^K user subsets in $\mathcal{K}$, with subset sizes ranging from 0 to K. Denote $\mathcal{K}_0 \triangleq \mathcal{K} \cup \{0\}$. Among all the user subsets, there are $\binom{K}{l}$ different user subsets with the same size $l \in \mathcal{K}_0$ ($l = 0$ corresponds to the empty subset $\emptyset$ in $\mathcal{K}$). They form a cache subgroup that contains all user subsets of size l, defined as $\mathcal{A}^l \triangleq \{\mathcal{S} : |\mathcal{S}| = l,\ \mathcal{S} \subseteq \mathcal{K}\}$ with $|\mathcal{A}^l| = \binom{K}{l}$, for $l \in \mathcal{K}_0$. For the N files, partition each file W_n into 2^K non-overlapping subfiles, one for each unique user subset $\mathcal{S} \subseteq \mathcal{K}$, denoted by $W_{n,\mathcal{S}}$ (it can be $\emptyset$). Each user $k \in \mathcal{S}$ stores subfile $W_{n,\mathcal{S}}$ in its local cache (for $\mathcal{S} = \emptyset$, subfile $W_{n,\emptyset}$ is not cached to any user, but only kept in the server). For any caching scheme, each file should be reconstructed by combining all its subfiles. Thus, we have the file partitioning constraint

$$\sum_{l=0}^{K} \sum_{\mathcal{S} \in \mathcal{A}^l} |W_{n,\mathcal{S}}| = F, \quad n \in \mathcal{N}. \tag{3.1}$$

It is shown in [37, Theorem 1] that for each file W_n, the size of its subfile $W_{n,\mathcal{S}}$ only depends on $|\mathcal{S}|$. This implies that $|W_{n,\mathcal{S}}|$ is the same for any $\mathcal{S} \in \mathcal{A}^l$ of the same size l. Based on this property, for each file W_n, its subfiles are grouped into file subgroups, each denoted by $\mathcal{W}_n^l = \{W_{n,\mathcal{S}} : \mathcal{S} \in \mathcal{A}^l\}$, for $l \in \mathcal{K}_0$. There are $\binom{K}{l}$ subfiles of the same size in $\mathcal{W}_n^l$ (intended for user subsets in cache subgroup $\mathcal{A}^l$), and there are total $K+1$ file subgroups.

Let $a_{n,l}$ denote the size of subfiles in $\mathcal{W}_n^l$, as a fraction of the file size F bits: $a_{n,l} \triangleq |W_{n,\mathcal{S}}|/F$ (for $\forall \mathcal{S} \in \mathcal{A}^l$), $l \in \mathcal{K}_0$, $n \in \mathcal{N}$. Note that $a_{n,0}$ represents the fraction of file W_n

that is not stored at any user's cache but only remains in the server. Then, the file partition constraint (3.1) is simplified to

$$\sum_{l=0}^{K} \binom{K}{l} a_{n,l} = 1, \quad n \in \mathcal{N}. \tag{3.2}$$

Recall that in file partitioning, each subfile is intended for a unique user subset. During the cache placement, user k stores all the subfiles in $\mathcal{W}_n^l$ that are intended for user subsets it belongs to, *i.e.,* $\{W_{n,\mathcal{S}} : \mathcal{S} \in \mathcal{A}^l \text{ and } k \in \mathcal{S}\} \subseteq \mathcal{W}_n^l$, for $l \in \mathcal{K}$. Note that in each $\mathcal{A}^l$, $l \in \mathcal{K}$, there are total $\binom{K-1}{l-1}$ different user subsets containing the same user k. Thus, there are $\sum_{l=1}^{K} \binom{K-1}{l-1}$ subfiles in each file W_n that a user can possibly store in its local cache. With subfile size $a_{n,l}$, this means that each user caches a total of $\sum_{l=1}^{K} \binom{K-1}{l-1} a_{n,l}$ fraction of file W_n. For cache size M at each user, we have the following local cache constraint

$$\sum_{n=1}^{N} \sum_{l=1}^{K} \binom{K-1}{l-1} a_{n,l} \leq M. \tag{3.3}$$

We point out that the above construction through subfile and user subset partitioning to represent an uncoded cache placement is general, *i.e.,* any uncoded cache placement scheme can be equivalently represented by the specific values of $\{a_{n,l} : n \in \mathcal{N}, l \in \mathcal{K}_0\}$.

Content Delivery via Coded Multicasting

For content delivery by the CCS, the server multicasts a unique coded message to each user subset. The message is formed by bitwise XOR operation of subfiles as

$$C_{\mathcal{S}} \triangleq \bigoplus_{k \in \mathcal{S}} W_{d_k, \mathcal{S}\backslash\{k\}}. \tag{3.4}$$

Note that the CCS originally proposed in [17] is shown to be a valid caching scheme for cache size $M = \{0, N/K, 2N/K, ..., N\}$. This conclusion can be straightforwardly extended to any cache size M, using the delivery strategy of the decentralized CCS in [18].

With nonuniform file popularities, the cache placement may be different for files with different popularities. This means the file partitioning may be different among these files, and the subfile size $a_{n,l}$ is a function of n. Note that when the sizes of subfiles are not

equal, zero padding is needed to code the subfiles together for multicasting in (3.4). As a result, the size of coded message $C_{\mathcal{S}}$ is determined by the largest subfile among subfiles in the delivery group (user subset) $\mathcal{S}$, *i.e.,*

$$|C_{\mathcal{S}}| = \max_{k \in \mathcal{S}} a_{d_k,l}, \;\; \mathcal{S} \in \mathcal{A}^{l+1}, l = 0, \ldots, K-1. \tag{3.5}$$

With (3.4) and (3.5), each user in $\mathcal{S}$ can retrieve the subfile of its requested file from the coded message $C_{\mathcal{S}}$.

3.2 Cache Placement Optimization Formulation

Based on (5.7), the average rate $\bar{R}$ of data delivery by the CCS is given by

$$\bar{R} = \mathbb{E}_{\mathbf{d}}\left[\sum_{\mathcal{S} \subseteq \mathcal{K}, \mathcal{S} \neq \emptyset} |C_{\mathcal{S}}|\right] = \mathbb{E}_{\mathbf{d}}\left[\sum_{l=0}^{K-1} \sum_{\mathcal{S} \in \mathcal{A}^{l+1}} \max_{k \in \mathcal{S}} a_{d_k,l}\right] \tag{3.6}$$

where $\mathbb{E}_{\mathbf{d}}[\cdot]$ is taken w.r.t. demand vector $\mathbf{d}$.

Let $\mathbf{a}_n = [a_{n,0}, \ldots, a_{n,K}]^T$ denote the $(K+1) \times 1$ cache placement vector for file $W_n, n \in \mathcal{N}$. The cache placement optimization problem for the CCS is formulated as obtaining the optimal $\{\mathbf{a}_n\}$ to minimize the average rate $\bar{R}$, given by[1]

$$\begin{aligned} \mathbf{P0}: \; & \min_{\{\mathbf{a}_n\}} \; \bar{R} \\ & \text{s.t.} \;\; (3.2), (3.3), \text{ and} \\ & \qquad \mathbf{a}_n \succcurlyeq \mathbf{0}, \; n \in \mathcal{N}. \end{aligned} \tag{3.7}$$

The optimization problem **P0** is complicated to solve. In the following, we provide a few simplifications to the average rate objective and the constraints and transform **P0** into a simplified equivalent problem.

3.2.1 Problem Reformulation

For nonuniform file popularities, it is shown that the optimal cache placement under the CCS has a *popularity-first* property [37]. Specifically, it states in [37, Theorem 2] that

[1]Note that **P0** is formulated for the CCS, which is based on uncoded cache placement and one-shot coded delivery with zero padding, as described in Section 3.1.2.

for file popularities $p_1 \geq \ldots \geq p_N$, under the optimal cache placement, the following condition holds for the cached subfiles

$$a_{n,l} \geq a_{n+1,l}, \quad l \in \mathcal{K},\ n \in \mathcal{N}\backslash\{N\}, \tag{3.8}$$

where the amount of cache assigned to a file is monotonic with the file popularity.

Without loss of the optimality, we now explicitly impose constraint (3.8) and have the following equivalent problem to **P0**

$$\begin{aligned} \textbf{P1}: \ & \min_{\{\mathbf{a}_n\}} \ \bar{R} \\ & \text{s.t.} \ \ (3.2), (3.3), (3.7), (3.8). \end{aligned}$$

At the optimality of **P1**, the local cache constraint (3.3) is attained with equality, *i.e.,* the cache memory is always fully utilized. To see this, note that at optimality if there is any unused memory, we can always modify the assumed optimal caching placement by adding any uncached portion of files into the unused memory. This leads to reduced $\bar{R}$, contradicting the assumption that there is unused cache memory at optimality. Thus, we replace constraint (3.3) with the equality constraint

$$\sum_{n=1}^{N}\sum_{l=1}^{K}\binom{K-1}{l-1}a_{n,l} = M. \tag{3.9}$$

Next, we show the following lemma for constraint (3.7).

Lemma 1. Under constraint (3.8), constraint (3.7) is equivalent to the following two constraints

$$a_{N,l} \geq 0,\ l \in \mathcal{K} \tag{3.10}$$

$$a_{1,0} \geq 0. \tag{3.11}$$

Proof. If $a_{N,l} \geq 0, \forall l \in \mathcal{K}$, by the popularity-first condition (3.8), we have

$$a_{n,l} \geq 0,\ \forall l \in \mathcal{K},\ \forall n \in \mathcal{N}. \tag{3.12}$$

Recall that subfile size $a_{n,0}$ represents the fraction of W_n that is not stored at any user cache. From (3.2), we have

$$a_{n,0} = 1 - \sum_{l=1}^{K} \binom{K}{l} a_{n,l},\ n \in \mathcal{N}. \tag{3.13}$$

Combining (3.8) and (3.13), we have $a_{1,0} \leq \ldots \leq a_{N,0}$. If $a_{1,0} \geq 0$ in (3.11) holds, then $a_{n,0} \geq 0$, $\forall n \in \mathcal{N}$. Combining this with (3.12), we have $\mathbf{a}_n \succcurlyeq 0$, $\forall n \in \mathcal{N}$, which is constraint (3.7). □

By Lemma 1, constraints (3.7) in **P1** can be equivalently replaced by constraints (3.10) and (3.11).

Let Y_m, $m = 1, \ldots, K$, denote the mth smallest file index in the demand vector $\mathbf{d}$. The probability distribution of Y_m is obtained in [33, Lemma 2] (the expression of Y_m is provided in Appendix A.1 for completeness). By the popularity-first property of the optimal cache placement, the average rate $\bar{R}$ in (3.6) is shown to have the following expression [33]

$$\bar{R} = \sum_{n=1}^{N} \sum_{l=1}^{K-1} \sum_{m=1}^{K} \binom{K-m}{l} \Pr[Y_m = n] a_{n,l} \ + \ \sum_{n=1}^{N} \sum_{m=0}^{K-1} \Pr[Y_{K-m} = n] a_{n,0}, \tag{3.14}$$

where $\Pr[Y_m = n]$ is not a function of $\mathbf{a}_n$. The above expression shows that $\bar{R}$ is a weighted sum of $a_{n,l}$'s (for each cache subgroup l).

From (3.14), define $\mathbf{g}_n \triangleq [g_{n,0}, \ldots, g_{n,K}]^{\mathrm{T}}$, $n \in \mathcal{N}$, where

$$\begin{aligned} g_{n,l} &\triangleq \sum_{m=1}^{K} \binom{K-m}{l} \Pr[Y_m = n], \quad l \in \mathcal{K}, \\ g_{n,0} &\triangleq \sum_{m=0}^{K-1} \Pr[Y_{K-m} = n]. \end{aligned} \tag{3.15}$$

Also, from (3.2) and (3.9), define $\mathbf{b} \triangleq [b_0, \ldots, b_K]^T$, with $b_l \triangleq \binom{K}{l}$, and $\mathbf{c} \triangleq [c_0, \ldots, c_K]^T$, with $c_l \triangleq \binom{K-1}{l+1}$, $l \in \mathcal{K}_0$. Combining the results from (3.9) to (3.15), we reformulate the cache placement optimization problem **P1** into the following equivalent LP problem

$$\textbf{P2:} \min_{\{\mathbf{a}_n\}} \ \sum_{n=1}^{N} \mathbf{g}_n^T \mathbf{a}_n$$

$$\text{s.t. } (3.8), (3.10), (3.11), \text{ and}$$

$$\mathbf{b}^T \mathbf{a}_n = 1, \ n \in \mathcal{N}, \tag{3.16}$$

$$\sum_{n=1}^{N} \mathbf{c}^T \mathbf{a}_n = M. \tag{3.17}$$

Note that compared to **P1** with $2N(K+1) - K + 1$ constraints, **P2** has $N(K+1) + 2$ constraints. Reducing the constraints facilitates us to explore the Karush-Kuhn-Tucker (KKT) optimality conditions [94] in the problem and obtain the inherent structure in the optimal cache placement.

3.3 The Optimal Cache Placement

In this section, we derive the optimal cache placement solution to **P2**. We first present a structural property of the optimal cache placement solution for **P2**. It is obtained by exploring the KKT conditions for **P2**. Based on this property, we identify several possible optimal solution structures. By analyzing each solution structure along with the file partition and cache memory constraints, we obtain the closed-form cache placement solution under each solution structure. Finally, we develop a simple low-complexity algorithm using these obtained candidate solutions to obtain the optimal solution for **P2**. We first give the definition of *file group* below.

Definition 1. (*File group*) A file group is a subset of $\mathcal{N}$ that contains all files with the same cache placement vector, *i.e.,* for any two files W_n and $W_{n'}$, if their placement vectors $\mathbf{a}_n = \mathbf{a}_{n'}$, then they belong to the same file group.

For N files, there could be potential as many as N file groups (*i.e.,* all $\mathbf{a}_n$'s are different), which makes the design of optimal cache placement a major challenge. File grouping is a popular method proposed for the CCS [16, 34–36] to simplify the cache placement design under nonuniform file popularity. Having fewer file groups reduces the complexity in determining the placement vectors $\{\mathbf{a}_n\}$. However, existing file grouping schemes are suboptimal. Our main result in Theorem 1 below describes the structural property, in terms of file groups, of the optimal cache placement for the CCS.

Theorem 1. For N files with any file popularity distribution $\mathbf{p}$, and for any K and $M \leq N$, there are at most three file groups under the optimal cache placement $\{\mathbf{a}_n\}$ for **P1**.

Proof. Since **P2** is an LP, we explore the KKT conditions for **P2** to derive the file group property. See Appendix A.2. □

Theorem 1 indicates that, regardless of the values of N, $\mathbf{p}$, K, and M, there are only three possible file group structures under the optimal cache placement, *i.e.,* one to three file groups. This implies that there are at most three unique vectors among the optimal cache placement vectors $\{\mathbf{a}_n\}$, one for each file group. This property drastically reduces the complexity in solving the cache placement problem, and in turn, it allows us to explore the solution structure to obtain the optimal solution $\{\mathbf{a}_n\}$ analytically. The result of at most three file groups, regardless of file popularity distribution $\mathbf{p}$ among N files, is somewhat surprising. We will provide some insight into this result in Section 3.3.4, after the cache placement structure and solution are obtained.

Remark 1. Existing file grouping strategies [16, 34–36] are either suboptimal or designed for a specific file popularity distribution. Some of these suboptimal file grouping strategies [34–36] were shown to be a constant factor away from the optimum in terms of the average rate. Since the constant factor is relatively large, it remains unclear how close their performance is to that under the optimal cache placement strategy for the CCS. Furthermore, under a file grouping strategy, the specific cache placement for each group is needed. Existing works use the symmetric decentralized cache placement strategy for each group. In contrast, by Theorem 1, in the following, we will discuss each of the three file grouping cases to obtain the corresponding optimal placement.

Following Theorem 1, we will examine all three cases of file groups for **P2** to obtain the placement solution. We first introduce the following notations to be used later:

- Denote $\bar{\mathbf{a}}_n = [a_{n,1}, \ldots, a_{n,K}]^T$ as the sub-placement vector in $\mathbf{a}_n$. It specifies only the size of each subfile stored in the local cache, while $a_{n,0}$ specifies the subfile kept at the server.

- We use notation $\bar{\mathbf{a}}_n \succcurlyeq_1 \mathbf{0}$ to indicate that there is at least one positive element in $\bar{\mathbf{a}}_n$ and the other elements are 0; otherwise, $\bar{\mathbf{a}}_n = \mathbf{0}$. Similarly, $\bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_2}$ denotes that at least one element in $\bar{\mathbf{a}}_{n_1}$ is greater than that in $\bar{\mathbf{a}}_{n_2}$ and all the rest elements in $\bar{\mathbf{a}}_{n_1}$ and $\bar{\mathbf{a}}_{n_2}$ are equal; otherwise $\bar{\mathbf{a}}_{n_1} = \bar{\mathbf{a}}_{n_2}$.

With the above notations, we establish the following equivalence on the placement vectors:

1. By (3.2), for any two files n_1 and n_2, we have

$$\mathbf{a}_{n_1} = \mathbf{a}_{n_2} \;\Leftrightarrow\; \bar{\mathbf{a}}_{n_1} = \bar{\mathbf{a}}_{n_2} \tag{3.18}$$

   where "$\Leftrightarrow$" denotes being equivalent.

2. By (3.8) and (3.13), for any two files with their indexes $n_1 < n_2$, we have

$$\mathbf{a}_{n_1} \neq \mathbf{a}_{n_2} \;\Leftrightarrow\; \bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_2} \text{ and } a_{n_1,0} < a_{n_2,0}. \tag{3.19}$$

In the following, we consider each case of file groups, and identify the complete structure of the cache placement vector and obtain the optimal solution for this case.

3.3.1 One File Group

With a single file group, the cache placement vectors are the same for all files. Let $\mathbf{a}_1 = \cdots = \mathbf{a}_N = \mathbf{a}$. In this case, we can simplify the expressions in **P2**. Denote $\tilde{\mathbf{g}} \triangleq [\tilde{g}_0, \ldots, \tilde{g}_K]^T$ with $\tilde{g}_l = \binom{K}{l+1}, l \in \mathcal{K}_0$. Then, **P2** is simplified into the following equivalent problem

$$\begin{aligned}
\textbf{P3:} \min_{\mathbf{a}} \;& \tilde{\mathbf{g}}^T\mathbf{a} \\
\text{s.t.} \;& \mathbf{b}^T\mathbf{a} = 1, && (3.20)\\
& \mathbf{c}^T\mathbf{a} = \frac{M}{N}, && (3.21)\\
& \mathbf{a} \succcurlyeq \mathbf{0}. && (3.22)
\end{aligned}$$

Note that **P3** is the same as the cache placement optimization problem for the uniform file popularity case (the same placement vector $\mathbf{a}$ for all files), of which the optimal solution has been obtained in [33] in closed-form. To summarize, the optimal $\mathbf{a}$ for **P3** is given as follows:

i) If $MK/N \in \mathbb{N}$: The optimal $\mathbf{a}$ has only one nonzero element: $a_{l_o} = 1/\binom{K}{l_o}$, $l_o = MK/N$, and $a_l = 0, \forall\, l \neq l_o$.

ii) If $MK/N \notin \mathbb{N}$: The optimal $\mathbf{a}$ has two nonzero adjacent elements: Let $v \triangleq \frac{KM}{N}$. Then,

$$\begin{aligned} &a_{l_o} = \frac{1 + \lfloor v \rfloor - v}{\binom{K}{\lfloor v \rfloor}}, \;\; a_{l_o+1} = \frac{v - \lfloor v \rfloor}{\binom{K}{\lceil v \rceil}}, \quad l_o = \lfloor v \rfloor \\ &a_l = 0, \quad \forall\, l \neq l_o \text{ or } l_o + 1. \end{aligned} \tag{3.23}$$

Note that Case i) is a special case of Case ii): In Case ii), if $l_o = v$, $a_{l_o+1} = 0$, the solution in (3.23) reduces to that of case i). Thus, the optimal solution of **P3** can be simply summarized in (3.23).

The above shows that the optimal $\mathbf{a}$ has at most two nonzero elements. When MK/N is an integer, $\mathbf{a}$ has only one nonzero element, which means each file is partitioned into equal subfiles of size a_{l_o}. Otherwise, $\mathbf{a}$ has two nonzero adjacent elements, which means each file is partitioned into subfiles of two different sizes a_{l_o} and a_{l_o+1}. Each subfile is cached into its intended user subset of size l_o or $l_o + 1$, as described in Section 3.1.2. Fig. 3.2 illustrates the optimal $\mathbf{a}$ in the one-file-group case.

3.3.2 Two File Groups

For the case of two file groups, there are only two unique placement vectors in $\{\mathbf{a}_n\}$. By (3.8), this implies that $\{\mathbf{a}_n\}$ has the following structure: $\mathbf{a}_1 = \ldots = \mathbf{a}_{n_o} \neq \mathbf{a}_{n_o+1} = \ldots = \mathbf{a}_N$, for some $n_o \in \{1, \ldots, N-1\}$. By (3.18) and (3.19), this is equivalent to

$$\begin{cases} \bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_N \\ a_{1,0} = \ldots = a_{n_o,0} < a_{n_o+1,0} = \cdots = a_{N,0} \end{cases} \tag{3.24}$$

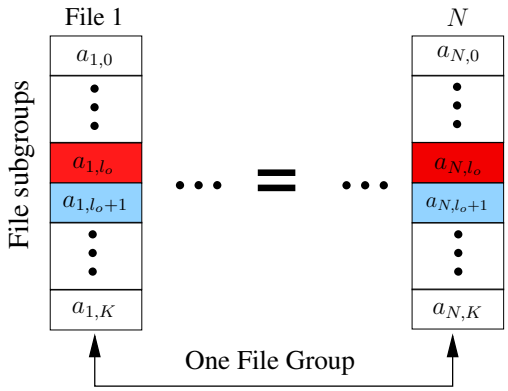

Figure 3.2: An example of the optimal cache placement for one file group: $\mathbf{a}_n = \mathbf{a}$, $\forall n$, with $a_{l_o}, a_{l_o+1} > 0$ and $a_l = 0, \forall l \neq l_o, l_o + 1$. (The same color indicates the same value of a_l)

for some $n_o \in \{1, \ldots, N-1\}$. It immediately follows that $a_{n_o+1,0} = \cdots = a_{N,0} > 0$. We use $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$ to represent the two unique placement vectors for the first and the second file group, respectively. We first characterize the structure of the placement vector $\mathbf{a}_{n_o+1}$ for the second file group below.

Proposition 1. If there are two file groups under the optimal cache placement $\{\mathbf{a}_n\}$, the optimal sub-placement vector $\bar{\mathbf{a}}_{n_o+1}$ for the second file group has at most one nonzero element.

Proof. See Appendix A.3. □

Proposition 1 indicates that either $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$ or $\bar{\mathbf{a}}_{n_o+1}$ has only one nonzero element. For the former, it means the files in the second file group are not cached but remain at the server only. Note that two file groups were considered for placement strategies in [35,36], where the second file group containing less popular files remains at the server, and the location of n_o for the grouping was proposed in different heuristic ways. These file grouping methods fall into the case of $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$. However, the case of allocating cache to the second file group, *i.e.,* $\bar{\mathbf{a}}_{n_o+1} \neq \mathbf{0}$, has never been considered in the literature.

Following Proposition 1, we obtain the optimal cache placement in each of the two cases for $\bar{\mathbf{a}}_{n_o+1}$ below:

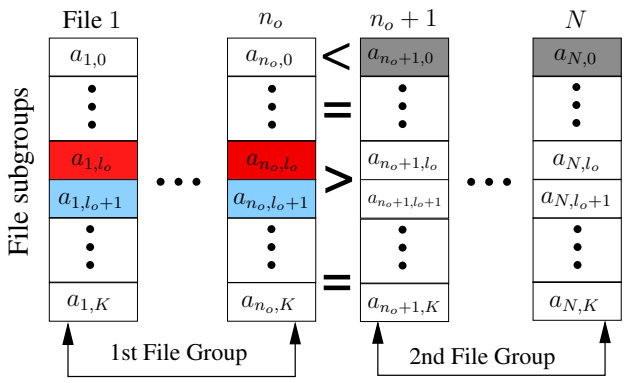

Figure 3.3: An example of the optimal cache placement for two file groups with $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$. The 1st file group: $a_{n,l_o} > 0$, $a_{n,l_o+1} > 0$, for $n = 1, \ldots, n_o$, and the rest are all 0's. The second file group: $a_{n_o+1,0} = \cdots = a_{N,0} = 1$.

$\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$

By (3.2), we have $a_{n_o+1,0} = 1$. It means that no cache is allocated to the second file group, and the entire cache is given to the first file group. It follows that the cache placement problem for $\mathbf{a}_{n_o}$ of the first group is reduced to that in the one-file-group case in Section 3.3.1. Specifically, we can treat the first file group as a new database consisting of these n_o files, for some $n_o \in \{1, \ldots, N-1\}$. Then, the cache placement optimization problem for $\mathbf{a}_{n_o}$ is the same as **P3**, except that N is replaced by n_o in constraint (3.21). It follows that, the optimal solution is the same as in (3.23), except that N is replaced by n_o, and $v = MK/n_o$, *i.e.,*

$$\begin{cases} a_{n_o,l_o} = \dfrac{1 + \lfloor v \rfloor - v}{\binom{K}{\lfloor v \rfloor}}, \; a_{n_o,l_o+1} = \dfrac{v - \lfloor v \rfloor}{\binom{K}{\lceil v \rceil}}, \; l_o = \lfloor v \rfloor \\ a_{n_o,l} = 0, \quad \forall\, l \neq l_o \text{ or } l_o + 1. \end{cases} \tag{3.25}$$

An example of the placement $\{\mathbf{a}_n\}$ of files in this case is shown in Fig. 3.3, where $\mathbf{a}_{n_o}$ for the first file group has two adjacent nonzero elements. In addition, for this case, Fig. 3.4 illustrates the actual file partitions and cached contents in user 1.

Based on the similarity of the solutions in (3.23) and (3.25), we can extend the two-file-group case to also include one file group as a special case where $n_o = N$. As a result, for

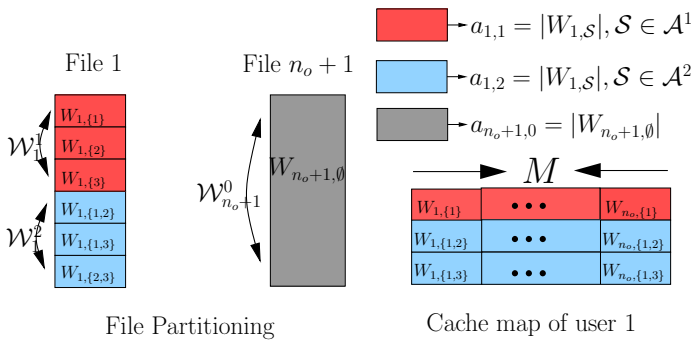

Figure 3.4: An illustration of file partition and cache placement based on the placement structure in Fig. 3.3, for $K = 3$ users, and $l_o = 1$. File W_1 in the 1st file group is partitioned into subfiles of two sizes $a_{1,1}$ and $a_{1,2}$. Subfiles in file subgroup $\mathcal{W}_1^1$ with size $a_{1,1} = |W_{1,\mathcal{S}}|/F$ (red) is placed in user subset $\mathcal{S} \in \mathcal{A}^1 = \{\{1\},\{2\},\{3\}\}$; Subfiles in file subgroup $\mathcal{W}_1^2$ with size $a_{1,2} = |W_{1,\mathcal{S}}|/F$ (blue) is placed in user subset $\mathcal{S} \in \mathcal{A}^2 = \{\{1,2\},\{1,3\},\{2,3\}\}$. For file W_{n_o+1} in the second file group, the entire file is stored solely in the server: $W_{n_o+1,\emptyset} = W_{n_o+1}$, $a_{n_o+1,0} = 1$. The cache memory map of user 1 shows the stored subfiles of the 1st file group $\{W_1, \ldots, W_{n_o}\}$.

Algorithm 1 The Cache Placement for the Extended Two-File-Group Case with $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$ (including one file group)

Input: K, M, N, and $\mathbf{p}$.
Output: $(\bar{R}_{\min}, n_o^*)$

1: **for** n_o = 1 to N **do**
2: Set $l_o = \lfloor \frac{MK}{n_o} \rfloor$; Set $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$, if $n_o < N$.
3: Determine $\mathbf{a}_{n_o}$ by (3.25).
4: Compute $\bar{R}_1(n_o)$ using (3.14), by replacing N with n_o in (3.14).
5: **end for**
6: Compute $\bar{n}_o^* = \text{argmin}_{n_o \in \mathcal{N}} \bar{R}_1(n_o)$; Set $\bar{R}_{\min} = \bar{R}_1(n_o^*)$.

the extended two-file-group case, the optimal cache placement solution is given by (3.25), for $n_o \in \{1, \ldots, N\} = \mathcal{N}$. What remains is to obtain the optimal n_o^* to determine $\{\mathbf{a}_n\}$ that minimizes the average rate objective in **P2**. The optimal n_o^* is the location to determine the file groups. It depends on $(N, \mathbf{p}, M, K)$ and is challenging to obtain analytically. Nonetheless, $\bar{R}$ can be easily computed using (3.25) for $n_o \in \mathcal{N}$, and we can conduct a search for n_o to determine n_o^* that gives the minimum $\bar{R}$. The algorithm to obtain the placement solution $\{\mathbf{a}_n\}$ in this case is summarized in Algorithm 1. Through a 1-D search for the optimal n_o^*, the algorithm computes $\bar{R}$ using the closed-form expression in (3.14) by N times.

$\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$

In this case, by Proposition 1, $\bar{\mathbf{a}}_{n_o+1}$ has only one nonzero element. Assume $a_{n_o+1,l_o} > 0$, for some $l_o \in \mathcal{K}$, and $a_{n_o+1,l} = 0$, $\forall l \neq l_o$, $l \in \mathcal{K}$. We have the following propositions describing the properties of $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$. Proposition 2 specifies the differences of $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ for the two file groups, and Proposition 3 characterizes the placement $\mathbf{a}_{n_o}$ for the first file group.

Proposition 2. If there are two file groups under the optimal cache placement $\{\mathbf{a}_n\}$, and $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$, for some $n_o \in \{1, \ldots, N-1\}$, then $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ are different by only one element.

Proof. See Appendix A.4. □

Proposition 3. If there are two file groups under the optimal cache placement $\{\mathbf{a}_n\}$, and $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$, for some $n_o \in \{1, \ldots, N-1\}$, then $a_{n_o,0} = 0$.

Proof. See Appendix A.5. □

Proposition 3 indicates that each file in the first file group has all its subfiles cached among K users, and no subfile solely remains in the server. Recall in this case that $\bar{\mathbf{a}}_{n_o+1}$ has only one nonzero element $a_{n_o+1,l_o} > 0$. By Proposition 2, the different element between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ can be either at index l_o or some l_1, for $l_1 \neq l_o$. By the popularity-first property in (3.8), either of the following two cases holds: 2.i) $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$; or 2.ii) $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$, for some $l_1 \neq l_o$, $l_1 \in \mathcal{K}$. The structure of $\{\mathbf{a}_n\}$ in Case 2.i) and Case 2.ii) is illustrated in Figs. 3.5 and 3.6, respectively. We point out that l_o and l_1 are not necessarily adjacent to each other. Now we derive the solution $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ in each of these two cases:

Case 2.i) $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$:
In this case, $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ are only different at the l_oth nonzero element in $\bar{\mathbf{a}}_{n_o+1}$. It follows that $a_{n_o,l} = a_{n_o+1,l} = 0$, $\forall l \neq l_o$, $l \in \mathcal{K}$. By Proposition 3, we conclude that a_{n_o,l_o} is the only nonzero element in $\mathbf{a}_{n_o}$. From (3.16) and (3.17), we have

$$b_{l_o} a_{n_o,l_o} = 1$$

$$n_o c_{l_o} a_{n_o,l_o} + (N - n_o) c_{l_o} a_{n_o+1,l_o} = M. \tag{3.26}$$

Solving (3.26) and substituting the expressions of b_{l_o} and c_{l_o} defined below (3.15), we have

$$a_{n_o,l_o} = \frac{1}{\binom{K}{l_o}}, \quad a_{n_o+1,l_o} = \frac{1}{\binom{K}{l_o}} \left(\frac{\frac{KM}{l_o N} - \frac{n_o}{N}}{1 - \frac{n_o}{N}} \right). \tag{3.27}$$

By the condition of Case 2.i) $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$, (3.27) is only valid if $n_o < KM/l_o < N$, for $l_o \in \mathcal{K}$. Thus, the range of l_o for this case to be a valid candidate for the optimal placement is

$$\left\lfloor \frac{KM}{N} \right\rfloor + 1 \leq l_o \leq \min \left\{ K, \left\lceil \frac{KM}{n_o} \right\rceil - 1 \right\}. \tag{3.28}$$

Finally, $a_{n,0}$'s can be obtained by (3.13). To summarize, the placement solution $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ in this case is given by

$$a_{n_o,l_o} = \frac{1}{\binom{K}{l_o}}, \quad a_{n_o,l} = 0, \ \forall\, l \neq l_o \tag{3.29}$$

$$\begin{cases} a_{n_o+1,0} = \dfrac{1 - \frac{KM}{l_o N}}{1 - \frac{n_o}{N}}, \quad a_{n_o+1,l_o} = \dfrac{1}{\binom{K}{l_o}} \left(\dfrac{\frac{KM}{l_o N} - \frac{n_o}{N}}{1 - \frac{n_o}{N}} \right) \\ a_{n_o+1,l} = 0, \ \forall\, l \neq 0 \text{ or } l_o \end{cases} \tag{3.30}$$

where l_o satisfies (3.28), and $n_o \in \{1, \ldots, N-1\}$.

Fig. 3.5 illustrates the above result in this case under two file groups as the optimal placement, where different color blocks indicate the different values of $\{a_{n,l}\}$.

Case 2.ii) $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$, $l_1 \neq l_o$:

In this case, the l_oth element in $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ are identical, and we have $a_{n_o,l_o} = a_{n_o+1,l_o} > 0$. Since $a_{n_o,0} = 0$ by Proposition 3, we conclude that $\mathbf{a}_{n_o}$ has two nonzero elements a_{n_o,l_o} and a_{n_o,l_1}. Also, recall from (3.24) that $a_{n_o+1,0} > 0$. Thus, $\mathbf{a}_{n_o+1}$ has two nonzero elements $a_{n_o+1,0}$ and a_{n_o+1,l_o}. The rest elements $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$ are all zeros. The placement structure of $\{\mathbf{a}_n\}$ in this case is illustrated in Fig. 3.6, where nonzero elements in $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$ are shown as colored blocks and zero elements as uncolored blocks. Given the structure of $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$, by (3.16) and (3.17), we have

$$b_{l_o} a_{n_o,l_o} + b_{l_1} a_{n_o,l_1} = 1, \ a_{n_o+1,0} + b_{l_o} a_{n_o,l_o} = 1 \tag{3.31}$$

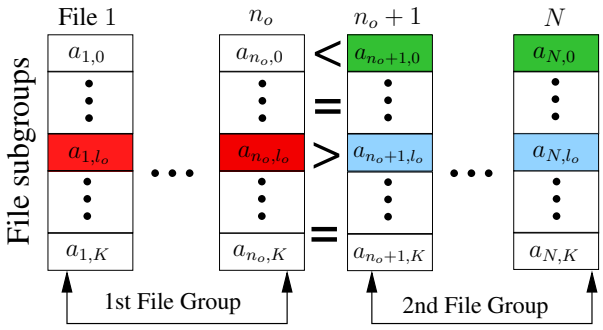

Figure 3.5: An example of the optimal cache placement for two file groups with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$: i) $0 = a_{n_o,0} < a_{n_o+1,0} < 1$. ii) Between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$: $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$; $a_{n_o,l} = a_{n_o+1,l} = 0, \forall l \in \mathcal{K}, l \neq l_o$.

$$Nc_{l_o}a_{n_o,l_o} + n_o c_{l_1} a_{n_o,l_1} = M. \tag{3.32}$$

Solving (3.31) and (3.32), and substituting the expressions of b_l and c_l given below (3.15), we obtain the solution of $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ as

$$\begin{cases} a_{n_o,l_o} = \dfrac{1}{\binom{K}{l_o}} \dfrac{\frac{KM}{l_oN} - \frac{l_1 n_o}{l_o N}}{1 - \frac{l_1 n_o}{l_o N}}, \; a_{n_o,l_1} = \dfrac{1}{\binom{K}{l_1}} \dfrac{1 - \frac{KM}{l_o N}}{1 - \frac{l_1 n_o}{l_o N}} \\ a_{n_o,l} = 0, \; \forall \, l \neq l_o \text{ or } l_1, \end{cases} \tag{3.33}$$

$$\begin{cases} a_{n_o+1,l_o} = a_{n_o,l_o}, \qquad a_{n_o+1,0} = \dfrac{1 - \frac{KM}{l_o N}}{1 - \frac{l_1 n_o}{l_o N}} \\ a_{n_o+1,l} = 0, \forall \, l \neq 0 \text{ or } l_o \end{cases} \tag{3.34}$$

where for a_{n_o,l_o}, a_{n_o,l_1}, and $a_{n_o+1,0}$ being all positive, l_o and l_1 should satisfy one of the following constraints

C1) $l_o > KM/N$ and $l_1 < KM/n_o$, or

C2) $l_o < KM/N$ and $l_1 > KM/n_o$.

Note that, if $n_o \leq M$, only constraint (C1) is valid.

In summary, for the case of two file group with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$, by (3.24), the placement $\{\mathbf{a}_n\}$ are determined via $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ in Cases 2.i) and 2.ii) for given (n_o, l_o) or (n_o, l_o, l_1), respectively. Since (n_o, l_o) can be viewed as a special case of (n_o, l_o, l_1) for $l_1 = l_o$, to unify the notations for different cases, we define $(n_o, l_o, l_o) \triangleq (n_o, l_o)$. As a result, the

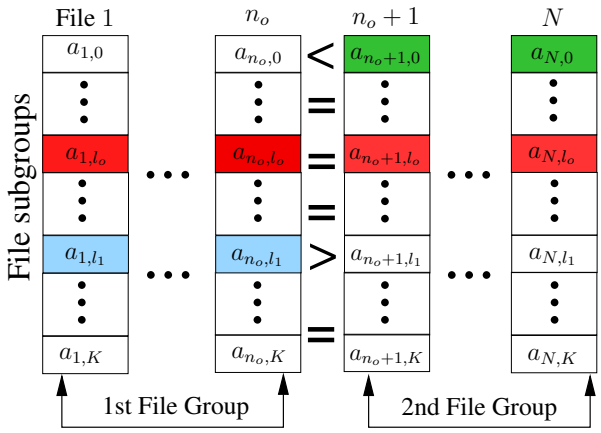

Figure 3.6: An example of the optimal cache placement $\{\mathbf{a}_n\}$ in the case of two file groups with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$: $a_{n_o+1,0} > a_{n_o,0} = 0$. Between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$: 1) $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$; 2) $a_{n_o,l_o} = a_{n_o+1,l_o} > 0$; 3) $a_{n_o,l} = a_{n_o+1,l} = 0, \forall l \in \mathcal{K}, l \neq l_o, l_1$.

average rate $\bar{R}$ in **P2** is a function of (n_o, l_o, l_1). To obtain the best tuple (n_o, l_o, l_1) that results in minimum $\bar{R}$, we can search over all possible values of $n_o \in \{1, \ldots, N-1\}$ and $l_o, l_1 \in \mathcal{K}$ within their respective range constraint in each case. The detail of obtaining the best solution $\{\mathbf{a}_n\}$ is summarized in Algorithm 2. In the algorithm, we express $\bar{R}$ explicitly as $\bar{R}(n_o, l_o, l_1)$ to emphasize its dependency on (n_o, l_o, l_1). It computes $\bar{R}(n_o, l_o, l_1)$ using the closed-form expression in (3.14) for at most $(N-1)K^2$ times in the worst case at different (n_o, l_o, l_1), which can be done in parallel. Thus, the complexity of the algorithm is very low.

Remark 2. In the case of two file groups, the first possible structure of the optimal placement $\{\mathbf{a}_n\}$ is described in Section 3.3.2: All the cache is allocated to the first group, and the cache placement for files in this group is identical, *i.e.,* symmetric placement, regardless of having different file popularities among them. As mentioned earlier, this file grouping case has been considered in [35] and [36] for a decentralized cache placement, with different methods proposed to determine the location of n_o. In [35], for files with Zipf distribution, the selection of n_o results in the performance being a constant away from that of the optimal placement. In [36], for an arbitrary file popularity distribution, the choice of n_o results in a suboptimal caching strategy. In contrast, we provide the optimal cache placement $\{\mathbf{a}_n\}$ in

Algorithm 2 The Cache Placement for Two File Groups with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$

Input: K, M, N, and $\mathbf{p}$
Output: $(\bar{R}_{\min}, n_o^*, l_o^*, l_1^*)$

for $n_o = 1$ to $N-1$ **do**
  for $l_o = \lfloor \frac{KM}{N} \rfloor + 1$ to $\min\{K, \lceil \frac{KM}{n_o} \rceil - 1\}$ **do**
    Compute $\{\mathbf{a}_n\}$ by (3.29) and (3.30).
    Compute $\bar{R}(n_o, l_o, l_o)$ by (3.14).
  end for
  for $l_o = \lfloor \frac{KM}{N} \rfloor + 1$ to K **do**
    for $l_1 = 1$ to $\min\{K, \lceil \frac{KM}{n_o} \rceil - 1\}$ **do**
      Compute $\{\mathbf{a}_n\}$ by (3.33) and (3.34).
      Compute $\bar{R}(n_o, l_o, l_1)$ by (3.14).
    end for
  end for
  for $l_o = 1$ to $\lfloor \frac{KM}{N} \rfloor$ **do**
    for $l_1 = \lceil \frac{KM}{n_o} \rceil$ to K **do**
      Compute $\{\mathbf{a}_n\}$ by (3.33) and (3.34).
      Compute $\bar{R}(n_o, l_o, l_1)$ by (3.14).
    end for
  end for
end for
Compute $(n_o^*, l_o^*, l_1^*) = \text{argmin}_{(n_o, l_o, l_1)} \bar{R}(n_o, l_o, l_1)$.
Set $\bar{R}_{\min} = \bar{R}(n_o^*, l_o^*, l_1^*)$.

Algorithm 1. The second possible structure of $\{\mathbf{a}_n\}$ is shown in Section 3.3.2 in two possible cases, where each file in the second file group is partly cached and partly remains at the server. Different from the first structure, in this case, coding opportunity between the two file groups is explored to minimize the average rate. We provide Algorithm 2 to determine the optimal cache placement $\{\mathbf{a}_n\}$. This placement structure has never been considered in the literature. Depending on $(N, \mathbf{p}, M, K)$, this placement structure may lead to a higher caching gain and lower rate than the first one, as we will show in the simulation.

3.3.3 Three File Groups

Similar to the case of two file groups, when there are three file groups under the optimal cache placement $\{\mathbf{a}_n\}$, we have three unique values among $\mathbf{a}_n$'s as $\mathbf{a}_1 = \ldots = \mathbf{a}_{n_o} \neq \mathbf{a}_{n_o+1} = \ldots = \mathbf{a}_{n_1} \neq \mathbf{a}_{n_1+1} = \ldots = \mathbf{a}_N$, for $1 \leq n_o < n_1 \leq N-1$. We use $\mathbf{a}_{n_o}$, $\mathbf{a}_{n_1}$

and $\mathbf{a}_{n_1+1}$ to represent the three unique placement vectors for the first, second, and third file group, respectively. We first determine the cache placement $\mathbf{a}_{n_1+1}$ in the 3rd file group below.

Proposition 4. If there are three file groups under the optimal cache placement $\{\mathbf{a}_n\}$, the optimal placement vector $\mathbf{a}_{n_1+1}$ for the third file group is given by $\bar{\mathbf{a}}_{n_1+1} = \mathbf{0}$, and $a_{n_1+1,0} = 1$.

Proof. See Appendix A.6. □

Proposition 4 indicates that when there are three file groups under the optimal placement, all the cache will be allocated to the first two file groups; the files in the 3rd file group solely remain in the server and are not cached to any user. Following this, we only need to obtain the two unique cache placement vectors $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_1}$ in the first two groups, respectively.

Note that since $\mathbf{a}_{n_1} \neq \mathbf{a}_{n_1+1}$, similar to (3.24), we have $\bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_1+1} = \mathbf{0}$ and $a_{n_1,0} < a_{n_1+1,0} = 1$. As a result, the cache placement $(\mathbf{a}_{n_o}, \mathbf{a}_{n_1})$ is the same as that of the two-file-group case with $\bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \mathbf{0}$ for the second file group in Section 3.3.2, where N is replaced by n_1. Specifically, for $\bar{\mathbf{a}}_{n_1} \succcurlyeq_1 0$, by Propositions 1 and 2, we conclude that $\bar{\mathbf{a}}_{n_1}$ has one nonzero element, and $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_1}$ are different by one element. Assume $a_{n_1,l_o} > 0$, for some $l_o \in \mathcal{K}$. The different element in $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_1}$ can be either at l_o with $a_{n_o,l_o} > a_{n_1,l_o}$ (as shown in Fig. 3.7), or at $l_1 \neq l_o$ for $l_1 \in \mathcal{K}$, with $a_{n_o,l_1} > a_{n_1,l_1} = 0$ (as shown in Fig. 3.8). Detailed solution for $(\mathbf{a}_{n_o}, \mathbf{a}_{n_1})$ in each case can be obtained from Section 3.3.2, summarized as follows:

When $a_{n_o,l_o} > a_{n_1,l_o} > 0$

Following (3.29) and (3.30), we have

$$a_{n_o,l_o} = \frac{1}{\binom{K}{l_o}}, \quad a_{n_o,l} = 0, \ \forall\, l \neq l_o \tag{3.35}$$

$$\begin{cases} a_{n_1,0} = \dfrac{1 - \frac{KM}{l_o n_1}}{1 - \frac{n_o}{n_1}}, \ a_{n_1,l_o} = \dfrac{1}{\binom{K}{l_o}} \left(\dfrac{\frac{KM}{l_o n_1} - \frac{n_o}{n_1}}{1 - \frac{n_o}{n_1}} \right) \\ a_{n_1,l} = 0, \ \forall\, l \neq 0 \text{ or } l_o \end{cases} \tag{3.36}$$

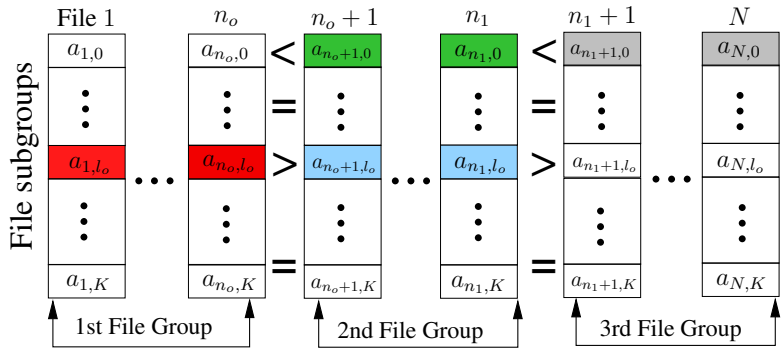

Fig. 3.7: An example of the optimal cache placement $\{\mathbf{a}_n\}$ in the case of three file groups. No cache is allocated to the 3rd file group: $a_{n_1+1,0} = 1$. For $\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1}$ in the first and second groups: $1 > a_{n_o+1,0} > a_{n_o,0} = 0$; $a_{n_o,l_o} > a_{n_o+1,l_o} > 0, l_o \in \mathcal{K}$; $a_{n_o,l} = a_{n_o+1,l} = 0$, $\forall l \in \mathcal{K}, l \neq l_o$.

where $\left\lfloor \frac{KM}{n_1} \right\rfloor + 1 \leq l_o \leq \min\left\{K, \left\lceil \frac{KM}{n_o} \right\rceil - 1\right\}$ for this case to be valid. Note that the condition for l_o can be satisfied only if $n_1 > M$. Thus, this case is possible for the optimal placement $\{\mathbf{a}_n\}$ only if $n_1 > M$.

When $a_{n_o,l_1} > a_{n_1,l_1} = 0$

From (3.33) and (3.34), we have

$$\begin{cases} a_{n_o,l_o} = \dfrac{1}{\binom{K}{l_o}} \dfrac{\frac{KM}{l_o n_1} - \frac{n_o}{n_1}}{1 - \frac{n_o}{n_1}}, \; a_{n_o,l_1} = \dfrac{1}{\binom{K}{l_1}} \dfrac{1 - \frac{KM}{l_o n_1}}{1 - \frac{l_1 n_o}{l_o n_1}} \\ a_{n_o,l} = 0, \; \forall \, l \neq l_o \text{ or } l_1, \end{cases} \tag{3.37}$$

$$\begin{cases} a_{n_1,l_o} = a_{n_o,l_o}, \qquad a_{n_1,0} = \dfrac{1 - \frac{KM}{l_o N}}{1 - \frac{l_1 n_o}{l_o n_1}} \\ a_{n_1,l} = 0, \forall \, l \neq 0 \text{ or } l_o \end{cases} \tag{3.38}$$

where l_o and l_1 need to satisfy one of the two conditions

C1') $l_o > KM/n_1$ and $l_1 < KM/n_o$, or

C2') $l_o < KM/n_1$ and $l_1 > KM/n_o$.

Since $l_o, l_1 \in \mathcal{K}$, to further analyze the above two conditions for l_o and l_1, we note that

- If $n_o < n_1 \leq M$: neither C1') nor C2') can be satisfied;

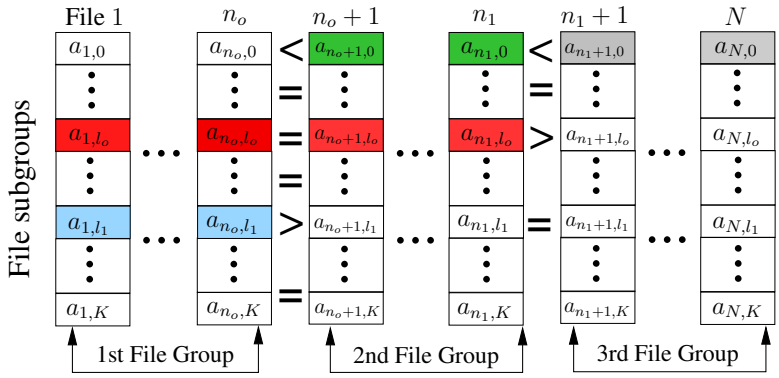

Fig. 3.8: An example of the optimal cache placement $\{\mathbf{a}_n\}$ in the case of three file groups. No cache is allocated to the 3rd file group: $a_{n_1+1,0} = 1$. For $\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1}$ in the first and second groups: 1) $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$; 2) $a_{n_o,l_o} = a_{n_o+1,l_o} > 0$; 3) $a_{n_o,l} = a_{n_o+1,l} = 0$, $\forall l \in \mathcal{K}, l \neq l_o, l_1$.

- If $n_o \leq M < n_1$: only C1') can be satisfied;
- If $M < n_o < n_1$: both C1') and C2') are possible.

As a result, Case 2) is only possible for the optimal placement $\{\mathbf{a}_n\}$ if $n_1 > M$.

The structure of $\{\mathbf{a}_n\}$ in Cases 1) and 2) are illustrated in Figs. 3.7 and 3.8, respectively, where the colored blocks indicate the nonzero elements in $\mathbf{a}_n$.

Remark 3. From Cases 1) and 2) above, we conclude that if the optimal placement results in three file groups, we must have $n_1 > M$. This result is consistent with our intuition: By Proposition 4, all the cache is allocated to the first two file groups. To maximally use the cache, the files to be cached (in the first two groups) must be no less than M files.

Based on the above discussion, for the case of three file groups, given (n_o, n_1, l_o, l_1), the solution $\{\mathbf{a}_n\}$ is obtained in closed-form, and so the average rate $\bar{R}$ in **P2** can be computed by (3.14) as a function of (n_o, n_1, l_o, l_1). Again, we can search over all possible values of $n_1 \in \{M+1, N-1\}$, $n_o \in \{1, \ldots, n_1 - 1\}$, and $l_o, l_1 \in \mathcal{K}$ within the range specified in Cases 1) and 2), to obtain the best tuple (n_o, n_1, l_o, l_1) that gives minimum $\bar{R}$. Algorithm 3 summarizes the steps to obtain the best placement solution $\{\mathbf{a}_n\}$ for three file groups. It uses Algorithm 2 to obtain the best tuple (n_o, l_o, l_1) in the two-

Algorithm 3 The Cache Placement for Three File Groups

Input: K, M, N, and $\mathbf{p}$

Output: $(\bar{R}_{\min}, n_o^*, n_1^*, l_o^*, l_1^*)$

1: **for** $n_1 = M$ to $N-1$ **do**
2: $\bar{R}_1(n_o, n_1, l_o, l_1) =$
3: Algorithm 2$(K, M, n_1, [p_1, \ldots, p_{n_1}]^T)$;
4: $\bar{R}_2(n_1) = \sum_{n=n_1+1}^{N} g_{n,0}$;
5: Compute $\bar{R}(n_o, n_1, l_o, l_1) = \bar{R}_1 + \bar{R}_2$
6: **end for**
7: Compute $(n_o^*, n_1^*, l_o^*, l_1^*) = \text{argmin}_{n_o, n_1, l_o, l_1} \bar{R}(n_o, n_1, l_o, l_1)$;
8: Set $\bar{R}_{\min} = \bar{R}(n_o^*, n_1^*, l_o^*, l_1^*)$.

Algorithm 4 The Optimal Cache Placement Solution for **P1**

Input: K, M, N, and $\mathbf{p}$

Output: $\bar{R}_{\min}$, $\{\mathbf{a}_1, \ldots, \mathbf{a}_N\}$

1: Run Algorithms 1, 2 and 3.
2: Find the minimum output $\bar{R}_{\min}$ among the outputs of Algorithms 1–3.
3: Set the corresponding placement $\{\mathbf{a}_1, \ldots, \mathbf{a}_N\}$ for $\bar{R}_{\min}$ as the optimal $\{\mathbf{a}_1, \ldots, \mathbf{a}_N\}$.

file-group subproblem, for each $n_1 \in \{M+1, \ldots, N-1\}$. The algorithm simply computes $\bar{R}$ for different (n_o, n_1, l_o, l_1) using the closed-form expression in (3.14) for at most $(N-1)(N-M-1)K^2/2$ times in the worst case (depending on the values of (N, M, K)) . They can be computed efficiently in parallel.

Remark 4. We point out that there is no three-file-group caching scheme proposed for the CCS in the literature. Only [36] has considered adding a specific three-file-group case heuristically as part of a mixed caching scheme, where the second file group contains only one file. However, uncoded caching is used for the case of three file groups, *i.e.,* the content delivery is uncoded, and the case is used for very rare occasions. In the simulation, we will show that the three-file-group cache placement for coded caching is optimal and outperforms the two-group strategy even for files with Zipf distribution.

3.3.4 The Optimal Cache Placement Solution

By Theorem 1, the optimal cache placement problem **P1** (or **P2**) is reduced to three subproblems, *i.e.,* one, two, or three file groups, respectively. The possible structure of the

optimal cache placement in each subproblem is given in Sections 3.3.1 to 3.3.3. These results lead to a simple algorithm to obtain the optimal placement solution $\{\mathbf{a}_n\}$ for **P1**: Each file-group case returns the candidate optimal solution $\{\mathbf{a}_n\}$ with the minimum $\bar{R}$ for this subproblem. The optimal $\{\mathbf{a}_n\}$ can then be obtained by taking the one that gives the minimum $\bar{R}$ among the three subproblems. The details are summarized in Algorithm 4. It uses Algorithms 1–3 and selects $\{\mathbf{a}_n\}$ that returns the minimum $\bar{R}$ as the optimal solution. Again, we point out obtaining the optimal $\{\mathbf{a}_n\}$ in Algorithm 4 requires minimum complexity. Algorithms 1–3 each involves computing a closed-form expression of $\bar{R}$ multiple times, and all can be done in parallel. In total, $\bar{R}$ is computed for at most $(N-1)(N-M+1)K^2/2+N$ times in the worst case.[2]

How to determine the file groups depends on $(\mathbf{p}, N, K, M)$. Although Sections 3.3.1 to 3.3.3 provide the possible structure of the optimal cache placement in three file grouping cases, analytically determining the final optimal file grouping, *i.e.,* the number of file groups and the group partition (n_o for two groups, and (n_o, n_1) for three groups), is still challenging. The same for the location of nonzero element(s) l_o (and l_1) in $\mathbf{a}_n$, *i.e.,* the choice of cache subgroup(s) for subfiles. They depend on the file popularity distribution $\mathbf{p}$, the number of users K and the relative cache size to the database size (M vs. N). Our proposed Algorithm 4 that combines Algorithms 1–3 provides a simple and efficient method to obtain the optimal file grouping. Using the obtained file group structures, Algorithms 1–3 significantly simplify the solving of **P1**, by providing a set of candidate solutions in closed-form in each case.

3.3.5 Discussion on the Optimal File Group Structure

The result in Theorem 1 of having at most three file groups in the optimal cache placement for the CCS, regardless of file distribution $\mathbf{p}$, is somewhat surprising. Based on the results obtained in Sections 3.3.1 to 3.3.3, we provide some insights into the optimal file group

[2]Under the optimal placement, files with the same popularity have identical placement, *i.e.,* $a_{n,l} = a_{n',l}$, $\forall l$, if $p_n = p'_n$. This means that the files with the same popularity are in the same file group (*e.g.,* a single file group for files with uniform popularity). This may further reduce the set of candidate solutions in Algorithms 1–3 by only considering possible values of n_o (and n_1) only for $p_{n_o} > p_{n_o+1}$ (and $p_{n_1} > p_{n_1+1}$).

structure. We can recognize the three file groups as three categories of "most popular," "moderately popular," and "non-popular" files. Regardless of file popularity distribution $\mathbf{p}$, the caching method only distinguishes files by one of these three categories. The three categories reflect the caching strategies: From the structure of optimal $\{\mathbf{a}_n\}$ obtained in Sections 3.3.1 to 3.3.3, the optimal caching strategy is to 1) cache all subfiles of the "most popular" files (among K users); 2) for the deemed "moderately popular" files, cache only a portion of each file, and leave the rest solely at the server; 3) if there are "non-popular" files, they are not cached but only stored in the server.

Note that a file belongs to which category is a relative notion: the mapping of files into these three categories, *i.e.,* file grouping, depends on the file popularity distribution $\mathbf{p}$ and the ratio of global cache size to the database size KM/N. To further understand the file grouping phenomenon and the case of three file groups, we provide numerical examples in Section 3.6.1 through Tables 3.1–3.6 to show how the file group structure changes (see Section 3.6.1 for the detailed discussion). As M/N increases (*e.g.,* from 10% to 80%), we observe that the optimal number of file groups changes as follows: $2 \to 3 \to 2 \to 3 \to 1$. Intuitively, increasing the cache memory allows more files to be cached. As a result, a file deemed "non-popular" for small cache size may be deemed "most popular" for large cache size. Thus, when M/N increases, more files are shifted from the "non-popular" group (only stored in the server) to the "most popular" group (all cached), with fewer files in the "non-popular" group. During this transition, the "moderately popular" file group (partly cached) appears, as the cache size is large enough to partly store some file but not all of it (among users). This explains why and when three file groups become optimal for the cache placement.

The existing two-file-group schemes proposed in [35, 36] have "non-popular" and "most popular" groups and use a suboptimal strategy to decide the file groups. They can be viewed as reflections of the two-file-group scenario. However, these two-file-group schemes cannot capture the "moderately popular" group during the transition stage mentioned above. In contrast, the optimal solution we obtain captures all possible file groups,

which provides the highest resolution in determining the optimal cache placement, leading to the minimum rate.

Remark 5. Note that the optimal cache placement is obtained in a centralized scenario, where the solution and the determination of file groups requires the knowledge of K. The knowledge of K is also required for the existing file grouping strategies [35, 36] for a bounded performance. In practice, the system can estimate K if it is unknown. A careful estimation of K based on some prior information will enable us to directly apply the optimal cache placement solution obtained in this work. For the effect of K on the cache placement, in general, the cache placement for the CCS is related to the ratio KM/N. A larger K value means higher KM/N. This leads to more files being shifted from the "non-popular" group to the "most popular" group and stored in the user caches. The inaccurate knowledge of K may result in a mismatch to the optimal file groups and a loss from the optimal performance. Quantifying the effect of overestimating or underestimating K on the performance loss is non-trivial and needs further study as a future work. To this end, it would be also interesting to study the optimal cache placement design and its gap to the lower bound for unknown K under nonuniform file popularity.

3.4 Converse Bound

In this section, we show that the structure of the optimal cache placement solution for **P1** obtained earlier can be used to obtain a tighter information-theoretic lower bound on the average rate $\bar{R}$ for any coded caching scheme (with uncoded or coded cache placement), under arbitrary file popularity. This converse bound is obtained using a genie-based method. Some existing works [16, 35, 36] have used this genie-based method to derive the lower bounds on the average rate with different tightness. This genie-based method constructs a virtual system, where only a group of popular files are delivered to the users via the shared link, and the rest (unpopular) files are delivered by a genie instead of using the shared link. Furthermore, the virtual system treats this group of popular files as if they have uniform popularity, leading to the symmetric cache placement strategy with the same placement for

all these files. The average rate of the original system under any coded caching scheme is shown to be lower bounded by that of this virtual system [18].

In deriving a lower bound using the genie-based method, the determination of the group of popular files plays an important role in the tightness of the bound. Let p' be the probability threshold to decide the group of popular files, where file W_n belongs to this group if $p_n \geq p'$. Let $N_{p'}$ denote the number of popular files in the group. The general result for the lower bound shows that, for K users requesting files independently, the average rate is lower bounded by [36]

$$\bar{R} \geq \bar{R}^{\text{lb}} = \frac{1}{11} K p' (N_{p'} - M). \tag{3.39}$$

Heuristical methods are used to decide the group of popular files to derive the converse bounds. In [35], specific for the Zipf distribution, the choice of $N_{p'}$ is proposed for different Zipf parameter values, file sizes, and cache sizes. In [36], the value of p' is proposed for an arbitrary file popularity distribution. To tighten the bound further, a file merging approach is proposed in [36]: Those files not belonging to the group of popular files, but deemed moderately popular, are merged into new virtual files to be included in the group of popular files. Specifically, by the definition of $N_{p'}$, we have $p_{N_{p'}+1} < p'$. From file $W_{N_{p'}+1}$ and afterwards, subsequent files are merged into a new virtual file until the accumulated popularity of these merged files exceeds p'. The procedure repeats until all the rest files are considered. Let $N_{p'}^{\text{m}}$ denote the number of virtual files generated by the merging procedure. With these additional virtual files, there are $N_{p'} + N_{p'}^{\text{m}}$ popular files. Using (3.39), [36] shows a tighter lower bound given by

$$\bar{R}^{\text{lb}} = \frac{1}{11} K p' (N_{p'} + N_{p'}^{\text{m}} - M), \tag{3.40}$$

and the number of virtual files is $N_{p'}^{\text{m}} = \lfloor \frac{\sum_{n>N_{p'}} p_n}{2/p'} + \frac{1}{2} \rfloor$. The file merging approach allows some moderately popular files to be considered in deriving the converse bound. As a result, the bound in (3.40) is by far the tightest converse bound.

The value of p' for the converse bound in (3.40) obtained in [36] is determined by combining a heuristic method and the exhaustive search. The method sets $p' = p_1 \triangleq$

$\frac{1}{K\max\{3,M\}}$, which results in N_{p_1} popular files and $N_{p_1}^{\mathrm{m}}$ virtual files. To avoid trivial negative lower bound in (3.40), when $N_{p_1}+N_{p_1}^{\mathrm{m}}<M$, the exhaustive search of p' ($N_{p'}$) is used by searching over the rest of files with popularity less than p_1, *i.e.,* $\{W_n: N_{p_1}+1\leq n\leq N\}$. As a result, the converse bound is given by [36]

$$\bar{R}^{\mathrm{lb}}=\frac{1}{11}\max\left\{\frac{1}{\max\{3,M\}}(N_{p_1}+N_{p_1}^{\mathrm{m}}-M),\max_{N_{p_1}+1\leq n\leq N}Kp_n(N_{p_n}+N_{p_n}^{\mathrm{m}}-M)\right\} \tag{3.41}$$

where the second term provides a possible improved converse bound through the exhaustive search for $N_p'\in\{N_{p_1}+1,\ldots,N\}$.

Interestingly, the use of popular files to derive the lower bound echoes the structure of the optimal cache placement solution for **P1**. As discussed in Section 3.3.4, the n_o files in the first file group are the most popular files for caching. Based on this observation, we determine $N_{p'}$ by the optimal cache placement for the CCS. The group of the most popular files is obtained from Algorithm 4 with size n_o, with the corresponding file popularity threshold set as $p'=p_{n_o}$. The number of virtual files is $N_{p_{n_o}}^{\mathrm{m}}$ accordingly. Then, we obtain the lower bound $\bar{R}^{\mathrm{lb}}$ as follows.

Proposition 5. Let n_o be the number of files in the first file group by the optimal cache placement solution for **P1**. The average delivery rate is lower bounded by

$$\bar{R}\geq\bar{R}^{\mathrm{lb}}=\frac{1}{11}Kp_{n_o}(n_o+N_{p_{n_o}}^{\mathrm{m}}-M). \tag{3.42}$$

We point out that the difference of $\bar{R}^{\mathrm{lb}}$ in (3.42) from the existing methods [35, 36] is that, instead of determining the popular files heuristically or through an exhaustive search, we obtain the number of most popular files n_o from the optimal file group structure in the cache placement optimization. In the simulation, we show that the lower bound in (3.42) is tighter than the existing ones, especially for a smaller cache size when the average rate is more sensitive to cache placement. This shows that using the file groups given by the optimal cache placement for the CCS provides a more accurate method in determining the popular files than existing methods.

3.5 Subpacketization Upper Bound

The subpacketization level, *i.e.,* the number of subfiles in each file required for caching, is an important issue for the practical implementation of coded caching. Since the optimal cache placement has not been characterized before, there is no clear quantification of the number of subfiles generated by the CCS. In this section, we explore the properties in the optimal cache placement solution for **P1** to characterize the subpacketization structure and derive an upper bound on the subpacketization level under the optimal cache placement, for any file popularity distribution $\mathbf{p}$ and memory size M.

Recall from Section 3.1.2 that each file can be partitioned into 2^K subfiles, which are divided into $K+1$ file subgroups $\mathcal{W}_n^l$, $l \in \mathcal{K} \cup \{0\}$. There are $\binom{K}{l}$ subfiles in $\mathcal{W}_n^l$, each with size $a_{n,l}$. They will be stored in corresponding user subsets with size l, provided that $a_{n,l} > 0$. The subpacketization level L_n of file n is directly related to its placement vector $\mathbf{a}_n$ as $L_n = \sum_{l \in \mathcal{K} \cup \{0\}: a_{n,l} > 0} \binom{K}{l}$. Based on the structure of the optimal cache placement $\{\mathbf{a}_n\}$ presented in Section 3.3, it is straightforward to conclude the following property of $\mathbf{a}_n$.

Corollary 1. For N files with any file popularity distribution $\mathbf{p}$, the optimal cache placement $\mathbf{a}_n$ of any file n for **P1** has at most two nonzero elements.

Following this property, we bound the worst-case maximum subpacketization level, defined by $L^{\max} = \max_n L_n$, for the CCS.

Proposition 6. For given $(N, \mathbf{p}, M, K)$, the maximum subpacketization level $L^{\max}$ under the optimal cache placement for the CCS is bounded by

$$L^{\max} \leq \binom{K}{\lfloor K/2 \rfloor} + \binom{K}{\lfloor K/2 \rfloor + 1} \leq \sqrt{\frac{8}{\pi}} e^{\frac{1}{12K}} \frac{2^K}{\sqrt{K}}. \tag{3.43}$$

Proof. From Corollary 1, by the optimal cache placement solution, the subfiles of any file belong to at most two file subgroups of different sizes. There are $\binom{K}{l}$ subfiles need to be cached into the user subsets with size $l \in \mathcal{K} \cup \{0\}$. Then, for $l = \lfloor K/2 \rfloor$ and $\lfloor K/2 \rfloor + 1$,

the number of subfiles is the highest. Consequently, we have $L^{\max} \leq \binom{K}{\lfloor K/2 \rfloor} + \binom{K}{\lfloor K/2 \rfloor + 1}$. Based on the Stirling's approximation [95], we have

$$\sqrt{2\pi K}\left(\frac{K}{e}\right)^K \leq K! \leq \sqrt{2\pi K}\left(\frac{K}{e}\right)^K e^{\frac{1}{12K}},$$

where the bounds become tight as K increases. Assuming $K = 2m$, $m \in \mathbb{N}^+$, we have

$$\binom{K}{\frac{K}{2}} = \frac{K!}{\frac{K}{2}! \cdot \frac{K}{2}!} \leq \frac{\sqrt{2\pi} e^{\frac{1}{12K}} K^{K+\frac{1}{2}} e^{-K}}{2\pi \left(\frac{K}{2}\right)^{K+1} e^{-K}} \leq \sqrt{\frac{2}{\pi}} e^{\frac{1}{12K}} \frac{2^K}{\sqrt{K}},$$

and we have (3.43). □

Proposition 6 indicates that the maximum number of subfiles in the worst-case grows as $\mathcal{O}(2^K/\sqrt{K})$. The actual subpacketization level of a file group depends on the location of nonzero element l_o (l_1) in $\mathbf{a}_n$. Although we cannot explicitly obtain l_o (l_1) for the optimal placement, in general, for given K, it is a function of the cache size relative to the database size M/N. Recall that for smaller l, subfiles in $\mathcal{W}_n^l$ are cached to smaller user subsets $\mathcal{S}$'s ($|\mathcal{S}| = l$), and vice versa. Intuitively, this means that the location l_o (l_1) of the nonzero element tends to be smaller for smaller cache size and becomes larger as M/N increase. This intuition is confirmed by experiments. In the simulation, we show that, depending on M/N, the actual subpacketization level of the optimal cache placement typically can be much less than the upper bound in (3.43).

Remark 6. The tradeoff between the average rate and the subpacketization level has been studied in [53] via a numerical search over different subpacketization levels. We point out that the upper bound in Proposition 6 provides the exact subpacketization level, for which increasing it further no longer leads to a rate reduction.

3.6 Numerical Results

In this section, we evaluate the performance of the optimal cache placement by the proposed algorithm and the corresponding subpacketization level for different system setups. Further, we also evaluate the information-theoretic lower bound based on the file grouping strategy in the optimal cache placement solution.

l	Cache placement vector of each file								
	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$	$\mathbf{a}_8$	$\mathbf{a}_9$
0	0	0	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0	0	0	0	0	0	0	0	0
2	0.0317	0.0317	0.0317	0	0	0	0	0	0
3	0.0095	0.0095	0.0095	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0

Table 3.1: Cache placement matrix for $K = 7$, $N = 9$, $\theta = 1.5$, $M = 1$.

3.6.1 The Optimal Cache Placement

We first verify the structure of the optimal cache placement solution for the CCS obtained in Section 3.3. To do so, we obtain the placement solution $\{\mathbf{a}_n\}$ by Algorithm 4 and verify that they match the optimal $\{\mathbf{a}_n\}$ obtained by numerically solving **P1**. For example, we generate user random demands using Zipf distribution, where file n is requested with probability $p_n = \frac{n^{-\theta}}{\sum_{i=1}^{N} i^{-\theta}}$, with $\theta > 0$ being the Zipf parameter. For $N = 9$, $\theta = 1.5$, and $K = 7$, Tables 3.1 - 3.6 show the optimal $\{\mathbf{a}_n\}$ for cache size $M = 1, 2.5, 4, 5.5, 6, 7$, respectively. They cover the possible cases of the optimal cache placement structure discussed in Section 3.3. As the cache size increases from small to large, different file groups and subfile partition strategies under the optimal placement solution can be observed. In all these results, the cache placement vectors $\{\mathbf{a}_n\}$ have at most two nonzero elements, as stated in Corollary 1.

Tables 3.1 shows the optimal cache placement solution $\{\mathbf{a}_n\}$ for $M = 1$. There are two file groups under the optimal solution, as in the case discussed in Section 3.3.2 (Fig. 3.3). They are deemed "most popular" and "non-popular" files. The placement vector of the first file group has two nonzero elements (*e.g.,* file W_1 is partitioned into two subfile groups $\mathcal{W}_1^2$ and $\mathcal{W}_1^3$, containing subfiles of size 0.0317 and 0.0095, respectively), and the files in the second group are only stored at the server.

As M is increased to 2.5, Tables 3.2 shows that the optimal placement divides files into three file groups, verifying the structure of the optimal $\{\mathbf{a}_n\}$ described in Section 3.3.3

l	Cache placement vector of each file								
	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$	$\mathbf{a}_8$	$\mathbf{a}_9$
0	0	0	0	0	0.2500	0.2500	1.0000	1.0000	1.0000
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0.0214	0.0214	0.0214	0.0214	0.0214	0.0214	0	0	0
4	0.0071	0.0071	0.0071	0.0071	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0

Table 3.2: Cache placement matrix for $K = 7$, $N = 9$, $\theta = 1.5$, $M = 2.5$.

l	Cache placement vector of each file								
	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$	$\mathbf{a}_8$	$\mathbf{a}_9$
0	0	0	0	0	0	0	0	1.0000	1.0000
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0.0286	0.0286	0.0286	0.0286	0.0286	0.0286	0.0286	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0

Table 3.3: Cache placement matrix for $K = 7$, $N = 9$, $\theta = 1.5$, $M = 4$.

and illustrated in Fig. 3.8. The "moderately popular" file group ($\{W_5, W_6\}$) is included in this case, for which the increased cache size allows more room to cache a portion of these files, while leaving the rest portion at the server. Between the first two groups, we observe that the sub-placement vectors $\bar{\mathbf{a}}_n$'s are only different by one element.

When M is further increased to 4, we observe from Table 3.3 that the placement results in two file groups, similar to that for $M = 1$. However, compared to $M = 1$, larger cache memory allows more files to be considered in the "most popular" file group to be cached. For these files, $\mathbf{a}_n$ has only one nonzero element, indicating they are all partitioned into subfiles of equal length.

For $M = 5.5$ in Table 3.4, the files are divided into three groups, where file W_8 is now considered "moderately popular" and partly cached, instead of "non-popular" as in the case of $M = 4$. Table 3.4 is the case described in Fig. 3.7 of Section 3.3.3. As we keep increasing M, we see from Tables 3.5 that for $M = 6$, the result is as described in Fig. 3.5, where files are considered either "most popular" or "moderately popular" and are

l	Cache placement vector of each file								
	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$	$\mathbf{a}_8$	$\mathbf{a}_9$
0	0	0	0	0	0	0	0	0.3000	1.0000
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0.0476	0.0476	0.0476	0.0476	0.0476	0.0476	0.0476	0.0333	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0

Table 3.4: Cache placement matrix for $K = 7$, $N = 9$, $\theta = 1.5$, $M = 5.5$.

l	Cache placement vector of each file								
	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$	$\mathbf{a}_8$	$\mathbf{a}_9$
0	0	0	0	0	0	0	0	0	0.6000
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0.0476	0.0476	0.0476	0.0476	0.0476	0.0476	0.0476	0.0476	0.019
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0

Table 3.5: Cache placement matrix for $K = 7$, $N = 9$, $\theta = 1.5$, $M = 6$.

stored among users accordingly. For $M = 7$, Table 3.6 shows that when there is enough cache at users, all files are considered "most popular" with identical cache placement as discussed in Section 3.3.1. This single file group resembles the placement under uniform file popularity.

From $M = 1$ to $M = 7$, we notice that the location of the nonzero element in $\mathbf{a}_n$ (the value of l_o and l_1) is increasing. This indicates that as M increases, each subfile is stored into a larger user subset. This trend confirms our intuition that the optimal l_o (l_1) increases as more cache memory is added.

Note that the optimal placement solutions in Tables 3.2, 3.4, and 3.5 show three or two file groups that have not been considered in the existing suboptimal schemes. For example, in [35], only two file groups are considered, with the second group of files kept at the server. As a result, these existing schemes cannot always guarantee the minimum rate.

$\mathbf{a}$	Cache placement vector of each file								
	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$	$\mathbf{a}_8$	$\mathbf{a}_9$
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0.0265	0.0265	0.0265	0.0265	0.0265	0.0265	0.0265	0.0265	0.0265
6	0.0635	0.0635	0.0635	0.0635	0.0635	0.0635	0.0635	0.0635	0.0635
7	0	0	0	0	0	0	0	0	0

Table 3.6: Cache placement matrix for $K = 7$, $N = 9$, $\theta = 1.5$, $M = 7$.

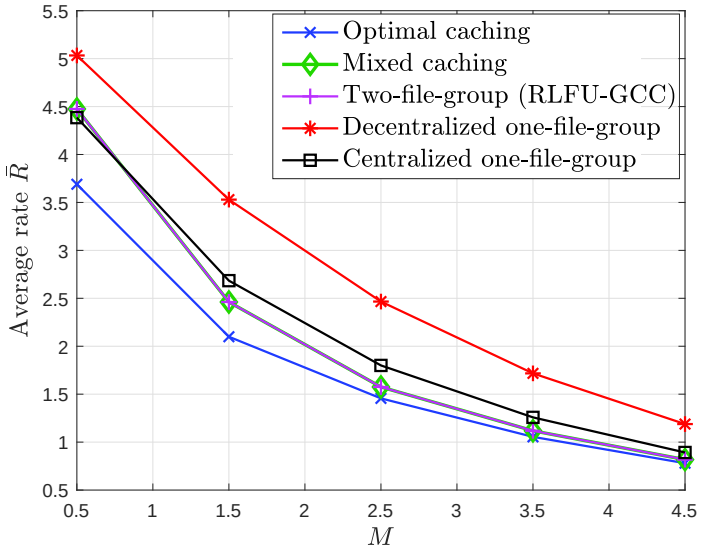

Figure 3.9: Average rate $\bar{R}$ vs. cache size M ($N = 10$, $K = 6$, Zipf distribution: $\theta = 1.5$).

3.6.2 Performance of Average Rate

To evaluate the performance of the optimal cache placement scheme obtained by Algorithm 4, we plot the average rate $\bar{R}$ vs. M for file popularity using Zipf distribution and a step function in Figs. 3.9 and 3.10, respectively. For comparison, we consider the centralized [17] and decentralized [18] symmetric cache placement schemes designed for uniform file popularity (*i.e.,* one file group), the RLFU-GCC scheme with two file groups [35], and the mixed caching scheme in [36]. In Fig. 3.9, we set $N = 10$, $K = 6$, and Zipf parameter $\theta = 1.5$. The optimal cache placement by Algorithm 4 results in the lowest $\bar{R}$ among all the schemes. As expected, the fixed one-file-group scheme, designed for uniform

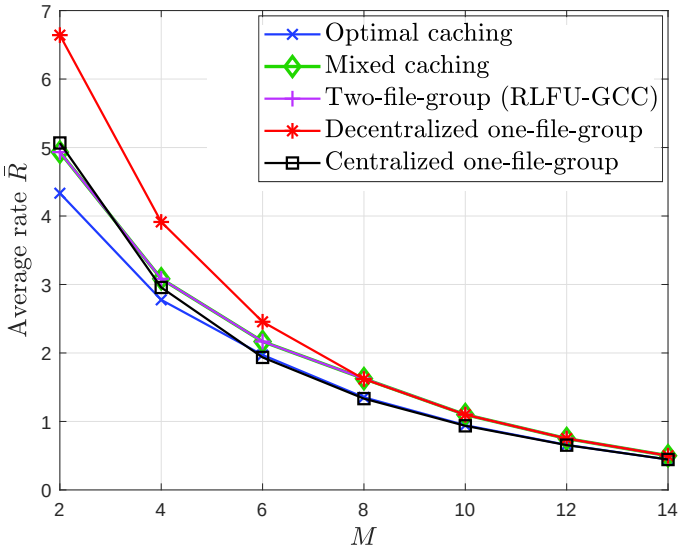

Figure 3.10: Average rate $\bar{R}$ vs. cache size M ($N = 21$, $K = 12$, step-function distribution).

popularity, has the worst performance. The two-file-group scheme (RLFU-GCC) and the mixed caching scheme have almost identical performance. The performance gap between the two-file-group scheme (RLFU-GCC) and the optimal solution is more noticeable for smaller M, and reduces as M increases.

In Fig. 3.10, we consider a case studied in [36] with $N = 21$, $K = 12$, and a non-Zipf step-function file popularity distribution given as: $p_1 = 5/9$, $p_n = 1/30$, for $n = 2, \ldots, 11$, and $p_n = 1/90$, for $n = 12, \ldots, 21$. Again, the average rate under the optimal cache placement is lower than that of all other schemes, with the gap more noticeable for smaller M. As an example, for $M = 2$, the optimal $\{\mathbf{a}_n\}$ results in three file groups for coded caching that has not been considered in any existing scheme.

3.6.3 Converse Bound

We now compare our proposed lower bound in (3.42) with those proposed in [35] and [36], as well as the average rate under the optimal caching scheme by Algorithm 4. In Fig. 3.11, we set $N = 10$, $K = 6$, and Zipf parameter $\theta = 1.5$. We observe that our proposed lower bound is the highest for all values of M, and the gap is larger for smaller M. In particular,

		Two-file-group [36]	Exhaustive Search [36]	Proposed
$N=5$	$N_{p'}$	4	5	3
	$N_{p'}^m$	0	0	1
	$\bar{R}^{\text{lb}}$	0.0909	0.1109	0.1789
$N=7$	$N_{p'}$	4	5	3
	$N_{p'}^m$	1	1	1
	$\bar{R}^{\text{lb}}$	0.1212	0.1296	0.1673
$N=9$	$N_{p'}$	4	5	3
	$N_{p'}^m$	2	1	2
	$\bar{R}^{\text{lb}}$	0.1515	0.1242	0.2138

Table 3.7: Comparison of different schemes to compute the lower bound in (3.40) ($M=1$, $K=6$, Zipf distribution: $\theta=1.5$. For $N=5,7,9$).

our bound in (3.42) based on the optimal file groups in the cache placement is higher than the one in (3.41) from [36].

As discussed in Section 3.4, the difference in the lower bounds comes from how the values of $p', N_{p'}, N_{p'}^m$ in (3.40) are set by each scheme (*i.e.,* (3.41) and (3.42)). To see the difference between our scheme and two other schemes in [36], including the two-file-group-based method and the exhaustive search, we show the values of $N_{p'}$, $N_{p'}^m$, and $\bar{R}^{\text{lb}}$ in (3.40) for each scheme in Table 3.7. We consider Zipf distribution with $\theta=1.5$, $K=6$, $M=1$, and compare the performance for $N=5,7,9$. Again, our scheme always leads to the tightest lower bound $\bar{R}^{\text{lb}}$. Note that for the popular file group, the optimal cache placement in our scheme always gives smaller $N_{p'}$. This indicates that a smaller number of the most popular files are selected, in contrast to the two-file-group based method and the exhaustive search. This shows that even the exhaustive search used in (3.41) is not enough to find the optimal number of the most popular files since it only searches a subgroup of the possible cases.

3.6.4 Subpacketization Level

Define the average subpacketization level among N files by $\bar{L}=\frac{1}{N}\sum_n L_n$. For $N=20$ and $K=10$, we obtain both $L^{\max}$ and $\bar{L}$ under the optimal cache placement by solving **P1** for different M and θ. Note that smaller θ indicates a more uniform popularity distribution

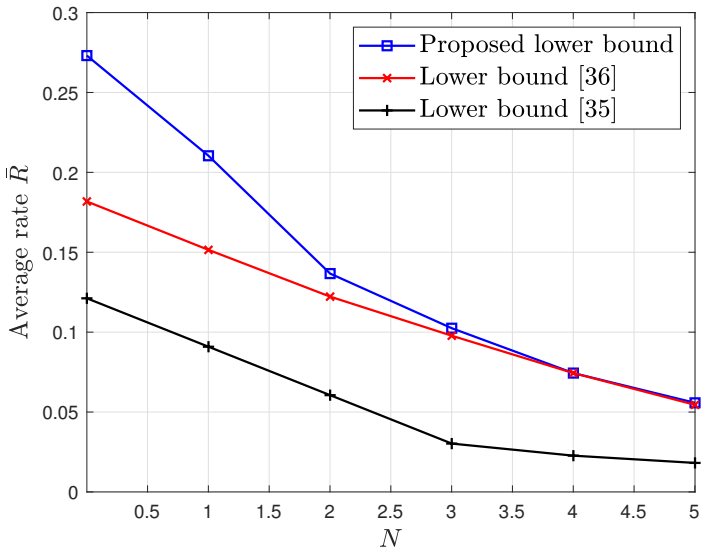

Figure 3.11: The lower bound $\bar{R}^{\text{lb}}$ vs. cache size M ($N = 10$, $K = 6$, Zipf distribution: $\theta = 1.5$).

and vice versa. Fig. 3.12 Top shows $L^{\max}$, the worst-case level (the upper bound in (3.43)), and the maximum possible number of subfiles (2^K), over different M, for Zipf parameter $\theta = 0.4, 1.4, 2.4$. We see that except for a small range of M, for most of the values of M, $L^{\max}$ is much lower than the worst-case level. Fig. 3.12 Bottom shows $\bar{L}$ over M. The general trend is similar to that of $L^{\max}$, except that $\bar{L}$ can be much less than $L^{\max}$ at the lower range of M, especially for $\theta = 2.4$, where there are only a few highly popular files. For both $L^{\max}$ and $\bar{L}$, they tend to increase then decrease with M. This is because the location l_o of the nonzero element in $\mathbf{a}_n$ increases as M becomes larger, as seen in Tables 3.1–3.5. As a result, the number of subfiles $\binom{K}{l_o}$ increases then decreases. In general, the subpacketization level is low for smaller or larger M/N and higher for moderate M/N.

3.7 Summary

In this chapter, we formulated an optimization problem to obtain the optimal cache placement of the CCS. We identified the inherent file group structure under the optimal solution, where there are at most three file groups under the optimal solution, regardless of file pop-

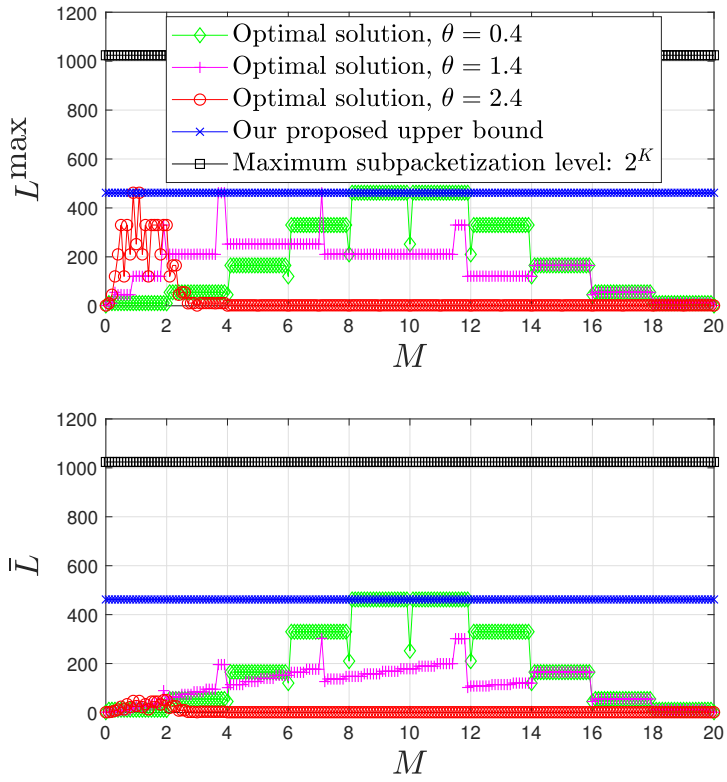

Figure 3.12: The subpacketization level under the optimal cache placement vs. cache size M ($N = 20$, $K = 10$). Top: The worst-case subpacketization level $L^{\max}$. Bottom: The average subpacketization level $\bar{L}$.

ularity or other system parameters. For each possible file grouping case, we obtained the cache placement solution in closed-form. Following this, we developed a simple and efficient algorithm to obtain the optimal cache placement solution by comparing a set of candidate closed-form solutions that can be computed in parallel. Our insight into at most three possible file groups links the caching method to popular, moderately popular, and non-popular file categories. Furthermore, the optimal cache placement may explore coding opportunities across file groups. Based on the optimal file grouping, we provided a new converse bound for caching under any placement that is tighter than any existing ones. We also quantified the subpacketization level of the optimal cache placement, where we showed that the worst-case subpacketization level grows as $\mathcal{O}(2^K/\sqrt{K})$. The simulation study verified the file group structure and the optimal cache placement solution by our pro-

posed simple algorithm. A lower average rate achieved by the optimal cache placement than that of existing schemes was shown.

Chapter 4

Memory-Rate Tradeoff for Caching with Uncoded Placement under Nonuniform Demands

In this chapter, for a caching system with nonuniform file popularity, we study the memory-rate tradeoff for caching with uncoded placement. We focus on the MCCS with cache placement optimized in the class of popularity-first placement for average rate minimization. We then propose a general lower bound and a popularity-first-based lower bound for caching with uncoded placement. We also compare the optimized MCCS with the lower bound to study the memory-rate tradeoff. Finally, we extend our study of memory-rate tradeoff to the case where files are nonuniform in both popularity and sizes.

4.1 Cache Placement Optimization of the MCCS for Rate Minimization

We consider a cache-aided transmission system with a server connecting to K cache-equipped users over a shared error-free link, as described in section 3.1.1. For any coded caching scheme, cache placement is a key design issue, which needs to be optimized to minimize the delivery rate. The MCCS is a coded caching scheme recently proposed [44], where the delivery strategy is improved over the original CCS [17] to reduce the delivery rate further. In this section, we formulate the rate minimization problem for the MCCS under the cache placement optimization.

4.1.1 Cache Placement

The cache placement construction for the MCCS is based on file partitioning. For K users, there are total 2^K user subsets in $\mathcal{K}$, with subset sizes ranging from 0 to K (including the empty set). Grouping the user subsets based on their sizes, we form a cache subgroup that contains all user subsets of size l, defined as $\mathcal{A}^l \triangleq \{\mathcal{S} : |\mathcal{S}| = l,\ \mathcal{S} \subseteq \mathcal{K}\}$ with $|\mathcal{A}^l| = \binom{K}{l}$, for $l = 0, \ldots, K$. Partition each file W_n into 2^K non-overlapping subfiles. Each subfile is for a unique user subset $\mathcal{S} \subseteq \mathcal{K}$, denoted by $W_{n,\mathcal{S}}$, and it is stored at the local cache of each user in subset $\mathcal{S}$. It is possible that $W_{n,\mathcal{S}} = \emptyset$ for a given $\mathcal{S}$, and also for $\mathcal{S} = \emptyset$, subfile $W_{n,\emptyset}$ is only kept in the server and not stored in any user's cache. A caching scheme specifies how files are partitioned for storage. Regardless of the scheme used, each file should be able to be reconstructed by combining all its subfiles. Thus, we have

$$\sum_{l=0}^{K} \sum_{\mathcal{S} \in \mathcal{A}^l} |W_{n,\mathcal{S}}| = F, \quad n \in \mathcal{N}. \tag{4.1}$$

There are 2^K subfile sizes to be determined for each file. To reduce the number of variables and simplify the cache placement problem for its tractability, the following condition is imposed:

C1) For each file W_n, the size of its subfile $W_{n,\mathcal{S}}$ only depends on $|\mathcal{S}|$, *i.e.,* $|W_{n,\mathcal{S}}|$ is the same for any $\mathcal{S} \in \mathcal{A}^l$ of the same size.

The above condition is in fact proven to be the property of the optimal cache placement for the CCS [37]. For the MCCS, although it is more difficult to prove analytically, it is numerically verified in [53] that imposing this condition results in no loss of optimality.[1] As a result, the subfiles of file W_n are grouped into file subgroups according to user subset size l, each denoted by $\mathcal{W}_n^l = \{W_{n,\mathcal{S}} : \mathcal{S} \in \mathcal{A}^l\}$, for $l = 0, \ldots, K$. Note that there are $\binom{K}{l}$ subfiles of the same size in $\mathcal{W}_n^l$, and there are $K+1$ file subgroups. Following this, let $a_{n,l}$ denote the size of subfiles in $\mathcal{W}_n^l$ as a fraction of file W_n size F bits, *i.e.,* $a_{n,l} \triangleq |W_{n,\mathcal{S}}|/F$, for $\forall \mathcal{S} \in \mathcal{A}^l$, $l = 0, \ldots, K$, $n \in \mathcal{N}$. In particular, $a_{n,0}$ represents the fraction of file W_n

[1] In Section 4.3, we are able to prove that imposing Condition C1) does not incur loss of optimality in some specific cases.

that is not stored at any user's cache but only remains in the server. Then, the file partition constraint (4.1) is simplified to

$$\sum_{l=0}^{K} \binom{K}{l} a_{n,l} = 1,\ n \in \mathcal{N}. \tag{4.2}$$

Recall that each subfile is intended for a unique user subset. For the cache placement, user k stores all the subfiles in $\mathcal{W}_n^l$ that are intended for it, *i.e.,* $\{W_{n,\mathcal{S}} : k \in \mathcal{S} \text{ and } \mathcal{S} \in \mathcal{A}^{l+1}\} \subseteq \mathcal{W}_n^l$, for $l = 0, \ldots, K-1$. Note that in each cache subgroup $\mathcal{A}^l$, there are $\binom{K-1}{l-1}$ different user subsets containing the same user k. Thus, there are $\sum_{l=1}^{K} \binom{K-1}{l-1}$ subfiles in each file W_n that a user can store in its local cache. This means that, in total, a fraction $\sum_{l=1}^{K} \binom{K-1}{l-1} a_{n,l}$ of file W_n is cached by a user. With cache size M at each user, we have the following cache constraint

$$\sum_{n=1}^{N} \sum_{l=1}^{K} \binom{K-1}{l-1} a_{n,l} \leq M. \tag{4.3}$$

For nonuniform file popularity, even with Condition C1, the cache placement is still a complicated problem. To further simplify the cache placement problem and the average rate expression, we consider the popularity-first cache placement approach described below.

Popularity-first cache placement

A popularity-first cache placement is to allocate more cache memory to a more popular file: With file popularity $p_1 \geq \cdots \geq p_N$, the cached subfiles satisfies

$$a_{n,l} \geq a_{n+1,l}, \quad l \in \mathcal{K},\ n \in \mathcal{N} \backslash \{N\}. \tag{4.4}$$

Remark 7. The popularity-first cache placement approach has been used for both the CCS [33, 37] and the MCCS [53] to simplify the cache placement problem. For the CCS, the popularity-first placement has been proven to be the property of the optimal cache placement [37]. For the MCCS, the same is difficult to prove analytically, but the optimality of the popularity-first placement has been verified numerically [53]. In Section 4.3, we will prove the optimality of popularity-first placement in some cases.

4.1.2 Content Delivery

In the content delivery phase, the server multicasts coded messages to different user subsets. Each coded message corresponds to a user subset $\mathcal{S}$, formed by the bitwise XOR operation of subfiles as

$$C_{\mathcal{S}} \triangleq \bigoplus_{k\in\mathcal{S}} W_{d_k,\mathcal{S}\setminus\{k\}}. \tag{4.5}$$

In the original CCS [17], the server simply delivers the coded message formed by each user subset, for any demand vector $\mathbf{d}$. However, under random demands, multiple users may request the same (popular) file, causing redundant coded messages transmitted separately multiple times. To address this, in the MCCS [44], a modified coded delivery strategy is proposed to remove this redundancy and reduce the average delivery rate further. Let $\widetilde{N}(\mathbf{d})$ denote the number of distinct file requests for demand vector $\mathbf{d}$, where $\widetilde{N}(\mathbf{d}) \leq K$. To describe the delivery strategy in the MCCS, we provide the following four definitions:

Definition 2 (Leader group)**.** The leader group $\mathcal{U}$ is a user subset of size $|\mathcal{U}| = \widetilde{N}(\mathbf{d})$, with the users in $\mathcal{U}$ having exactly $\widetilde{N}(\mathbf{d})$ distinct file requests.

Definition 3 (Redundant group)**.** Given the leader group $\mathcal{U}$, any user subset $\mathcal{S} \subseteq \mathcal{K}$ with $\mathcal{S} \cap \mathcal{U} = \emptyset$ is called a redundant group; otherwise, $\mathcal{S}$ is a non-redundant group.

Definition 4 (Redundant request)**.** A file request d_k by any user k in the redundant group $\mathcal{S}$ is a redundant request.

Definition 5 (Redundant message)**.** Any coded message $C_{\mathcal{S}}$ corresponding to a redundant group $\mathcal{S}$ is a redundant message; otherwise, it is a non-redundant message.

Based on the above definitions, any user subset is either a redundant group or a non-redundant group. In the MCCS, only the non-redundant messages are multicasted to both non-redundant and redundant groups. For nonuniform file popularity, file partitioning may be different for different files, leading to different subfile sizes. In formulating the coded

message $C_{\mathcal{S}}$ in (4.5), the following technique is commonly used for the subfiles of different lengths in the XOR operation.

Zero-padding: With different subfile sizes, subfiles in coded message $C_{\mathcal{S}}$ are zero-padded to the size of the largest subfile in $C_{\mathcal{S}}$ for transmission.

Consider zero-padding for the coded message $C_{\mathcal{S}}$ for subgroup $\mathcal{S}$. The size of $C_{\mathcal{S}}$ in (4.5) is given by

$$|C_{\mathcal{S}}| = \max_{k\in\mathcal{S}} a_{d_k,l}, \quad \mathcal{S} \in \mathcal{A}^{l+1},\ l = 0, \ldots, K-1. \tag{4.6}$$

Remark 8. Zero-padding is a technique commonly used to form coded messages in the existing works [33, 37, 47, 48, 53]. However, the impact of zero-padding on coded caching is unknown. Intuitively, zero-padding introduces extra waste bits that may degrade the performance. In Section 4.3, we will provide our findings on this issue.

4.1.3 Cache Placement Optimization

Let $\mathbf{a}_n \triangleq [a_{n,0}, \ldots, a_{n,K}]^T$ denote the $(K+1)\times 1$ cache placement vector for file W_n, $n \in \mathcal{N}$, and let $\mathbf{a} \triangleq [\mathbf{a}_1^T, \cdots, \mathbf{a}_N^T]^T$ represent the entire placement for N files. For demand vector $\mathbf{d}$, the delivery rate is the total size of the non-redundant messages, given by

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a}) = \sum_{\mathcal{S}\subseteq\mathcal{K},\mathcal{S}\cap\mathcal{U}\neq\emptyset} |C_{\mathcal{S}}| = \sum_{\mathcal{S}\subseteq\mathcal{K},\mathcal{S}\cap\mathcal{U}\neq\emptyset} \max_{k\in\mathcal{S}} a_{d_k,l}. \tag{4.7}$$

The average delivery rate $\bar{R}_{\text{MCCS}}$ is given by

$$\bar{R}_{\text{MCCS}}(\mathbf{a}) = \mathbb{E}_{\mathbf{d}}\left[R_{\text{MCCS}}(\mathbf{d};\mathbf{a})\right] = \mathbb{E}_{\mathbf{d}}\left[\sum_{\mathcal{S}\subseteq\mathcal{K},\mathcal{S}\cap\mathcal{U}\neq\emptyset} \max_{k\in\mathcal{S}} a_{d_k,l}\right] \tag{4.8}$$

where $\mathbb{E}_{\mathbf{d}}[\cdot]$ is taken with respect to $\mathbf{d}$.

From (4.4), define the set of all popularity-first placements by $\mathcal{Q} \triangleq \{\mathbf{a} :\ a_{n,l} \geq a_{n+1,l},\ l \in \mathcal{K}, n \in \mathcal{N}\backslash\{N\}\}$. Aiming to obtain the minimum average rate for the MCCS, we optimize the cache placement $\mathbf{a} \in \mathcal{Q}$ to minimize $\bar{R}_{\text{MCCS}}$, given by

$$\mathbf{P4}: \min_{\mathbf{a}\in\mathcal{Q}}\ \bar{R}_{\text{MCCS}}(\mathbf{a}) \tag{4.9}$$

$$\text{s.t. } (4.2), (4.3), \text{ and}$$

$$\mathbf{a}_n \succcurlyeq \mathbf{0},\ n \in \mathcal{N}. \tag{4.10}$$

Note that in **P4**, we restricted the cache placement optimization within the set of popularity-first placements, for the reason discussed in Remark 7. In the following, we first focus on analyzing how optimal the MCCS in **P4** is under nonuniform file popularity, by comparing it with the lower bounds we develop for caching with uncoded placement. Then, in Section 4.4, we describe the optimal cache placement solution to **P4** and its inherent structure.

4.2 Converse Bound for Uncoded Placement

In this section, we first introduce a lower bound on the average rate for any caching with uncoded placement. Then, we develop a popularity-first-based lower bound by restricting the uncoded placement to the set of popularity-first placements.

Let $\mathcal{D}$ denote the set of the distinct file indexes in demand vector $\mathbf{d}$, *i.e.,* $\mathcal{D} = \text{Unique}(\mathbf{d}) \subseteq \mathcal{N}$, where $\text{Unique}(\mathbf{d})$ is to extract the unique elements in $\mathbf{d}$. By the definition of the leader group $\mathcal{U}$ in Definition 2, we have $|\mathcal{D}| = |\mathcal{U}| = \tilde{N}(\mathbf{d})$, for a given $\mathbf{d}$. The following lemma gives a lower bound on the average rate under any uncoded placement.

Lemma 2. For the caching problem described in Section 3.1, the following optimization problem provides a lower bound on the average rate for caching with uncoded placement

$$\textbf{P5:}\ \min_{\mathbf{a}}\ \bar{R}_{\text{lb}}(\mathbf{a}) \triangleq \sum_{\mathcal{D}\subseteq\mathcal{N}} \sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})} \prod_{k=1}^{K} p_{d_k} R_{\text{lb}}(\mathcal{D};\mathbf{a}) \tag{4.11}$$

$$\text{s.t.}\quad (4.2), (4.3), \text{ and } (4.10)$$

where $\mathcal{T}(\mathcal{D}) \triangleq \{\mathbf{d} : \text{Unique}(\mathbf{d}) = \mathcal{D},\ \mathbf{d} \in \mathcal{N}^K\}$, and $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ is the lower bound for the distinct file set $\mathcal{D}$ with the placement vectors $\{\mathbf{a}_n, n \in \mathcal{D}\}$, given by

$$R_{\text{lb}}(\mathcal{D};\mathbf{a}) \triangleq \max_{\pi:\mathcal{I}_{|\mathcal{D}|}\to\mathcal{D}} \sum_{l=0}^{K-1} \sum_{i=1}^{\tilde{N}(\mathbf{d})} \binom{K-i}{l} a_{\pi(i),l} \tag{4.12}$$

where $\pi : \mathcal{I}_{|\mathcal{D}|} \to \mathcal{D}$ is any bijective map from $\mathcal{I}_{|\mathcal{D}|}$ to $\mathcal{D}$.

Proof. See Appendix B.1. □

Note that **P5** is a min-max problem. It can be cast in its epigraph form by moving (4.12) to the constraints. The resulting equivalent problem is a linear program (LP), which can be solved by standard LP solvers.

Given that the popularity-first placement approach has been considered in the existing works under nonuniform file popularity, we also develop a popularity-first-based lower bound for caching, assuming that the popularity-first placement approach is used for the uncoded placement.

Lemma 3. (Popularity-first-based lower bound) For the caching problem described in Section 3.1, the following optimization problem provides a lower bound on the average rate for caching under popularity-first cache placement

$$\textbf{P6:} \min_{\mathbf{a}\in\mathcal{Q}} \ \bar{R}_{\text{lb}}(\mathbf{a}) \triangleq \sum_{\mathcal{D}\subseteq\mathcal{N}} \sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})} \prod_{k=1}^{K} p_{d_k} R_{\text{lb}}(\mathcal{D};\mathbf{a}) \tag{4.13}$$
$$\text{s.t.} \quad (4.2), (4.3), \text{ and } (4.10)$$

where $R_{\text{lb}}(\mathcal{D};\mathbf{a})$, for $\mathbf{a}\in\mathcal{Q}$, is given by

$$R_{\text{lb}}(\mathcal{D};\mathbf{a}) \triangleq \sum_{l=0}^{K-1} \sum_{i=1}^{\tilde{N}(\mathbf{d})} \binom{K-i}{l} a_{\phi(i),l}, \quad \mathbf{a}\in\mathcal{Q} \tag{4.14}$$

where $\phi : \mathcal{I}_{|\mathcal{D}|} \to \mathcal{D}$ is the bijective map in the decreasing order of file popularity, *i.e.,* $p_{\phi(1)} \geq \cdots \geq p_{\phi(|\mathcal{D}|)}$.

Proof. See Appendix B.2. □

Note that for $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.12), by restricting $\mathbf{a}\in\mathcal{Q}$, we can remove the max operation and simplify the expression to (4.14). Due to the restriction $\mathbf{a}\in\mathcal{Q}$ in **P6**, the lower bound given by **P5** is also a lower bound for **P6**. In the following theorem, we show that for $K=2$ users, there is no loss of optimality by restricting to popularity-first placements.

Theorem 2. For $K=2$, **P5** and **P6** are equivalent.

Proof. See Appendix B.3 □

Theorem 2 indicates that, with nonuniform file popularity, for $K = 2$ users, the popularity-first placement is an optimal cache placement, regardless of the values of N, $\mathbf{p}$, and M. The general lower bound for any caching with uncoded placement given by **P5** is attained by some $\mathbf{a} \in \mathcal{Q}$. For $K > 2$ users, the same conclusion turns out to be challenging to prove. Although the same conclusion cannot be shown analytically, in Section 4.6, we provide numerical results to show that the two lower bounds by **P5** and **P6** are equal in general.

4.3 Memory-Rate Tradeoff Characterization

In this section, we compare the average rate of the optimized MCCS in **P4** and that of the popularity-first-based lower bound given by **P6** to show the tightness of the bound, as well as the how optimal the MCCS is. Note that the difference between **P4** and **P6** is only in the expression of the average rate objective. Thus, it is sufficient to compare $\bar{R}_{\text{MCCS}}(\mathbf{a})$ and $\bar{R}_{\text{lb}}(\mathbf{a})$, for $\mathbf{a} \in \mathcal{Q}$.

Consider N files with arbitrary popularity distribution $\mathbf{p}$ and local cache size M. We compare $\bar{R}_{\text{MCCS}}(\mathbf{a})$ and $\bar{R}_{\text{lb}}(\mathbf{a})$ in three regions:

Region 1: $K = 2$;

Region 2: $K > 2$, $\tilde{N}(\mathbf{d}) = K$ (no redundant file requests);

Region 3: $K > 2$, $\tilde{N}(\mathbf{d}) < K$ (with redundant file requests).

Note that Region 2 is possible only when $K \leq N$, and Region 3 is when there are multiple users requesting the same file. We summarize our results below:

- For both Regions 1 and 2, we prove that the popularity-first-based lower bound by **P6** is tight, *i.e.,* the optimized MCCS by **P4** attains this lower bound. In particular, in Region 1, by Theorem 2, the result implies the optimality of the MCCS (with the optimized popularity-first placement) for any caching with uncoded placement. Also, the tight bound reveals that there is no loss of optimality by zero-padding in coded messages in the MCCS in Regions 1 and 2.

- For Region 3, we show that there may be a performance gap between the optimized MCCS and the popularity-first-based lower bound by **P6**. The loss is due to zero-padding in the coded messages in the delivery phase. Nonetheless, numerical results from our experiments show that, in general, the loss only appears in limited scenarios and is very small.

4.3.1 Expression of $\bar{R}_{\mathbf{MCCS}}(\mathbf{a})$

We first rewrite the expression of $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.8) for the MCCS. Given placement vector $\mathbf{a} \in \mathcal{Q}$, we rewrite $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.7) for demand vector $\mathbf{d}$ as

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a}) = \sum_{l=0}^{K-1} \sum_{\mathcal{S}\in\mathcal{A}^{l+1},\mathcal{S}\cap\mathcal{U}\neq\emptyset} \max_{k\in\mathcal{S}} a_{d_k,l} \tag{4.15}$$

where we regroup the terms in $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ based on the size $|\mathcal{S}|$ of the non-redundant groups, and $\mathcal{U}$ is the leader group for $\mathbf{d}$. Define $\psi : \mathcal{I}_{|\mathcal{U}|} \to \mathcal{U}$ as a bijective map from $\mathcal{I}_{|\mathcal{U}|}$ to the leader group $\mathcal{U}$, such that the requested (distinct) files by the users in $\mathcal{U}$ are ordered in decreasing popularity, *i.e.,* $p_{d_{\psi(1)}} \geq \ldots \geq p_{d_{\psi(|\mathcal{U}|)}}$. Recall that $\phi : [|\mathcal{D}|] \to \mathcal{D}$ defined in Lemma 3 maps the indexes of distinct files with the same file popularity order. By the relation of $\mathcal{D}$ and $\mathcal{U}$, the two mappings $\psi(\cdot)$ and $\phi(\cdot)$ are based on the same file popularity order, and we have $d_{\psi(i)} = \phi(i)$, $i = 1,\ldots,\tilde{N}(\mathbf{d})$. Also, since $\mathbf{a} \in \mathcal{Q}$, we have

$$a_{d_{\psi(1)},l} \geq \ldots \geq a_{d_{\psi(\tilde{N}(\mathbf{d}))},l}, \quad l \in \mathcal{K}. \tag{4.16}$$

To evaluate the inner max operation in (4.15), we partition the coded messages into different categories according to the user subsets. Recall that cache subgroup $\mathcal{A}^{l+1}$ is the set of all $\binom{K}{l+1}$ user subsets with size $|\mathcal{S}| = l+1$. Among these user subsets, there are $\binom{K-1}{l}$ subsets containing user $\psi(1)$ in $\mathcal{U}$. Let $\bar{a}_{\psi(1),l}$ denote the size of the coded message corresponding to each of these $\binom{K-1}{l}$ subsets containing user $\psi(1)$. From (4.16), we have

$$\bar{a}^{\mathcal{S}}_{\psi(1),l} = \max_{k\in\mathcal{S}} a_{d_k l}, \text{ for } \mathcal{S} \in \mathcal{A}^{l+1}, \psi(1) \in \mathcal{S}\cap\mathcal{U}. \tag{4.17}$$

Similarly, there are $\binom{K-2}{l}$ user subsets in $\mathcal{A}^{l+1}$ that contain user $\psi(2)$ but not $\psi(1)$ in $\mathcal{U}$. Denote the size of the coded message corresponding to each of these subsets as $\bar{a}_{\psi(2),l}$,

then

$$\bar{a}^{\mathcal{S}}_{\psi(2),l} = \max_{k\in\mathcal{S}} a_{d_k,l}, \text{ for } \mathcal{S}\in\mathcal{A}^{l+1}, \psi(1)\notin\mathcal{S}, \psi(2)\in\mathcal{S}\cap\mathcal{U}. \tag{4.18}$$

Following the above, in general, the number of user subsets in $\mathcal{A}^{l+1}$ that include $\psi(i)$ but not $\psi(1),\ldots,\psi(i-1)$ is $\binom{K-i}{l}$. Let $\bar{a}_{\psi(i),l}$ denote the size of the coded message corresponding to each of these subsets. Then, we have

$$\bar{a}^{\mathcal{S}}_{\psi(i),l} = \max_{k\in\mathcal{S}} a_{d_k,l}, \quad \text{for } \mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1} \tag{4.19}$$

where $\tilde{\mathcal{A}}_i^{l+1} \triangleq \{\mathcal{S}\in\mathcal{A}^{l+1}\colon \{\psi(1),\ldots,\psi(i-1)\}\cap\mathcal{S}=\emptyset, \psi(i)\in\mathcal{S}\cap\mathcal{U}\}$, with $|\tilde{\mathcal{A}}_i^{l+1}| = \binom{K-i}{l}$.

Based on (4.19), we can rewrite (4.15) as

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a}) = \sum_{l=0}^{K-1}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}} \bar{a}^{\mathcal{S}}_{\psi(i),l}. \tag{4.20}$$

Averaging $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.20) over $\mathbf{d}$, $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.8) can be rewritten as

$$\bar{R}_{\text{MCCS}}(\mathbf{a}) = \sum_{\mathcal{D}\subseteq\mathcal{N}}\sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})}\prod_{k=1}^{K} p_{d_k} R_{\text{MCCS}}(\mathbf{d};\mathbf{a}). \tag{4.21}$$

where $\mathcal{T}(\mathcal{D})$ is defined below **P5** in Lemma 2. With the expression in (4.21), we now can directly compare the minimum average rate in **P4** with **P5** or **P6**.

4.3.2 Region 1: $K = 2$

In this region, We have the following result on the optimality of the MCCS.

Theorem 3. For a caching problem of N files with popularity distribution $\mathbf{p}$ and local cache size M, for $K = 2$ users, the minimum average rate of the optimized MCCS in **P4** attains the lower bound given by **P5**. Thus, the MCCS is optimal for caching with uncoded placement.

Proof. We first show that **P4** and **P6** are equivalent, *i.e.,* $\bar{R}_{\text{MCCS}}(\mathbf{a}) = \bar{R}_{\text{lb}}(\mathbf{a})$, for $\mathbf{a}\in\mathcal{Q}$. Comparing the two expressions in (4.13) and (4.21), we only need to examine $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$

and $R_{\text{lb}}(\mathcal{D};\mathbf{a})$. For $\mathcal{K}=\{1,2\}$, we have $|\mathcal{D}|=\tilde{N}(\mathbf{d})=1$ or 2. We consider these two cases separately below.

Case 1) $\tilde{N}(\mathbf{d})=1$: In this case, two users request the same file. Based on the relation of two mappings $\psi(\cdot)$ and $\phi(\cdot)$ discussed in Section 4.3.1, we have $\mathcal{D}=\{\phi(1)\}$, and $d_{\psi(1)}=d_{\psi(2)}=\phi(1)$. For $K=2$, $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.14) is given by

$$R_{\text{lb}}(\mathcal{D};\mathbf{a})=a_{\phi(1),0}+a_{\phi(1),1}. \tag{4.22}$$

For $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.20), based on $\tilde{\mathcal{A}}_i^{l+1}$ defined below (4.19), for $\tilde{N}(\mathbf{d})=1$, we have $\tilde{\mathcal{A}}_1^1=\{\psi(1)\}$ and $\tilde{\mathcal{A}}_1^2=\{\psi(1),\psi(2)\}$. Thus, from (4.16) and $\bar{a}^{\mathcal{S}}_{\psi(1),l}$ in (4.17), we have

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_1^1}\bar{a}^{\mathcal{S}}_{\psi(1),0}+\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_1^2}\bar{a}_{\psi(1),1}=a_{d_{\psi(1)},0}+a_{d_{\psi(1)},1}. \tag{4.23}$$

Since $d_{\psi(1)}=\phi(1)$, from (4.22) and (4.23), we have

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=a_{\phi(1),0}+a_{\phi(1),1}=R_{\text{lb}}(\mathcal{D};\mathbf{a}). \tag{4.24}$$

Case 2) $\tilde{N}(\mathbf{d})=2$: In this case, two users request two different files, *i.e.,* $\mathcal{D}=\{\phi(1),\phi(2)\}$. By the popularity-first placement in (4.4), we have $a_{\phi(1),l}\geq a_{\phi(2),l}$, $l=1,2$. Following this, for $K=2$ and $\tilde{N}(\mathbf{d})=2$, $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.14) is given by

$$R_{\text{lb}}(\mathcal{D};\mathbf{a})=a_{\phi(1),0}+a_{\phi(2),0}+a_{\phi(1),1}. \tag{4.25}$$

For $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.20), we have

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=\sum_{l=0}^{1}\sum_{i=1}^{2}\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}}\bar{a}^{\mathcal{S}}_{\psi(i),l}. \tag{4.26}$$

Note that by definition, $\tilde{\mathcal{A}}_2^2$ contains those user subsets of size two that include $\psi(2)$ but not $\psi(1)$. However, for $K=2$, when excluding $\psi(1)$, the size of the user subset can only be 1. Thus, we have $\tilde{\mathcal{A}}_2^2=\emptyset$ for $K=2$. In addition, we have $\tilde{\mathcal{A}}_1^1=\{\psi(1)\}$, $\tilde{\mathcal{A}}_2^1=\{\psi(2)\}$, and $\tilde{\mathcal{A}}_1^2=\{\psi(1),\psi(2)\}$. Thus, $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.26) is given by

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_1^1}\bar{a}^{\mathcal{S}}_{\psi(1),0}+\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_2^1}\bar{a}^{\mathcal{S}}_{\psi(2),0}+\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_1^2}\bar{a}^{\mathcal{S}}_{\psi(1),1}$$

$$= a_{d_{\psi(1)},0} + a_{d_{\psi(2)},0} + a_{d_{\psi(1)},1}$$
$$= a_{\phi(1),0} + a_{\phi(2),0} + a_{\phi(1),1} = R_{\text{lb}}(\mathcal{D}; \mathbf{a}) \tag{4.27}$$

where the second equality is due to (4.16)–(4.18), and the third equality is because of $d_{\psi(i)} = \phi(i)$. From (4.24) and (4.27), we conclude that for $K = 2$, $R_{\text{MCCS}}(\mathbf{d}; \mathbf{a}) = R_{\text{lb}}(\mathcal{D}; \mathbf{a})$, for $\mathbf{a} \in \mathcal{Q}$. Thus, **P4** and **P6** are equivalent. By Theorem 2, **P4** and **P5** are equivalent for $K = 2$, and we complete the proof. □

For $K = 2$, Theorem 3 shows that the lower bound given by **P5** is tight. It also indicates two types of optimality for the MCCS: 1) the optimality of the popularity-first cache placement for the MCCS; and 2) the optimality of the MCCS for caching with uncoded placement. The tight bound enables us to characterize the exact memory-rate tradeoff for caching with uncoded placement. Also, as discussed in Section 4.1.2, zero-padding is commonly used in constructing coded messages. The optimality of the MCCS reveals that there is no loss of optimality to use zero-padding in the MCCS for the coded message.

4.3.3 Region 2: $K > 2$, $\tilde{N}(\mathbf{d}) = K$

When $K \leq N$, this region may occur if every user requests a different file, *i.e.,* $|\mathcal{D}| = |\mathcal{U}| = \tilde{N}(\mathbf{d}) = K$. Note that under this condition, the probability of each file being requested has changed. Let $p_{i|K}$ denote the conditional probability of file i being requested, given $\tilde{N}(\mathbf{d}) = K$. Then, in this case, $\bar{R}_{\text{lb}}(\mathbf{a})$ in (4.13) of **P6** is rewritten as

$$\bar{R}_{\text{lb}}(\mathbf{a}) = \sum_{\mathcal{D} \subseteq \mathcal{N}^K} \sum_{\mathbf{d} \in \mathcal{T}(\mathcal{D})} \prod_{k=1}^{K} p_{d_k|K} R_{\text{lb}}(\mathcal{D}; \mathbf{a}), \tag{4.28}$$

and $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.21) for the MCCS is rewritten as

$$\bar{R}_{\text{MCCS}}(\mathbf{a}) = \sum_{\mathcal{D} \subseteq \mathcal{N}^K} \sum_{\mathbf{d} \in \mathcal{T}(\mathcal{D})} \prod_{k=1}^{K} p_{d_k|K} R_{\text{MCCS}}(\mathbf{d}; \mathbf{a}). \tag{4.29}$$

Comparing the expressions in (4.28) and (4.29) in **P4** and **P6**, respectively, we have the following result.

Theorem 4. For the caching problem of N files with distribution $\mathbf{p}$ and local cache size M, in Region 2, the optimized MCCS in **P4** attains the popularity-first-based lower bound given by **P6**.

Proof. To prove the result, we only need to show that $R_{\text{lb}}(\mathcal{D};\mathbf{a}) = R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$, for $\mathbf{a} \in \mathcal{Q}$ and $\tilde{N}(\mathbf{d}) = K$. Consider $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.20). Since every user requests a distinct file, the leader group includes all users and can be written as $\mathcal{U} = \{\psi(1),\ldots,\psi(K)\}$. Thus, for any user subset $\mathcal{S}$, we have $\mathcal{S} \subseteq \mathcal{U}$. From the definition of $\tilde{\mathcal{A}}_i^{l+1}$ below (4.19), this means that for any $\mathcal{S} \in \tilde{\mathcal{A}}_i^{l+1}$, $\psi(i) \in \mathcal{S} \subseteq \mathcal{U}$, and $d_{\psi(i)}$ is the most popular file requested in $\mathcal{S}$. By (4.16), we have $\max_{k\in\mathcal{S}} a_{d_k,l} = a_{d_{\psi(i)},l}$. Thus, $\bar{a}^{\mathcal{S}}_{\psi(i),l}$ in (4.19) is given by

$$\bar{a}^{\mathcal{S}}_{\psi(i),l} = \max_{k\in\mathcal{S}} a_{d_k,l} = a_{d_{\psi(i)},l}, \quad \text{for } \mathcal{S} \in \tilde{\mathcal{A}}_i^{l+1}. \tag{4.30}$$

Since $|\tilde{\mathcal{A}}_i^{l+1}| = \binom{K-i}{l}$, $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.20) is given by

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a}) = \sum_{l=0}^{K-1}\sum_{i=1}^{K}\binom{K-i}{l} a_{d_{\psi(i)},l} = \sum_{l=0}^{K-1}\sum_{i=1}^{K}\binom{K-i}{l} a_{\phi(i),l}, \tag{4.31}$$

which is the same as $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.14) for $\tilde{N}(\mathbf{d}) = K$. Thus, we conclude that $\bar{R}_{\text{lb}}(\mathbf{a}) = \bar{R}_{\text{MCCS}}(\mathbf{a})$ in Region 2. □

Based on Theorem 4, we provide the following remarks.

The optimality of popularity-first placement

We point out that in Region 2, the MCCS and the CCS are identical. Recall from Section 4.1.2 that the MCCS uses a modified coded delivery strategy: it removes the redundant coded messages in that of the CCS when there are redundant requests. In Region 2, since all file requests are distinct, there is no redundant message in the delivery phase. As a result, the MCCS is the same as the CCS. Following this, and combining it with the result in Region 1, we have the following corollary.

Corollary 2. In Regions 1 and 2, the optimal cache placement for the MCCS is a popularity-first placement, *i.e.,* there is no loss of optimality by restricting $\mathbf{a} \in \mathcal{Q}$ in **P4**.

Proof. In Region 1, the claim immediately follows Theorem 3, as discussed at the end of Section 4.3.2. In Region 2, the MCCS is the same as the CCS. It has been shown that the optimal placement for the CCS is a popularity-first placement [37]. Thus, the claim immediately follows for the MCCS in Region 2. □

The optimality of Condition C1

Note that in Section 4.1.1, we have imposed Condition C1 for the cache placement to simplify the problem. The popularity-first placement is defined under Condition C1. Corollary 2 reveals that Condition C1, in fact, holds true for the optimal cache placement of the MCCS, *i.e.,* the size of each subfile $W_{n,\mathcal{S}}$ only depends on the size of the user subset $|\mathcal{S}|$. This is summarized in the following corollary.

Corollary 3. Condition C1 is the property of the optimal cache placement for the MCCS in Regions 1 and 2.

Note that the optimality of Condition C1 for the MCCS has only been demonstrated through numerical results in [53]. We show analytically in Corollary 3 that this property holds for the MCCS in Regions 1 and 2.

Effect of zero-padding in coded caching

As mentioned earlier, for nonuniform file popularity, zero-padding is a common technique to form the coded messages for both the CCS [33, 37, 47, 48] and the MCCS [48, 53]. However, the effect of using it has never been discussed or studied. The tight lower bounds shown in Theorems 3 and 4 indicate that the use of zero-padding does not cause any loss, as stated below.

Corollary 4. Zero-padding in the coded messages incurs no loss of optimality for the MCCS in Regions 1 and 2.

The optimality of the CCS

The above analysis focuses on the MCCS. The result in Theorem 4 also leads to several conclusions on the optimality of the CCS, as discussed below:

Zero-padding Note that the delivery strategy of the CCS does not distinguish whether file requests are the same or different, *i.e.,* it treats all the requested files as distinct files to form the coded messages.[2] This means that for any demand vector $\mathbf{d}$, the CCS is effectively equivalent to the case when $\tilde{N}(\mathbf{d}) = K$ for the MCCS in Region 2, where all users request different files. In other words, the average rate of the CCS (averaged over *all* $\mathbf{d}$'s) is equal to that of the MCCS in Region 2.[3]

Assuming all files are *treated* as distinct in the delivery phase, we can also construct a lower bound under popularity-first placement. Given how **P6** in Region 2 is formulated, this lower bound is equivalent to **P6**. By Theorem 4, it follows that the average rate of the CCS attains the popularity-first-based lower bound, assuming all files are treated as distinct. Since the popularity-first placement is optimal for the CCS under nonuniform file popularity [37], this tight lower bound also means that zero-padding used in the CCS incurs no loss. We state this conclusion below.

Corollary 5. Using zero-padding for the CCS incurs no loss of optimality under nonuniform file popularity.

The optimality of Condition C1 Following the discussion above, the tight lower bound also implies that Condition C1 is the property of the optimal cache placement for the CCS. This is by the similar argument used in the zero-padding discussion. Although this optimality has been proven in [37], the method used there is more involved. Our results in Theorem 4 provides a simpler alternative proof of this result.

The optimality of the CCS Following the discussion in a), since the CCS attains the lower bound given by **P6**, we also conclude that *the CCS is optimal in terms of the average rate for caching under popularity-first placement, if all file requests are distinct (i.e., the*

[2]The CCS was originally proposed for the worst-case peak rate consideration, where all file requests are distinct.

[3]This result should not be confused with the conclusion that the CCS and the MCCS being the same in Region 2 in Section 4.3.3. Here, there may be multiple requests for the same file, although the CCS does not distinguish them. In Section 4.3.3, the comparison is restricted to Region 2 where all file requests are indeed distinct.

worst-case). Note that in the literature of caching with uncoded placement, for uniform file popularity, the optimality of the CCS in terms of the worst-case peak rate in the case of $K \leq N$ has been proven [45, 46]. For nonuniform file popularity, although many existing works study the cache placement for the CCS [16, 33–37, 47, 48], the optimality of the CCS in this case has never been studied or known. Our result sheds some light on the optimality of the CCS under nonuniform file popularity.

4.3.4 Region 3: $K > 2$, $\tilde{N}(\mathbf{d}) < K$

This region reflects the scenario when there exist multiple users request the same file. In the following, we show that, in general, there may exist a gap between $R_{\text{MCCS}}(\mathbf{d}; \mathbf{a})$ and $R_{\text{lb}}(\mathcal{D}; \mathbf{a})$, for $\mathbf{a} \in \mathcal{Q}$. The main cause of the gap is the zero-padding used in the MCCS.

Examining the expressions of $R_{\text{MCCS}}(\mathbf{d}; \mathbf{a})$ in (4.20) and $R_{\text{lb}}(\mathcal{D}; \mathbf{a})$ in (4.14), we see that they contain the same number of coded messages, which is $\sum_{l=0}^{K-1} \sum_{i=1}^{\tilde{N}(\mathbf{d})} \binom{K-i}{l}$. The only difference between $R_{\text{MCCS}}(\mathbf{d}; \mathbf{a})$ and $R_{\text{lb}}(\mathcal{D}; \mathbf{a})$ is the size of each coded message $|C_{\mathcal{S}}|$, *i.e.,* $\bar{a}^{\mathcal{S}}_{\psi(i),l}$ and $a_{\phi(i),l}$. Thus, we need to examine whether $\bar{a}^{\mathcal{S}}_{\psi(i),l}$ is the same as $a_{\phi(i),l}$, for any $\mathcal{S}$. To better explain our result, in the following, we first use an example to show that $\bar{a}^{\mathcal{S}}_{\psi(i),l}$ and $a_{\phi(i),l}$ may be different, which is due to zero-padding.

Example: Assume that two users request file $\phi(1)$, the most popular file in the requests. One user is in the leader group $\mathcal{U}$, denoted by $\psi(1)$ and the other from a redundant group, denoted by $k' \notin \mathcal{U}$, where $d_{k'} = \phi(1)$. From $R_{\text{lb}}(\mathcal{D}; \mathbf{a})$ in (4.14), for all $\binom{K-2}{l}$ user subsets that include user $\psi(2)$ but not user $\psi(1)$, the size of coded messages corresponding to these subsets is always $a_{\phi(2),l}$ ($d_{\varphi(2)} = \phi(2)$). Now, for $R_{\text{MCCS}}(\mathbf{d}; \mathbf{a})$ in (4.20), consider user subset $\mathcal{S}$ that includes users $\psi(2)$ and k' but not user $\psi(1)$. From (4.18), the size of coded messages for $\mathcal{S}$ is $\bar{a}^{\mathcal{S}}_{\psi(2),l} = a_{d_{k'},l} = a_{\phi(1),l}$, due to zero-padding. Since $a_{\phi(1),l} \geq a_{\phi(2),l}$, in this case, zero-padding by the MCCS results in a longer coded message for user subsets that include user k' but not the leader user $\psi(1)$, as it always zero-pads the subfile to the longest one in the subset.

Similar to the above example, in general, $\bar{a}^{\mathcal{S}}_{\psi(i),l}$ and $a_{\phi(i),l}$ may be different for a coded message corresponding to user subset $\mathcal{S}$, where $\mathcal{S}$ includes a user from a redundant group

who requests a file that is more popular than the rest requested by all other users in $\mathcal{S}$. For the MCCS using the popularity-first placement in (4.4), the coded message is zero-padded to the size of the subfile requested by that user from the redundant group (the largest). In contrast, for the lower bound $R_{\text{lb}}(\mathcal{D};\mathbf{a})$, the size of the coded message is that of the subfile of the most popular file ($\phi(i)$, for some i) among files requested by those users in the leader group, *i.e.,* $\mathcal{S}\cap\mathcal{U}$. This mismatch between $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ and $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ leads to a possible gap between the average rate of the optimized MCCS and the lower bound in **P6**. To further quantify the difference between $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ and $R_{\text{lb}}(\mathcal{D};\mathbf{a})$, we re-express $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in the following lemma.

Lemma 4. For any demand vector $\mathbf{d}$, let $\hat{N}(i)$ denote the total number of redundant requests for files $\{\phi(1),\ldots,\phi(i)\}$, for $i=1,\ldots,\tilde{N}(\mathbf{d})$, and $\hat{N}(0)=0$. Then, $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.20) can be rewritten as

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=\sum_{l=0}^{K-1}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\left[\sum_{j=i}^{\tilde{N}(\mathbf{d})}\binom{K-j-\hat{N}(i-1)}{l}-\sum_{j=i+1}^{\tilde{N}(\mathbf{d})}\binom{K-j-\hat{N}(i)}{l}\right]a_{\phi(i),l}. \tag{4.32}$$

Proof. See Appendix B.4. □

The expression in (4.32) clearly shows the difference between $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ and $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.14): the gap between the two is determined by $\hat{N}(i), i=1,\ldots,\tilde{N}(\mathbf{d})$, *i.e.,* the number of redundant requests for files in $\mathcal{D}$. From this, we identify the following two cases where $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=R_{\text{lb}}(\mathcal{D};\mathbf{a})$:

Case i): $|\mathcal{D}|=|\mathcal{U}|=\tilde{N}(\mathbf{d})=1$. In this case, all users request the same file. We have $a_{d_{\psi(1)},l}=\ldots=a_{d_{\psi(K)},l}=a_{\phi(1),l}$, since only one file is requested. Then, (4.32) is given by

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=\sum_{l=0}^{K-1}\binom{K-1}{l}a_{\phi(1),l}=R_{\text{lb}}(\mathcal{D};\mathbf{a}).$$

Case ii): $\hat{N}(i)=0$, $i=1,\ldots,\tilde{N}(\mathbf{d})-1$. In this case, only file $\phi(\tilde{N}(\mathbf{d}))$, *i.e.,* the least popular file in $\mathcal{D}$, has redundant requests. In other words, all the users in the redundant

group request file $\phi(\tilde{N}(\mathbf{d}))$. From (4.32), it is straightforward to show

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a}) = \sum_{l=0}^{K-1}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\binom{K-i}{l}a_{\phi(i),l} = R_{\text{lb}}(\mathcal{D};\mathbf{a}).$$

Finally, we point out that although $\bar{R}_{\text{MCCS}}(\mathbf{a})$ and $\bar{R}_{\text{lb}}(\mathbf{a})$ are generally not the same due to the existence of redundant requests as discussed above, the optimal placement solution $\mathbf{a}^*$ to **P4**, **P5**, and **P6** can still be the same for some values of M and $\mathbf{p}$. This also leads to the following case where $\bar{R}_{\text{MCCS}}(\mathbf{a}) = \bar{R}_{\text{lb}}(\mathbf{a})$.

Case iii): If **P4**, **P5** *and* **P6** *result in the same optimal solution* $\mathbf{a}^*$ *that satisfies* $\mathbf{a}_1^* = \cdots = \mathbf{a}_N^*$, *then* $\bar{R}_{MCCS}(\mathbf{a}^*) = \bar{R}_{lb}(\mathbf{a}^*)$. In this case, all the subfiles have the same size. Thus, there is no zero-padding and no potential waste. One obvious example is the special case of uniform file popularity, where the optimal $\mathbf{a}_n^*$'s are all identical and the same for **P4**, **P5**, and **P6**. In this case, the same result $\bar{R}_{\text{MCCS}}(\mathbf{a}^*) = \bar{R}_{\text{lb}}(\mathbf{a}^*)$ has been shown in [44]. Below, we provide a simple proof of our statement.

Since $a_{1,l}^* = \ldots = a_{N,l}^*$, $l = 0,\ldots,K$, $\bar{a}_{\psi(i),l}^{\mathcal{S}}$ in (4.19) can be written as

$$\bar{a}_{\psi(i),l}^{\mathcal{S}} = \max_{k\in\mathcal{S}} a_{d_k,l}^* = a_{1,l}^*, \quad \text{for } \mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}. \tag{4.33}$$

Since $|\tilde{\mathcal{A}}_i^{l+1}| = \binom{K-i}{l}$, $R_{\text{MCCS}}(\mathbf{d};\mathbf{a}^*)$ in (4.20) is written as

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a}^*) = \sum_{l=0}^{K-1}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\binom{K-i}{l}a_{1,l}^*. \tag{4.34}$$

Similarly, $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.12) of **P5** and (4.14) of **P6** can both be rewritten as

$$R_{\text{lb}}(\mathcal{D};\mathbf{a}^*) = \sum_{l=0}^{K-1}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\binom{K-i}{l}a_{1,l}^* = R_{\text{MCCS}}(\mathbf{d};\mathbf{a}^*). \tag{4.35}$$

Thus, from (4.11) and (4.21), we conclude that for $\mathbf{a}_1^* = \ldots = \mathbf{a}_N^*$,

$$\bar{R}_{\text{MCCS}}(\mathbf{a}^*) = \bar{R}_{\text{lb}}(\mathbf{a}^*) = \sum_{\mathcal{D}\subseteq\mathcal{N}}\sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})}\prod_{k=1}^{K}p_{d_k}\sum_{l=0}^{K-1}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\binom{K-i}{l}a_{1,l}^*. \tag{4.36}$$

Remark 9. As discussed above, when there are redundant file requests, zero-padding the message to the largest subfile may lead to a loss in the average rate. One possible solution

to avoid this loss is to create as many subfiles of equal size as possible during the placement phase. Coincidentally, such an approach was exploited in [52] for $N = 2$ files, where a placement scheme was proposed to create equal subfile size and was shown to be an optimal caching scheme with uncoded placement for two files.

4.4 Optimal Cache Placement for the MCCS

Due to the more complicated delivery scheme by the MCCS, finding the optimal cache placement for the MCCS under nonuniform file popularity is challenging, and the problem has not been solved. The optimal cache placement for the CCS under nonuniform file popularity has recently been obtained in Chapter 3. In this section, through reformulating **P4**, we show that the cache placement problem has a similar structure to that for the CCS. As a result, the optimal cache placement structure for the MCCS inherits that for the CCS characterized in Chapter 3. Extending the results from the CCS, we present a simple algorithm to compute the optimal cache placement solution for the MCCS. In the following, we first reformulate **P4** and then describe the optimal cache placement solution structure.

4.4.1 Reformulation of P4

It is straightforward to see that at the optimum, the cache memory is fully utilized, and the local cache constraint (4.3) is attained with equality, which can be replaced by

$$\sum_{n=1}^{N}\sum_{l=1}^{K}\binom{K-1}{l-1}a_{n,l} = M. \tag{4.37}$$

Next, according to Lemma 1 in Chapter 3, for any popularity-first placement $\mathbf{a} \in \mathcal{Q}$, constraint (4.10) can be equivalently replaced by the following two constraints:

$$a_{1,0} \geq 0 \quad \text{and} \quad a_{N,l} \geq 0, \quad l \in \mathcal{K}. \tag{4.38}$$

Finally, for $\mathbf{a} \in \mathcal{Q}$, the expression of the average rate $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.8) can be simpli-

fied as [53][4]

$$\bar{R}_{\text{MCCS}}(\mathbf{a}) = \sum_{l=0}^{K-1}\binom{K}{l+1}\sum_{n=1}^{N}\left(\left(\sum_{n'=n}^{N}p_{n'}\right)^{l+1}-\left(\sum_{n'=n+1}^{N}p_{n'}\right)^{l+1}\right)a_{n,l} - \sum_{u=1}^{\min\{N,K\}}\sum_{l=0}^{K-u-1}\binom{K-u}{l+1}\sum_{i=1}^{K-u}\binom{K-u-i}{l}\sum_{n=1}^{N}P_{i,u,n}a_{n,l} \tag{4.39}$$

where $P_{i,u,n}$ is the joint probability of i) having u distinct file requests; and ii) file W_n being the i-th least popular file among files requested by all the users that are not in the leader group $\{d_k : k \in \mathcal{K}\backslash\mathcal{U}\}$. The expression of $P_{i,u,n}$ is derived in [53], which is lengthy and non-essential in developing our results. Therefore, we omit it here but only point out that $P_{i,u,n}$ is not a function of $\mathbf{a}$.

The expression of $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.39) is a weighted sum of $a_{n,l}$'s. Define $\mathbf{g}_n \triangleq [g_{n,0},\ldots,g_{n,K}]^{\mathrm{T}}$, with

$$g_{n,l} \triangleq \binom{K}{l+1}\left(\left(\sum_{n'=n}^{N}p_{n'}\right)^{l+1}-\left(\sum_{n'=n+1}^{N}p_{n'}\right)^{l+1}\right) - \sum_{u=1}^{\min\{N,K\}}\binom{K-u}{l+1}\sum_{i=1}^{K-u}\binom{K-u-i}{l}P_{i,u,n}. \tag{4.40}$$

Also, from (4.2) and (4.37), define $\mathbf{b} \triangleq [b_0,\ldots,b_K]^T$ with $b_l \triangleq \binom{K}{l}$, and $\mathbf{c} \triangleq [c_0,\ldots,c_K]^T$ with $c_l \triangleq \binom{K-1}{l-1}$. The cache placement optimization problem **P4** can be reformulated into the following equivalent LP problem

$$\begin{aligned}\textbf{P7:}\ \min_{\mathbf{a}\in\mathcal{Q}}\quad & \sum_{n=1}^{N}\mathbf{g}_n^{\mathrm{T}}\mathbf{a}_n \\ \text{s.t.}\quad & (4.38),\ \text{and} \\ & \mathbf{b}^{\mathrm{T}}\mathbf{a}_n = 1,\ n\in\mathcal{N}, \qquad (4.41)\\ & \sum_{n=1}^{N}\mathbf{c}^{\mathrm{T}}\mathbf{a}_n = M. \qquad (4.42)\end{aligned}$$

[4]The expression in (4.8) can be simplified to (4.39) by utilizing the properties of the popularity-first placement with ordered $a_{n,l}$'s to eliminate the max operation in (4.8). The expression of $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.39) can be evaluated in polynomial time instead of the exponential time required in (4.8), which simplifies **P4**.

Connection to the cache placement problem for the CCS

As described in Section 4.1.2, the MCCS only delivers the non-redundant messages, while the CCS delivers the coded messages corresponding to all the user subsets. The expression of $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.39) explicitly shows this difference by grouping all the messages into the first term and the redundant messages into the second term. The cache placement optimization problem for the CCS was formulated in problem **P2**. It essentially minimizes the first term in (4.39), with all the constraints on $\mathbf{a}$ being the same as in **P7**, except that the expression of $\mathbf{g}_n$ is different. Based on this structural similarity, the structural properties of the placement obtained for the CCS can be straightforwardly extended to **P7** for the MCCS.

Since the cache placement result is the direct extension from that of the CCS in Chapter 3, in the following, we focus on describing the placement structure and omit the details of derivations or proofs.

4.4.2 Optimal Cache Placement: Solution Structure

A major challenge in solving **P7** is that the placement vectors in the optimal cache placement $\mathbf{a}$ can be all different, depending on the file popularity distribution. It turns out that the number of distinct placement vectors in the optimal $\mathbf{a}$ is limited to at most three. Similar to Chapter 3, we first define *file group* below.

Definition 6 (File group)**.** A file group is a subset of $\mathcal{N}$ that contains all files with the same cache placement vector, *i.e.,* for any two files W_n and $W_{n'}$, if their placement vectors $\mathbf{a}_n = \mathbf{a}_{n'}$, then they belong to the same file group.

The first structural property, in terms of file groups, of the optimal cache placement for the MCCS is provided in the following theorem.

Theorem 5. For N files with any file popularity distribution $\mathbf{p}$, and for any K and $M \leq N$, there are at most three file groups under the optimal cache placement $\{\mathbf{a}_n\}$ for **P7**.

Proof. The only differences between **P7** and the the cache placement optimization problem for the CCS in **P2** are the expressions of $\mathbf{g}_n$, which does not affect the arguments used in the

proof of Theorem 1. Thus, the same result straightforwardly applies to the optimal solution of **P7** for the MCCS. □

Theorem 5 indicates that, regardless of N, $\mathbf{p}$, and M, there are at most three unique values among the optimal placement vectors $\{\mathbf{a}_n\}$. This leads to three possible cases: one, two, or three file groups. In the following, we provide the optimal placement solution $\{\mathbf{a}_n\}$ for each case.

One file group

In this case, the optimal cache placement vectors are identical for all files, *i.e.,* $\mathbf{a}_1 = \cdots = \mathbf{a}_N$. Under this structure, the cache placement problem is reduced to that under uniform file popularity (*i.e.,* all files have the same cache placement vector), of which the optimal solution has been obtained in closed-form [96], which is identical to that for the CCS under uniform file popularity [33]. Specifically, the optimal placement $\mathbf{a}_n$, for any file $n \in \mathcal{N}$, has at most two nonzero elements, which is given as follows:

$$\begin{cases} a_{n,l_o} = \frac{1+\lfloor v \rfloor - v}{\binom{K}{\lfloor v \rfloor}},\ a_{n,l_o+1} = \frac{v-\lfloor v \rfloor}{\binom{K}{\lceil v \rceil}}, & l_o = \lfloor v \rfloor \\ a_{n,l} = 0, & \forall\, l \neq l_o \text{ or } l_o+1 \end{cases} \tag{4.43}$$

where $v \triangleq \frac{MK}{N}$. In particular, when v is an integer, the optimal $\mathbf{a}_n$ has only one nonzero element: $a_{l_o} = 1/\binom{K}{l_o}$ for $l_o = MK/N$, and $a_l = 0, \forall\, l \neq l_o$.

Two file groups

In this case, there are only two unique placement vectors in $\{\mathbf{a}_n\}$, *i.e.,* $\mathbf{a}_1 = \cdots = \mathbf{a}_{n_o} \neq \mathbf{a}_{n_o+1} = \cdots = \mathbf{a}_N$, for some $n_o \in \{1, \ldots, N-1\}$. We use $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$ to represent the two unique placement vectors for the first and the second file group, respectively. Define $\bar{\mathbf{a}}_n \triangleq [a_{n,1}, \ldots, a_{n,K}]^T$as the sub-placement vector in $\mathbf{a}_n$. It specifies the subfiles stored in the local cache, and $a_{n,0}$ specifies the subfile kept at the server. Let $\bar{\mathbf{a}}_n \succcurlyeq_1 \mathbf{0}$ denote that there is only one positive element in $\bar{\mathbf{a}}_n$, and the rest are all 0's.

There are several structural properties of the optimal placement in the two-file-group case. They are all direct extensions from the optimal cache placement of the CCS in Chapter 3. We summarize them below.

Proposition 7. If there are two file groups under the optimal cache placement $\{\mathbf{a}_n\}$, the following three properties hold:

Property 1 (Proposition 1): The optimal sub-placement vector $\bar{\mathbf{a}}_{n_o+1}$ for the second file group has at most one nonzero element.

Property 2 (Proposition 2): If $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$, then $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ are different by only one element.

Property 3 (Proposition 3): If $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$, then $a_{n_o,0} = 0$.

Following the properties in Proposition 7, the optimal placement solution for **P7** can be one of the following two structures: 1) $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$; 2) $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$. For the completeness, we briefly present the optimal placement solutions below.

Case 1: $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$. This condition means that no cache is allocated to the files in the second group. By (4.2), we have $\mathbf{a}_{n_o+1} = [1, 0, 0, \ldots]^T$. To determine $\mathbf{a}_{n_o}$ for the first group, we treat the first n_o files as a new database. Then, the cache placement problem is reduced to the one in the one-file-group case in Section 4.4.2. Therefore, the solution is the same as in (4.43), except that N is replaced by n_o, and thus, $v = MK/n_o$.

Note that this two-file-group case and the one-file-group case in Section 4.4.2 can be combined as follows: The optimal $\mathbf{a}_{n_o}$ is given by (4.43), for $v = MK/n_o$ with $n_o \in \mathcal{N}$. The optimal n_o^* depends on $(N, \mathbf{p}, M, K)$, which can be determined via a 1-D search over n_o that gives the minimum $\bar{R}_{\text{MCCS}}(\mathbf{a})$, where $\bar{R}_{\text{MCCS}}(\mathbf{a})$ is computed using the closed-form expression in (4.39). For the overall algorithm, please refer to Algorithm 1, with the only exception that the average rate is computed using $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.39).

Case 2: $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$. In this case, only one element in $\bar{\mathbf{a}}_{n_o+1}$ is nonzero. Assume $a_{n_o+1,l_o} > 0$, for some $l_o \in \mathcal{K}$, and $a_{n_o+1,l} = 0$, $\forall l \neq l_o$, $l \in \mathcal{K}$. By Property 2 in Proposition 7, the element that is different between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ can be either at index l_o or some l_1, for $l_1 \neq l_o$. Thus, for $\mathbf{a} \in \mathcal{Q}$, there are only two possible cases for $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$: *2.i)* $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$; or *2.ii)* $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$, for some $l_1 \neq l_o$, $l_1 \in \mathcal{K}$. We present the solution in each of these two cases:

Case 2.i) $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$: In this case, the only different element between $\bar{\mathbf{a}}_{n_o}$

and $\bar{\mathbf{a}}_{n_o+1}$ is at position l_o, *i.e.,* the nonzero element a_{n_o+1,l_o} in $\bar{\mathbf{a}}_{n_o+1}$. Also, by Property 3 in Proposition 7, a_{n_o,l_o} is the only nonzero element in $\mathbf{a}_{n_o}$, and $a_{n_o,l} = a_{n_o+1,l} = 0$, for $\forall l \neq l_o, l \in \mathcal{K}$. For $\bar{\mathbf{a}}_{n_o+1}$, there are two unknown nonzero elements $a_{n_o+1,0}$ and a_{n_o+1,l_o}. Solving the unknown elements in $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$ using constraints (4.41) and (4.42), the optimal $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ is given by

$$\begin{cases} a_{n_o,l_o} = \frac{1}{\binom{K}{l_o}}, \quad a_{n_o,l} = 0, \ \forall\, l \neq l_o \\ a_{n_o+1,0} = 1 - \left(\dfrac{\frac{KM}{l_o} - n_o}{N - n_o}\right), \ a_{n_o+1,l_o} = \dfrac{1}{\binom{K}{l_o}} \left(\dfrac{\frac{KM}{l_o} - n_o}{N - n_o}\right) \\ a_{n_o+1,l} = 0, \ \forall\, l \neq 0 \text{ or } l_o \end{cases} \tag{4.44}$$

where $n_o \in \{1, \ldots, N-1\}$, and the nonzero element position l_o is limited to $\left\lfloor \frac{KM}{N} \right\rfloor + 1 \leq l_o \leq \min\left\{K, \left\lceil \frac{KM}{n_o} \right\rceil - 1\right\}$.

Case 2.ii) $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$, $l_1 \neq l_o$: In this case, the only different element between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ is at position l_1, which is one of the zero elements in $\bar{\mathbf{a}}_{n_o+1}$. It follows that $a_{n_o,l_o} = a_{n_o+1,l_o} > 0$. Since $a_{n_o,0} = 0$ (by Property 3 in Proposition 7), $\mathbf{a}_{n_o}$ has two nonzero elements, a_{n_o,l_o} and a_{n_o,l_1}, and the rest are all 0's. For $\mathbf{a}_{n_o+1}$, there are two unknown nonzero elements, $a_{n_o+1,0}$ and $a_{n_o+1,l_o} = a_{n_o,l_o,}$ and the rest are all 0's. Thus, we have three unknown elements $a_{n_o,l_o} = a_{n_o+1,l_o}$, a_{n_o,l_1}, and $a_{n_o+1,0}$ to determine in $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$. Solving these unknown elements using constraints (4.41) and (4.42), the optimal $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ is given by

$$\begin{cases} a_{n_o,l_o} = \dfrac{1}{\binom{K}{l_o}} \dfrac{\frac{KM}{l_1} - n_o}{\frac{l_o}{l_1}N - n_o}, \ a_{n_o,l_1} = \dfrac{1}{\binom{K}{l_1}} \dfrac{\frac{l_o}{l_1}N - \frac{KM}{l_1}}{\frac{l_o}{l_1}N - n_o} \\ a_{n_o,l} = 0, \ \forall\, l \neq l_o \text{ or } l_1 \\ a_{n_o+1,l_o} = a_{n_o,l_o}, \qquad a_{n_o+1,0} = 1 - \dfrac{\frac{KM}{l_1} - n_o}{\frac{l_o}{l_1}N - n_o} \\ a_{n_o+1,l} = 0, \forall\, l \neq 0 \text{ or } l_o \end{cases} \tag{4.45}$$

where positions l_o and l_1 satisfy either of the following two conditions: i) $l_o > KM/N$ and $l_1 < KM/n_o$, or ii) $l_o < KM/N$ and $l_1 > KM/n_o$. Note that since $l_o, l_1 \leq K$, if $n_o \leq M$, only the condition in i) is valid.

In summary, we can consider Case 2.i) as the special case when $l_1 = l_o$. Then, for given (n_o, l_o, l_1), the closed-form solution in (4.44) of Case 2.i), or (4.45) of Case 2.ii)

completely determines the optimal $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$. We can search over all possible values of $n_o \in \{1, \ldots, N-1\}$ and $l_o, l_1 \in \mathcal{K}$ for the optimal tuple (n_o, l_o, l_1) that gives minimum $\bar{R}_{\text{MCCS}}$. For the overall algorithm, please refer to Algorithm 2.

Three file groups

Under this structure, there are three unique placement vectors in $\{\mathbf{a}_n\}$, *i.e.,* $\mathbf{a}_1 = \ldots = \mathbf{a}_{n_o} \neq \mathbf{a}_{n_o+1} = \ldots = \mathbf{a}_{n_1} \neq \mathbf{a}_{n_1+1} = \ldots = \mathbf{a}_N$, for $1 \leq n_o < n_1 \leq N-1$. We use $\mathbf{a}_{n_o}$, $\mathbf{a}_{n_o+1}$ and $\mathbf{a}_{n_1+1}$ to represent the three unique cache placement vectors for the first, second, and the third file group, respectively. Following the proof of Proposition 4, it is straightforward to show the same result in the following holds for the MCCS as well: All the memory is allocated to the first two file groups and the optimal cache placement vector for the third group is $\mathbf{a}_{n_1+1} = [1, 0, 0, \ldots]^T$.

Following the above, we treat those n_1 files in the first two groups as a new database. The cache placement problem for these first two groups is essentially reduced to the previous two-file-group case. Since $\mathbf{a}_{n_o+1} \neq \mathbf{a}_{n_1+1}$, we have $\bar{\mathbf{a}}_{n_o+1} \neq \bar{\mathbf{a}}_{n_1+1} = \mathbf{0}$. This means that $\bar{\mathbf{a}}_{n_o+1}$ includes at least one nonzero element. Therefore, the cache placement $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ belongs to the case of two file groups with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$ (for the second group) in Case 2 of Section 4.4.2. For given n_1, the optimal solution for $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ can be obtained from (4.44) or (4.45), except that N is replaced by $n_1 \in \{2, \ldots, N\}$.

The final optimal $\{\mathbf{a}_n\}$ is obtained by searching over all possible values of $n_1 \in \{2, \ldots, N\}$,[5] $n_o \in \{1, \ldots, n_1 - 1\}$, and $l_o, l_1 \in \mathcal{K}$ for the optimal tuple (n_o, n_1, l_o, l_1) that results in minimum $\bar{R}_{\text{MCCS}}$. For the overall algorithm, please refer to Algorithm 3. The algorithm simply computes $\bar{R}_{\text{MCCS}}$ using a closed-form expression for at most $(N-1)(N-2)K^2/2$ times with different values of (n_o, n_1, l_o, l_1), which can be computed efficiently in parallel.

[5]Further analysis shows that we can limit the range of search for n_1 within $n_1 \in \{M+1, \ldots, N-1\}$ [48].

The Optimal Cache Placement Solution

In summary, by Theorem 5, the optimal cache placement problem **P7** is reduced to a search among three possible file grouping structures (from one to three file groups). From Sections 4.4.2 to 4.4.2, using the closed-form expressions for $\{\mathbf{a}_n\}$ and $\bar{R}_{\text{MCCS}}(\mathbf{a})$, we obtain the candidate optimal placement for each file-group case. The final optimal placement is the one in these three solutions that results in the minimum $\bar{R}_{\text{MCCS}}(\mathbf{a})$. Since the overall algorithm only involves parallel evaluations of closed-form expressions, the algorithm is very efficient with low computational complexity.

Remark 10. We point out that, for nonuniform file popularity, although the MCCS and the CCS have the same set of candidate optimal cache placement structures and solutions, the final optimal cache placement for the two schemes may be different due to different expressions of the average rate (*e.g.,* $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in **P4**). We will demonstrate this in our numerical studies in Section 4.6. Note that this is in contrast to uniform file popularity, where the optimal cache placements of the MCCS [44] and the CCS [33] are exactly the same.

4.5 Memory-rate Tradeoff For Nonuniform File Popularity and Size

In the previous sections, we have focused on files of equal size.[6] In this section, we extend our study on the memory-rate tradeoff in caching to the more general case where both file popularity and sizes are nonuniform among files. We extend the cache placement optimization formulation in Section 4.1 to this case and propose a lower bound for caching with uncoded placement. By comparing the average rates of the optimized MCCS and the lower bound, we characterize the exact memory-rate tradeoff for $K = 2$ users.

[6] In practice, files with nonuniform sizes could also be tailored into files with uniform size which are treated separately with different popularities [97–99].

4.5.1 The Optimized MCCS

Consider each file of different sizes. We assume that file W_n has F_n bits. Recall in Section 3.1 that, for uniform file sizes, subfile size $a_{n,l}$ and cache size M are normalized by the file size and defined in the unit of file. For files with different sizes, we remove this normalization. Instead, for each file $n \in \mathcal{N}$, we define the size of each subfile in bits: $a_{n,l} \triangleq |W_{n,\mathcal{S}}|$, for $|\mathcal{S}| = l$. Likewise, the cache size M is now defined in bits. Accordingly, we rewrite file partition constraint (4.2) as

$$\sum_{l=0}^{K} \binom{K}{l} a_{n,l} = F_n,\ n \in \mathcal{N}. \tag{4.46}$$

With the above redefinitions of $a_{n,l}$ and M, the expressions of the cache size constraint (4.3) and the average delivery rate $\bar{R}_{\text{MCCS}}(\mathbf{a})$ in (4.8) still remain the same under the nonuniform file popularity and size. The cache placement optimization problem for the MCCS to minimize $\bar{R}_{\text{MCCS}}(\mathbf{a})$ under nonuniform file popularity and size is formulated as follows:

$$\begin{aligned} \textbf{P8}: \ \min_{\mathbf{a}} \ & \bar{R}_{\text{MCCS}}(\mathbf{a}) \\ \text{s.t.} \ & (4.3), (4.10), \text{and } (4.46) \end{aligned}$$

where $\bar{R}_{\text{MCCS}}(\mathbf{a})$ is given in (4.8), and the objective and constraint functions are now all expressed in bits.

Note that different from **P4**, which is restricted to the popularity-first placement $\mathbf{a} \in \mathcal{Q}$, the problem size of **P8** grows exponentially with K. We will not further study the simplification of **P4** and its performance in this dissertation. For the CCS, a similar problem under nonuniform file popularity and sizes has been studied, and tractable techniques have been developed to simplify the optimization problem with good performances [33]. The techniques can be adopted here for **P4** for the MCCS, due to the similarity between the two caching schemes. In the following, we focus on the characterization of the memory-rate tradeoff under nonuniform file popularity and size, which is unknown in the literature.

4.5.2 Memory-Rate Tradeoff Characterization

To characterize the memory-rate tradeoff for nonuniform file popularity and size, we first propose a lower bound on average rate under uncoded placement. The lower bound is a straightforward extension of the lower bound in **P5** by considering nonuniform file popularity and sizes, instead of nonuniform file popularity only.

Recall that for nonuniform file popularity and size, $a_{n,l}$ and M are defined in bits in Section 4.5.1. For a given $\mathbf{a}$, the expression of the lower bound on the average rate $\bar{R}_{\text{lb}}(\mathbf{a})$ in (4.11) remains unchanged (except that it is in bits). Since constraints (4.3), (4.10), and (4.46) remain the same, we can formulate an optimization problem to minimize $\bar{R}_{\text{lb}}(\mathbf{a})$ to obtain the lower bound for caching with uncoded placement under nonuniform file popularity and sizes. The result is given by the following lemma.

Lemma 5. For the caching problem with nonuniform file popularity and sizes, the following optimization problem provides a lower bound on the average rate for caching with uncoded placement:

$$\begin{aligned} \textbf{P9:} \min_{\mathbf{a}} \quad & \bar{R}_{\text{lb}}(\mathbf{a}) \\ \text{s.t.} \quad & (4.3), (4.10), \text{ and } (4.46) \end{aligned}$$

where $\bar{R}_{\text{lb}}(\mathbf{a})$ is given in (4.11), and the objective and constraints are all in bits.

Comparing **P8** and **P9**, we obtain the following result on the optimized MCCS for the two-user case.

Theorem 6. For the caching problem of N files with nonuniform file popularity and sizes, for $K = 2$ users, the minimum average rate for the optimized MCCS in **P8** attains the lower bound given by **P9**.

Proof. See Appendix B.5. □

Remark 11. The tight lower bound shown in Theorem 6 shows the optimality of the optimized MCCS for $K = 2$ users. It enables us to characterize the exact memory-rate tradeoff

for $K = 2$ users under nonuniform file popularity and sizes. The optimality of the MCCS also indicates that there is no loss of optimality by zero-padding. For the general case of $K > 2$ users, our numerical studies in Section 4.6 will show that the gap between the optimized MCCS (**P8**) and the lower bound (**P9**) is very small in general.

4.6 Numerical Results

In this section, we provide numerically studies on the optimized MCCS and the lower bounds obtained for caching with uncoded placement. We first consider files of the same size but with nonuniform popularity. We study the performance of the optimized MCCS (under the optimal cache placement obtained in Section 4.4), the lower bound in **P5**, and the popularity-first-based lower bound in **P6**. For comparison, we also consider a few existing strategies proposed for the CCS, including i) the optimized CCS [48], ii) a two-file-group scheme named RLFU-GCC [35], and iii) the mixed caching strategy [36].

Let $\bar{R}$ denote the average rate obtained by different schemes or the lower bound. Fig. 4.1 shows the average rate $\bar{R}$ vs. M for $N = 7$ and $K = 4$. We generate the file popularities using the Zipf distribution with $p_n = n^{-\theta}/\sum_{i=1}^{N} i^{-\theta}$, where θ is the Zipf parameter. We set $\theta = 0.56$ (used in [33, 100]). We see that, among all the caching strategies, the optimized MCCS results in the lowest average rate for all values of M. The two lower bounds in **P5** and **P6** are numerically identical, indicating the optimality of popularity-first placement. Comparing the optimized MCCS with the lower bounds, we see that the gap between them is very small and only appears at a small range of cache size values $M \in [2.5, 3.5]$. The gap between the average rates of the optimized MCCS and the optimized CCS mainly exists for small cache sizes $M \in [0, 2]$ and shrinks as M increases.

As discussed in Section 4.4.2, although the candidate solutions of the optimal cache placement for the MCCS and the CCS are the same, the optimal placements may be different for the two schemes. To see this difference, for the same setting considered in Fig. 4.1, we show the optimal $\{\mathbf{a}_n\}$ for the two schemes for $M = 1, 2, 6$ in Tables 4.1, 4.2, and 4.3, respectively, representing small, moderate, and large cache sizes. For a small cache size

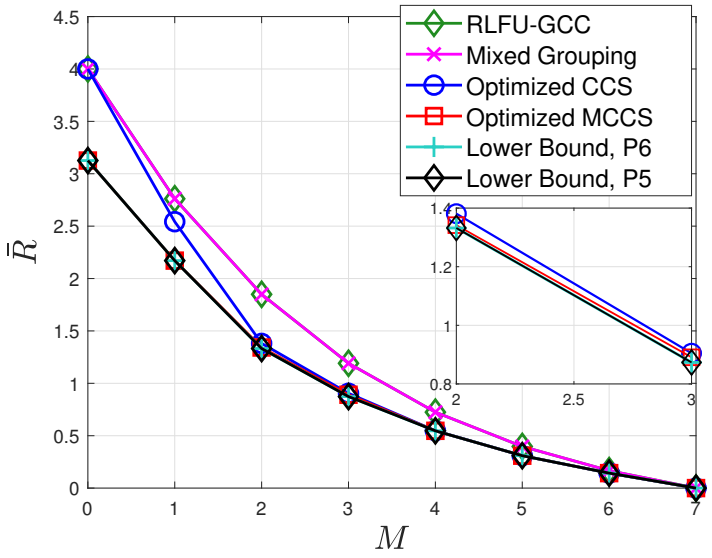

Figure 4.1: Average rate $\bar{R}$ vs. cache size M ($N = 7$, $K = 4$, equal file sizes, file popularity Zipf distribution with $\theta = 0.56$).

	l	Optimal cache placement vectors for the files						
		$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$
CCS	0	0	0	0	0	1.0000	1.0000	1.0000
	1	0.2500	0.2500	0.2500	0.2500	0	0	0
	2	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0
MCCS	0	0.4286	0.4286	0.4286	0.4286	0.4286	0.4286	0.4286
	1	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429
	2	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0

Table 4.1: The optimal cache placement vectors $\{\mathbf{a}_n\}$ for the MCCS and the CCS ($M = 1$, $N = 7$, $K = 4$, $\theta = 0.56$).

($M = 1$), the optimal placements for the MCCS and the CCS in Table 4.1 are different. For the MCCS, all files have the identical placement, where each file is partitioned into subfiles of two sizes, with one stored at the server ($a_{n,0}$) and the rest in each user's local cache. In contrast, for the CCS, files $\{W_5, W_6, W_7\}$ are solely stored at the server, and files $\{W_1, \cdots, W_4\}$ are stored in each user's local cache. This difference on the placement is the main cause of the performance gap between the MCCS and the CCS in Fig. 4.1. For

	l	Optimal cache placement vectors for the files						
		$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$
CCS	0	0	0	0	0	0	0	0
	1	0.2143	0.2143	0.2143	0.2143	0.2143	0.2143	0.2143
	2	0.0238	0.0238	0.0238	0.0238	0.0238	0.0238	0.0238
	3	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0
MCCS	0	0	0	0	0	0	0	0
	1	0.2143	0.2143	0.2143	0.2143	0.2143	0.2143	0.2143
	2	0.0238	0.0238	0.0238	0.0238	0.0238	0.0238	0.0238
	3	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0

Table 4.2: The optimal cache placement vectors $\{\mathbf{a}_n\}$ for the MCCS and the CCS ($M = 2$, $N = 7$, $K = 4$, $\theta = 0.56$).

	l	Optimal cache placement vectors for the files						
		$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$
CCS	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0
	3	0.4286	0.4286	0.4286	0.4286	0.4286	0.4286	0.4286
	4	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429
MCCS	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0
	3	0.4286	0.4286	0.4286	0.4286	0.4286	0.4286	0.4286
	4	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429

Table 4.3: The optimal cache placement vectors $\{\mathbf{a}_n\}$ for the MCCS and the CCS ($M = 6$, $N = 7$, $K = 4$, $\theta = 0.56$).

moderate to large cache sizes ($M = 2, 6$), Tables 4.2 and 4.3 show that the optimal cache placements are the same for the MCCS and the CCS. However, we see from Fig. 4.1 that for $M = 2$, there is a small observable gap between the average rates of the two schemes, with that of the MCCS being lower; and for $M = 6$, the average rates of the two are nearly identical. The explanation for this trend is that there exist more redundant messages for $M = 2$ with the placement in Table 4.2 than those for $M = 6$ with the placement in Table 4.3. To elaborate more on this, note that for given demand $\mathbf{d}$, the number of redundant groups in cache subgroup $\mathcal{A}^{l+1}$ is $\binom{K-\tilde{N}(\mathbf{d})}{l}$, which decreases with l. They determine the number of redundant messages. The indexes of the nonzero elements in $\mathbf{a}_n$ are $l = 1, 2$ for $M = 2$,

M	l	Optimal cache placement vectors for the files								
		$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$	$\mathbf{a}_5$	$\mathbf{a}_6$	$\mathbf{a}_7$	$\mathbf{a}_8$	$\mathbf{a}_9$
3	0	0	0	0	0	1.000	1.000	1.000	1.000	1.000
	1	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0
	3	0.250	0.250	0.250	0.250	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0.667	1.000	1.000	1.000
	1	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0
	3	0.250	0.250	0.250	0.250	0.250	0.083	0	0	0
	4	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0
	3	0.222	0.222	0.222	0.222	0.222	0.222	0.222	0.222	0.222
	4	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111

Table 4.4: file grouping structures of the optimal cache placement $\{\mathbf{a}_n\}$ for the MCCS ($N = 9$, $K = 4$, $\theta = 1.2$).

and $l = 3, 4$ for $M = 6$. As a result, for $M = 6$, there are only a very small number of redundant messages that are removed by the MCCS. Thus, the performance of the MCCS and the CCS are almost identical. Finally, the larger improvement of the MCCS over the CCS (and the MCCS almost attains the lower bounds) for $M \in [0, 2]$ indicates that, at a small cache size, coded caching is more sensitive to the cache placement to achieve the largest caching gain.

To evaluate the performance with other file popularity distribution, we consider the case studied in [36] with $N = 12$, $K = 5$, and a step-function for file popularity distribution: $p_1 = 7/12$, $p_n = 1/18$, $n = 2, \ldots, 7$, and $p_n = 1/60$, $n = 8, \ldots, 12$. Fig. 4.2 shows the average rate $\bar{R}$ vs. M by different caching schemes and the lower bounds. Similar to Fig. 4.1, for all values of M, the optimized MCCS achieves the lowest $\bar{R}$ among all the strategies, which is very close to the lower bounds. The two lower bounds in **P5** and **P6** are equal for different values of M, with the only exception for $M = 2$, where $\bar{R}$ for **P5** is 10^{-4} smaller than that of **P6**. The gap between the MCCS and the CCS again only exists for small values of M. To show the performance at different levels of popularity distribution, we show in Fig. 4.3 the average rate $\bar{R}$ vs. Zipf parameter θ. We set $N = 7$, $K = 4$. We

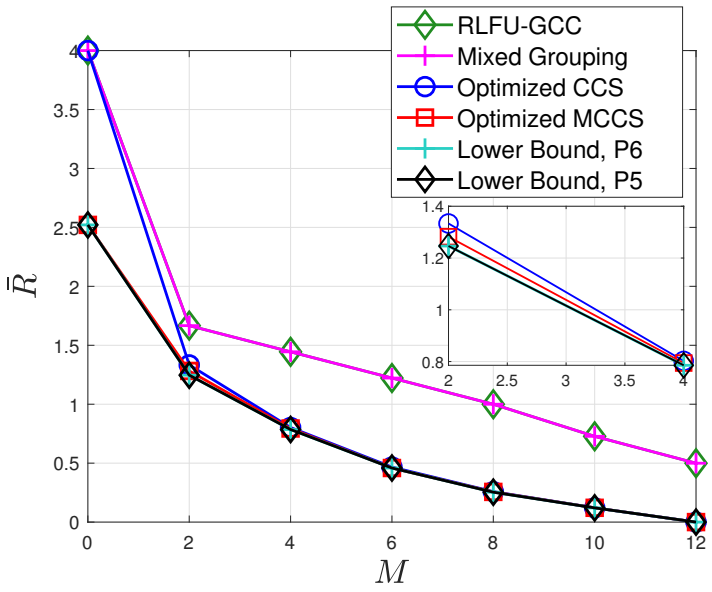

Figure 4.2: Average rate $\bar{R}$ vs. cache size M ($N = 12$, $K = 4$, equal file sizes, file popularity distribution: step function).

choose a small cache size $M = 1$ to show clearly the performance gap between the caching schemes and lower bounds. The optimized MCCS always performs the best among all the caching strategies for any θ. The lower bound in **P5** and the popularity-first-based lower bound in **P6** are numerically identical. Also, we observe that the gap between the average rates of the optimized MCCS and the lower bounds only exists at a moderate range of θ and is very small in general. In contrast, the gap between the MCCS and the CCS is obvious at all values of θ. This demonstrates the advantage of the MCCS over other caching schemes at a small value of M.

We now verify the structure of the optimal cache placement for the MCCS described in Section 4.4.2. We generate file popularity using Zipf distribution with $\theta = 1.2$. We obtain the optimal placement solution $\{\mathbf{a}_n\}$ using our proposed algorithm and verify that it matches the optimal solution obtained by solving **P0** numerically. As an example, for $N = 9$ and $K = 4$, Table 4.4 shows the optimal $\{\mathbf{a}_n\}$ that is obtained by solving **P0** numerically, for $M = 3, 4, 7$. For $M = 3$, we see that there are two file groups $\{W_1, \ldots, W_4\}$ and $\{W_5, \ldots, W_9\}$ under the optimal placement. This structure matches Case 1 in Section

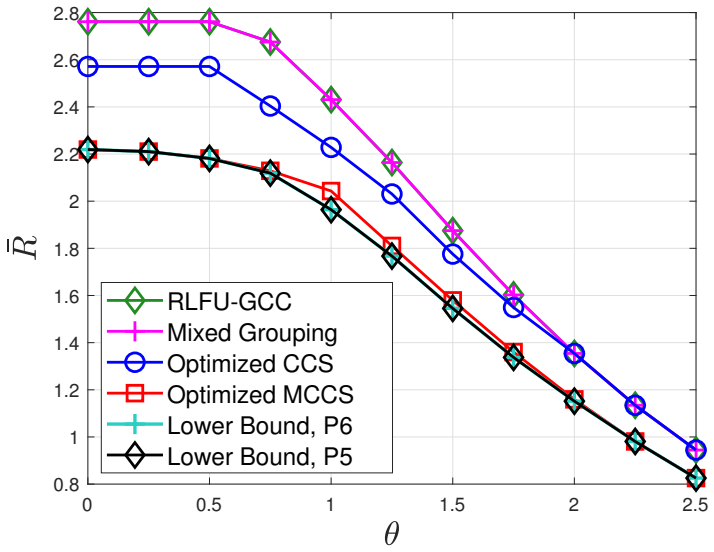

Figure 4.3: Average rate $\bar{R}$ vs. Zipf parameter θ ($N = 7$, $K = 4$, $M = 1$, equal file sizes).

4.4.2, where the cache is entirely allocated to the first file group with the most popular files, and the files in the second file group are only stored at the server ($a_{5,0} = \ldots = a_{9,0} = 1$). The optimal $\mathbf{a}_n$'s for the first group are identical with only one nonzero element. This means those files are partitioned into subfiles of the same size and are stored at users' local caches. With a small cache size, this placement result is intuitive: only a few popular files are cached, and the rest remain in the server; thus, the optimal cache placement results in two file groups. For $M = 4$, a different cache placement strategy is shown, where the files are divided into three file groups. The optimal $\{\mathbf{a}_n\}$ is as described in Section 4.4.2 for the three-file-group case: no cache is allocated to the third file group $\{W_7, W_8, W_9\}$, and a portion of the file is cached for W_6 in the second file group; for the first file group, files are partitioned into subfiles of a single size and are all stored at different users. For $M = 7$, the optimal placement has only a single file group, where all the files have the same placement as discussed in Section 4.4.2. From Table 4.4, we see that the file popularity differences are more critical for the placement when the case size M is limited (relative to the total files in the database). As M becomes large, all the files tend to have the same placement into the user caches.

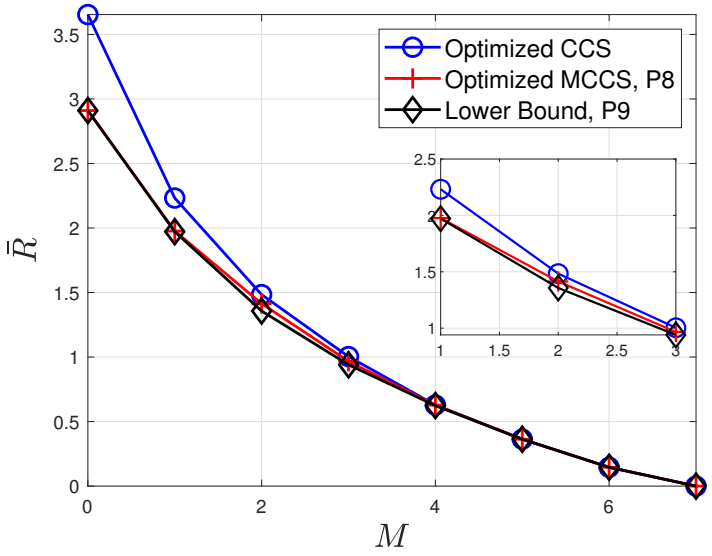

Figure 4.4: Average rate $\bar{R}$ vs. cache size M (kbits) ($N = 7$, $K = 4$, file popularity distribution: $\mathbf{p} = [0.0888, 0.0968, 0.1072, 0.1215, 0.2640, 0.1427, 0.1791]$, file sizes: $[F_1, \ldots, F_N] = [9/6, 8/6, 7/6, 6/6, 5/6, 4/6, 3/6]$ kbits.

Finally, we consider the scenario of nonuniform file popularity and sizes. We generate the file popularity using Zipf distribution with $\theta = 0.56$, which gives $\mathbf{p} = [0.0888, 0.0968, 0.1072, 0.1215, 0.2640, 0.1427, 0.1791]$. The file sizes are set as $[F_1, \ldots, F_N] = [9/6, 8/6, 7/6, 6/6, 5/6, 4/6, 3/6]$ kbits. The file sizes and popularity combinations are chosen similar to those used in [33], which simulate a practical scenario where file popularity and size are relatively uncorrelated. In Fig. 4.4, we compare the optimized MCCS in **P8**, the lower bound in **P9**, and the optimized CCS [33]. The gap between the optimized MCCS and the lower bound only exists for $M \in [1, 3]$ and is very small. Moreover, the optimized MCCS outperforms the optimized CCS. The gap between the two again is obvious at small values of M, and it reduces to zero as M becomes large.

4.7 Summary

In this chapter, we first formulated an optimization problem to obtain the optimal cache placement for the MCCS under nonuniform popularity. We then provided a general lower

bound and a popularity-first-based lower bound for caching with uncoded placement, which are shown to be identical for $K = 2$ users. We compared the MCCS with the lower bounds to study the memory-rate tradeoff. For $K = 2$ users, the lower bounds are achieved by the MCCS, providing exact memory-rate tradeoff. For $K > 2$ users with distinct requests, the optimized MCCS attains the popularity-first-based lower bound. For $K > 2$ users with redundant requests, our analysis showed that a gap might exist between the optimized MCCS and the lower bounds due to zero-padding. However, numerical results showed that such loss only exists in some limited cases and is very small in general. Finally, we extended our study of memory-rate tradeoff to the case where files are nonuniform in both popularity and sizes. We showed that the optimized MCCS attains the lower bound for $K = 2$ users and characterizes the exact memory-rate tradeoff.

Chapter 5

Memory-Rate Tradeoff for Decentralized Caching with Nonuniform Demands

In this chapter, we study the memory-rate tradeoff for decentralized caching with nonuniform demands. Focusing on the D-MCCS, we formulate a cache placement optimization problem that is non-convex and develop two approximate algorithms to solve it. We further propose a lower bound through a non-convex optimization problem that is solved through an approximate algorithm. Finally, we compare the optimized D-MCCS with the lower bound to study the memory-rate tradeoff for decentralized caching.

5.1 Decentralized Modified Coded Caching Scheme

In this section, we describe the cache placement and content delivery procedures of the D-MCCS under nonuniform file popularity.

5.1.1 Decentralized Cache Placement

A salient feature of decentralized caching is that the active user set $\mathcal{A}$ (both the size and the user identities) is unknown during the cache placement phase. We consider the following decentralized placement procedure: each user $k \in \mathcal{K}$ independently and randomly selects and caches $q_n F$ bits of file $W_n, n \in \mathcal{N}$, where q_n is the portion of the file the user wants to

cache, *i.e.,*

$$0 \leq q_n \leq 1, \quad n \in \mathcal{N}. \tag{5.1}$$

We define $\mathbf{q} \triangleq [q_1, \ldots, q_N]^T$ as the cache placement vector of all the files in $\mathcal{N}$. For uniform file popularity, the symmetrical decentralized placement is optimal for the D-MCCS [44], *i.e.,* $q_1 = \cdots = q_N = \frac{M}{N}$. For nonuniform file popularity, the cache placement may be different for different files, which complicates the cache placement design of the D-MCCS. In this work, we aim to optimize the cache placement vector $\mathbf{q}$ for the D-MCCS to minimize the average delivery rate. Since a subset of file $n \in \mathcal{N}$ of $q_n F$ bits are cached by each user of cache size M, we have the cache size constraint given by

$$\sum_{n=1}^{N} q_n F \leq MF. \tag{5.2}$$

Note that the server knows the cached contents by each user.

5.1.2 Content Delivery

During the delivery process, the server receives the information of the active user set $\mathcal{A}$ and their demand vector $\mathbf{d}_{\mathcal{A}}$. Based on these, the server knows the cached contents among the users in $\mathcal{A}$. We define subfile $W_{n,\mathcal{S}}$ as the chunk of file W_n that is cached by the active user subset $\mathcal{S} \subseteq \mathcal{A}$ but not by the rest users in $\mathcal{A}$, *i.e.,* $\mathcal{A} \backslash \mathcal{S}$. We use $W_{n,\emptyset}$ to represent the portion of file n that is not cached by any user in $\mathcal{A}$. Under the decentralized placement, for file size F being sufficiently large, by the law of large numbers, q_n is approximately the probability of one bit in file n being selected by a user. Following this, the size of subfile $W_{n,\mathcal{S}}$ is approximately given by [18]

$$|W_{n,\mathcal{S}}| \approx q_n^s (1 - q_n)^{A-s} F, \quad \mathcal{S} \subseteq \mathcal{A}, |\mathcal{S}| = s \tag{5.3}$$

where $A \triangleq |\mathcal{A}|$. According to (5.3), the size of subfile $W_{n,\mathcal{S}}$ depends on $|\mathcal{S}|$, *i.e.,* the number of the users who cache it.

For any file demand vector $\mathbf{d}_{\mathcal{A}}$, the D-MCCS multicasts coded messages to different user subsets in $\mathcal{A}$. Each coded message is intended for an unique active user subset $\mathcal{S} \subseteq \mathcal{A}$

and is formed by the bitwise XOR operation of total $|\mathcal{S}|$ subfiles, one from each requested file, given by

$$C_{\mathcal{S}} \triangleq \bigoplus_{k\in\mathcal{S}} W_{d_k,\mathcal{S}\backslash\{k\}}, \quad \mathcal{S}\subseteq\mathcal{A}, \mathcal{S}\neq\emptyset. \tag{5.4}$$

From (5.4), each of the subfiles in $\mathcal{C}_{\mathcal{S}}$ belongs to file d_k requested by user $k \in \mathcal{S}$ and is cached by users in $\mathcal{S}\backslash\{k\}$ exclusively. Note that the coded messages can only be formed for the nonempty active user subset $\mathcal{S} \neq \emptyset$.

Since the portion q_n of file n cached by the users may be different for files with different popularities, the subfiles forming the coded message $\mathcal{C}_{\mathcal{S}}$ in (5.4) may not have equal size. In this case, zero-padding is adopted for the XOR operation such that subfiles are zero-padded to the size of the longest subfile. Thus, the size of $C_{\mathcal{S}}$ is determined by the largest subfile in $\mathcal{C}_{\mathcal{S}}$, *i.e.,*

$$|C_{\mathcal{S}}| = \max_{k\in\mathcal{S}} |W_{d_k,\mathcal{S}\backslash\{k\}}| = \max_{k\in\mathcal{S}} q_{d_k}^s (1-q_{d_k})^{A-s} F, \quad \mathcal{S}\subseteq\mathcal{A}, |\mathcal{S}| = s+1, s = 0,\ldots,A-1. \tag{5.5}$$

Remark 12. For nonuniform file popularity, cache placement may be different for different files, resulting in generating subfiles of nonequal sizes. The existence of nonequal subfiles complicates the cache placement design. Zero-padding is a common technique to handle the nonequal files in formulating the coded messages for both centralized [33, 48, 53] and decentralized coded caching [61]. However, its impact on decentralized coded caching has never been studied and is unknown. In Section 5.3.2, by developing a matching converse bound, we show that there is no loss of optimality by using zero-padding in decentralized coded caching if the size of the active user set is no more than two, $A \leq 2$.

In the original D-CCS [18], for any file demand vector $\mathbf{d}_{\mathcal{A}}$, the server transmits the coded messages corresponding to all the active user subsets $\{\mathcal{C}_{\mathcal{S}} : \forall \mathcal{S} \subseteq \mathcal{A}\}$ to the active users. For the D-MCCS, the server only transmits coded messages corresponding to certain selected active user subsets [44]. We first provide the following two definitions:

Algorithm 5 Decentralized modified coded caching scheme

Decentralized cache placement procedure:

1: **for** $n \in \mathcal{N}$ **do**
2: Each user randomly caches $q_n F$ bit of file W_n.
3: **end for**

Coded delivery procedure:

1: **for** $\mathcal{S} \subseteq \mathcal{A}$ and $\mathcal{S} \cap \mathcal{U}_{\mathcal{A}} \neq \emptyset$ **do**
2: Server generates $\mathcal{C}_{\mathcal{S}}$ and multicasts it to $\mathcal{S}$.
3: **end for**

Definition 7. *Leader group*: For any demand vector $\mathbf{d}_{\mathcal{A}}$ with $\widetilde{N}(\mathbf{d}_{\mathcal{A}})$ distinct requests, the leader group $\mathcal{U}_{\mathcal{A}}$ is a subset of the active user set $\mathcal{A}$, *i.e.,* $\mathcal{U}_{\mathcal{A}} \subseteq \mathcal{A}$, that satisfies $|\mathcal{U}_{\mathcal{A}}| = \widetilde{N}(\mathbf{d}_{\mathcal{A}})$ and the users in $\mathcal{U}_{\mathcal{A}}$ have exactly $\widetilde{N}(\mathbf{d}_{\mathcal{A}})$ distinct requests.

Definition 8. *Redundant group*: Given $\mathcal{U}_{\mathcal{A}}$, any active user subset $\mathcal{S} \subseteq \mathcal{A}$ is called a redundant group if $\mathcal{S} \cap \mathcal{U}_{\mathcal{A}} = \emptyset$; otherwise, $\mathcal{S}$ is a non-redundant group.

The delivery procedure of the D-MCCS improves upon that of the D-CCS by only multicasting coded messages corresponding to the non-redundant groups, *i.e.,* $\{C_{\mathcal{S}} : \forall \mathcal{S} \subseteq \mathcal{A}$ and $\mathcal{S} \cap \mathcal{U}_{\mathcal{A}} \neq \emptyset\}$, to both non-redundant and redundant groups.[1] As a result, the D-MCCS achieves a lower delivery rate than the D-CCS. Note that the rate reduction only occurs when there exist redundant groups, *i.e.,* there are multiple requests of the same file among the active users.

We summarize both the cache placement and coded delivery procedures of the D-MCCS in Algorithm 5. With the cached contents at each user via the decentralized cache placement described in Section 5.1.1 and the coded messages $\{C_{\mathcal{S}} : \forall \mathcal{S} \subseteq \mathcal{A}$ and $\mathcal{S} \cap \mathcal{U}_{\mathcal{A}} \neq \emptyset\}$ multicasted by the server, each user in $\mathcal{A}$ can retrieve all the subfiles of its requested file [44].

[1] Note that this coded delivery strategy follows that of the centralized MCCS [44], which has been shown to be a valid scheme, *i.e.,* a user can reconstruct any requested file.

5.2 Decentralized Cache Placement Optimization

In this section, we first formulate the cache placement design for the D-MCCS under nonuniform file popularity as an optimization problem to minimize the average delivery rate. We then develop two algorithms to solve the problem.

5.2.1 Problem Formulation

Based on the delivery procedure in the D-MCCS described in Section 5.1.2, for a given demand vector $\mathbf{d}_{\mathcal{A}}$, the delivery rate is the total number of bits in the coded messages corresponding to all the non-redundant groups $\{C_{\mathcal{S}} : \forall \mathcal{S} \subseteq \mathcal{A} \text{ and } \mathcal{S} \cap \mathcal{U}_{\mathcal{A}} \neq \emptyset\}$, expressed as

$$R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; \mathbf{q}) = \sum_{\mathcal{S} \subseteq \mathcal{A}, \mathcal{S} \cap \mathcal{U}_{\mathcal{A}} = \emptyset} |C_{\mathcal{S}}|. \tag{5.6}$$

Define $\mathcal{Q}^s \triangleq \{\mathcal{S} \subseteq \mathcal{A} : \mathcal{S} \cap \mathcal{U}_{\mathcal{A}} = \emptyset, |\mathcal{S}| = s\}$ as the set of the non-redundant groups with $|\mathcal{S}| = s$ users for $s = 1, \ldots, K$. Based on (5.5), we can rewrite (5.6) as

$$R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; \mathbf{q}) = \sum_{s=0}^{A-1} \sum_{\mathcal{S} \in \mathcal{Q}^{s+1}} \max_{k \in \mathcal{S}} q_{d_k}^s (1 - q_{d_k})^{A-s} F. \tag{5.7}$$

By taking the expectation of $R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; \mathbf{q})$ over all the possible $\mathbf{d}_{\mathcal{A}} \in \mathcal{N}^A$ and $\mathcal{A} \subseteq \mathcal{K}$, the average rate of the D-MCCS as a function of $\mathbf{q}$ is given by

$$\bar{R}_{\text{MCCS}}(\mathbf{q}) = E_{\mathcal{A}}\left[E_{\mathbf{d}_{\mathcal{A}}}[R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; \mathbf{q})]\right] = E_{\mathcal{A}}\left[\sum_{\mathbf{d}_{\mathcal{A}} \in \mathcal{N}^A} \left(\prod_{k \in \mathcal{A}} p_{d_k}\right) R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; \mathbf{q})\right]. \tag{5.8}$$

Thus, we formulate the cache placement optimization problem for the D-MCCS under nonuniform file popularity as

$$\textbf{P10}: \quad \min_{\mathbf{q}} \bar{R}_{\text{MCCS}}(\mathbf{q}) \quad \text{s.t.} \quad (5.1), (5.2).$$

P10 is a non-convex optimization problem w.r.t. $\mathbf{q}$, which is difficult to solve. In the following subsection, we propose two different algorithms to solve **P10**.

5.2.2 Optimal Decentralized Cache Placement Solutions

We first develop an algorithm that converges to a stationary point of **P10** through solving a series of Geometric Programming (GP) problems. To reduce the computational complexity, we further propose an approximate solution with very low complexity to compute.

Successive GP Approximation Algorithm

We reformulate **P10** into an equivalent Complementary GP (CGP) problem, which is an intractable NP-hard problem [101]. It is shown that a stationary point of a CGP can be obtained through the generic successive approximation method [102].

To reformulate **P10** into an equivalent CGP problem, we first introduce auxiliary variables $x_n, n \in \mathcal{N}$, and add the following inequality constraint for $(1 - q_n)$ in (5.7).

$$1 - q_n \leq x_n, \quad n \in \mathcal{N}. \tag{5.9}$$

We also introduce auxiliary variables $w_{\mathbf{d}_\mathcal{A},\mathcal{S}}$ for $\mathcal{A} \subseteq \mathcal{K}$, $\mathbf{d}_\mathcal{A} \in \mathcal{N}^A$ and $\mathcal{S} \in \mathcal{Q}^{s+1}, s = 0, \ldots, K-1$. By (5.9), we replace $\max_{k\in\mathcal{S}} q_{d_k}^s (1 - q_{d_k})^{A-s} F$ in (5.7) with $w_{\mathbf{d}_\mathcal{A},\mathcal{S}}$ and add the following constraints

$$q_{d_k}^s x_{d_k}^{A-s} F \leq w_{\mathbf{d}_\mathcal{A},\mathcal{S}}, \quad k \in \mathcal{S} \tag{5.10}$$

for given $\mathcal{S} \in \mathcal{Q}^{s+1}, \mathbf{d}_\mathcal{A} \subseteq \mathcal{N}^A, \mathcal{A} \subseteq \mathcal{K}$. As a result, we reformulate **P10** into the following equivalent problem

$$\textbf{P11}: \min_{\mathbf{q},\mathbf{x},\mathbf{w}\succcurlyeq 0} E_\mathcal{A}\left[\sum_{\mathbf{d}_\mathcal{A}\in\mathcal{N}^A}\left(\prod_{k\in\mathcal{A}} p_{d_k}\right)\sum_{s=0}^{A-1}\sum_{\mathcal{S}\in\mathcal{Q}^{s+1}} w_{\mathbf{d}_\mathcal{A},\mathcal{S}}\right]$$

$$\text{s.t. } q_n \leq 1, \qquad n \in \mathcal{N}, \tag{5.11}$$

$$\sum_{n=1}^{N} q_n M^{-1} \leq 1, \tag{5.12}$$

$$\frac{1}{q_n + x_n} \leq 1, \qquad n \in \mathcal{N}, \tag{5.13}$$

$$w_{\mathbf{d}_\mathcal{A},\mathcal{S}}^{-1} \cdot q_{d_k}^s x_{d_k}^{A-s} F \leq 1,\ k \in \mathcal{S}, \mathcal{S} \in \mathcal{Q}^{s+1}, \quad s = 0, \ldots, K-1, \mathbf{d}_\mathcal{A} \in \mathcal{N}^A, \mathcal{A} \subseteq \mathcal{K} \tag{5.14}$$

where $\mathbf{x} \triangleq (x_n)_{n\in\mathcal{N}}$ and $\mathbf{w} \triangleq (w_{\mathbf{d}_\mathcal{A},\mathcal{S}})_{\mathcal{S}\in\mathcal{Q}^{s+1},\mathbf{d}_\mathcal{A}\subseteq\mathcal{N}^A,\mathcal{A}\subseteq\mathcal{K}}$. Note that constraints (5.12), (5.13) and (5.14) are direct reformulations of (5.2), (5.9) and (5.10), respectively. **P11** minimizes a posynomial subject to upper bound three inequality constraints (5.11), (5.12) and (5.14) that are posynomials and the inequality constraint (5.13) that is on the ratio between two posynomials. Thus, **P11** is a CGP problem. For a CGP problem, an approach was developed using a sequence of approximate GPs to obtain a stationary point of the problem [102]. We adopt this approach to solve **P11** by solving $(\mathbf{q}, \mathbf{x}, \mathbf{w})$ iteratively via a sequence of approximate GP problems. Define the objective function of **P11** by $\bar{R}^{\text{CGP}}_{\text{MCCS}}$. In the ith iteration, given $(\mathbf{q}^{(i)}, \mathbf{x}^{(i)})$ obtained from previous iteration, we have the following approximate GP problem of **P11**.

$$\begin{aligned}\mathbf{P12}(\mathbf{q}^{(i)},\mathbf{x}^{(i)}):(\mathbf{q}^{(i+1)}, \mathbf{x}^{(i+1)}, \mathbf{w}^{(i+1)}) =& \operatorname{argmin}_{\mathbf{q},\mathbf{x},\mathbf{w}\succcurlyeq 0} \bar{R}^{\text{CGP}}_{\text{MCCS}}(\mathbf{q}, \mathbf{x}, \mathbf{w})\\ \text{s.t. } & (5.11), (5.12), (5.14),\\ & \frac{1}{\left(q_n^{(i)}+x_n^{(i)}\right)\left(\frac{q_n}{q_n^{(i)}}\right)^{\alpha_n^{(i)}}\left(\frac{x_n}{x_n^{(i)}}\right)^{\beta_n^{(i)}}} \leq 1, \quad n \in \mathcal{N} \end{aligned} \tag{5.15}$$

where $\alpha_n^{(i)} \triangleq \frac{q_n^{(i)}}{q_n^{(i)}+x_n^{(i)}}$ and $\beta_n^{(i)} \triangleq \frac{x_n^{(i)}}{q_n^{(i)}+x_n^{(i)}}$. Note that constraint (5.15) is formed using (5.13) and the arithmetic-geometric mean inequality [102]

$$q_n + x_n \geq \left(\frac{q_n}{\alpha_n^{(i)}}\right)^{\alpha_n^{(i)}} \left(\frac{x_n}{\beta_n^{(i)}}\right)^{\beta_n^{(i)}} \geq 1. \tag{5.16}$$

Note that $\mathbf{P12}(\mathbf{q}^{(i)}, \mathbf{x}^{(i)})$ is a standard GP problem, which can be solved using a standard convex optimization solver. The above approach of iteratively solving $\mathbf{P12}(\mathbf{q}^{(i)}, \mathbf{x}^{(i)})$ is guaranteed to converge to a stationary point of **P11** [102]. This successive GP approximation algorithm is summarized in Algorithm 6. By the equivalence of **P10** and **P11**, we can compute a stationary point of **P10** using Algorithm 6. Note that as the size of $\mathbf{P12}(\mathbf{q}^{(i)}, \mathbf{x}^{(i)})$ grows exponentially with K, the computational complexity of Algorithm 6 can be very high. To address this, next, we develop an alternative algorithm to solve the problem with very low complexity.

Algorithm 6 The successive GP approximation algorithm for **P11**

Input: K, M, N, $\mathbf{p}$, $\mathbf{p}_{\mathrm{a}}$.

1: **Initialization:** Choose initial feasible point $\left(\mathbf{q}^{(0)}, \mathbf{x}^{(0)}, \mathbf{w}^{(0)}\right)$. Set $i = 0$.
2: **repeat**
3: Solve **P12**$(\mathbf{q}^{(i)}, \mathbf{x}^{(i)})$ to obtain $\left(\mathbf{q}^{(i+1)}, \mathbf{x}^{(i+1)}, \mathbf{w}^{(i+1)}\right)$.
4: Set $i = i + 1$.
5: **until** $\bar{R}^{\text{CGP}}_{\text{MCCS}}(\mathbf{q}^{(i)}, \mathbf{x}^{(i)}, \mathbf{w}^{(i)})$ converges.
6: $\bar{R}^{*}_{\text{MCCS}} = \bar{R}^{\text{CGP}}_{\text{MCCS}}(\mathbf{q}^{(i)}, \mathbf{x}^{(i)}, \mathbf{w}^{(i)})$; $\mathbf{q}^{*} = \mathbf{q}^{(i)}$.

Output: $\bar{R}^{*}_{\text{MCCS}}$, $\mathbf{q}^{*}$.

Low-Complexity File-Group-Based Approach

We now propose an algorithm that computes an approximate solution for **P10**. The algorithm is based on a particular cache placement structure. In particular, we consider the two-file-group based placement scheme below.

Two-file-group-based placement: In the cache placement phase, the N files are partitioned into two groups – based on their popularity distribution $\mathbf{p}$. Define $\mathcal{N}_1 \triangleq \{1, \ldots, N_1\}$, for $N_1 \in \mathcal{N}$, and $\mathcal{N}_2 \triangleq \mathcal{N} \backslash \mathcal{N}_1$ as the file index sets of first and second file groups, respectively. The first group $\mathcal{N}_1$ contains more popular files. For each user $k \in \mathcal{K}$, its entire cache is allocated to the first group $\mathcal{N}_1$ and and is equally split among these files in $\mathcal{N}_1$. Thus, the cached portion of each file in the two-file-group based placement is given by

$$q_n = \begin{cases} M/N_1, & n \in \mathcal{N}_1, \\ 0, & n \in \mathcal{N}_2. \end{cases} \tag{5.17}$$

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ denote the sets of active users who request the files in $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively. Note that $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$ and $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$. Denote $A_i = |\mathcal{A}_i|$, for $i = 1, 2$. Accordingly, the number of distinct file requests from $\mathcal{A}_i$ is $\widetilde{N}(\mathbf{d}_{\mathcal{A}_i})$. Note that $\mathcal{A}_i$ and $\widetilde{N}(\mathbf{d}_{\mathcal{A}_i})$, $i = 1, 2$ are all functions of N_1. Under this two-file-group-based placement structure, in the following lemma, **P10** can be reformulated as an optimization problem w.r.t. N_1 to minimize the average rate.

Proposition 8. Consider the decentralized caching problem of N files with popularity distribution $\mathbf{p}$ and K users, where each user k has cache size M and is with probability $p_{\mathrm{a},k}$ of being active. The minimum average rate under the two-file-group-based decentralized

Algorithm 7 Two-file-group based approximate solution

Input: K, M, N, $\mathbf{p}$, $\mathbf{p}_{\mathrm{a}}$.

1: **for** $N_1 = 1$ to N **do**
2: Compute $\bar{R}^{\text{FG-2}}_{\text{MCCS}}(N_1)$ by (5.18).
3: **end for**
4: Compute $N_1^* = \text{argmin}_{N_1} \bar{R}^{\text{FG-2}}_{\text{MCCS}}(N_1)$.
5: Compute $\bar{R}^{\text{FG-2}}_{\text{MCCS}}(N_1^*)$.

Output: $\bar{R}^{\text{FG-2}}_{\text{MCCS}}(N_1^*)$.

cache placement (5.17) for the D-MCCS is $\min_{N_1 \in \mathcal{N}} \bar{R}^{\text{FG-2}}_{\text{MCCS}}(N_1)$, where $\bar{R}^{\text{FG-2}}_{\text{MCCS}}(N_1)$ is given by

$$\bar{R}^{\text{FG-2}}_{\text{MCCS}}(N_1) \triangleq E_{\mathcal{A}} \left[\sum_{\mathbf{d}_{\mathcal{A}} \in \mathcal{N}^A} \left(\prod_{k \in \mathcal{A}} p_{d_k} \right) R^{\text{FG-2}}_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; N_1) \right] \tag{5.18}$$

where $R^{\text{FG-2}}_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; N_1)$ is the delivery rate under two-file-group-based placement given $\mathbf{d}_{\mathcal{A}}$ and N_1, expressed as

$$R^{\text{FG-2}}_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; N_1) = \sum_{s=0}^{A-1} \left(\sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})} \binom{A-i}{s} - \sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}_2})} \binom{A_2-i}{s} \right) \cdot \left(\frac{M}{N_1} \right)^s \left(1 - \frac{M}{N_1} \right)^{A-s} F + \widetilde{N}(\mathbf{d}_{\mathcal{A}_2}) F. \tag{5.19}$$

Proof. See Appendix C.2. □

By Proposition 8, under the two-file-group-based placement, the optimal $N_1 \in \mathcal{N}$ that results in the minimum average rate can be obtained through search in $\mathcal{N}$. Our algorithm is summarized in Algorithm 7. For each $N_1 \in \mathcal{N}$, the average rate is computed directly using the objective function in (5.18) using the closed-form expression in (5.19). Thus, the computational complexity of Algorithm 7 is much lower as compared with Algorithm 6. Interestingly, our numerical study in Section 5.4 shows that the average rate achieved by Algorithm 7 is very close to that of Algorithm 6 and, in some cases, could be even lower than that of Algorithm 6.

Remark 13. Assuming the active user set $\mathcal{A}$ is known, a similar two-file-group-based placement has been considered in [35] [36] for the D-CCS, where the size of the first

group N_1 was proposed through heuristics. Our work is different from them in the following aspects: First, we develop our placement solution for the case of unknown active user set $\mathcal{A}$. Second, the D-MCCS is different from the D-CCS considered in [35,36] in terms of the delivery procedure. Specifically, the delivery procedure of the D-MCCS removes the redundancy that exists in the coded messages of the D-CCS. Furthermore, [35, 36] apply a user-grouping-based coded message generation method, where the coded message in (4) is formed by files within the same file group, and there is no coding across file groups. In contrast, we explore the coded caching gain among all the requested files in Algorithm 5. Note that as it has been shown for the D-CCS, the average rate of the coded delivery that explores the coded caching gain among all files is a lower bound to that of the user-grouping-based delivery [47].

5.3 Memory Rate Tradeoff for Decentralized Caching

In this section, we characterize the memory-rate tradeoff for decentralized caching under nonuniform file popularity by proposing a lower bound and comparing it with the average rate of the optimized D-MCCS in **P10**.

5.3.1 Lower Bound for Decentralized Caching

The general idea for developing the lower bound for decentralized caching is to divide all the possible file demand vectors into different types and then derive a lower bound for each type separately [44]. Given any active user set $\mathcal{A} \subseteq \mathcal{K}$, we categorize all the possible demand vectors $\mathbf{d}_{\mathcal{A}} \in \mathcal{N}^A$ based on the distinct file requests in $\mathbf{d}_{\mathcal{A}}$. We use Unique$(\mathbf{d}_{\mathcal{A}})$ to denote extracting the distinct file requests in $\mathbf{d}_{\mathcal{A}}$, and the resulting index set of distinct files is denoted as $\mathcal{D}_{\mathcal{A}} \triangleq$ Unique$(\mathbf{d}_{\mathcal{A}})$. Recall that the leader group $\mathcal{U}_{\mathcal{A}}$ contains $\widetilde{N}(\mathbf{d}_{\mathcal{A}})$ users requesting all the distinct files in $\mathbf{d}_{\mathcal{A}}$ and thus we have $|\mathcal{D}_{\mathcal{A}}| = |\mathcal{U}_{\mathcal{A}}| = \widetilde{N}(\mathbf{d}_{\mathcal{A}})$.

We present a lower bound on the average rate for decentralized caching under nonuniform file popularity in the following theorem.

Theorem 7. Consider the decentralized caching problem of N files with popularity distribution $\mathbf{p}$ and K users, where each user k has cache size M and is with probability $p_{\mathrm{a},k}$ of being active. The following optimization problem provides a lower bound on the average rate:

$$\mathbf{P13}: \min_{\mathbf{q}} \bar{R}_{\mathrm{lb}}(\mathbf{q}) \triangleq E_{\mathcal{A}}\left[\sum_{\mathcal{D}_{\mathcal{A}}\subseteq\mathcal{N}} \sum_{\mathbf{d}_{\mathcal{A}}\in\mathcal{T}(\mathcal{D}_{\mathcal{A}})} \left(\prod_{k\in\mathcal{A}} p_{d_k}\right) R_{\mathrm{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})\right] \quad \text{s.t. (5.1), (5.2)} \tag{5.20}$$

where $\mathcal{T}(\mathcal{D}_{\mathcal{A}}) \triangleq \{\mathbf{d}_{\mathcal{A}} : \mathrm{Unique}(\mathbf{d}_{\mathcal{A}}) = \mathcal{D}_{\mathcal{A}},\ \mathbf{d}_{\mathcal{A}} \in \mathcal{N}^A\}$, and $R_{\mathrm{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})$ is the lower bound on the rate for given $\mathbf{q}$ and $\mathcal{D}_{\mathcal{A}}$, given by

$$R_{\mathrm{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q}) \triangleq \max_{\pi:\mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|}\to\mathcal{D}_{\mathcal{A}}} \sum_{s=0}^{A-1} \sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})} \binom{A-i}{s} q_{\pi(i)}^{s} (1-q_{\pi(i)})^{A-s} F \tag{5.21}$$

where $\mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|} \triangleq \{1,\ldots,|\mathcal{D}_{\mathcal{A}}|\}$ and $\pi : \mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|} \to \mathcal{D}_{\mathcal{A}}$ is any bijective map from $\mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|}$ to $\mathcal{D}_{\mathcal{A}}$.

Proof. See Appendix C.1 □

P13 is a non-convex optimization problem, and the only difference between **P13** and **P10** are their objective functions $\bar{R}_{\mathrm{MCCS}}(\mathbf{q})$ and $\bar{R}_{\mathrm{lb}}(\mathbf{q})$.

Following the similar approach in Section 5.2.2, we first formulate **P13** into an equivalent CGP problem. With the same auxiliary variables $x_n, n \in \mathcal{N}$, we add the same inequality constraints (5.9). Also, we introduce auxiliary variable $r_{\mathcal{D}_{\mathcal{A}},\pi}$ for $\pi : \mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|} \to \mathcal{D}_{\mathcal{A}}$, $\mathcal{D}_{\mathcal{A}} \subseteq \mathcal{N}$ and $\mathcal{A} \subseteq \mathcal{K}$. By (5.9), we replace the expression in (5.21) with $r_{\mathcal{D}_{\mathcal{A}},\pi}$ and add the following constraint

$$\sum_{s=0}^{A-1} \sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})} \binom{A-i}{s} \left(q_{\pi(i)}\right)^{s} \left(x_{\pi(i)}\right)^{A-s} F \leq r_{\mathcal{D}_{\mathcal{A}},\pi} \tag{5.22}$$

for given $\pi : \mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|} \to \mathcal{D}_{\mathcal{A}}$, $\mathcal{D}_{\mathcal{A}} \subseteq \mathcal{N}$ and $\mathcal{A} \subseteq \mathcal{K}$. Similar to the reformulation of **P10** to **P11**, with (5.22), we can reformulate **P13** into the following CGP.

$$\mathbf{P14}: \min_{\mathbf{q},\mathbf{x},\mathbf{r}\succcurlyeq 0} E_{\mathcal{A}}\left[\sum_{\mathcal{D}_{\mathcal{A}}\subseteq\mathcal{N}} \sum_{\mathbf{d}_{\mathcal{A}}\in\mathcal{T}(\mathcal{D}_{\mathcal{A}})} \left(\prod_{k\in\mathcal{A}} p_{d_k}\right) r_{\mathcal{D}_{\mathcal{A}},\pi}\right]$$

s.t. (5.11), (5.12), (5.13) and

$$(r_{\mathcal{D}_\mathcal{A},\pi})^{-1}\sum_{s=0}^{A-1}\sum_{i=1}^{\widetilde{N}(\mathbf{d}_\mathcal{A})}\binom{A-i}{s}\left(q_{\pi(i)}\right)^s\left(x_{\pi(i)}\right)^{A-s}F\leq 1,\quad \mathcal{A}\subseteq\mathcal{K},\mathcal{D}_\mathcal{A}\subseteq\mathcal{N},\forall\pi:\mathcal{I}_{|\mathcal{D}_\mathcal{A}|}\rightarrow\mathcal{D}_\mathcal{A}. \tag{5.23}$$

Let $\bar{R}_{\text{lb}}^{\text{CGP}}(\mathbf{q},\mathbf{x},\mathbf{r})$ denote the objective function of **P14**. Following similar approach in Section 5.2.2, in the ith iteration, for given $(\mathbf{q}^{(i)},\mathbf{x}^{(i)})$, we formulate the following approximate optimization problem of **P14**.

$$\begin{aligned}\textbf{P15}\left(\mathbf{q}^{(i)},\mathbf{x}^{(i)}\right):\left(\mathbf{q}^{(i+1)},\mathbf{x}^{(i+1)},\mathbf{r}^{(i+1)}\right)=&\operatorname{argmin}_{\mathbf{q},\mathbf{x},\mathbf{r}\succcurlyeq 0}\bar{R}_{\text{lb}}^{\text{CGP}}(\mathbf{q},\mathbf{x},\mathbf{r})\\ &\text{s.t. (5.11), (5.12), (5.15), and (5.23).}\end{aligned}$$

Thus, we again use the successive GP approximation algorithm by iteratively solving **P15**$\left(\mathbf{q}^{(i)},\mathbf{x}^{(i)}\right)$ to obtain a stationary point of **P14**, which is summarized in Algorithm 8. Finally, by the equivalence of **P13** and **P14**, we obtain the stationary point of **P13** by Algorithm 8.

Algorithm 8 The Successive GP Approximation Algorithm for **P14**

Input: K, M, N, $\mathbf{p}$, $\mathbf{p}_\text{a}$
Output: $\bar{R}_{\text{lb}}^*$, $\mathbf{q}^*$

1: **Initialization:** Choose initial feasible point $\left(\mathbf{q}^{(0)},\mathbf{x}^{(0)},\mathbf{r}^{(0)}\right)$, set $i=0$.
2: **repeat**
3: Solve **P15**$\left(\mathbf{q}^{(i)},\mathbf{x}^{(i)}\right)$ to obtain $\left(\mathbf{q}^{(i+1)},\mathbf{x}^{(i+1)},\mathbf{r}^{(i+1)}\right)$.
4: Set $i=i+1$.
5: **until** $\bar{R}_{\text{lb}}^{\text{CGP}}(\mathbf{q}^{(i)},\mathbf{x}^{(i)},\mathbf{r}^{(i)})$ converges.
6: $\bar{R}_{\text{lb}}^*=\bar{R}_{\text{lb}}^{\text{CGP}}(\mathbf{q}^{(i)},\mathbf{x}^{(i)},\mathbf{r}^{(i)})$; $\mathbf{q}^*=\mathbf{q}^{(i)}$.

5.3.2 Memory-Rate Tradeoff Characterization

We now compare the optimized D-MCCS in **P10** with the lower bound in **P13** and demonstrate the equivalence of the two problems in some specific cases. Since the difference between **P10** and **P13** is only in the average rate objective expression, it is sufficient to compare $\bar{R}_{\text{MCCS}}(\mathbf{q})$ and $\bar{R}_{\text{lb}}(\mathbf{q})$.

We first consider a special case where there are at most two users being active at the same time, *i.e.,* $A\leq 2$. Conditional on $A\leq 2$, $\bar{R}_{\text{MCCS}}(\mathbf{q})$ in (5.8) and $\bar{R}_{\text{lb}}(\mathbf{q})$ in (5.20) are

rewritten as

$$\bar{R}_{\text{MCCS}}(\mathbf{q}) = E_{\mathcal{A}}\left[\sum_{\mathbf{d}_{\mathcal{A}}\in\mathcal{N}^A}\left(\prod_{k\in\mathcal{A}} p_{d_k}\right) R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q})\big|A\leq 2\right], \tag{5.24}$$

$$\bar{R}_{\text{lb}}(\mathbf{q}) = E_{\mathcal{A}}\left[\sum_{\mathcal{D}_{\mathcal{A}}\subseteq\mathcal{N}}\sum_{\mathbf{d}_{\mathcal{A}}\in\mathcal{T}(\mathcal{D}_{\mathcal{A}})}\left(\prod_{k\in\mathcal{A}} p_{d_k}\right) R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})\big|A\leq 2\right]. \tag{5.25}$$

Comparing (5.24) and (5.25), we show in the following theorem that the lower bound in **P13** is tight.

Theorem 8. For the special case of no more than two actives users at the same time, *i.e.,* $A \leq 2$, the average rate of the optimized D-MCCS in **P10** attains the lower bound in **P13**.

Proof. To show the equivalence of **P10** and **P13**, it is sufficient to show that $\bar{R}_{\text{MCCS}}(\mathbf{q})$ and $\bar{R}_{\text{lb}}(\mathbf{q})$ in (5.24) and (5.25) are equivalent. Comparing $\bar{R}_{\text{MCCS}}(\mathbf{q})$ and $\bar{R}_{\text{lb}}(\mathbf{q})$, we only need to examine $R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q})$ and $R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})$ in (5.7) and (4.12). We compare $R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q})$ and $R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})$ for the cases of $A = 1$ and $A = 2$ separately below.

Case 1: $A = 1$. Denote $\mathcal{A} = \{u_1\}$. In this case, $R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q})$ in (5.7) can be straightforwardly rewritten as

$$R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}) = \sum_{\mathcal{S}=\{u_1\}} \max_{k\in\mathcal{S}} q_{d_k}^s (1-q_{d_k})^{1-s} F = 1 - q_{d_{u_1}}.$$

For $\mathcal{D}_{\mathcal{A}} = \{d_{u_1}\}$, we rewrite $R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})$ in (4.12) as

$$R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q}) = 1 - q_{d_{u_1}} = R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}), \quad |\mathcal{A}| = 1. \tag{5.26}$$

Case 2: $A = 2$. Denote $\mathcal{A} = \{u_1, u_2\}$. In this case, the two active users can either have the same or distinct file requests. We discuss the two cases in the following.

$d_{u_1} = d_{u_2}$

Two users request the same file, and we have $\widetilde{N}(\mathbf{d}_{\mathcal{A}}) = 1$. Without loss the generality, we denote leader group $\mathcal{U}_{\mathcal{A}} = \{u_1\}$. By definition, the set of non-redundant groups is $\{\{u_1\}, \{u_1, u_2\}\}$. We can rewrite (5.7) as

$$R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}) = \sum_{\mathcal{S}\in\{\{u_1\},\{u_1,u_2\}\}} \max_{k\in\mathcal{S}} q_{d_k}^s (1-q_{d_k})^{2-s} F = (1-q_{d_{u_1}})^2 + q_{d_{u_1}}(1-q_{d_{u_1}}).$$

Given the leader group $\mathcal{U}_{\mathcal{A}} = \{u_1\}$, we have $\mathcal{D}_{\mathcal{A}} = \{d_{u_1}\}$ and it is straightforward to rewrite (4.12) as

$$R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q}) = (1-q_{d_{u_1}})^2 + q_{d_{u_1}}(1-q_{d_{u_1}}) = R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}). \tag{5.27}$$

$d_{u_1} \neq d_{u_2}$

When two users request different files and we have $\widetilde{N}(\mathbf{d}_{\mathcal{A}}) = 2$. The leader group is $\mathcal{U}_{\mathcal{A}} = \{u_1, u_2\}$. Thus, we can rewrite $R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q})$ in (5.7) as

$$\begin{aligned} R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}) &= \sum_{\mathcal{S}\in\{\{u_1\},\{u_2\},\{u_1,u_2\}\}} \max_{k\in\mathcal{S}} q_{d_k}^s (1-q_{d_k})^{2-s} F \\ &= (1-q_{d_{u_1}})^2 + (1-q_{d_{u_2}})^2 + \max\{q_{d_{u_1}}(1-q_{d_{u_1}}), q_{d_{u_2}}(1-q_{d_{u_2}})\}. \end{aligned}$$

Moreover, we also rewrite $R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})$ in (4.12) as

$$\begin{aligned} R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q}) &= \max\big\{(1-q_{d_{u_1}})^2 + (1-q_{d_{u_2}})^2 + q_{d_{u_1}}(1-q_{d_{u_1}}), \\ &\qquad (1-q_{d_{u_1}})^2 + (1-q_{d_{u_2}})^2 + q_{d_{u_2}}(1-q_{d_{u_2}})\big\} \\ &= R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}). \end{aligned} \tag{5.28}$$

From (5.27) and (5.28), we conclude $\bar{R}_{\text{MCCS}}(\mathbf{q}) = \bar{R}_{\text{lb}}(\mathbf{q})$ for $A = 2$. Combining the result in (5.26), we prove that $\bar{R}_{\text{MCCS}}(\mathbf{q}) = \bar{R}_{\text{lb}}(\mathbf{q})$ for $A = 1, 2$. □

Theorem 8 shows that if there are no more than two active users at the same time, then the optimized D-MCCS is an optimal decentralized caching scheme. In this case, the optimized D-MCCS characterizes the exact memory-rate tradeoff for decentralized caching under nonuniform file popularity. Moreover, the result also implies that zero-padding used in the delivery phased of D-MCCS incurs no loss of optimality in this case.

In the general scenario where the active user set is not limited to two users, although $\bar{R}_{\text{MCCS}}(\mathbf{q})$ and $\bar{R}_{\text{lb}}(\mathbf{q})$ may not be the same, the optimal placement solution $\mathbf{q}^*$ to **P10**, **P13** may still be the same for certain system configuration $(N, K, M, \mathbf{p}, \mathbf{p}_{\text{a}})$. The following proposition describes the result in this case.

Proposition 9. If $\mathbf{q}^*$ with $q_1^* = \cdots = q_N^*$ is the optimal solution to both **P10** and **P13**, then $\bar{R}_{\text{MCCS}}(\mathbf{q}^*) = \bar{R}_{\text{lb}}(\mathbf{q}^*)$. *i.e.,* the lower bound in **P13** is attained by the optimized D-MCCS in **P10**.

Proof. For $q_1^* = \cdots = q_N^*$, based on (5.5), the length of coded message $\mathcal{C}_{\mathcal{S}}$ corresponding to $\mathcal{S} \in \mathcal{Q}^{s+1}$ is given by

$$\max_{k\in\mathcal{S}}(q_{d_k}^*)^s(1-q_{d_k}^*)^{A-s}F = (q_1^*)^s(1-q_1^*)^{A-s}F.$$

Following this, based on (5.7), we have

$$R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}^*) = \sum_{s=0}^{A-1}\sum_{\mathcal{S}\in\mathcal{Q}^{s+1}}(q_1^*)^s(1-q_1^*)^{A-s}F. \tag{5.29}$$

The two summations in (5.29) represents the number of all the non-redundant groups. By the definition, the non-redundant groups are the active user subset include at least one users in the leader group $\mathcal{U}_{\mathcal{A}}$. Denote $\mathcal{U}_{\mathcal{A}} \triangleq \{u_1,\ldots,u_{\widetilde{N}(\mathbf{d}_{\mathcal{A}})}\}$. Among all user subsets in $\mathcal{Q}^{s+1}$, there are $\binom{A-i}{s}$ subsets including user $\{u_1,\ldots,u_i\}$. By considering $i = 1,\ldots,\widetilde{N}(\mathbf{d}_{\mathcal{A}})$, we can rewrite (5.29) as

$$R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}};\mathbf{q}^*) = \sum_{s=0}^{A-1}\sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})}\binom{A-i}{s}(q_1^*)^s(1-q_1^*)^{A-s}F = R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q}^*). \tag{5.30}$$

Thus, we can conclude that $\bar{R}_{\text{MCCS}}(\mathbf{q}^*) = \bar{R}_{\text{lb}}(\mathbf{q}^*)$. □

Proposition 9 indicates that if the optimal placement $\mathbf{q}^*$ is symmetric for all files, $q_1^* = \cdots = q_N^*$, the optimized D-MCCS is an optimal decentralized caching scheme that characterizes the exact memory-rate tradeoff. One known example is the special case of uniform file popularity. In this case, the optimized D-MCCS (**P10**) and the lower bound (**P13**) have the same optimal solution $\mathbf{q}^*$ with q_n^*'s being all identical, and the result $\bar{R}_{\text{MCCS}}(\mathbf{q}^*) = \bar{R}_{\text{lb}}(\mathbf{q}^*)$ has been shown in [44].

Finally, we point out that our numerical study in Section 3.6 shows that the gap between the optimized D-MCCS and the lower bound in **P13** is very small in general. This indicates that the performance of the optimized D-MCCS is very close to the optimal decentralized caching.

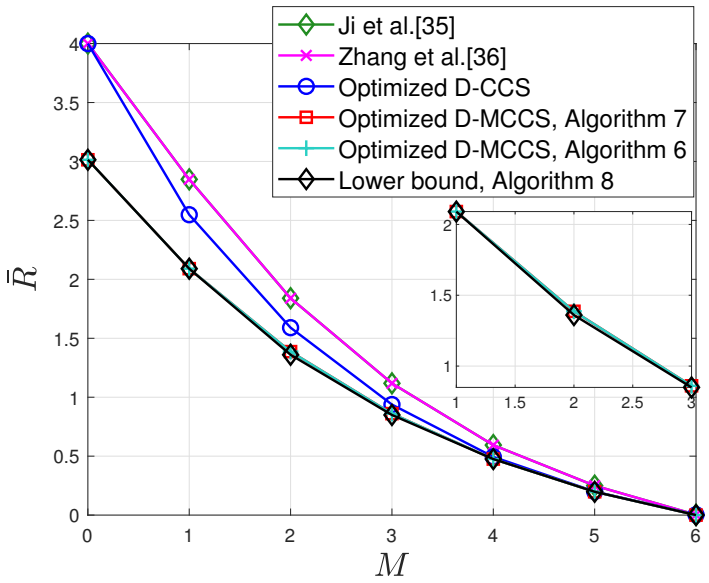

Figure 5.1: Average rate $\bar{R}$ vs. cache size M ($N = 6$, $K = 4$, Zipf file popularity distribution with $\theta = 0.56$).

5.4 Numerical Results

We now provide our numerical study on the performance of the optimized D-MCCS in **P10** and the proposed lower bound for decentralized caching in **P13**. We set $N = 6$ files and $K = 4$ users. We generate the nonuniform file popularities using the Zipf distribution with $p_n = n^{-\theta}/\sum_{i=1}^{N} i^{-\theta}$, where θ is the Zipf parameter. We set the probability of each user being active as $p_{\mathrm{a},k} = 0.5, k \in \mathcal{K}$. We use $\bar{R}$ to generally indicate the average rate obtained by various schemes considered. We consider our proposed two schemes (Algorithms 6 and 7) for solving **P10** to optimize D-MCCS, Algorithm 8 for the lower bound in **P13**. We set the convergence criterion for both Algorithms 6 and 8 as the difference in the average rate over consecutive iterations is less than $1e{-}4$. For comparison, we also consider the existing well-known decentralized scheme based on D-CCS, including the optimized D-CCS [61] and the file-grouping-based schemes in [35] and [36].

In Fig. 5.1, we plot $\bar{R}$ vs. cache size M under different schemes for $\theta = 0.56$ (the same as [33]). For the optimized D-MCCS in **P10**, we observe that the average rate $\bar{R}$ achieved by Algorithms 6 and 7 are nearly identical. This indicates that the low-complexity two-

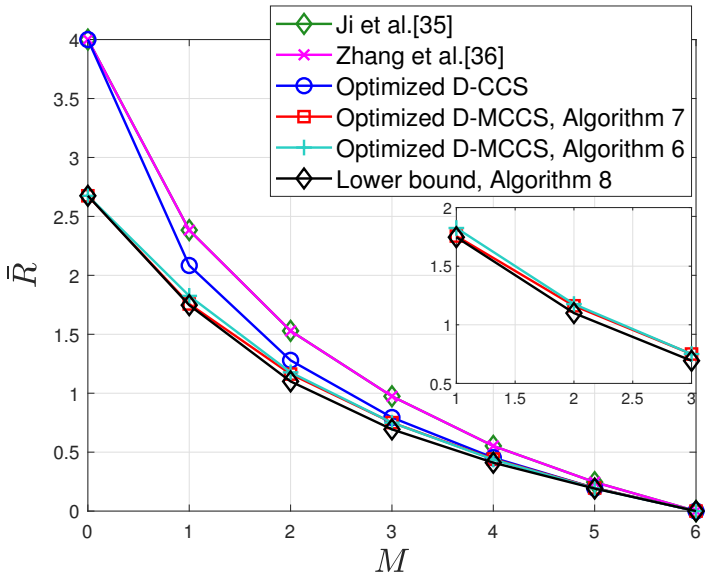

Figure 5.2: Average rate $\bar{R}$ vs. cache size M ($N = 6$, $K = 4$, Zipf file popularity distribution with $\theta = 1.2$).

file-group-based solution is close to the stationary points obtained by the successive GP approximation algorithm. Among the caching schemes compared, the optimized D-MCCS provides the lowest average rate for all values of M. The gap between the optimized D-MCCS and the optimized D-CCS reduces as M increases. This is mainly because there are more redundant groups in the coded delivery phase for a smaller value of M, and, as a result, the D-MCCS improved upon the D-CCS more. The average rates obtained by Algorithms 6 and 7 are both very close to the lower bound in **P13**, with only a very small gap observed for $M \in [1, 3]$. This shows that the performance of the optimized D-MCCS is close to that under the optimal decentralized caching.

In Fig. 5.2, we consider a larger value of $\theta = 1.2$ for a more skewed file popularity distribution to study $\bar{R}$ vs. M. The average rates obtained by Algorithms 6 and 7 are again nearly identical. However, it is interesting to see that for $M = 1$ and 2, Algorithm 7 achieves a lower average rate than Algorithm 6 does. This shows that in some cases, the two-file-group-based solution could perform even better than the successive GP approximation algorithm, which has a much higher computational complexity. Among all the schemes compared, the optimized D-MCCS again achieves the lowest average rate for all

values of M. The gap between optimized D-MCCS and the lower bound is again very small in general. This demonstrates the optimized D-MCCS is close to optimal for decentralized caching.

5.5 Summary

In this chapter, we first formulated the cache placement optimization problem for the D-MCCS to minimize the average rate and developed two algorithms to solve this non-convex problem: a successive GP approximation algorithm to compute a stationary point of the optimization problem, and a low-complexity simple two-file-group-based approximate algorithm. To study the memory-rate tradeoff, we proposed a lower bound through a non-convex optimization problem, for which we developed an algorithm to compute a stationary point. We compared the optimized D-MCCS with the lower bound, showing that for the special case of no more than two active users, the optimized D-MCCS attains the lower bound. For general cases, we identified a condition for the optimized D-MCCS to attain the lower bound. Our numerical study showed that the gap between the average rate of the optimized D-MCCS and the lower bound is very small in general.

Chapter 6

Coded Distributed Computing with Nonuniform File Popularity

In this chapter, we study the heterogeneous CDC for arbitrary number of files with nonuniform popularity, assuming nonuniform mapping and reducing loads.

6.1 System Model

We consider a distributed computing framework with a MapReduce structure, where the network aims to process jobs from a set of N input files using K distributed workers. Denote $\mathcal{K} \triangleq \{1, \ldots, K\}$ and $\mathcal{N} \triangleq \{1, \ldots, N\}$ as the worker and file indexes, respectively. Each file $n \in \mathcal{N}$ is of size F bits. We assume files have nonuniform popularity to be used by the jobs, which is common in practical systems and can be measured by the frequency of the a file being accessed by the jobs [55,56,60]. Denote $\mathbf{p} = [p_1, \ldots, p_N]$ as the popularity distribution vector of the N files, where p_n is the probability of file n being accessed by a job, and $\sum_{n=1}^{N} p_n = 1$. The K workers are expected to accomplish different jobs cooperatively. The processing of a job needs access to a subset of files $\mathcal{D} \subseteq \mathcal{N}$, where $\mathcal{D} \neq \emptyset$. We refer $\mathcal{D}$ as the input files of the job. Also define $D = |\mathcal{D}|$ as the number of input files in $\mathcal{D}$. By the MapReduce framework, each job is split into Q target functions, denoted by $\{\phi_1(\mathcal{D}), \ldots, \phi_Q(\mathcal{D})\}$, where each target function maps all the input files in $\mathcal{D}$ to an output stream Φ_q of B bits: $\Phi_q = \phi_q(\mathcal{D})$. Let $\mathcal{Q} \triangleq \{1, \ldots, Q\}$ denote the indexes of the target functions. These target functions are distributed to different workers to compute.

The computing of the Q target functions $\{\phi_q(\mathcal{D}), q \in \mathcal{Q}\}$ is decomposed into a combination of Map and Reduce functions. Specifically, the Q target functions are first split into QD Map functions. For a given target function $\phi_q(\mathcal{D}), q \in \mathcal{Q}$ and an input file $n \in \mathcal{D}$, each Map function, denoted by $g_{q,n}$ outputs an Intermediate Value (IV) $V_{q,n}$ of T bits. Define $V_{q,n}$ as the IV generated by $g_{q,n}$ that is of T bits. Each target function $\phi_q(\mathcal{D}), q \in \mathcal{Q}$, is computed by a Reduce function h_q from the corresponding IVs for files in $\mathcal{D}$ as

$$\phi_q(\mathcal{D}) = h_q(\{V_{q,n}, n \in \mathcal{D}\}), \quad q \in \mathcal{Q}.$$

Each worker $k \in \mathcal{K}$ computes a subset of the QD Map functions based on its stored input files and generates the local IVs. Let $M_k \in \mathbb{N}^+$ denote the number of files that can be stored by worker k, referred to as the mapping load. Denote $M \triangleq [M_1, \ldots, M_K]$ as the mapping load vector of the workers. Following the common practice [26–30, 32], we assume $\sum_{k=1}^{K} M_k \geq N$, which guarantees that all the files can be stored among the workers. Denote $\mathcal{M}_k \subseteq \mathcal{N}$ as the set of files stored at worker k. Each worker $k \in \mathcal{K}$ computes the Map functions of its stored input files in $\mathcal{M}_k \cap \mathcal{D}$ for each $q \in \mathcal{Q}$, $\{g_{q,n}, q \in \mathcal{Q}, n \in \mathcal{M}_k \cap \mathcal{D}\}$.

We consider a heterogeneous target function assignment where workers may be assigned with different number of target functions. Denote $\mathcal{W}_k$ as the set of target functions assigned to worker k where $W_k \triangleq |\mathcal{W}_k|/Q$ is the reducing load of worker k normalized by Q. Denote $W \triangleq [W_1, \ldots, W_K]$ as the reducing load vector of the workers. Note that each target function is computed by one worker, *i.e.,* $\sum_{k=1}^{K} W_k = Q$.

In summary, the distributed computing network computes any given job in three phases: *Map, Shuffle* and *Reduce*.

- In the Map phase, each worker $k \in \mathcal{K}$ computes the Map functions for all the target functions and its stored input files $\mathcal{M}_k \cap \mathcal{D}$ to generate local IVs, given by $\{V_{q,n} : q \in \mathcal{Q}, n \in \mathcal{M}_k \cap \mathcal{D}\}$.

- In the Shuffle phase, each worker $k \in \mathcal{K}$ generates message X_k using its local IVs and multicasts it to all the other workers. Each worker k receives the multicasted

messages from the other workers $\{X_i, i \in \mathcal{K}\backslash\{k\}\}$ to obtain required IVs that are not computed locally, *i.e.,* $\{V_{q,n} : q \in \mathcal{W}_k, n \notin \mathcal{M}_k, n \in \mathcal{D}\}$.

- In the Reduce phase, each worker $k \in \mathcal{K}$ computes its assigned target functions $\mathcal{W}_k$ through the Reduce functions, using its local IVs and other required IVs obtained during the Shuffle phase, given by $\{h_q(\{V_{q,n}, n \in \mathcal{D}\}), q \in \mathcal{W}_k\}$.

6.2 Heterogeneous Coded Distributed Computing

The original CDC scheme [26] is constructed for a single job, which requires a sufficiently large number N of input files, assuming uniform mapping load and reducing load. The main idea there is to form coded messages to explore coded multicasting opportunities in the Shuffle phase. Specifically, in the Map phase, each worker is assigned the same number of target functions. The files are partitioned into file subsets. A unique file subset (which can be empty) is assigned to a unique worker subset $\mathcal{S} \subseteq \mathcal{K}$, and worker subsets of the same size contains the same number of input files. Each worker $k \in \mathcal{S}$ generates the same number of IVs, and these local IVs among the workers in $\mathcal{S}$ are mutually exclusive. In the Shuffle phase, each worker $k \in \mathcal{S}$ generates a coded message containing the IVs needed by all the rest workers in $\mathcal{S}$ and multicast to them. Note that this scheme requires N to be large enough for files to be partitioned into required subsets.

In this work, we consider a heterogeneous CDC scenario for N input files with nonuniform popularity. We propose a file placement strategy for any number N of files. With nonuniform file popularity, the size of file subsets placed at the worker subsets of the same size may be different. Thus, the number of IVs needed by different workers in the same worker subset may be different. This causes variable lengths of messages at each worker that need to be coded together for multicasting. As a results, some IVs cannot be encoded into the same coded message. One straightforward solution is to unicast these remaining IVs. However, it leads to a loss due to not taking advantage of coded multicasting opportunities [27–29]. To overcome this, we propose a nested coded shuffling strategy to encode all IVs, by adopting a similar approach proposed in [30] to process a job. A different

heterogeneous CDC scenario is considered in [30], where although all files are needed to process a job, workers have different mapping loads and reducing loads. Intuitively, this strategy exploits extra coded multicasting opportunities for those remaining IVs, leading to reduce the shuffling load as compare with the unicast solution.

6.2.1 The Mapping Strategy

During the Map phase, the files are placed at the workers so that the Map functions are computed to generate local IVs. For K worker, there are $2^K - 1$ non-empty worker subsets. Each of the N files is placed in a unique worker subset. Let $t_{n,\mathcal{S}}$ be an indicator to indicate whether file n is exclusively placed in the worker subset $\mathcal{S}$:

$$t_{n,\mathcal{S}} \in \{0,1\}, \quad n \in \mathcal{N}, \mathcal{S} \subseteq \mathcal{K}, \mathcal{S} \neq \emptyset. \tag{6.1}$$

Then, we have the file placement constraint

$$\sum_{\mathcal{S}\subseteq\mathcal{K},\mathcal{S}\neq\emptyset} t_{n,\mathcal{S}} = 1, \quad n \in \mathcal{N}. \tag{6.2}$$

Furthermore, the files placed at each worker $k \in \mathcal{K}$ cannot exceed the mapping load, and we have the mapping load constraint

$$\sum_{n=1}^{N} \sum_{\mathcal{S}\subseteq\mathcal{K},\mathcal{S}\neq\emptyset,k\in\mathcal{S}} t_{n,\mathcal{S}} \leq M_k, \quad k \in \mathcal{K}. \tag{6.3}$$

Consider a given job requiring the set of input files $\mathcal{D}$. Let $\mathcal{A}_{\mathcal{S}}^{\mathcal{D}} \subseteq \mathcal{D}$ denote the set of files placed exclusively at worker subset $\mathcal{S}$, and let $a_{\mathcal{S}}^{\mathcal{D}} = |\mathcal{A}_{\mathcal{S}}^{\mathcal{D}}|$. Then, we have

$$a_{\mathcal{S}}^{\mathcal{D}} = \sum_{n\in\mathcal{D}} t_{n,\mathcal{S}}, \quad \mathcal{S} \subseteq \mathcal{K}, \mathcal{S} \neq \emptyset. \tag{6.4}$$

Let $\mathcal{T} \triangleq \{t_{n,\mathcal{S}} : n \in \mathcal{N}, \mathcal{S} \subseteq \mathcal{K}, \mathcal{S} \neq \emptyset\}$ denote the file placement strategy. Based on the file placement strategy $\mathcal{T}$, the workers compute the Map functions to generate local IVs. The IVs generated exclusively by worker subset $\mathcal{S}$ are given by $\{V_{q,n} : q \in \mathcal{Q}, n \in \mathcal{A}_{\mathcal{S}}^{\mathcal{D}}\}$. We aim to optimize the file placement strategy to minimize the expected shuffling load.

Remark 14. Note that the existing homogeneous and heterogeneous CDC schemes [26–32] require the number of input files being sufficiently large. Also, they assume each job requires all input files and do not consider the heterogeneity of file popularity for different jobs. In contrast, we propose a mapping strategy for arbitrary number of input files with nonuniform popularity to be used by jobs.

6.2.2 The Nested Coded Shuffling Strategy

In the Shuffle phase, each worker k needs to obtain the IVs that are not computed locally $\{V_{q,n} : q \in \mathcal{W}_k, n \notin \mathcal{M}_k, n \in \mathcal{D}\}$, in order to compute the assigned target functions. As discussed in earlier, in a CDC scheme, each worker combines the IVs needed by other workers in its worker subset in a coded message and multicast it to them. However, in a heterogeneous scenario where input file has different popularity, the number of the IVs needed by different workers in the same worker subset may be different, which cannot be directly coded into the same message. This complicates the design of coded shuffling strategy.

To maximize the benefit of coded multicasting to reduce the shuffling load, in the following, we propose a nested coded shuffling strategy that explores coded multicasting opportunities for all IVs. Our strategy follows a similar approach in [30], which is designed for processing a job that requires all input files, while workers have different loads. We extend that approach to a more general heterogeneous scenario, where input files have nonuniform popularity to be used by multiple jobs.

We first describe the shuffling strategy for worker k in given worker subset $\mathcal{S}$, where $|\mathcal{S}| \geq 2$.[1] The same strategy is then applied to all other $\mathcal{S}$'s that work k is in, and to all other workers. Based on the file placement, each unique worker subset $\mathcal{S}$ is assigned an exclusive input file subset. Thus, worker k needs IVs from some worker subset in $\{\mathcal{S}' : \mathcal{S}' \subseteq \mathcal{K}, k \notin \mathcal{S}'\}$.

For a job with input file set $\mathcal{D}$, the nested coded shuffling strategy consists of three steps: Step 1: Identify the IVs to be sent to each worker in $\mathcal{S}$; Step 2: Recursively regroups

[1]Note that for worker subset $\mathcal{S}$ containing a single worker $|\mathcal{S}| = 1$, there is no IVs need to be shuffled.

portion of IVs of the same size so that they can be coded into the same message to be multicasted in the subset of $\mathcal{S}$; Step 3: Generate coded messages to be multicasted to other workers. We have the following steps.

Step 1: For any $\mathcal{S} \subseteq \mathcal{K}$, consider worker $k \in \mathcal{S}$, and consider input files placed in $\mathcal{S}_{-k} \triangleq \mathcal{S} \backslash \{k\}$. Worker k needs to obtain the IVs computed by the workers in $\mathcal{S}_{-k}$, given by $\mathcal{V}^{\mathcal{D}}_{k,\mathcal{S}_{-k}} = \{V_{q,n} : q \in \mathcal{W}_k, n \in \mathcal{A}^{\mathcal{D}}_{\mathcal{S}_{-k}}\}$. Recall from (6.4) that $a^{\mathcal{D}}_{\mathcal{S}_{-k}} = |\mathcal{A}^{\mathcal{D}}_{\mathcal{S}_{-k}}|$. Thus, the total size of $\mathcal{V}^{\mathcal{D}}_{k,\mathcal{S}_{-k}}$ in bits is $|\mathcal{V}^{\mathcal{D}}_{k,\mathcal{S}_{-k}}| = TW_kQa^{\mathcal{D}}_{\mathcal{S}_{-k}} = TW_kQ\sum_{n\in\mathcal{D}} t_{n,\mathcal{S}_{-k}}$. Note that $|\mathcal{V}_{k,\mathcal{S}_{-k}}|$ may be different for each $k \in \mathcal{S}$.

On the other hand, for shuffling, any worker $j \in \mathcal{S}$ needs to code the IVs required by other workers in $\mathcal{S}$ into a message. Since the total size of the IVs may be different for different workers, we take a fraction of the same size from the IVs for each worker and encode them into the message. This leaves some *residual* IVs at worker j, which need to be sent to other workers in $\mathcal{S}$.

Let $\mathcal{S}_{+i} \triangleq \mathcal{S} \cup \{i\}$, for $i \in \mathcal{K}\backslash\mathcal{S}$. Let $d^{\mathcal{S}}_{k,\mathcal{S}_{+i}}(\mathcal{D})$ denote the total size (in bits normalized by Q) of those residual IVs that are needed by worker k in worker subset $\mathcal{S}_{+i}$. Our proposed strategy is to encode those residual IVs into coded messages and send them in $\mathcal{S}$ or the subset of $\mathcal{S}$. In other words, those residual IVs needed by worker k from all those worker subsets of size $|\mathcal{S}| + 1$, where k is in, will be encoded and sent in $\mathcal{S}$ or subsets of $\mathcal{S}$ in Step 2. Following this, the total size of those residual IVs from $\mathcal{S}_{+i}$, for $i \in \mathcal{K}\backslash\mathcal{S}$, is $Q\sum_{i\in\mathcal{K}\backslash\mathcal{S}} d^{\mathcal{S}}_{k,\mathcal{S}_{+i}}(\mathcal{D})$.

Let $\mathcal{I}^{\mathcal{D}}_{k,\mathcal{S}}$ denote the set of IVs to be sent to worker k within subset $\mathcal{S}$. It consists of the IVs in $\mathcal{V}^{\mathcal{D}}_{k,\mathcal{S}_{-k}}$ and the residual IVs from all $\mathcal{S}_{+i}$'s, for $i \in \mathcal{K}\backslash\mathcal{S}$. Thus, the size of $\mathcal{I}^{\mathcal{D}}_{k,\mathcal{S}}$ in bits is given by

$$|\mathcal{I}^{\mathcal{D}}_{k,\mathcal{S}}| = W_kTQ\sum_{n\in\mathcal{D}} t_{n,\mathcal{S}_{-k}} + Q\sum_{i\in\mathcal{K}\backslash\mathcal{S}} d^{\mathcal{S}}_{k,\mathcal{S}_{+i}}(\mathcal{D}). \tag{6.5}$$

Note again that $|\mathcal{I}^{\mathcal{D}}_{k,\mathcal{S}}|$ may be different for each $k \in \mathcal{S}$.

Step 2: Let $\mathcal{I}^{j}_{k,\mathcal{S}}(\mathcal{D})$ denote the portion of $\mathcal{I}^{\mathcal{D}}_{k,\mathcal{S}}$ that are locally computed by worker $j \in \mathcal{S}$, $j \neq k$, and sent to other workers in $\mathcal{S}$. We require $\mathcal{I}^{j}_{k,\mathcal{S}}(\mathcal{D})$'s, for $k \in \mathcal{S}, k \neq j$, to be the same size, such that worker j can encodes them into a message and multicasts it

to all other workers in $\mathcal{S}$. This ensures that all IVs sent in $\mathcal{S}$ are via multicasting. Define $L^{\mathcal{D}}_{j,\mathcal{S}} \triangleq |\mathcal{I}^j_{k,\mathcal{S}}(\mathcal{D})|/Q$, for all $k \in \mathcal{S}, k \neq j$.

After the above step, if there are residual IVs in $\mathcal{I}^{\mathcal{D}}_{k,\mathcal{S}}$, we need to identify the IVs to be encoded in worker subset $\mathcal{S}' \subset \mathcal{S}$ of size $|\mathcal{S}|-1$. This essentially follows the same strategy to encode the residual IVs as described in Step 1. Recall from Step 1 that $d^{\mathcal{S}_{-i}}_{k,\mathcal{S}}(\mathcal{D})$ is the size (in bits normalized by Q) of the IVs in $\mathcal{I}^{\mathcal{D}}_{k,\mathcal{S}}$ needed by worker k in $\mathcal{S}$ and are encoded and sent in $\mathcal{S}_{-i}$ or subsets of $\mathcal{S}_{-i}$, for $i \in \mathcal{S}, i \neq k$. Then, the total size of these residual IVs from $\mathcal{S}$ is $Q\sum_{i\in\mathcal{S},i\neq k} d^{\mathcal{S}_{-i}}_{k,\mathcal{S}}(\mathcal{D})$. Then, this nested process will continue until all IVs can be encoded and sent via multicasting and no residual IVs remaining.

To ensure this nested strategy of regrouping of the residual IVs to be sent in the subsets of current worker subset to be feasible, from (6.5), we have the following equality constraint

$$W_k T\sum_{n\in\mathcal{D}} t_{n,\mathcal{S}_{-k}} + \sum_{i\in\mathcal{K}\backslash\mathcal{S}} d^{\mathcal{S}}_{k,\mathcal{S}_{+i}}(\mathcal{D}) = \sum_{j\in\mathcal{S},j\neq k} L^{\mathcal{D}}_{j,\mathcal{S}} + \sum_{i\in\mathcal{S},i\neq k} d^{\mathcal{S}_{-i}}_{k,\mathcal{S}}(\mathcal{D}),$$
$$k \in \mathcal{S}, \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 2, \mathcal{D} \subseteq \mathcal{N}. \tag{6.6}$$

Note that $\{L^{\mathcal{D}}_{j,\mathcal{S}}\}$ and $\{d^{\mathcal{S}_{-i}}_{k,\mathcal{S}}(\mathcal{D})\}$ are the design variables for coded shuffling. These variables along with the file placement $\mathcal{T}$ are to be jointly optimized to minimize the expected shuffling load, which will be described in Section 6.3.

Step 3: Each worker $j \in \mathcal{S}$ generates a coded message via bitwise XOR operation of $\mathcal{I}^j_{i,\mathcal{S}}(\mathcal{D})$'s, for all $i \in \mathcal{S}, i \neq j$, as

$$C^j_{\mathcal{S}}(\mathcal{D}) \triangleq \bigoplus_{i\in\mathcal{S},i\neq j} \mathcal{I}^j_{i,\mathcal{S}}(\mathcal{D}). \tag{6.7}$$

The coded message $C^j_{\mathcal{S}}(\mathcal{D})$ is then multicasted by worker j to the rest workers in $\mathcal{S}$.

To decode $C^j_{\mathcal{S}}(\mathcal{D})$ at worker $k \in \mathcal{S}$, we note that by the definition of $\mathcal{I}^j_{i,\mathcal{S}}(\mathcal{D})$ in Step 2, each worker $k \in \mathcal{S}$ has already locally generated all the IVs in $\mathcal{I}^j_{i,\mathcal{S}}(\mathcal{D})$, for $i \in \mathcal{S},\ i \neq k, j$. Thus, worker k can successfully decode $\mathcal{I}^j_{k,\mathcal{S}}(\mathcal{D})$ from $C^j_{\mathcal{S}}(\mathcal{D})$ from worker $j \in \mathcal{S}$, $j \neq k$. Since $L^{\mathcal{D}}_{j,\mathcal{S}} = |\mathcal{I}^j_{k,\mathcal{S}}(\mathcal{D})|/Q$, for all $k \in \mathcal{S}, k \neq j$, the size of the coded message $C^j_{\mathcal{S}}(\mathcal{D})$ normalized by Q is

$$|C^j_{\mathcal{S}}(\mathcal{D})| = L^{\mathcal{D}}_{j,\mathcal{S}}, \quad j \in \mathcal{S}. \tag{6.8}$$

The overall nested coded shuffling strategy applies the above Steps 1-3 to each worker subset in $\{\mathcal{S} : \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 2\}$ to let each worker generates the coded message and multicasts it to the other works in the same worker subset. The following proposition shows the validity of the proposed nested shuffling strategy.

Proposition 10. The proposed file placement strategy and coded shuffling strategy for the heterogeneous CDC with nonuniform file popularity described in Section 6.1 are valid for any number N of input files.

Proof. To prove the proposed strategies are valid, we need to show that for each job with its required set of input files $\mathcal{D}$, each worker $k \in \mathcal{K}$ can obtain all of the IVs it needs from $\mathcal{D}$. Recall that for each worker subset $\mathcal{S}$ containing at least two workers ($|\mathcal{S}| \geq 2$), the file placement strategy $\mathcal{T}$ and the shuffling strategy with design variables $\{L^{\mathcal{D}}_{k,\mathcal{S}}\}$ and $\{d^{\mathcal{S}_{-i}}_{k,\mathcal{S}}(\mathcal{D})\}$ must satisfy equality constraint (6.6). Summing both side of (6.6) over all $\mathcal{S}$'s, where $k \in \mathcal{S}$, and $\mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 2$, we have

$$\sum_{\substack{\mathcal{S}\subseteq\mathcal{K},\\ |\mathcal{S}|\geq 2, k\in\mathcal{S}}} W_k T \sum_{n\in\mathcal{D}} t_{n,\mathcal{S}_{-k}} + \sum_{\substack{\mathcal{S}\subseteq\mathcal{K},\\ |\mathcal{S}|\geq 2, k\in\mathcal{S}}} \sum_{i\in\mathcal{K}\backslash\mathcal{S}} d^{\mathcal{S}}_{k,\mathcal{S}_{+i}}(\mathcal{D}) = \sum_{\substack{\mathcal{S}\subseteq\mathcal{K},\\ |\mathcal{S}|\geq 2, k\in\mathcal{S}}} \sum_{j\in\mathcal{S}, j\neq k} L^{\mathcal{D}}_{j,\mathcal{S}} + \sum_{\substack{\mathcal{S}\subseteq\mathcal{K},\\ |\mathcal{S}|\geq 3, k\in\mathcal{S}}} \sum_{i\in\mathcal{S}, i\neq k} d^{\mathcal{S}_{-i}}_{k,\mathcal{S}}(\mathcal{D}). \tag{6.9}$$

Note that the second terms on both sides of (6.9) are equal. Thus, we have

$$\sum_{\substack{\mathcal{S}\subseteq\mathcal{K},\\ |\mathcal{S}|\geq 2, k\in\mathcal{S}}} W_k T \sum_{n\in\mathcal{D}} t_{n,\mathcal{S}_{-k}} = \sum_{\substack{\mathcal{S}\subseteq\mathcal{K},\\ |\mathcal{S}|\geq 2, k\in\mathcal{S}}} \sum_{j\in\mathcal{S}, j\neq k} L^{\mathcal{D}}_{j,\mathcal{S}}. \tag{6.10}$$

Note that the left hand side of (6.10) is the total size of all the IVs needed by worker k in all the worker subsets in $\{\mathcal{S} : \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 2, k \in \mathcal{S}\}$, which contains all the IVs worker k needs from other workers, as discussed at the beginning of Section 6.2.2. The right hand side of (6.10) is the total size of all the IVs multicasted by all the workers other than worker k in each of the worker subset in $\{\mathcal{S} : \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 2, k \in \mathcal{S}\}$. The equality (6.10) shows that, under the strategies satisfying constraint (6.6), each worker k can obtain its needed IVs from the multicasted coded messages by all the other workers. □

6.3 File Placement and Coded Shuffling Optimization

In this section, we formulate an optimization problem to optimize the file placement and nested coded shuffling strategies proposed in Section 6.2.1 and 6.2.2.

For a given job, the shuffling load within a worker subset $\mathcal{S}$ equals to the coded messages multicasted by all the workers in $\mathcal{S}$ and the overall shuffling load $L(\mathcal{D})$ is the sum over all the worker subsets in $\{\mathcal{S} : \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 2\}$, given by

$$L(\mathcal{D}) = \sum_{\mathcal{S},j:\mathcal{S}\subseteq\mathcal{K},|\mathcal{S}|\geq 2,j\in\mathcal{S}} |C_{\mathcal{S}}^{j}(\mathcal{D})| = \sum_{\mathcal{S},j:\mathcal{S}\subseteq\mathcal{K},|\mathcal{S}|\geq 2,j\in\mathcal{S}} L_{j,\mathcal{S}}^{\mathcal{D}}.$$

We define $p_{\mathcal{D}}$ as the probability of a nonempty input file set $\mathcal{D}$ being accessed by a job, which is a function of the file popularity distribution $\mathbf{p}$. By averaging $L(\mathcal{D})$ over all the possible nonempty input file sets $\mathcal{D} \subseteq \mathcal{N}, \mathcal{D} \neq \emptyset$, the expected shuffling load is given by

$$\bar{L} = \sum_{\mathcal{D}\subseteq\mathcal{N},\mathcal{D}\neq\emptyset} p_{\mathcal{D}} L(\mathcal{D}).$$

Following this, we formulate **P16** that jointly optimizes the file placement strategy $\mathcal{T}$ and nested coded shuffling strategy $\{L_{k,\mathcal{S}}^{\mathcal{D}}, d_{k,\mathcal{S}}^{\mathcal{S}_{-i}}(\mathcal{D})\}$ to minimize $\bar{L}$.

$$\mathbf{P16}: \min_{\{t_n,\mathcal{S},L_{k,\mathcal{S}}^{\mathcal{D}},d_{k,\mathcal{S}}^{\mathcal{S}_{-i}}(\mathcal{D})\}} \bar{L}$$

$$\text{s.t. } (6.1), (6.2), (6.3), (6.6), \text{ and}$$

$$L_{k,\mathcal{S}}^{\mathcal{D}} \geq 0,\ k \in \mathcal{S}, \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 2, \mathcal{D} \subseteq \mathcal{N} \tag{6.11}$$

$$d_{k,\mathcal{S}}^{\mathcal{S}_{-i}}(\mathcal{D}) \geq 0, k \in \mathcal{S}, i \in \mathcal{S}_{-k}, \mathcal{D} \subseteq \mathcal{N}, \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| \geq 3 \tag{6.12}$$

where (6.11) and (6.12) guarantee that the size of the coded messages are non-negative. Due to constraint (6.1), **P16** is a mixed integer linear programming (MILP) problem, which in general is NP-hard [103]. Although approximate solution can be obtained by existing optimization solvers using the branch-and-cut method (*e.g.,* Mosek, Gurobi etc.), the computational complexity remains very high. In the following, we resort to develop a heuristic approximate algorithm to solve the problem.

6.3.1 Approximate Solution for P16

The constraint on the file placement vector in (6.1) makes **P16** a MILP problem. To eliminate this constraint, we first propose a two-file-group-based file placement strategy.

Two-file-group-based file placement: We partition the files $\mathcal{N}$ into two non-overlapping groups. The first group contains the N_1 popular files in $\mathcal{N}$, defined as $\mathcal{N}_1 \triangleq \{1, \ldots, N_1\}$ for $N_1 \in \mathcal{N}$. The second group contains the rest of the less popular files, denoted by $\mathcal{N}_2 \triangleq \mathcal{N} \backslash \mathcal{N}_1$. We describe the two-file-group-based placement strategy in Algorithm 9. For a given N_1, the output of Algorithm 9 is a two-file-group-based file placement strategy $\{\bar{\mathcal{M}}_k^{N_1}, k \in \mathcal{K}\}$, where $\bar{\mathcal{M}}_k^{N_1}$ is the set of files placed at worker k. We start with the placement of the files in $\mathcal{N}_2$ in lines 1 – 8. The idea is to place each file in $n \in \mathcal{N}_2$ at exactly one worker. Starting from $k = 1$ and $n = N_1 + 1$, we place file n at worker k in line 5 and update its mapping load M_k in line 6. Then we move to next available worker $(k+1) \bmod K$ to place file $n+1$ until finish the placement for $\mathcal{N}_2$. The placement strategy of the files in $\mathcal{N}_1$ is described in lines 9 – 17. Starting from $k = 1$, for each worker k with $M_k > 0$, we place the files in $\mathcal{N}_1$ cyclically at worker k until its mapping load fulfilled and then move to the next available worker.

Comments on the two-file-group-based file placement. The proposition of the two-file-group-based file placement strategy is inspired by the widely used two-file-group-based cache placement for coded caching under nonuniform file popularity [36]. Note that the CDC scheme is an extension of the coded caching to the distributed computing problem, with the similar idea of exploiting the coded multicasting opportunities that significantly reduce the delivery/shuffling load [26]. For coded caching under nonuniform popularity, the two-file-group-based cache placement has shown to have close-to-optimal performance [36, 48]. The two-file-group-based cache placement keeps the less popular files solely in the server and devotes all the caches to the popular files through a decentralized placement. Our two-file-group-based file placement for the heterogeneous CDC employs the similar idea: We place each of the less popular files at one worker so that the minimum mapping loads are used, and devote all the rest mapping loads to the popular files by

Algorithm 9 Two-file-group-based file placement algorithm

Input: K, M, N, N_1
Output: $\bar{\mathcal{M}}_1^{N_1}, \ldots, \bar{\mathcal{M}}_K^{N_1}$

#File placement for the second file group $\mathcal{N}_2$
$\bar{\mathcal{M}}_k^{N_1} = \emptyset$, $k \leftarrow 1$
for $n = N_1 + 1$ to N **do**
  if $M_k > 0$ **then**
    $\bar{\mathcal{M}}_k^{N_1} \leftarrow \bar{\mathcal{M}}_k^{N_1} \cup \{n\}$
    $M_k \leftarrow M_k - 1$
  end if
  $k \leftarrow (k+1) \bmod K$
end for
#File placement for the first file group $\mathcal{N}_1$
$n \leftarrow 0$
for $k = 1$ to K **do**
  if $M_k > 0$ **then**
    for $m = 1$ to M_k **do**
      $\bar{\mathcal{M}}_k^{N_1} \leftarrow \bar{\mathcal{M}}_k^{N_1} \cup \{(n+m) \bmod N_1\}$
    end for
    $n \leftarrow (M_k + n) \bmod N_1$
  end if
end for

placing them in a round-robin manner.

For a given two-file-group-based placement $\{\bar{\mathcal{M}}_k^{N_1}\}$ computed by Algorithm 9, we redefine $\mathcal{T}$ as a two-file-group-based file placement vector, where the cache placement vector for file $n \in \mathcal{N}$ is given by

$$t_{n,\mathcal{S}} = \begin{cases} 1, & \mathcal{S} = \{k : k \in \mathcal{K}, n \in \bar{\mathcal{M}}_k^{N_1}\} \\ 0, & \text{otherwise.} \end{cases}$$

Also redefine $\bar{L}$ as the expected shuffling load and $\{L_{k,\mathcal{S}}^{\mathcal{D}}, d_{k,\mathcal{S}}^{\mathcal{S}_{-i}}(\mathcal{D})\}$ as the nested coded shuffling strategy for given $\mathbf{t}$, we can convert **P16** into a coded shuffling optimization problem for a given two-file-group base file placement strategy $\{\bar{\mathcal{M}}_k^{N_1}\}$, given by

$$\textbf{P17}: \min_{\{L_{k,\mathcal{S}}^{\mathcal{D}}, d_{k,\mathcal{S}}^{\mathcal{S}_{-i}}(\mathcal{D})\}} \quad \bar{L} \quad \text{s.t. (6.6), (6.11), (6.12).}$$

Note that **P17** is a LP problem with $\sum_{n=1}^{N} \binom{N}{n} \left(K^2 + \sum_{k=3}^{K} k^2 - K \right)$ variables, which can be solved by standard optimization solvers.

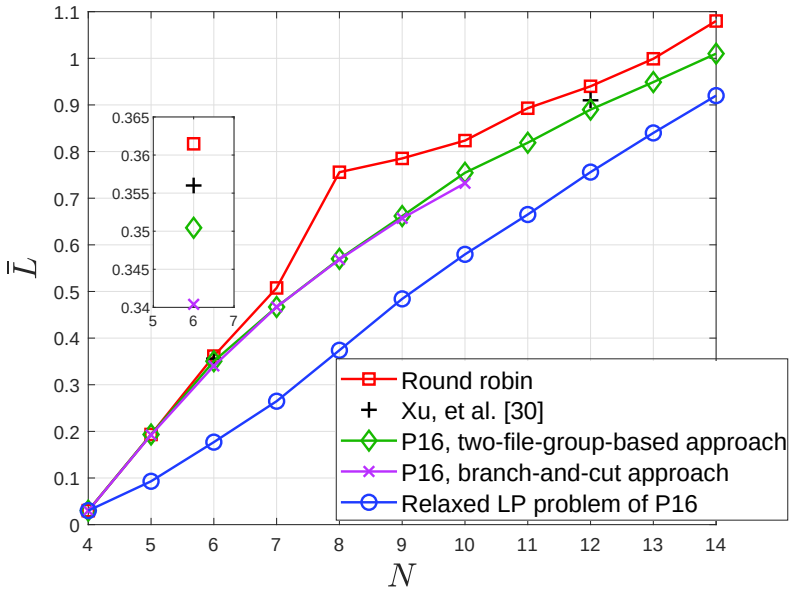

Figure 6.1: Average rate $\bar{R}$ vs. cache size M ($N = 6$, $K = 4$, Zipf file popularity distribution with $\theta = 0.56$).

For a given two-file-group-based placement strategy $\{\bar{\mathcal{M}}_k^{N_1}\}$, **P17** optimizes the coded shuffling strategy to minimize the expected shuffling load. Following this, we can further search for the optimal $N_1 \in \mathcal{N}$ that gives us the minimum shuffling load among all possible two-file-group-based placement strategies $\{\{\bar{\mathcal{M}}_k^{N_1}\}, N_1 \in \mathcal{N}\}$. In Section 6.4, numerical studies show that the optimal two-file-group-based solution has close performance to the conventional branch-and-cut method.

6.4 Numerical Results

In this section, we provide numerical studies on the performance of the proposed heterogeneous CDC scheme in **P16**. We solve **P16** using the proposed two-file-group-based approach described in Section 6.3.1 and the conventional branch-and-cut method. We obtain the two-file-group-based file placement using Algorithm 9. We also compare with the file placement strategy in [30], the round-robin placement strategy commonly used by uncoded distributed computing [60] and the relaxed LP problem of **P16**, which can be considered

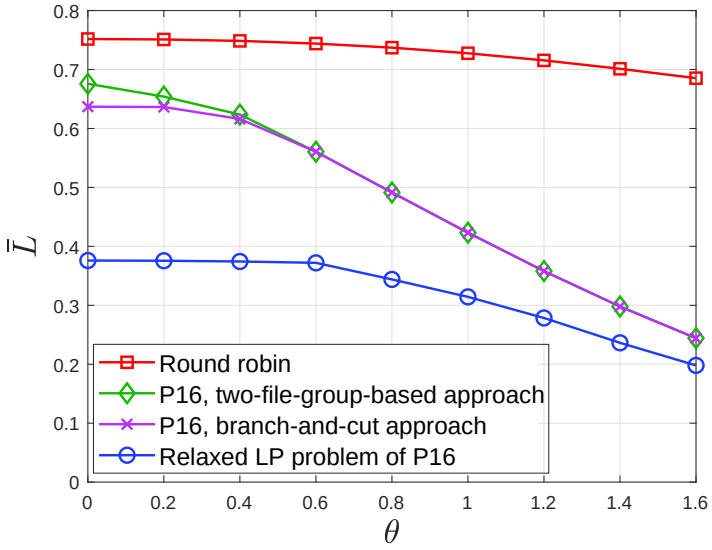

Figure 6.2: Average rate $\bar{R}$ vs. cache size M ($N = 6$, $K = 4$, Zipf file popularity distribution with $\theta = 1.2$).

as a lower bound for the proposed heterogeneous CDC scheme. We generate the file popularities using the Zipf distribution with $p_n = n^{-\theta}/\sum_{i=1}^{N} i^{-\theta}$ where θ is the Zipf parameter. We assume the files popularities are independently distributed and derive the probability of file subset $\mathcal{D}$ being accessed by a job as

$$p_{\mathcal{D}} = \frac{\prod_{n\in\mathcal{D}} p_n \prod_{n\notin\mathcal{D}}(1-p_n)}{1-\prod_{n\in\mathcal{N}}(1-p_n)}, \quad \mathcal{D}\subseteq\mathcal{N}, \mathcal{D}\neq\emptyset.$$

We consider a CDC network of $K = 4$ workers with nonuniform mapping and normalized reducing loads, given by $M = [3, 4, 4, 5]$ and $W = [1/8, 1/4, 1/4, 3/8]$, respectively. We also set the size of the IVs as $T = 1$.

In Fig. 6.1, we show the expected load $\bar{L}$ vs. the file number N for $\theta = 0.56$ (used in [100]). We solve **P16** by the branch-and-cut method only for $N \leq 10$, due to its high computational complexity. From Fig. 6.1, although the branch-and-cut method achieves the lowest $\bar{L}$ among all the schemes, the two-file-group-based approach has very close performance, with slightly higher $\bar{L}$'s observed at specific points. The placement strategy in [30] only works for $N = 6$ and 12, where the $\bar{L}$'s are higher than the proposed scheme. The gap between **P16** and its relaxed LP problem is increasing with N for $N \leq 6$ and remains approximately unchanged for $N > 6$. In Fig. 6.2, we set $N = 8$ and plot $\bar{L}$ vs.

Zipf parameter θ. Comparison different schemes to compute the lower bound in method only exists for $\theta < 0.6$ and is decreasing with θ. This shows that the two-file-group-based placement performs better for larger θ's, *i.e.,* more diverse distributions. When θ increases, the solution of **P16** (by both the two-file-group-based and branch-and-cut approaches) is getting closer to that of its relaxed LP problem. The $\bar{L}$ of the round-robin approach has small changes for different Zipf parameter θ's and is always produce the highest $\bar{L}$.

6.5 Summary

In this chapter, for heterogeneous CDC with an arbitrary number of files with nonuniform popularity, we proposed a file placement strategy and a nested coded shuffling strategy to exploit coded multicasting opportunities. We formulated an optimization problem to jointly optimize the proposed file placement and shuffling strategies. The problem is a generally NP-hard MILP problem and we developed a two-file-group-based approximate approach to solve it. Numerical studies showed that the proposed approximate approach achieves an average load close to that of the conventional branch-and-cut method which has high computational complexity.

Chapter 7

Conclusion and Future Works

In this chapter, summarize the main results in this dissertation and outline some possible future works.

7.1 Coded Caching

In this dissertation, we mainly focused on studying the caching problems for the system of multiple cache-enabled users connected to a single server through a shared error-free link. In Chapter 3, we characterized the optimal cache placement for both the CCS. Using the optimal cache placement structure, we derived a lower bound for caching and characterized the subpacketization in the optimal CCS. In Chapter 4, we studied the memory-rate tradeoff for caching under nonuniform demands. We formulated an optimization problem to optimize the cache placement for the MCCS and characterized the optimal cache placement structure. To characterize the memory-rate tradeoff, we proposed two lower bounds. We compared the optimized MCCS with the lower bounds in three regions and provide insights. decentralized caching under nonuniform demands. In Chapter 5, we studied the memory-rate tradeoff for the decentralized caching paradigm. We formulated cache placement optimization problem to obtain the optimal cache placement for the D-MCCS and compared the optimized D-MCCS with the proposed lower bound to characterize the memory-rate tradeoff.

One possible direction of our future works is to extend our studies of the optimal cache

placement for coded caching to the device-to-device (D2D) network and the interference network with multiple transmitters and users. Existing works on the coded caching for D2D and interference networks mainly focus on uniform demand [21–23, 104–108], we can formulate cache placement optimization problems to study the optimal cache placement of the CCS with nonuniform demands under D2D and interference networks networks. It is also interesting to study the memory-rate tradeoff for caching with nonuniform demands under D2D and interference networks, which remains unknown in the literature. To accomplish this, efforts are needed to develop new lower bounds for caching under the aforementioned network structures and then compare with the optimized coded caching schemes.

7.2 Coded Distributed Computing

In Chapter 6, we investigated the MapReduce based heterogenous CDC for arbitrary number of files with nonuniform popularity. We proposed a general file placement strategy and adopted a nested coded shuffling strategy to exploit extra coded multicasting opportunities. We formulated an optimization problem to optimize the proposed file placement and shuffling strategies. The problem is a generally NP-hard MILP problem and we developed an two-file-group-based approximate approach to solve it. Numerical study showed that the approximate approach performs close to the conventional branch-and-cut method.

The study of heterogeneous CDC is still at an early stage. First of all, the computation-communication tradeoff remains unknown under this more general system setup. One possible future work is to develop a lower bound on the average shuffling load to characterize the computation-communication tradeoff for the heterogeneous CDC with nonuniform file popularity. Other than this, the application of CDC in different scenarios can be considered in the future. One direction is to study the online CDC where the jobs arrive at different times and have different deadlines of being finished. In this case, it is interesting to study the scheduling of the job execution and coded shuffling for CDC. In [87], the wireless CDC has been studied for the homogeneous system where both unlink and downlink communication load have been considered. We can also consider extending our work to the wireless

heterogeneous CDC for an arbitrary number of files with nonuniform popularity.

Appendix A

Appendices for Chapter 3

A.1 Probability Distribution of Y_m

For Y_m being the mth smallest file index in the demand vector $\mathbf{d}$, $m = 1, \ldots, K$, the probability distribution of Y_m is given as follows [33, Lemma 2]:

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

$$\Pr[Y_2 = i] = \Pr[Y_1 = i] + K\Big[\Big(\sum_{l=1}^{i-1} p_l\Big)\Big(\sum_{l=i}^{N} p_l\Big)^{K-1} - \Big(\sum_{l=1}^{i} p_l\Big)\Big(\sum_{l=i+1}^{N} p_l\Big)^{K-1}\Big];$$

For $3 \le m \le K$,

$$\Pr[Y_m = 1] = \sum_{k=0}^{K-m} \binom{K}{m+k} p_1^{m+k} (1-p_1)^{K-m-k},$$

$$\Pr[Y_m = i] = \binom{K}{K-m+1} \left(\Big(\sum_{l=i}^{N} p_l\Big)^{K-m+1} - \Big(\sum_{l=i+1}^{N} p_l\Big)^{K-m+1} \right) \left(\sum_{l=1}^{i-1}\right)^{m-1}$$
$$+ \sum_{k=0}^{K-2} \sum_{b=\max\{0,m-2-k\}}^{\min\{m,K-k\}-2} \left(\frac{K!\, p_i^{2+k}}{(2+k)!\, b!\, (K-2-k-b)!} \left(\sum_{l=1}^{i-1} p_l\right)^b \left(\sum_{l=i+1}^{N} p_l\right)^{K-2-k-b} \right),$$
for $i = 2, \ldots, N$.

A.2 Proof of Theorem 1

We prove Theorem 1 by exploiting the properties in the Karush-Kuhn-Tucker (KKT) conditions for **P2**. The Lagrangian associate with **P2** is given by

$$L = \sum_{n=1}^{N} \mathbf{g}_n^T \mathbf{a}_n - \sum_{n=1}^{N-1} \sum_{l=1}^{K} \gamma_{n,l}(a_{n,l} - a_{n+1,l}) - \sum_{l=1}^{K} \rho_l a_{N,l} - \rho_0 a_{1,0} + \lambda(\sum_{n=1}^{N} \mathbf{c}^T \mathbf{a}_n - M) + \sum_{n=1}^{N} \nu_n(\mathbf{b}^T \mathbf{a}_n - 1) \tag{A.1}$$

where $\{\gamma_{n,l}\}$ are the Lagrange multipliers for constraint (3.8), $\{\rho_0, \ldots, \rho_K\}$ are the Lagrange multipliers for constraints in (3.10), $\{\nu_n\}$ are the Lagrange multipliers for constraint (3.16), and λ is the Lagrange multiplier for constraint (3.3). Since **P2** is an LP, the KKT conditions hold for **P2**, which are listed below:

$$\mathbf{b}^T \mathbf{a}_n = 1, \tag{A.2}$$

$$\sum_{n=1}^{N} \mathbf{c}^T \mathbf{a}_n = M, \tag{A.3}$$

$$a_{N,l} \geq 0, \quad l \in \mathcal{K}, \tag{A.4}$$

$$\rho_l \cdot a_{N,l} = 0, \rho_l \geq 0, \quad l \in \mathcal{K}, \tag{A.5}$$

$$a_{1,0} \geq 0, \tag{A.6}$$

$$\rho_0 \cdot a_{1,0} = 0, \tag{A.7}$$

$$a_{n,l} - a_{n+1,l} \geq 0, \quad n \in \mathcal{N}\backslash\{N\}, l \in \mathcal{K}, \tag{A.8}$$

$$\gamma_{n,l}(a_{n,l} - a_{n+1,l}) = 0, \gamma_{n,l} \geq 0, \quad n \in \mathcal{N}\backslash\{N\}, l \in \mathcal{K} \tag{A.9}$$

$$\frac{\partial L}{\partial a_{n,l}} = g_{n,l} - \gamma_{n,l} + \gamma_{n-1,l} + \lambda c_l + \nu_n b_l = 0, \quad n \in \mathcal{N}\backslash\{1, N\}, l \in \mathcal{K} \tag{A.10}$$

$$\frac{\partial L}{\partial a_{1,l}} = g_{1,l} - \gamma_{1,l} + \lambda c_l + \nu_1 b_l = 0, \quad l \in \mathcal{K} \tag{A.11}$$

$$\frac{\partial L}{\partial a_{N,l}} = g_{N,l} - \rho_l + \gamma_{N-1,l} + \lambda c_l + \nu_N b_l = 0, \tag{A.12}$$

$$\frac{\partial L}{\partial a_{n,0}} = g_{n,0} + \lambda c_0 + \nu_n b_0 = 0, \quad n \in \mathcal{N}\backslash\{1\}, \tag{A.13}$$

$$\frac{\partial L}{\partial a_{1,0}} = g_{1,0} - \rho_0 + \lambda c_0 + \nu_1 b_0 = 0. \tag{A.14}$$

From (A.10) and (A.11), we have, for $m = 1, \ldots, N-1$,

$$\sum_{n=1}^{m} \frac{\partial L}{\partial a_{n,l}} = \sum_{n=1}^{m} g_{n,l} - \gamma_{m,l} + m\lambda c_l + \sum_{n=1}^{m} \nu_n b_l = 0. \tag{A.15}$$

Based on the above KKT conditions, we prove Theorem 1 by contradiction. Assume that there exists an optimal solution $\{\mathbf{a}_n\}$ that divides the files into four file groups. The structure of the sub-placement vectors $\bar{\mathbf{a}}_n$'s can be expressed as $\bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_1+1} = \ldots = \bar{\mathbf{a}}_{n_2} \succcurlyeq_1 \bar{\mathbf{a}}_{n_2+1} = \ldots = \bar{\mathbf{a}}_N$, for $1 \leq n_o < n_1 < n_2 \leq N-1$. By the property in (3.8), we assume $a_{n_o,l_o} > a_{n_o+1,l_o}$, $a_{n_1,l_1} > a_{n_1+1,l_1}$ and $a_{n_2,l_2} > a_{n_2+1,l_2}$, for some $l_o, l_1, l_2 \in \mathcal{K}$. From (A.9), we have

$$\gamma_{n_o,l_o} = \gamma_{n_1,l_1} = \gamma_{n_2,l_2} = 0. \tag{A.16}$$

Since $c_0 = 0$ and $b_0 = 1$, from (A.13), we have

$$\nu_n = -g_{n,0}, \quad n \in \mathcal{N}\backslash\{1\}. \tag{A.17}$$

Following (A.15), let $m = n_o, n_1, n_2$, we have

$$\sum_{n=1}^{n_o} g_{n,l_o} - \gamma_{n_o,l_o} + n_o \lambda c_{l_o} + \sum_{n=1}^{n_o} \nu_n b_{l_o} = 0, \tag{A.18}$$

$$\sum_{n=1}^{n_1} g_{n,l_1} - \gamma_{n_1,l_1} + n_1 \lambda c_{l_1} + \sum_{n=1}^{n_1} \nu_n b_{l_1} = 0, \tag{A.19}$$

$$\sum_{n=1}^{n_2} g_{n,l_2} - \gamma_{n_2,l_2} + n_2 \lambda c_{l_2} + \sum_{n=1}^{n_2} \nu_n b_{l_2} = 0. \tag{A.20}$$

Substituting the values of $\gamma_{n_o,l_o}, \gamma_{n_1,l_1}, \gamma_{n_2,l_2}$ in (A.16) and ν_n in (A.17) into (A.18) - (A.20), we have

$$\lambda n_o c_{l_o} + \nu_1 b_{l_o} = -\sum_{i=2}^{n_o} g_{n,0} b_{l_o} - \sum_{n=1}^{n_o} g_{n,l_o}, \tag{A.21}$$

$$\lambda n_1 c_{l_1} + \nu_1 b_{l_1} = -\sum_{i=2}^{n_1} g_{n,0} b_{l_1} - \sum_{n=1}^{n_1} g_{n,l_1}, \tag{A.22}$$

$$\lambda n_2 c_{l_2} + \nu_1 b_{l_2} = -\sum_{i=2}^{n_2} g_{n,0} b_{l_2} - \sum_{n=1}^{n_2} g_{n,l_2}. \tag{A.23}$$

We rewrite (A.21) - (A.23) into a matrix form $\mathbf{A}\mathbf{x} = \mathbf{b}$ as

$$\begin{bmatrix} n_o c_{l_o} & b_{l_o} \\ n_1 c_{l_1} & b_{l_1} \\ n_2 c_{l_2} & b_{l_2} \end{bmatrix} \begin{bmatrix} \lambda \\ \nu_1 \end{bmatrix} = \begin{bmatrix} -\sum_{n=2}^{n_o} g_{n,0} b_{l_o} - \sum_{n=1}^{n_o} g_{n,l_o} \\ -\sum_{n=2}^{n_1} g_{n,0} b_{l_1} - \sum_{n=1}^{n_1} g_{n,l_1} \\ -\sum_{n=2}^{n_2} g_{n,0} b_{l_2} - \sum_{n=1}^{n_2} g_{n,l_2} \end{bmatrix}. \tag{A.24}$$

Note that $n_o \neq n_1 \neq n_2$, and by the definition of c_l and b_l below (3.15), the 3×2 coefficient matrix $\mathbf{A}$ in (A.24) is full rank, there is no feasible solution for λ, ν_1. This contradicts the assumption that there exists an optimal $\{\mathbf{a}_n\}$ with four file groups. A similar argument follows to show more than four file groups is not possible. Thus, we have the conclusion in Theorem 1.

A.3 Proof of Proposition 1

With two file groups, the sub-placement vectors have the following relation: $\bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_N$, for some $n_o \in \{1, \ldots, N-1\}$. Since there is at least one element that is different between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$, by (3.8), we assume $a_{n_o,l_o} > a_{n_o+1,l_o}$, for some $l_o \in \mathcal{K}$. Consequently, we have $\gamma_{n_o,l_o} = 0$ based on (A.9). From (A.15), we have

$$\sum_{n=1}^{n_o} g_{n,l} + n_o \lambda c_{l_o} + \sum_{n=1}^{n_o} \nu_n b_{l_o} = 0. \tag{A.25}$$

From (A.10)–(A.12), we have

$$\sum_{n=1}^{N} g_{n,l} - \rho_l + N\lambda c_l + \sum_{n=1}^{N} \nu_n b_l = 0, \quad l \in \mathcal{K}. \tag{A.26}$$

Assume $\bar{\mathbf{a}}_N$ has two nonzero elements at the l_1th and l_2th locations, *i.e.,* $a_{N,l_1} > 0$, $a_{N,l_2} > 0$, for $l_1 \neq l_2$, $l_1, l_2 \in \mathcal{K}$. Note that one of l_1 and l_2 can be l_o. Without loss of generality, we assume $l_2 \neq l_o$. We know from (A.5) that $\rho_{l_1} = \rho_{l_2} = 0$. Then, from (A.26), we have

$$\sum_{n=1}^{N} g_{n,l_1} + N\lambda c_{l_1} + \sum_{n=1}^{N} \nu_n b_{l_1} = 0, \tag{A.27}$$

$$\sum_{n=1}^{N} g_{n,l_2} + N\lambda c_{l_2} + \sum_{n=1}^{N} \nu_n b_{l_2} = 0. \tag{A.28}$$

Using the expression of ν_n in (A.17), we can rewrite (A.25)(A.27)(A.28) into a matrix form as

$$\begin{bmatrix} n_o c_{l_o} & b_{l_o} \\ N c_{l_1} & b_{l_1} \\ N c_{l_2} & b_{l_2} \end{bmatrix} \begin{bmatrix} \lambda \\ \nu_1 \end{bmatrix} = \begin{bmatrix} -\sum_{n=2}^{n_o} g_{n,0} b_{l_o} - \sum_{n=1}^{n_o} g_{n,l_o} \\ -\sum_{n=2}^{N} g_{n,0} b_{l_1} - \sum_{n=1}^{N} g_{n,l_1} \\ -\sum_{n=2}^{N} g_{n,0} b_{l_2} - \sum_{n=1}^{N} g_{n,l_2} \end{bmatrix}. \tag{A.29}$$

Similar to the argument in the proof of Theorem 1, since $n_o < N$, $l_2 \neq l_1$, $l_2 \neq l_o$, by the definition of c_l and b_l, the coefficient matrix of (A.29) is full rank, and there is no feasible solution for λ and ν_1, contradicting the assumption that the optimal $\bar{\mathbf{a}}_N$ has two nonzero elements. Similarly, we show the optimal $\bar{\mathbf{a}}_N$ cannot have more than two nonzero elements. Thus, we complete the proof.

A.4 Proof of Proposition 2

Since the optimal cache placement solution result in two file groups, the sub-placement vectors have the following structure: $\bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_N \succcurlyeq_1 \mathbf{0}$, for some

$n_o \in \{1, \ldots, N-1\}$. By Proposition 1, $\bar{\mathbf{a}}_{n_o+1}$ has only one nonzero element. Assume $a_{n_o+1,l_o} > 0$, for some $l_o \in \mathcal{K}$. From (A.5), we know that $\rho_{l_o} = 0$. Then from (A.26), we have

$$\sum_{n=1}^{N} g_{n,l_o} + \lambda N c_{l_o} + \sum_{n=1}^{N} \nu_n b_{l_o} = 0. \tag{A.30}$$

Assume that there are two elements in $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ being different: $a_{n_o,l_1} > a_{n_o+1,l_1}$ and $a_{n_o,l_2} > a_{n_o+1,l_2}$, for $l_1 \neq l_2$, $l_1, l_2 \in \mathcal{K}$. Without loss of generality, we assume $l_2 \neq l_o$. From (A.9), we have $\gamma_{n_o,l_1} = \gamma_{n_o,l_2} = 0$. As a result, from (A.15), we have

$$\sum_{n=1}^{n_o} g_{n,l_1} + \lambda n_o c_{l_1} + \sum_{n=1}^{n_o} \nu_n b_{l_1} = 0, \tag{A.31}$$

$$\sum_{n=1}^{n_o} g_{n,l_2} + \lambda n_o c_{l_2} + \sum_{n=1}^{n_o} \nu_n b_{l_2} = 0. \tag{A.32}$$

Again, using (A.17), we put (A.30)–(A.32) in a matrix form as

$$\begin{bmatrix} N c_{l_o} & b_{l_o} \\ n_o c_{l_1} & b_{l_1} \\ n_o c_{l_2} & b_{l_2} \end{bmatrix} \begin{bmatrix} \lambda \\ \nu_1 \end{bmatrix} = \begin{bmatrix} -\sum_{n=2}^{N} g_{n,0} b_{l_o} - \sum_{n=1}^{N} g_{n,l_o} \\ -\sum_{n=2}^{n_o} g_{n,0} b_{l_1} - \sum_{n=1}^{n_o} g_{n,l_1} \\ -\sum_{n=2}^{n_o} g_{n,0} b_{l_2} - \sum_{n=1}^{n_o} g_{n,l_2} \end{bmatrix}.$$

By the similar argument in the proof of Proposition 1, the coefficient matrix of (A.30)–(A.32) is full rank, and λ and ν_1 do not have any feasible solution, contradicting the assumption that $\bar{\mathbf{a}}_N$ has two nonzero elements. Similarly, we can proof that $\bar{\mathbf{a}}_N$ cannot have more than two nonzero elements.

A.5 Proof of Proposition 3

With two file groups, the sub-placement vectors have the following structure: $\bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_N$, for some $n_o \in \{1, \ldots, N-1\}$. Assume $\bar{a}_{n_o+1,l_o} > 0$, for $l_o \in \mathcal{K}$. By Proposition 2, only one element is different between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$. This element can be either at l_o, *i.e.,* $a_{n_o,l_o} > a_{n_o+1,l_o}$ (as shown in Fig. 3.5), or any $l_1 \neq l_o$, $l_1 \in \mathcal{K}$, *i.e.,* $a_{n_o,l_1} > a_{n_o+1,l_1}$ (as shown in Fig. 3.6). We discuss the two cases separately.

$a_{n_o,l_o} > a_{n_o+1,l_o} > 0$

If $a_{1,0} = \ldots = a_{n_o,0} > 0$, by (A.7) we have $\rho_0 = 0$. Combining this with (A.14), we have $g_{1,0} + \lambda c_0 + \nu_1 b_0 = 0$, which gives

$$\nu_1 = -g_{1,0}. \tag{A.33}$$

By (A.9), since $a_{n_o,l_o} > a_{n_o+1,l_o}$, we have $\gamma_{n_o,l_o} = 0$. Substituting the expression of ν_n in (A.17) and (A.33) into (A.15), we have

$$\lambda n_o c_{l_o} = -\sum_{n=1}^{n_o} g_{n,l_o} - \sum_{n=1}^{n_o} g_{n,0} b_{l_o}. \tag{A.34}$$

Combining (A.10) and (A.12), we have

$$\begin{aligned}\sum_{n=n_o+1}^{N} \frac{\partial L}{\partial a_{n,l_o}} &= \sum_{n=n_o+1}^{N} g_{n,l_o} - \rho_{l_o} + \gamma_{n_o,l_o} + (N-n_o)\lambda c_{l_o} + \sum_{n=n_o+1}^{N} \nu_n b_{l_o} \\ &= 0.\end{aligned} \tag{A.35}$$

For $a_{n_o+1,l_o} = a_{N,l_o} > 0$, from (A.5), we have $\rho_{l_o} = 0$. Along with $\gamma_{n_o,l_o} = 0$, (A.35) can be rewritten as

$$\lambda(N-n_o)c_{l_o} = -\sum_{n=n_o+1}^{N} g_{n,l_o} - \sum_{n=n_o+1}^{N} g_{n,0} b_{l_o}. \tag{A.36}$$

Examining (A.34) and (A.36), we see that there is no feasible solution for λ to satisfy both equations. This contradicts the assumption that $a_{1,0} = \ldots = a_{n_o,0} > 0$.

$a_{n_o,l_1} > a_{n_o+1,l_1} = 0$**, for** $l_1 \neq l_o$

From (A.9), we have $\gamma_{n_o,l_1} = 0$. Assuming $a_{1,0} = \ldots = a_{n_o,0} > 0$, we have (A.33). Similar to (A.34), we have

$$\lambda n_o c_{l_1} = -\sum_{n=1}^{n_o} g_{n,l_1} - \sum_{n=1}^{n_o} g_{n,0} b_{l_1}. \tag{A.37}$$

For $a_{n_o+1,l_o} = a_{N,l_o} > 0$, again we have $\rho_{l_o} = 0$, and ν_n in (A.17) and (A.33). Thus, from (A.26), we have

$$\lambda N c_{l_o} = -\sum_{n=2}^{N} g_{n,0} b_{l_o} - \sum_{n=1}^{N} g_{n,l_o}. \tag{A.38}$$

Again, there is no feasible solution for λ to satisfy both (A.37) and (A.38). This contradicts the assumption that $a_{1,0} = \ldots = a_{n_o,0} > 0$.

From both two cases above, we conclude if $a_{1,0} = \ldots = a_{n_o,0} > 0$, there is no feasible solution for λ and ν_1. Thus, we have $a_{1,0} = \ldots = a_{n_o,0} = 0$ for the optimal $\{\mathbf{a}_n\}$.

A.6 Proof of Proposition 4

With three file groups, the sub-placement vectors have the following structure: $\bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_1+1} = \ldots = \bar{\mathbf{a}}_N$, for $1 \leq n_o < n_1 \leq N-1$. Assume that $a_{n_o,l_o} > a_{n_o+1,l_o}$ and $a_{n_1,l_1} > a_{n_1+1,l_1}$, for $l_o, l_1 \in \mathcal{K}$. From (A.9), we have $\gamma_{n_o,l_o} = \gamma_{n_1,l_1} = 0$. Substitute the value of ν_n in (A.17) into (A.15), we have the following

$$\lambda n_o c_{l_o} + \nu_1 b_{l_o} = -\sum_{n=2}^{n_o} g_{n,0} b_{l_o} - \sum_{n=1}^{n_o} g_{n,l_o}, \tag{A.39}$$

$$\lambda n_1 c_{l_1} + \nu_1 b_{l_1} = -\sum_{n=2}^{n_1} g_{n,0} b_{l_1} - \sum_{n=1}^{n_1} g_{n,l_1}. \tag{A.40}$$

To show $\bar{\mathbf{a}}_{n_1+1} = \mathbf{0}$ by contradiction, assume that $\bar{\mathbf{a}}_N \succcurlyeq_1 \mathbf{0}$, *i.e.,* $\bar{\mathbf{a}}_{n_1+1} = \ldots = \bar{\mathbf{a}}_N$ has at least one nonzero element. Let $a_{N,l_2} > 0$ for some $l_2 \in \mathcal{K}$. Then, we have $\rho_{l_2} = 0$ by (A.5). Then, from (A.26), we have

$$N\lambda c_{l_2} + \nu_1 b_{l_2} = -\sum_{n=1}^{N} g_{n,l_2} - \sum_{n=2}^{N} g_{n,0} b_{l_2}. \tag{A.41}$$

Putting (A.39)–(A.41) into a matrix form, we have

$$\begin{bmatrix} n_o c_{l_o} & b_{l_o} \\ n_1 c_{l_1} & b_{l_1} \\ N c_{l_2} & b_{l_2} \end{bmatrix} \cdot \begin{bmatrix} \lambda \\ \nu_1 \end{bmatrix} = \begin{bmatrix} -\sum_{n=2}^{n_o} g_{n,0} b_{l_o} - \sum_{n=1}^{n_o} g_{n,l_o} \\ -\sum_{n=2}^{n_1} g_{n,0} b_{l_1} - \sum_{n=1}^{n_1} g_{n,l_1} \\ -\sum_{n=2}^{N} g_{n,0} b_{l_2} - \sum_{n=1}^{N} g_{n,l_2} \end{bmatrix}.$$

Using a similar argument as in the proof of Theorem 1, we conclude that λ and ν_1 do not have any feasible solution, which contradicts the assumption that $\bar{\mathbf{a}}_N \succcurlyeq_1 \mathbf{0}$. Thus, we complete the proof.

Appendix B

Appendices for Chapter 4

B.1 Proof of Lemma 2

The proof directly follows the proof of [52, Theorem 2] with a slight modification. In the proof of [52, Theorem 2], by a genie-based method, it is shown that the delivery rate for a distinct file set $\mathcal{D}$ satisfies

$$R(\mathcal{D}) \geq \max_{\pi:\mathcal{I}_{|\mathcal{D}|}\to\mathcal{D}} \sum_{l=0}^{K} \sum_{i=1}^{\tilde{N}(\mathbf{d})} \binom{K-i}{l} \frac{\check{a}_{n,l}}{\binom{K}{l}} \tag{B.1}$$

where $\check{a}_{n,l}$ is the total number of bits from file $n \in \mathcal{D}$ cached by exactly l users. In our definition of $a_{n,l}$, subscript l refers to the user subset size $|\mathcal{S}| = l$ in a cache subgroup $\mathcal{A}^l$. Thus, they are identical. Following the definitions of $\check{a}_{n,l}$ and $a_{n,l}$, we have $a_{n,l} = \check{a}_{n,l}/\binom{K}{l}$, since there are $\binom{K}{l}$ user subsets in $\mathcal{A}^l$. Thus, for a given $\mathbf{a}$, we can equivalently express (B.1) as

$$R(\mathcal{D}) \geq \max_{\pi:\mathcal{I}_{|\mathcal{D}|}\to\mathcal{D}} \sum_{l=0}^{K} \sum_{i=1}^{\tilde{N}(\mathbf{d})} \binom{K-i}{l} a_{n,l} \triangleq R_{\text{lb}}(\mathcal{D};\mathbf{a}). \tag{B.2}$$

Using the above expression, by averaging $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ over all possible $\mathcal{D} \subseteq \mathcal{N}$, we obtain the general lower bound $\bar{R}_{\text{lb}}(\mathbf{a})$ the average rate w.r.t $\mathbf{a}$ in (4.11). The final lower bound on average rate is obtained by optimizing $\mathbf{a}$ to minimize $\bar{R}_{\text{lb}}(\mathbf{a})$, which is shown in **P1**.

B.2 Proof of Lemma 3

The popularity-first-based lower bound is essentially a simplification of **P1** in Lemma 2, by restricting to the set of popularity-first placement vectors: $\mathbf{a} \in \mathcal{Q}$. We need to show that $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.12) can be simplified into (4.14) for $\mathbf{a} \in \mathcal{Q}$. For $\forall \mathcal{D} \subseteq \mathcal{N}$, since $\phi(\cdot)$ is

such that $p_{\phi(1)} \geq \cdots \geq p_{\phi(|\mathcal{D}|)}$, by the definition of popularity-first placement in (4.4), for $\mathbf{a} \in \mathcal{Q}$, we have

$$a_{\phi(i),l} \geq a_{\phi(i+1),l}, \quad l \in \mathcal{K},\ i = 1, \ldots, \tilde{N}(\mathbf{d}) - 1. \tag{B.3}$$

Since $\binom{K-i}{l}$ is a decreasing function of i, for $\mathbf{a} \in \mathcal{Q}$, we have

$$\max_{\pi:\mathcal{I}_{|\mathcal{D}|}\to\mathcal{D}} \sum_{l=0}^{K}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\binom{K-i}{l} a_{\pi(i),l} = \sum_{l=0}^{K}\sum_{i=1}^{\tilde{N}(\mathbf{d})}\binom{K-i}{l} a_{\phi(i),l}.$$

Thus, we remove the max operation in (4.12) to arrive at the simplified expression in (4.14), for $\mathbf{a} \in \mathcal{Q}$.

B.3 Proof of Theorem 2

For **P1**, consider a feasible cache placement vector $\hat{\mathbf{a}}$ and any $l_o \in \mathcal{K}$. Define $\varphi : \mathcal{I}_{|\mathcal{N}|} \to \mathcal{N}$ as a bijective map for $\hat{\mathbf{a}}$ such that $\hat{a}_{\varphi(1),l_o} \geq \ldots \geq \hat{a}_{\varphi(N),l_o}$. Note that $\varphi(\cdot)$ depends on $\hat{\mathbf{a}}$ and l_o.

Assume $p_{\varphi(i_o)} < p_{\varphi(i_o+1)}$, for some $i_o \in \mathcal{I}_{|\mathcal{N}|}\backslash\{N\}$. We construct another feasible cache placement vector $\tilde{\mathbf{a}}$ using $\hat{\mathbf{a}}$ by switching the values of $\hat{a}_{\varphi(i_o),l_o}$ and $\hat{a}_{\varphi(i_o+1),l_o}$. Specifically,
i) for $l \in \mathcal{K}$, we have

$$\begin{cases} \tilde{a}_{\varphi(i_o),l_o} = \hat{a}_{\varphi(i_o+1),l_o} \\ \tilde{a}_{\varphi(i_o+1),l_o} = \hat{a}_{\varphi(i_o),l_o} \\ \tilde{a}_{\varphi(i),l} = \hat{a}_{\varphi(i),l}, \quad i \neq i_o, i \in \mathcal{I}_{|\mathcal{N}|}; \end{cases} \tag{B.4}$$

ii) for $l = 0$, by (B.4) and file partition constraint (4.2), we have

$$\begin{cases} \tilde{a}_{\varphi(i_o),0} = 1 - \sum_{l\in\mathcal{K}\backslash\{l_o\}}\binom{K}{l}\hat{a}_{\varphi(i_o),l} - \binom{K}{l_o}\hat{a}_{\varphi(i_o+1),l_o} \\ \tilde{a}_{\varphi(i_o+1),0} = 1 - \sum_{l\in\mathcal{K}\backslash\{l_o\}}\binom{K}{l}\hat{a}_{\varphi(i_o+1),l} - \binom{K}{l_o}\hat{a}_{\varphi(i_o),l_o} \\ \tilde{a}_{\varphi(i),0} = \hat{a}_{\varphi(i),0}, \quad i \neq i_o, i_o+1,\ i \in \mathcal{I}_{|\mathcal{N}|}. \end{cases} \tag{B.5}$$

From (B.4), we have

$$\begin{aligned} &\tilde{a}_{\varphi(1),l_o} \geq \cdots \geq \tilde{a}_{\varphi(i_o),l_o}, \quad \tilde{a}_{\varphi(i_o+1),l_o} \geq \cdots \geq \tilde{a}_{\varphi(N),l_o}, \\ &\tilde{a}_{\varphi(i_o+1),l_o} \geq \tilde{a}_{\varphi(i_o),l_o}. \end{aligned} \tag{B.6}$$

From (B.5) and (4.2), we conclude that

$$\hat{a}_{\varphi(i_o),0} + \hat{a}_{\varphi(i_o+1),0} = \tilde{a}_{\varphi(i_o),0} + \tilde{a}_{\varphi(i_o+1),0} \tag{B.7}$$

$$\hat{a}_{\varphi(i_o),0} - \tilde{a}_{\varphi(i_o),0} = \binom{K}{l_o}(\hat{a}_{\varphi(i_o+1),l_o} - \hat{a}_{\varphi(i_o),l_o}) \tag{B.8}$$

$$\hat{a}_{\varphi(i_o+1),0} - \tilde{a}_{\varphi(i_o+1),0} = \binom{K}{l_o}(\hat{a}_{\varphi(i_o),l_o} - \hat{a}_{\varphi(i_o+1),l_o}). \tag{B.9}$$

Now, we show that $\bar{R}_{\text{lb}}(\hat{\mathbf{a}}) \geq \bar{R}_{\text{lb}}(\tilde{\mathbf{a}})$.

For $K = 2$, we have $\tilde{N}(\mathbf{d}) = |\mathcal{D}| \leq 2$. Define $\xi : \mathcal{I}_{|\mathcal{D}|} \to \mathcal{D}$ as a bijective map for $\mathbf{a}$ such that $a_{\xi(1),1} \geq a_{\xi(|\mathcal{D}|),1}$. Note that $\xi(\cdot)$ depends on $\mathbf{a}$. Then, $R_{\text{lb}}(\mathcal{D}; \mathbf{a})$ in (4.12) is given by

$$R_{\text{lb}}(\mathcal{D}; \mathbf{a}) = \max_{\pi:\mathcal{I}_{|\mathcal{D}|}\to\mathcal{D}} \Big\{ \sum_{i=1}^{|\mathcal{D}|} a_{\pi(i),0} + a_{\pi(1),1} \Big\} = \sum_{i=1}^{|\mathcal{D}|} a_{\xi(i),0} + a_{\xi(1),1} \tag{B.10}$$

Given $\hat{\mathbf{a}}$, the set $\mathcal{D}$ of distinct file indexes in demand vector $\mathbf{d}$ can be categorized into the following four types:

1. $\widetilde{\mathcal{D}}_{1,j} \triangleq \{\mathcal{D} \subseteq \mathcal{N} : \varphi(i_o) \in \mathcal{D}, \varphi(i_o + 1) \notin \mathcal{D}, \varphi(i_o) = \xi(j)\}$, for $j = 1, 2$.[1]
2. $\widetilde{\mathcal{D}}_{2,j} \triangleq \{\mathcal{D} \subseteq \mathcal{N} : \varphi(i_o + 1) \in \mathcal{D}, \varphi(i_o) \notin \mathcal{D}, \varphi(i_o + 1) = \xi(j)\}$, for $j = 1, 2$.
3. $\widetilde{\mathcal{D}}_3 \triangleq \{\ \{\varphi(i_o), \varphi(i_o + 1)\}\ \}$.
4. $\widetilde{\mathcal{D}}_4 \triangleq \{\mathcal{D} \subseteq \mathcal{N} \backslash \{\varphi(i_o), \varphi(i_o + 1)\}\ \}$.

Note that for any $\mathcal{D}$, its type is the same for $\hat{\mathbf{a}}$ and $\tilde{\mathbf{a}}$. To see this, consider $\mathcal{D} = \{\varphi(i_o), n'\}$, where $n' \in \mathcal{N}\backslash\{\varphi(i_o), \varphi(i_o + 1)\}$. Assume $\hat{a}_{\varphi(i_o),1} \geq \hat{a}_{n',1}$. Then for $\hat{\mathbf{a}}$, we have $\xi(1) = \varphi(i_o)$, and $\mathcal{D} \in \widetilde{\mathcal{D}}_{1,1}$. For $\tilde{\mathbf{a}}$, from (B.6), we also have $\tilde{a}_{\varphi(i_o),1} \geq \tilde{a}_{n',1}$. Thus, for the mapping $\xi(\cdot)$ for $\tilde{\mathbf{a}}$, we have $\xi(1) = \varphi(i_o)$, and in this case, we again have $\mathcal{D} \in \widetilde{\mathcal{D}}_{1,1}$. All other types of $\mathcal{D}$ can be verified using the similar argument.

Based on the above four categories of $\mathcal{D}$, we rewrite $\bar{R}_{\text{lb}}(\mathbf{a})$ in (4.11) as

$$\bar{R}_{\text{lb}}(\mathbf{a}) = \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{\mathcal{D}\in\widetilde{\mathcal{D}}_{i,j}}\sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})} p_{d_1}p_{d_2}R_{\text{lb}}(\mathcal{D}; \mathbf{a}) + \sum_{\mathcal{D}\in\widetilde{\mathcal{D}}_3\cup\widetilde{\mathcal{D}}_4}\sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})} p_{d_1}p_{d_2}R_{\text{lb}}(\mathcal{D}; \mathbf{a}).$$

Following the above, we have

$$\bar{R}_{\text{lb}}(\hat{\mathbf{a}}) - \bar{R}_{\text{lb}}(\tilde{\mathbf{a}}) = \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{\mathcal{D}\in\widetilde{\mathcal{D}}_{i,j}}\sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})} p_{d_1}p_{d_2}(R_{\text{lb}}(\mathcal{D}; \hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D}; \tilde{\mathbf{a}}))$$

[1] Set $\widetilde{\mathcal{D}}_{1,j}$ corresponds to the case where file $\varphi(i_o)$ is requested and $a_{\varphi(i_o),1}$ is ranked the jth in $\xi(\cdot)$.

$$+ \sum_{\mathcal{D}\in\widetilde{\mathcal{D}}_3\cup\widetilde{\mathcal{D}}_4}\sum_{\mathbf{d}\in\mathcal{T}(\mathcal{D})} p_{d_1}p_{d_2}(R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}})). \tag{B.11}$$

We now evaluate the differences between $R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}})$ and $R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}})$, for $l_o = 1, 2$.

Case 1: $l_o = 1$**.** We express $R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}})$ in (B.10) based on the types of $\mathcal{D}$. If $\mathcal{D} \in \widetilde{\mathcal{D}}_{1,1}$, then $\varphi(i_o) = \xi(1)$, and we have

$$R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) = \begin{cases} \hat{a}_{\varphi(i_o),0} + \hat{a}_{\varphi(i_o),1}, & |\mathcal{D}| = 1 \\ \hat{a}_{\varphi(i_o),0} + \hat{a}_{\xi(2),0} + \hat{a}_{\varphi(i_o),1}, & |\mathcal{D}| = 2, \end{cases} \tag{B.12}$$

or more compactly, we can express (B.12) as follows

$$R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) = \hat{a}_{\varphi(i_o),0} + s\cdot\hat{a}_{\xi(2),0} + \hat{a}_{\varphi(i_o),1},\ \mathcal{D}\in\widetilde{\mathcal{D}}_{1,1} \tag{B.13}$$

where $s \in \{0,1\}$ is an indicator defined by $s = \{0 : \text{if } |\mathcal{D}| = 1; 1 : \text{if } |\mathcal{D}| = 2\}$. Similarly, for any other types of $\mathcal{D}$, we can always rewrite $R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}})$ in (B.10) as in (B.13) by replacing $\xi(1)$ and $\xi(2)$ with $\varphi(i_o)$ and $\varphi(i_o+1)$, given as follows

$$R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) = \begin{cases} \hat{a}_{\varphi(i_o),0} + s\cdot\hat{a}_{\xi(2),0} + \hat{a}_{\varphi(i_o),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{1,1} \\ \hat{a}_{\xi(1),0} + \hat{a}_{\varphi(i_o),0} + \hat{a}_{\xi(1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{1,2} \\ \hat{a}_{\varphi(i_o+1),0} + s\cdot\hat{a}_{\xi(2),0} + \hat{a}_{\varphi(i_o+1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{2,1} \\ \hat{a}_{\xi(1),0} + \hat{a}_{\varphi(i_o+1),0} + \hat{a}_{\xi(1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{2,2} \\ \hat{a}_{\varphi(i_o),0} + \hat{a}_{\varphi(i_o+1),0} + \hat{a}_{\varphi(i_o),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{3} \\ \hat{a}_{\xi(1),0} + s\cdot\hat{a}_{\xi(2),0} + \hat{a}_{\xi(1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{4} \end{cases} \tag{B.14}$$

where the second and fourth cases are only for $|\mathcal{D}| = 2$.

Similar to $R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}})$ in (B.14), we can rewrite $R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}})$ in (B.10) as follows

$$R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}}) = \begin{cases} \tilde{a}_{\varphi(i_o),0} + s\cdot\tilde{a}_{\xi(2),0} + \tilde{a}_{\varphi(i_o),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{1,1} \\ \tilde{a}_{\xi(1),0} + \tilde{a}_{\varphi(i_o),0} + \tilde{a}_{\xi(1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{1,2} \\ \tilde{a}_{\varphi(i_o+1),0} + s\cdot\tilde{a}_{\xi(2),0} + \tilde{a}_{\varphi(i_o+1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{2,1} \\ \tilde{a}_{\xi(1),0} + \tilde{a}_{\varphi(i_o+1),0} + \tilde{a}_{\xi(1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{2,2} \\ \tilde{a}_{\varphi(i_o),0} + \tilde{a}_{\varphi(i_o+1),0} + \tilde{a}_{\varphi(i_o+1),1}, & \mathcal{D}\in\widetilde{\mathcal{D}}_{3} \\ \tilde{a}_{\xi(1),0} + s\cdot\tilde{a}_{\xi(2),0} + \tilde{a}_{\xi(1),1} & Dc\in\widetilde{\mathcal{D}}_{4}. \end{cases} \tag{B.15}$$

Comparing (B.14) and (B.15), we note that the only difference is the case of $\mathcal{D} \in \widetilde{\mathcal{D}}_3$, where $\hat{a}_{\xi(1),1} = \hat{a}_{\varphi(i_o),1}$, while $\tilde{a}_{\xi(1),1} = \tilde{a}_{\varphi(i_o+1),1}$ by (B.6).

For $K = 2$, $l_o = 1$, (B.8) and (B.9) are respectively given by

$$\hat{a}_{\varphi(i_o),0} - \tilde{a}_{\varphi(i_o),0} = 2(\hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1}), \tag{B.16}$$

$$\hat{a}_{\varphi(i_o+1),0} - \tilde{a}_{\varphi(i_o+1),0} = 2(\hat{a}_{\varphi(i_o),1} - \hat{a}_{\varphi(i_o+1),1}). \tag{B.17}$$

Based on (B.14)–(B.17), we now compute $R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}})$ for different types of $\mathcal{D}$. For $\mathcal{D} \in \widetilde{\mathcal{D}}_{1,1}$, we have

$$\begin{aligned} R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}}) &= \hat{a}_{\varphi(i_o),0} - \tilde{a}_{\varphi(i_o),0} + \hat{a}_{\varphi(i_o),1} - \tilde{a}_{\varphi(i_o),1} \\ &\overset{(a)}{=} 2\hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1} - \tilde{a}_{\varphi(i_o),1} \\ &\overset{(b)}{=} \hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1} \end{aligned} \tag{B.18}$$

where (a) is by (B.16) and (b) is due to (B.4). Similarly, using (B.6)(B.7)(B.16) and (B.17), we obtain the following for all the other types of $\mathcal{D}$

$$R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}}) = \begin{cases} \hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1}, & \mathcal{D} \in \widetilde{\mathcal{D}}_{1,1} \\ 2(\hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1}), & \mathcal{D} \in \widetilde{\mathcal{D}}_{1,2} \\ \hat{a}_{\varphi(i_o),1} - \hat{a}_{\varphi(i_o+1),1}, & \mathcal{D} \in \widetilde{\mathcal{D}}_{2,1} \\ 2(\hat{a}_{\varphi(i_o),1} - \hat{a}_{\varphi(i_o+1),1}), & \mathcal{D} \in \widetilde{\mathcal{D}}_{2,2} \\ 0, & \mathcal{D} \in \widetilde{\mathcal{D}}_3 \cup \widetilde{\mathcal{D}}_4, \end{cases}$$

or more compactly,

$$R_{\text{lb}}(\mathcal{D};\hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D};\tilde{\mathbf{a}}) = \begin{cases} j(\hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1}), & \mathcal{D} \in \widetilde{\mathcal{D}}_{1,j},\ j = 1,2 \\ j(\hat{a}_{\varphi(i_o),1} - \hat{a}_{\varphi(i_o+1),1}), & \mathcal{D} \in \widetilde{\mathcal{D}}_{2,j},\ j = 1,2 \\ 0, & \mathcal{D} \in \widetilde{\mathcal{D}}_3 \cup \widetilde{\mathcal{D}}_4. \end{cases} \tag{B.19}$$

Substituting (B.19) into (B.11), we have

$$\begin{aligned} \bar{R}_{\text{lb}}(\hat{\mathbf{a}}) - \bar{R}_{\text{lb}}(\tilde{\mathbf{a}}) = &\sum_{j=1}^{2} \sum_{\mathcal{D} \in \widetilde{\mathcal{D}}_{1,j}} \sum_{\mathbf{d} \in \mathcal{T}(\mathcal{D})} p_{d_1} p_{d_2} j(\hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1}) \\ &+ \sum_{j=1}^{2} \sum_{\mathcal{D} \in \widetilde{\mathcal{D}}_{2,j}} \sum_{\mathbf{d} \in \mathcal{T}(\mathcal{D})} p_{d_1} p_{d_2} j(\hat{a}_{\varphi(i_o),1} - \hat{a}_{\varphi(i_o+1),1}) \\ = &\big(p_{\varphi(i_o)}^2 + 2\sum_{n' \in \mathcal{N}'} p_{n'} p_{\varphi(i_o)}\big)(\hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1}) \\ &+ \big(2\sum_{n' \in \mathcal{N}''} p_{n'} p_{\varphi(i_o)}\big) 2(\hat{a}_{\varphi(i_o+1),1} - \hat{a}_{\varphi(i_o),1}) \\ &+ \big(p_{\varphi(i_o+1)}^2 + 2\sum_{n' \in \mathcal{N}'} p_{n'} p_{\varphi(i_o+1)}\big)(\hat{a}_{\varphi(i_o),1} - \hat{a}_{\varphi(i_o+1),1}) \\ &+ \big(2\sum_{n' \in \mathcal{N}''} p_{n'} p_{\varphi(i_o+1)}\big) 2(\hat{a}_{\varphi(i_o),1} - \hat{a}_{\varphi(i_o+1),1}) \end{aligned}$$

$$\geq 0. \tag{B.20}$$

where $\mathcal{N}' \triangleq \{\varphi(i_o+2), \ldots, \varphi(N)\}$ and $\mathcal{N}'' \triangleq \{\varphi(1), \ldots, \varphi(i_o-1)\}$, and the last inequality is due to the assumption that $p_{\varphi(i_o)} < p_{\varphi(i_o+1)}$ and $\hat{a}_{\varphi(i_o),1} \geq \hat{a}_{\varphi(i_o+1),1}$ for $l_o = 1$.

Case 2: $l_o = 2$**.** From the third case in (B.4), we have

$$\hat{a}_{n,1} = \tilde{a}_{n,1},\ n \in \mathcal{N}. \tag{B.21}$$

For $K = 2$, $l_o = 2$, (B.8) and (B.9) are respectively given by

$$\hat{a}_{\varphi(i_o),0} - \tilde{a}_{\varphi(i_o),0} = \hat{a}_{\varphi(i_o+1),2} - \hat{a}_{\varphi(i_o),2}, \tag{B.22}$$

$$\hat{a}_{\varphi(i_o+1),0} - \tilde{a}_{\varphi(i_o+1),0} = \hat{a}_{\varphi(i_o),2} - \hat{a}_{\varphi(i_o+1),2}. \tag{B.23}$$

We compare $R_{\text{lb}}(\mathcal{D}; \hat{\mathbf{a}})$ and $R_{\text{lb}}(\mathcal{D}; \tilde{\mathbf{a}})$ for different types of $\mathcal{D}$'s. For $\mathcal{D} \notin \widetilde{\mathcal{D}}_3$, it is straightforward to show that the expressions of $R_{\text{lb}}(\mathcal{D}; \hat{\mathbf{a}})$ and $R_{\text{lb}}(\mathcal{D}; \tilde{\mathbf{a}})$ are the same as the those for Case 1 ($l_o = 1$) in (B.14) and (B.15), respectively. Similar to Case 1, based on (B.14) (B.15) and (B.21) – (B.23), except for $\mathcal{D} \notin \widetilde{\mathcal{D}}_3$, we have

$$R_{\text{lb}}(\mathcal{D}; \hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D}; \tilde{\mathbf{a}}) = \begin{cases} \hat{a}_{\varphi(i_o+1),2} - \hat{a}_{\varphi(i_o),2}, & \mathcal{D} \in \widetilde{\mathcal{D}}_{1,j},\ j = 1, \ldots, |\mathcal{D}| \\ \hat{a}_{\varphi(i_o),2} - \hat{a}_{\varphi(i_o+1),2}, & \mathcal{D} \in \widetilde{\mathcal{D}}_{2,j},\ j = 1, \ldots, |\mathcal{D}| \\ 0, & \mathcal{D} \in \widetilde{\mathcal{D}}_4. \end{cases} \tag{B.24}$$

For $\mathcal{D} \in \widetilde{\mathcal{D}}_3$, we rewrite (B.10) for both $\hat{\mathbf{a}}$ and $\tilde{\mathbf{a}}$ as follows

$$R_{\text{lb}}(\mathcal{D}; \mathbf{a}) = a_{\varphi(i_o),0} + a_{\varphi(i_o+1),0} + \max\{a_{\varphi(i_o),1}, a_{\varphi(i_o+1),1}\}, \quad \mathcal{D} \in \widetilde{\mathcal{D}}_3,\ \mathbf{a} \in \{\hat{\mathbf{a}}, \tilde{\mathbf{a}}\}. \tag{B.25}$$

By (B.7), the sum of the first two terms in (B.25) is the same for $\hat{\mathbf{a}}$ and $\tilde{\mathbf{a}}$. By (B.21), the third term in (B.25) is identical for $\hat{\mathbf{a}}$ and $\tilde{\mathbf{a}}$. Thus, we have

$$R_{\text{lb}}(\mathcal{D}; \hat{\mathbf{a}}) - R_{\text{lb}}(\mathcal{D}; \tilde{\mathbf{a}}) = 0, \quad \mathcal{D} \in \widetilde{\mathcal{D}}_3. \tag{B.26}$$

Following (B.20), we substitute (B.24) and (B.26) into (B.11) and obtain the following

$$\begin{aligned} \bar{R}_{\text{lb}}(\hat{\mathbf{a}}) - \bar{R}_{\text{lb}}(\tilde{\mathbf{a}}) =& \Big(p_{\varphi(i_o)}^2 + \sum_{n' \in \mathcal{N}' \cup \mathcal{N}''} 2p_{n'}p_{\varphi(i_o)}\Big)(\hat{a}_{\varphi(i_o+1),2} - \hat{a}_{\varphi(i_o),2}) \\ &+ \Big(p_{\varphi(i_o+1)}^2 + \sum_{n' \in \mathcal{N}' \cup \mathcal{N}''} 2p_{n'}p_{\varphi(i_o+1)}\Big)(\hat{a}_{\varphi(i_o),2} - \hat{a}_{\varphi(i_o+1),2}) \\ \geq& 0. \end{aligned} \tag{B.27}$$

where $\mathcal{N}'$ and $\mathcal{N}''$ are defined below (B.20) and the inequality is due to the assumption that $p_{\varphi(i_o)} < p_{\varphi(i_o+1)}$ and $\hat{a}_{\varphi(i_o),2} \geq \hat{a}_{\varphi(i_o+1),2}$ for $l_o = 2$.

From the above results, we conclude that $\bar{R}_{\text{lb}}(\hat{\mathbf{a}}) - \bar{R}_{\text{lb}}(\tilde{\mathbf{a}}) \geq 0$, for any $l_o \in \{1, 2\}$. This means that, if $p_{\varphi(i_o)} < p_{\varphi(i_o+1)}$, we can always reduce $\bar{R}_{\text{lb}}(\hat{\mathbf{a}})$ by switching the values of $\hat{a}_{\varphi(i_o),l_o}$ and $\hat{a}_{\varphi(i_o+1),l_o}$. It follows that at the optimality of **P1**, we have $a_{n_1,l_o} \geq a_{n_2,l_o}$, $l_o = 1, 2$, for any $n_1, n_2 \in \mathcal{N}$ satisfying $p_{n_1} \geq p_{n_2}$, *i.e.,* the optimal $\mathbf{a}$ is a popularity-first cache placement. Thus, **P1** and **P2** are equivalent.

B.4 Proof of Lemma 4

We look at each inner term $\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}} \bar{a}^{\mathcal{S}}_{\psi(i),l}$ of $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (4.20), for $i = 1, \ldots, \tilde{N}(\mathbf{d})$. For cache subgroup $\mathcal{A}^{l+1}$, first consider $\tilde{\mathcal{A}}_1^{l+1}$, where for any user subset $\mathcal{S} \in \tilde{\mathcal{A}}_1^{l+1}$, $\mathcal{S}$ includes user $\psi(1)$. Based on the relation of mappings $\psi(\cdot)$ and $\phi(\cdot)$ discussed above (4.16), we have $a_{d_{\psi(1)},l} = a_{\phi(1),l}$, which is the size of the coded message for any user subset $\mathcal{S} \in \tilde{\mathcal{A}}_1^{l+1}$. By (4.16), (4.19), and $|\tilde{\mathcal{A}}_1^{l+1}| = \binom{K-1}{l}$, we have

$$\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_1^{l+1}} \bar{a}^{\mathcal{S}}_{\psi(1),l} = \binom{K-1}{l} a_{\phi(1),l}. \tag{B.28}$$

Denote $\ddot{N}_i$ as the number of users that request file $\phi(i)$ but are not in leader group $\mathcal{U}$. We have $\ddot{N}_i \leq N - \tilde{N}(\mathbf{d})$. For $\tilde{\mathcal{A}}_2^{l+1}$ (in which user subsets includes user $\psi(2)$ but not $\psi(1)$),. among the total of $\binom{K-2}{l}$ user subsets, there are $\binom{K-2-\ddot{N}_1}{l}$ user subsets that do not contain any user that requests file $\phi(1)$. The size of coded messages corresponding to these user subsets is $a_{d_{\psi(2)},l} = a_{\phi(2),l}$. For the rest of $\binom{K-2}{l} - \binom{K-2-\ddot{N}_1}{l}$ user subsets, since they contain at least one user k' from the redundant group that requests file $\phi(1)$, the size of coded message is $a_{d_{k'},l} = a_{\phi(1),l}$. Thus, the size of coded message for user subset $\mathcal{S} \in \tilde{\mathcal{A}}_2^{l+1}$ can be one of the above two cases, and we have

$$\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_2^{l+1}} \bar{a}^{\mathcal{S}}_{\psi(2),l} = \left(\binom{K-2}{l} - \binom{K-2-\ddot{N}_1}{l}\right) a_{\phi(1),l} + \binom{K-2-\ddot{N}_1}{l} a_{\phi(2),l}. \tag{B.29}$$

Following the similar arguments above, the size of coded message for user subset $\mathcal{S} \in \tilde{\mathcal{A}}_3^{l+1}$ (*i.e.,* including $\psi(3)$ but not $\psi(1)$,$\psi(2)$) can be one of the three types $a_{\phi(1),l}$, $a_{\phi(2),l}$ and $a_{\phi(3),l}$. It follows that

$$\begin{aligned}\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_3^{l+1}} \bar{a}^{\mathcal{S}}_{\psi(3),l} = &\left(\binom{K-3}{l} - \binom{K-3-\ddot{N}_1}{l}\right) a_{\phi(1),l} \\ &+ \left(\binom{K-3-\ddot{N}_1}{l} - \binom{K-3-\ddot{N}_1-\ddot{N}_2}{l}\right) a_{\phi(2),l}\end{aligned}$$

$$+\binom{K-3-\ddot{N}_1-\ddot{N}_2}{l}a_{\phi(3),l}. \tag{B.30}$$

The first term in the above expression corresponds to the coded messages for the user subsets that contain users from the redundant group requesting file $\phi(1)$. The second term is for the coded messages for the user subsets that contain users from the redundant group requesting file $\phi(2)$ but not $\phi(1)$. The third term represents the coded messages for all the rest user subsets in $\tilde{\mathcal{A}}_2^{l+1}$ that do not request either file $\phi(1)$ or $\phi(2)$.

Following the derivations above, we can obtain the general expression of $\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}}\bar{a}^{\mathcal{S}}_{\psi(i),l}$ with a recursive pattern. Let $\hat{N}(i)$ be the total number of redundant requests for files $\{\phi(1),\ldots,\phi(i)\}$ (*i.e.,* file requests by users in the redundant group). We have $\hat{N}(i)\triangleq\sum_{j=1}^{i}\ddot{N}_j$. Similar to (B.28)–(B.30), for the coded messages for $\mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}$, we have

$$\begin{aligned}\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}}\bar{a}^{\mathcal{S}}_{\psi(i),l}=&\left(\binom{K-i}{l}-\binom{K-i-\hat{N}(1)}{l}\right)a_{\phi(1),l}+\ldots\\&+\left(\binom{K-i-\hat{N}(i-2)}{l}-\binom{K-i-\hat{N}(i-1)}{l}\right)a_{\phi(i-1),l}\\&+\binom{K-i-\hat{N}(i-1)}{l}a_{\phi(i),l},\end{aligned} \tag{B.31}$$

for $i=1,\ldots,\tilde{N}(\mathbf{d})$. Assume that $\hat{N}(0)=0$. From (B.28) – (B.31), we have

$$\sum_{i=1}^{\tilde{N}(\mathbf{d})}\sum_{\mathcal{S}\in\tilde{\mathcal{A}}_i^{l+1}}\bar{a}^{\mathcal{S}}_{\psi(i),l}=\sum_{i=1}^{\tilde{N}(\mathbf{d})}\left[\sum_{j=i}^{\tilde{N}(\mathbf{d})}\binom{K-j-\hat{N}(i-1)}{l}-\sum_{j=i+1}^{\tilde{N}(\mathbf{d})}\binom{K-j-\hat{N}(i)}{l}\right]a_{\phi(i),l}. \tag{B.32}$$

Summing up both sides of (B.32) for $l=0,\ldots,K-1$, we have $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ as in (4.32).

B.5 Proof of Theorem 6

To show that **P4** and **P5** are equivalent for $K=2$, we will show that $\bar{R}_{\text{lb}}(\mathbf{a})=\bar{R}_{\text{MCCS}}(\mathbf{a})$, for any given $\mathbf{a}$. To do so, we only need to compare $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ and $R_{\text{lb}}(\mathcal{D};\mathbf{a})$. For $\mathcal{K}=\{1,2\}$, we have $|\mathcal{D}|=1$ or 2. We consider the two cases separately below.

For $|\mathcal{D}|=1$

Two users request the same file. We have $d_1=d_2$. Thus, we have $\mathcal{D}=\{d_1\}$. By $R_{\text{MCCS}}(\mathbf{d};\mathbf{a})$ in (5.7) and $R_{\text{lb}}(\mathcal{D};\mathbf{a})$ in (4.12), it is straightforward to show that

$$R_{\text{MCCS}}(\mathbf{d};\mathbf{a})=R_{\text{lb}}(\mathcal{D};\mathbf{a})=a_{d_1,0}+a_{d_1,1}. \tag{B.33}$$

For $|\mathcal{D}| = 2$

As shown in (B.10), $R_{\text{lb}}(\mathcal{D}; \mathbf{a})$ in (4.12) can be written as

$$R_{\text{lb}}(\mathcal{D}; \mathbf{a}) = a_{\xi(1),0} + a_{\xi(2),0} + a_{\xi(1),1} \tag{B.34}$$

where $\xi : \mathcal{I}_{|\mathcal{D}|} \to \mathcal{D}$ is defined as a bijective map such that $a_{\xi(1),1} \geq a_{\xi(2),1}$.

Since two users request different files, we have the leader group $\mathcal{U} = \{1, 2\}$. For $\mathcal{K} = \{1, 2\}$, $R_{\text{MCCS}}(\mathbf{d}; \mathbf{a})$ in (5.7) is given by

$$R_{\text{MCCS}}(\mathbf{d}; \mathbf{a}) = \sum_{\mathcal{S} \subseteq \{\{1\},\{2\},\{1,2\}\}} \max_{k \in \mathcal{S}} a_{d_k,l} = a_{d_1,0} + a_{d_2,0} + \max\{a_{d_1,1}, a_{d_2,1}\}. \tag{B.35}$$

By the definition of $\xi : [|\mathcal{D}|] \to \mathcal{D}$, we have $R_{\text{lb}}(\mathcal{D}; \mathbf{a}) = R_{\text{MCCS}}(\mathbf{d}; \mathbf{a})$. Thus, we conclude that $\bar{R}_{\text{lb}}(\mathbf{a}) = \bar{R}_{\text{MCCS}}(\mathbf{a})$, and **P4** and **P5** are equivalent .

Appendix C

Appendices for Chapter 5

C.1 Proof of Theorem 7

The proof follows the genie-based approach used in developing the lower bound for the centralized uncoded cache placement under uniform or nonuniform file popularity [44,109] or nonuniform cache sizes [66]. For a given file request vector $\mathcal{D}_{\mathcal{A}}$ by the active users in $\mathcal{A}$, the average delivery rate must satisfy [109]

$$R(\mathcal{D}_{\mathcal{A}};\mathbf{q}) \geq \max_{\pi:\mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|}\to\mathcal{D}_{\mathcal{A}}} \sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})} \sum_{s=1}^{A-1} \binom{A-s}{i} a_{\pi(i),s} \tag{C.1}$$

where $a_{\pi(i),s}$ is the number of bits of file $\pi(i)$ cached exclusively by any user subset $\mathcal{S} \in \mathcal{A}$ with $|\mathcal{S}| = s$ users. With decentralized placement, as shown in (5.3), the number of bits cached by any s active users in $\mathcal{A}$ is $a_{\pi(i),s} = q_{\pi(i)}^{s}(1-q_{\pi(i)})^{A-s}F$. Following this, under decentralized placement, we can rewrite (C.1) as

$$R(\mathcal{D}_{\mathcal{A}};\mathbf{q}) \geq \max_{\pi:\mathcal{I}_{|\mathcal{D}_{\mathcal{A}}|}\to\mathcal{D}_{\mathcal{A}}} \sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})} \sum_{s=1}^{A-1} \binom{A-s}{i} q_{\pi(i)}^{s}(1-q_{\pi(i)})^{A-s}F.$$

which is the lower bound on the delivery rate for a given $\mathcal{D}_{\mathcal{A}}$ and $\mathbf{q}$ defined in (4.12). By averaging $R_{\text{lb}}(\mathcal{D}_{\mathcal{A}};\mathbf{q})$ over all possible $\mathcal{D}_{\mathcal{A}} \subseteq \mathcal{N}$ and $\mathcal{A}$ over all possible $\mathcal{A} \subseteq \mathcal{K}$, we obtain the general lower bound $\bar{R}_{\text{lb}}(\mathbf{q})$ the average rate w.r.t $\mathbf{q}$ in (5.20). The final lower bound on average rate is obtained by optimizing $\mathbf{q}$ to minimize $\bar{R}_{\text{lb}}(\mathbf{q})$, which is the optimization problem **P4** as presented in Theorem 7.

C.2 Proof of Proposition 8

We divide all the transmitted coded messages into two different types based on the user subsets they corresponding to. Denote $\mathcal{Q}_1^{s+1} \triangleq \{\mathcal{S} \subseteq \mathcal{A} : \mathcal{S} \cap \mathcal{U}_{\mathcal{A}} \neq \emptyset, \mathcal{S} \cap \mathcal{A}_1 \neq \emptyset, |\mathcal{S}| =$

$s+1\}$ as the non-redundant groups of size $s+1$ that include the users in $\mathcal{A}_1$. Also denote $\mathcal{Q}_2 \triangleq \{\mathcal{S} \subseteq \mathcal{A} : \mathcal{S} \cap \mathcal{U}_{\mathcal{A}} \neq \emptyset, \mathcal{S} \cap \mathcal{A}_1 = \emptyset, |\mathcal{S}| = s+1\}$ as the rest of the non-redundant groups of size $s+1$, *i.e.,* the ones do not include any user in $\mathcal{A}_1$. By definition, we have $\mathcal{Q}^{s+1} = \mathcal{Q}_1^{s+1} \cup \mathcal{Q}_2^{s+1}$. Thus, we can rewrite $R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; \mathbf{q})$ in (5.7) as

$$R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}; \mathbf{q}) = \sum_{s=0}^{A-1} \sum_{\mathcal{S} \in \mathcal{Q}_1^{s+1}} |\mathcal{C}_{\mathcal{S}}| + \sum_{s=0}^{A-1} \sum_{\mathcal{S} \in \mathcal{Q}_2^{s+1}} |\mathcal{C}_{\mathcal{S}}|. \tag{C.2}$$

For the coded message corresponding to the user subset $\mathcal{S} \in \mathcal{Q}_1^{s+1}$, its length is given by

$$|\mathcal{C}_{\mathcal{S}}| = \left(\frac{M}{N_1}\right)^s \left(1 - \frac{M}{N_1}\right)^{A-s} F, \ \mathcal{S} \in \mathcal{Q}_1^{s+1}. \tag{C.3}$$

The number of user subsets in $\mathcal{Q}_1^{s+1}$ can be expressed as the total number of non-redundant groups minus the number of non-redundant groups that only include users in $\mathcal{A}_2$, given by $\left(\binom{A}{s+1} - \binom{A-\widetilde{N}(\mathbf{d}_{\mathcal{A}})}{s+1}\right) - \left(\binom{A_2}{s+1} - \binom{A_0-\widetilde{N}(\mathbf{d}_{\mathcal{A}_2})}{s+1}\right)$. Thus, the total length of the all the coded messages corresponding to the user subsets in $\mathcal{Q}_1^{s+1}$, *i.e.,* the first term of (C.2) can be expressed as

$$\begin{aligned}
&\sum_{\mathcal{S} \in \mathcal{Q}_1^{s+1}} |\mathcal{C}_{\mathcal{S}}| = \\
&\left(\binom{A}{s+1} - \binom{A-\widetilde{N}(\mathbf{d}_{\mathcal{A}})}{s+1} - \left(\binom{A_2}{s+1} - \binom{A_2-\widetilde{N}(\mathbf{d}_{\mathcal{A}_2})}{s+1}\right)\right) \left(\frac{M}{N_1}\right)^s \left(1 - \frac{M}{N_1}\right)^{A-s} F \\
&= \left(\sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})} \binom{A-i}{s} - \sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}_2})} \binom{A_2-i}{s}\right) \left(\frac{M}{N_1}\right)^s \left(1 - \frac{M}{N_1}\right)^{A-s} F
\end{aligned} \tag{C.4}$$

where the second equality is due to $\binom{A}{s+1} - \binom{A-\widetilde{N}(\mathbf{d}_{\mathcal{A}})}{s+1} = \sum_{i=1}^{\widetilde{N}(\mathbf{d}_{\mathcal{A}})} \binom{A-i}{s}$ that can be easily shown.

For the coded messages corresponding to $\mathcal{S} \in \mathcal{Q}_2^{s+1}, s = 0, \ldots, K-1$, since all the files requested by the users in the user subset are only in the server, we have

$$|\mathcal{C}_{\mathcal{S}}| = \begin{cases} F, & s = 0, \mathcal{S} \in \mathcal{Q}_2^{s+1}; \\ 0, & s \geq 1, \mathcal{S} \in \mathcal{Q}_2^{s+1}. \end{cases} \tag{C.5}$$

As a result, the length of the coded messages corresponding to user subsets in $\mathcal{Q}_2$, *i.e.,* the first term of (C.2) is given by

$$\sum_{s=0}^{A-1} \sum_{\mathcal{S} \in \mathcal{Q}_2^{s+1}} |\mathcal{C}_{\mathcal{S}}| = \widetilde{N}(\mathbf{d}_{\mathcal{A}_2}) F. \tag{C.6}$$

Thus, by plugging in the results in (C.4) and (C.6) into (C.2), we obtain the expression of $R_{\text{MCCS}}(\mathbf{d}_{\mathcal{A}}, N_1)$ in (5.19) and finish the proof.

Bibliography

[1] Cisco, "Global mobile data traffic forecast update, 2016-2021," 2017.

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

[3] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5g systems," *IEEE Commun. Mag.*, vol. 52, pp. 131–139, Feb. 2014.

[4] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE J. Sel. Areas Commun.*, vol. 36, pp. 1111–1125, Sep. 2018.

[5] R. Fagin, "Asymptotic miss ratios over independent references," *Journal of Computer and System Sciences*, vol. 14, pp. 222–250, Feb. 1977.

[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, 1999, pp. 126–134.

[7] M. Korupolu and M. Dahlin, "Coordinated placement and replacement for large-scale distributed caches," *IEEE Trans. on Knowledge and Data Engineering*, vol. 14, pp. 1317–1329, Jun. 2002.

[8] I. D. Baev and R. Rajaraman, "Approximation algorithms for data placement in arbitrary networks," in *the 20th annual ACM-SIAM symp. on Discrete algorithms*, 2001, pp. 661–670.

[9] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. on Netw.*, vol. 8, pp. 568–582, May 2000.

[10] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, Mar. 2010, pp. 1–9.

[11] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 176–189, Jan. 2016.

[12] K. Li, C. Yang, Z. Chen, and M. Tao, "Optimization and analysis of probabilistic caching in n -tier heterogeneous networks," *IEEE Trans. Commun.*, vol. 17, pp. 1283–1297, Feb. 2018.

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

[14] J. Song, H. Song, and W. Choi, "Optimal content placement for wireless femto-caching network," *IEEE Trans. Wireless Commun.*, vol. 16, pp. 4433–4444, Jul. 2017.

[15] S.-H. Park, O. Simeone, and S. Shamai Shitz, "Joint optimization of cloud and edge processing for fog radio access networks," *IEEE Trans. Wireless Commun.*, vol. 15, pp. 7621–7632, Sep. 2016.

[16] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inform. Theory*, vol. 63, pp. 1146–1158, Dec. 2017.

[17] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inform. Theory*, vol. 60, pp. 2856–2867, Mar. 2014.

[18] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, pp. 1029–1040, Aug. 2015.

[19] A. Sengupta, R. Tandon, and O. Simeone, "Fog-aided wireless networks for content delivery: Fundamental latency tradeoffs," *IEEE Trans. Inform. Theory*, vol. 63, pp. 6650–6678, Aug. 2017.

[20] M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, Jun. 2015, pp. 809–813.

[21] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inform. Theory*, vol. 62, pp. 849–869, Feb. 2016.

[22] Ç. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of D2D coded caching with uncoded cache placement and one-shot delivery," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8179–8192, Dec. 2019.

[23] F. Xu, M. Tao, and K. Liu, "Fundamental tradeoff between storage and latency in cache-aided wireless interference networks," *IEEE Trans. Inform. Theory*, vol. 63, pp. 7464–7491, Jun. 2017.

[24] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. of the ACM*, vol. 51, pp. 348–355, Jan. 2008.

[25] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Magazine*, vol. 55, pp. 34–40, Apr. 2017.

[26] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inform. Theory*, vol. 64, pp. 109–128, Jan. 2018.

[27] M. Kiamari, C. Wang, and A. S. Avestimehr, "On heterogeneous coded distributed computing," in *Proc. IEEE Global Telecommn. Conf. (GLOBECOM)*, Dec. 2017.

[28] F. Xu and M. Tao, "Heterogeneous coded distributed computing: Joint design of file allocation and function assignment," in *Proc. IEEE Global Telecommn. Conf. (GLOBECOM)*, Dec. 2019.

[29] N. Woolsey, R.-R. Chen, and M. Ji, "Coded distributed computing with heterogeneous function assignments," in *Proc. IEEE Int. Conf. Communications (ICC)*, Jun. 2020.

[30] F. Xu, S. Shao, and M. Tao, "New results on the computation-communication tradeoff for heterogeneous coded distributed computing," *IEEE Trans. Commun.*, vol. 69, pp. 2254–2270, Apr. 2021.

[31] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Transactions on Networking*, vol. 25, pp. 2643–2654, May 2017.

[32] Y. Dong, B. Tang, B. Ye, Z. Qu, and S. Lu, "Intermediate value size aware coded mapreduce," in *26th IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS)*, Dec. 2020, pp. 348–355.

[33] A. M. Daniel and W. Yu, "Optimization of heterogeneous coded caching," *IEEE Trans. Inform. Theory*, vol. 66, pp. 1893–1919, Mar. 2020.

[34] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inform. Theory*, vol. 63, pp. 3108–3141, Mar. 2017.

[35] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Trans. Inform. Theory*, vol. 63, pp. 3923–3949, Apr. 2017.

[36] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Trans. Inform. Theory*, vol. 64, pp. 349–366, Nov. 2018.

[37] S. Jin, Y. Cui, H. Liu, and G. Caire, "Structural properties of uncoded placement optimization for coded delivery," *arXiv preprint arXiv:1707.07146*, Jul. 2017.

[38] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite-length analysis of caching-aided coded multicasting," *IEEE Trans. Inform. Theory*, vol. 62, pp. 5524–5537, Aug. 2016.

[39] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Trans. Inform. Theory*, vol. 63, pp. 5821–5833, Jul. 2017.

[40] M. Cheng, Q. Yan, X. Tang, and J. Jiang, "Coded caching schemes with low rate and subpacketizations," *arXiv preprint arXiv:1703.01548*, Mar. 2017.

[41] L. Tang and A. Ramamoorthy, "Low subpacketization schemes for coded caching," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, Jun. 2017, pp. 2790–2794.

[42] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using Ruzsa-Szeméredi graphs," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, Jun. 2017, pp. 1237–1241.

[43] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE J. Sel. Areas Commun.*, vol. 36, pp. 1176–1188, Jun. 2018.

[44] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inform. Theory*, vol. 64, pp. 1281–1296, Feb. 2018.

[45] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, Jul. 2016.

[46] ——, "On the optimality of uncoded cache placement," in *IEEE Infor. Theory Workshop*, Sep. 2016.

[47] S. A. Saberali, L. Lampe, and I. F. Blake, "Full characterization of optimal uncoded placement for the structured clique cover delivery of nonuniform demands," *IEEE Trans. Inform. Theory*, vol. 66, no. 1, pp. 633–648, Jan. 2020.

[48] Y. Deng and M. Dong, "Fundamental structure of optimal cache placement for coded caching with heterogeneous demands," *arXiv preprint arXiv:1912.01082*, Apr. 2020.

[49] H. Cheng, C. Li, H. Xiong, and P. Frossard, "Optimal decentralized coded caching for heterogeneous files," in *the 25th European Signal Processing Conf.*, 2017.

[50] J. Zhang, X. Lin, C. Wang, and X. Wang, "Coded caching for files with distinct file sizes," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, 2015.

[51] J. Zhang, X. Lin, and C. Wang, "Closing the gap for coded caching with distinct file sizes," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, 2019.

[52] S. Sahraei, P. Quinton, and M. Gastpar, "The optimal memory-rate trade-off for the non-uniform centralized caching problem with two files under uncoded placement," *IEEE Trans. Inform. Theory*, vol. 65, no. 12, pp. 7756–7770, Dec. 2019.

[53] S. Jin, Y. Cui, H. Liu, and G. Caire, "Uncoded placement optimization for coded delivery," *arXiv preprint arXiv:1709.06462*, Jul. 2018.

[54] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. of the ACM*, vol. 51, pp. 107–113, Jan. 2008.

[55] Q. Chen, J. Yao, and Z. Xiao, "Libra: Lightweight data skew mitigation in mapreduce," *IEEE Trans. on Parallel and Distributed Systems*, vol. 26, pp. 2520–2533, Sep. 2015.

[56] W. Chen, B. Liu, I. Paik, Z. Li, and Z. Zheng, "Qos-aware data placement for mapreduce applications in geo-distributed data centers," *IEEE Trans. on Engineering Management*, vol. 68, pp. 120–136, Jan. 2021.

[57] F. Li, J. Chen, and Z. Wang, "Wireless mapreduce distributed computing," *IEEE Transactions on Information Theory*, vol. 65, pp. 6101–6114, Oct. 2019.

[58] K. Konstantinidis and A. Ramamoorthy, "Resolvable designs for speeding up distributed computing," *IEEE/ACM Transactions on Networking*, vol. 28, pp. 1657–1670, Apr. 2020.

[59] A. R. Elkordy, S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Compressed coded distributed computing," *IEEE Transactions on Communications*, vol. 69, pp. 2773–2783, May 2021.

[60] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, "Scarlett: Coping with skewed content popularity in mapreduce clusters," in *6th ACM Conf. on Computer Systems (EuroSys)*, Apr. 2011.

[61] Q. Wang, Y. Cui, S. Jin, J. Zou, C. Li, and H. Xiong, "Optimization-based decentralized coded caching for files and caches with arbitrary sizes," *IEEE Trans. Commun.*, Dec. 2019.

[62] C. Wang, S. H. Lim, and M. Gastpar, "A new converse bound for coded caching," in *Inf. Theory and Applications Workshop (ITA)*, 2016.

[63] C. Wang, S. Saeedi Bidokhti, and M. Wigger, "Improved converses and gap results for coded caching," *IEEE Trans. Inform. Theory*, vol. 64, no. 11, pp. 7051–7062, Nov. 2018.

[64] J. Zhang, X. Lin, and C. Wang, "Closing the gap for coded caching with distinct file sizes," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, 2019, pp. 687–691.

[65] Q. Yang and D. Gndz, "Coded caching and content delivery with heterogeneous distortion requirements," *IEEE Trans. Inform. Theory*, vol. 64, pp. 4347–4364, Jun. 2018.

[66] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Coded caching for heterogeneous systems: An optimization perspective," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5321–5335, Aug. 2019.

[67] A. S. Cacciapuoti, M. Caleffi, M. Ji, J. Llorca, and A. M. Tulino, "Speeding up future video distribution via channel-aware caching-aided coded multicast," *IEEE J. Sel. Areas Commun.*, vol. 34, pp. 2207–2218, Aug. 2016.

[68] M. Mohammadi Amiri and D. Gndz, "Cache-aided content delivery over erasure broadcast channels," *IEEE Trans. Commun.*, vol. 66, pp. 370–381, Jan. 2018.

[69] D. Cao, D. Zhang, P. Chen, N. Liu, W. Kang, and D. Gndz, "Coded caching with asymmetric cache sizes and link qualities: The two-user case," *IEEE Trans. Commun.*, vol. 67, no. 9, pp. 6112–6126, 2019.

[70] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *IEEE Trans. Inform. Theory*, vol. 65, pp. 647–663, Jan. 2019.

[71] C. Chang and C. Wang, "Coded caching with full heterogeneity: Exact capacity of the two-user/two-file case," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, 2019.

[72] ——, "Coded caching with heterogeneous file demand sets the insufficiency of selfish coded caching," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, 2019.

[73] S. Wang and B. Peleato, "Coded caching with heterogeneous user profiles," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, 2019.

[74] C. Chang, C. Wang, and B. Peleato, "On coded caching for two users with overlapping demand sets," in *Proc. IEEE Int. Conf. Communications (ICC)*, 2020.

[75] C. Zhang and B. Peleato, "On the average rate for coded caching with heterogeneous user profiles," in *Proc. IEEE Int. Conf. Communications (ICC)*, 2020.

[76] D. Cao, D. Zhang, P. Chen, N. Liu, W. Kang, and D. Gündüz, "Coded caching with asymmetric cache sizes and link qualities: The two-user case," *IEEE Trans. Commun.*, vol. 67, pp. 6112–6126, Sep. 2019.

[77] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Rate-memory trade-off for caching and delivery of correlated sources," *IEEE Trans. Inform. Theory*, vol. 66, pp. 2219–2251, Apr. 2020.

[78] Q. Yang and D. Gündüz, "Coded caching and content delivery with heterogeneous distortion requirements," *IEEE Trans. Inform. Theory*, vol. 64, pp. 4347–4364, Jun. 2018.

[79] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Rate-distortion-memory trade-offs in heterogeneous caching networks," *IEEE Trans. Wireless Commun.*, vol. 19, pp. 3019–3033, May 2020.

[80] E. Parrinello, A. Ünsal, and P. Elia, "Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching," *IEEE Trans. Inform. Theory*, vol. 66, pp. 2252–2268, Apr. 2020.

[81] M. Mohammadi Amiri, Q. Yang, and D. Gndz, "Decentralized caching and coded delivery with distinct cache capacities," *IEEE Trans. Commun.*, vol. 65, pp. 4657–4669, Nov. 2017.

[82] L. Zheng, Q. Chen, Q. Yan, and X. Tang, "Decentralized coded caching scheme with heterogeneous file sizes," *IEEE Trans. Veh. Technol.*, vol. 69, pp. 818–827, Jan. 2020.

[83] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, Apr. 2017.

[84] ——, "Coded mapreduce," in *Allerton Conference on Commun., Control, and Comput.*, Sep. 2015, pp. 964–971.

[85] N. Woolsey, R. Chen, and M. Ji, "A new combinatorial design of coded distributed computing," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, Jun. 2018, pp. 726–730.

[86] Q. Yan, S. Yang, and M. Wigger, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," in *IEEE Infor. Theory Workshop*, Nov. 2018, pp. 1–5.

[87] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2643–2654, Oct. 2017.

[88] F. Li, J. Chen, and Z. Wang, "Wireless mapreduce distributed computing," *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, vol. 65, no. 10, pp. 6101–6114, Oct. 2019.

[89] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inform. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

[90] M. Kiamari, C. Wang, and A. S. Avestimehr, "On heterogeneous coded distributed computing," in *Proc.IEEE Global Telecommn. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–7.

[91] N. Woolsey, R. Chen, and M. Ji, "Cascaded coded distributed computing on heterogeneous networks," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, Jul. 2019, pp. 2644–2648.

[92] N. Woolsey, R. Chen, and M. Ji, "Coded distributed computing with heterogeneous function assignments," *arXiv preprint arXiv:1902.10738*, Feb. 2019.

[93] F. Xu and M. Tao, "Heterogeneous coded distributed computing: Joint design of file allocation and function assignment," *arXiv preprint arXiv:1908.06715*, Aug. 2019.

[94] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[95] W. Feller, "Stirlings formula," *An introduction to probability theory and its applications*, vol. 1, no. 3, pp. 50–53, 1968.

[96] Y. Deng and M. Dong, "Optimal cache placement for modified coded caching with arbitrary cache size," in *Proc. IEEE Int. Workshop on Signal Processing advances in Wireless Commun.(SPAWC)*, Jul. 2019, pp. 1–5.

[97] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 836–845, Apr. 2016.

[98] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *Proc. IEEE Int. Conf. Communications (ICC)*, 2015.

[99] S. Park, O. Simeone, and S. Shamai Shitz, "Joint optimization of cloud and edge processing for fog radio access networks," *IEEE Trans. Wireless Commun.*, vol. 15, pp. 7621–7632, Nov. 2016.

[100] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network–measurements, models, and implications," *Comput. Netw.*, vol. 53, pp. 501–514, Apr. 2009.

[101] M. Avriel, *Advances in Geometric Programming*. New York: PlenumPress, 1980.

[102] M. Chiang, C. W. Tan, D. P. Palomar, D. O'neill, and D. Julian, "Power control by geometric programming," *IEEE Trans. Wireless Commun.*, vol. 6, pp. 2640–2651, Jul. 2007.

[103] M. Conforti, G. Cornuéjols, G. Zambelli *et al.*, *Integer programming*. Springer, 2014.

[104] J. Wang, M. Cheng, Q. Yan, and X. Tang, "Placement delivery array design for coded caching scheme in D2D networks," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3388–3395, May 2019.

[105] N. Woolsey, R.-R. Chen, and M. Ji, "Towards practical file packetizations in wireless device-to-device caching networks," *arXiv preprint arXiv:1712.07221*, Oct. 2017.

[106] X. Zhang and M. Ji, "A new design framework on device-to-device coded caching with optimal rate and significantly less subpacketizations," *arXiv preprint arXiv:1901.07057*, Jun. 2019.

[107] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Device-to-device coded caching with heterogeneous cache sizes," in *Proc. IEEE Int. Conf. Communications (ICC)*, May 2018, pp. 1–6.

[108] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Trans. Inform. Theory*, pp. 1–1, Feb. 2017.

[109] Y. Deng and M. Dong, "Memory-rate tradeoff for caching with uncoded placement under nonuniform file popularity," in *the 54th Asilomar Conf. on Signals, Systems and computers*, 2020.