FUNDAMENTAL LIMITS OF CACHING: SYMMETRY STRUCTURE AND CODED PLACEMENT SCHEMES

A Dissertation

by

KAI ZHANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2020

Major Subject: Electrical Engineering

ABSTRACT

Caching is a technique to reduce the communication load in peak hours by prefetching contents during off-peak hours. In 2014, Maddah-Ali and Niesen introduced a framework for coded caching, and showed that significant improvement can be obtained compared to uncoded caching. Considerable efforts have been devoted to identify the precise information theoretic fundamental limit of such systems, however the difficulty of this task has also become clear. One of the reasons for this difficulty is that the original coded caching setting allows multiple demand types during delivery, which in fact introduces tension in the coding strategy to accommodate all of them. We seek to develop a better understanding of the fundamental limit of coded caching.

In order to characterize the fundamental limit of the tradeoff between the amount of cache memory and the delivery transmission rate of multiuser caching systems, various coding schemes have been proposed in the literature. These schemes can largely be categorized into two classes, namely uncoded prefetching schemes and coded prefetching schemes. While uncoded prefetching schemes in general over order-wise optimal performance, coded prefetching schemes often have better performance at the low cache memory regime. At first sight it seems impossible to connect these two different types of coding schemes, yet finding a unified coding scheme that achieves the optimal memory-rate tradeoff is an important and interesting problem. We take the first step on this direction and provide a connection between the uncoded prefetching scheme proposed by Maddah Ali and Niesen (and its improved version by Yu et al.) and the coded prefetching scheme proposed by Tian and Chen. The intermediate operating points of this general scheme can in fact provide new memory-rate tradeoff points previously not known to be achievable in the literature. This new general coding scheme is then presented and analyzed rigorously, which yields a new inner bound to the memory-rate tradeoff for the caching problem.

While studying the general case can be difficult, we found that studying the single demand type systems will provide important insights. Motivated by these findings, we focus on systems where the number of users and the number of files are the same, and the demand type is when all files are being requested. A novel coding scheme is proposed, which provides several optimal memory transmission operating points. Outer bounds for this class of systems are also considered, and their relation with existing bounds is discussed.

Outer-bounding the fundamental limits of coded caching problem is difficult, not only because there are tons of information inequalities and problem specific equalities to choose from, but also because of identifying a useful subset (and often a quite small subset) from them and how to combine them to produce an improved outerbound is a hard problem. Information inequalities can be used to derive the fundamental limits of information systems. Many information inequalities and problem-specific constraints are linear equalities or inequalities of joint entropies, and thus outer bounding the fundamental limits can be viewed as and in principle computed through linear programming. However, for many practical engineering problems, the resultant linear program (LP) is very large, rendering such a computational approach almost completely inapplicable in practice. We provide a method to pinpoint this reduction by counting the number of orbits induced by the symmetry on the set of the LP variables and the LP constraints, respectively. We proposed a generic three-layer decomposition of the group structures for this purpose. This general approach can also be applied to various other problems such as extremal pairwise cyclically symmetric entropy inequalities and the regenerating code problem.

Decentralized coded caching is applicable in scenarios when the server is uninformed of the number of active users and their identities in a wireless or mobile environment. We propose a decentralized coded prefetching strategy where both prefetching and delivery are coded. The proposed strategy indeed outperforms the existing decentralized uncoded caching strategy in regimes of small cache size when the numbers of files is less than the number of users. Methods to manage the coding overhead are further suggested.

DEDICATION

To my mother, my father and my wife, who are always supportive and helped me in all things great and small.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Prof. Tian, because of his encouragement and patience for every single step when I am persuing my PhD degree. From courses choosen to PhD research topic selection, from guiding me writing papers word by word to suggestions and help on my job hunting process. Prof. Tian gave me careful guidance and full support for every important choice I made.

Over the years, Prof. Tian has given me careful guidance in my studies, strict requirements in my work and enough freedom to choose the research topic I'm interested in. I still remember the first few weeks I entered his research group, instead of assigning us tons of papers to read to get familar with this area before doing any research, prof. Tian assigned us a small project to do which is interesting and at the same time not too difficult to tackle so we could quickly devote ourselves into his research work. We could quickly grasp the critical problem in this area and with the his guidance along the way, we had some positive results very soon and published papers on that. This accomplishment brought a positive feedback and encouraged me to read more papers and books on this topic. I became more interested in devoting myself to this research area and discovering more things. Thanks to him for bring me into his research area in a smart way.

This work would not have been possible without the financial support of Prof. Tian. During the past five years, he always made sure we could get enough financial support thus we could devote our 100% energy on the research without worrying too much about life, and I'm so grateful for that. Thank him for trusting me and bringing me with him to a new university in a new city, where I could broaden by horizen by learning from so many great professors and other colleagues. Thank him for helping me finding internships and refering me, which makes my job hunting process much easier. Thanks for taking a whole day driving all the way to another city to introduce me to the professor of my internship program. I couldn't be more grateful to all of these things big or small he helped me. It's a great pleasure to have the chance to work with him for five years. From him, I could see how great a teacher, mentor, and a friend could be.

I would like to first say thank you to my supervisor Dr. Xiaoqian Jiang you for being a tremendous mentor for me, and for all the support he gave me during the three month summer internship. Thank him for leading me into a new research area and everything he taught me. Without his guidance and constant feedback I couldn't make the achievement using three months. He made my three months internship interesting and fruitful.

Besides my advisor, I also want to thank the other members of my dissertation committee, Prof. Tie Liu, Prof. Alex Sprintson, Prof. Andrew Jiang. They have provided me many useful suggestions for out on-going research during my preliminary exam. I would especially like to thank Prof. Tie Liu, for joining our group meeting and gave us the opportunity to learn from him. Even though we do not have collaborative research, I could still learn a lot from these times I listen to his talk. Thank Prof. Liu, Prof. Sprintson and Prof. Jiang's great support and invaluable advices and crucial remarks during my preliminary exam that shaped my final dissertation. I am quite appreciative of Prof. Anoosheh Heidarzadeh for agreeing to serve on my dissertation committee on such a short notice as a replacement for Prof. Sprintson.

Thanks to my collegues and friends: Hao Ye, Sara Mousavi, Tianli Zhou, Tao Guo, Lei Zheng, for being such good colleagues and friends. I value their friendship and support. Also thank Lucia, Shuo Shao, I haven't gotten a chance to get to really know them since I've been in a hole this past year but they've all been so friendly. I wish these younger students, Ruida and Wenjing, the best of luck and hopefully their time in graduate school.

I would like to specially thank Hao. He was the first friend I made after I came to United States. Thank him for all the help to us, especially during my first year when we don't have a vehicle. He took us to grocery stores, introduce us to all the nice people in the church where we became friends with so many amazing people. He is the friend that led me to get familiar with every life aspect here, and helped us pass the hard time when we had very few friends while needs lots of help. I wish him all the best with his research and his life.

Especially, thank Tianli and Lei, for a wonderful year's time we spent from 2017 to 2018. I still remember those fun times we spent together, dining together and play card games each week.

I can remember almost all the fun moments, and I occasionally take out our pictures and try to recall those nice moments.

Nobody has been more important to me in the pursuit of my PhD degree than the members of my family. I would like to thank my parents, whose love and guidance are with me in whatever I pursue. At the same time, I would like to apologize for being such selfish of going abroad for my degree and not able to be around these years. My hard-working parents have sacrificed their lives for me and provided unconditional love and care. I love them so much, and I would not have made it this far without them. They are always my ultimate role models.

Most importantly, I wish to thank my loving and supportive wife and my soul-mate, Dan, who spend the past five years with me when we have no families around but only have each other. I married the best person out there for me. There are no words to convey how much I love her. Dan has been a true and great supporter and has unconditionally loved me during my good and bad times. She has been non-judgmental of me and instrumental in instilling confidence. She has faith in me and my intellect even when I felt like digging hole and crawling into one because I didn't have faith in myself. These past several years have not been an easy ride, both academically and personally. During the job hunting process, I don't remember how many times I'm about to give up when facing all the interviews, along with graduation work and ongoing research work. She helped me so much by taking all the time-consuming application process to herself so I could focus on preparing for interviews. I truly thank Dan for sticking by my side, even when I was irritable and depressed. I feel that what we both learned a lot about life and strengthened our commitment and determination to each other and to live life to the fullest.

I also want to thank Carl and Paula, Gene and Pam. They are such nice people who taught us American culture, introduced us to all their families and friends, invited us to spend every holiday with their families. We couldn't get used to this new environment so quickly without them. Thank them for taking time to take my parents out for picnic when they visited me. We view them as part of our important family members. Thank Randy and Robin, for being such good friends, and for spending days helping me finding place to live.

I thank my beloved cat, Jiangjiang for his companion during the past three years.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Chao Tian and Professor Tie Liu, Professor Alex Sprintson of the Department of Electrical and Computer Engineering and Professor Anxiao (Andrew) Jiang of the Department of Computer Science and Engineering.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by the National Science Foundation under Grants CCF-15-26095 and CCF-18-32309.

NOMENCLATURE

OGAPS	Office of Graduate and Professional Studies at Texas A&M University
B/CS	Bryan and College Station
TAMU	Texas A&M University
ECEN	Electrical & Computer Engineering
WEB	Wisenbaker Engineering Building
RAID	Redundant Array of Inexpensive Disks or Drives
SSD	Solid State Drive
HDD	Hard Disk Drive
KKT	Karush–Kuhn–Tucker conditions
GP	Geometric Program
IID	Independent and Identically Distributed
DOS	Disk Operating System
HDMI	High Definition Multimedia Interface
LP	Linear Program
LAPACK	Linear Algebra PACKage
MAP	Maximum Aposteriori Probability
ML	Maximum Likelihood
QCQP	Quadratically Constrained Quadratic Program
QP	Quadratic Program
SDP	Semidefinite Program
SOS	Sum Of Squares
SVD	Singular Value Decomposition

L^1	Space of absolutely Lebesgue integrable functions; i.e., $\int \|f\| < \infty$
L^2	Space of square-Lebesgue-integrable functions, i.e., $\int \|f\|^2 < \infty$
$PC(S)$	Space of piecewise-continuous functions on S
GNU	GNU is Not Unix
GUI	Graphical User Interface
PID	Principal Integral Domain
MIP	Mixed Integer Program
EM	Expectation Maximization
AWGN	Additive White Gaussian Noise
BCH	Bose Chaudhuri Hocquenghem code
BSC	Binary Symmetric Channel
MCMC	Markov Chain Monte Carlo
CPU	Central Processing Unit
RAM	Random-access memory
RMS	Root Mean Square
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
MMSE	Minimum Mean Square Error
RMSE	Root Mean Squared Error
SVM	Support Vector Machine
KSVMs	Kernel Support Vector Machines
LSTM	Long Short-Term Memory
MDP	Markov decision process
MAE	Mean Absolute Error
KNN	K Nearest Neighborhood

BP	Back Propagation
ERM	Empirical Risk Minimization
FP	False Positive
TP	True Positive
FPR	False Positive Rate
FFN	Feedforward Neural Network
SGD	Stochastic Gradient Descent
SRM	Structural Risk Minimization

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION[*]

Caching is a technique that has been studied for almost half a decade and are now widely used in computer systems. CPU cache refers to a type of memory which is physically close to CPU, with smaller amount of memory size compared to the hard disk and RAM (Figure 1.1). The advantage of cache memory is that it can increase the speed of computer program by storing the most frequently visited data and instructions in the cache. The data access time for CPU can significantly reduce compared to the accessing time from RAM or the hard disk.

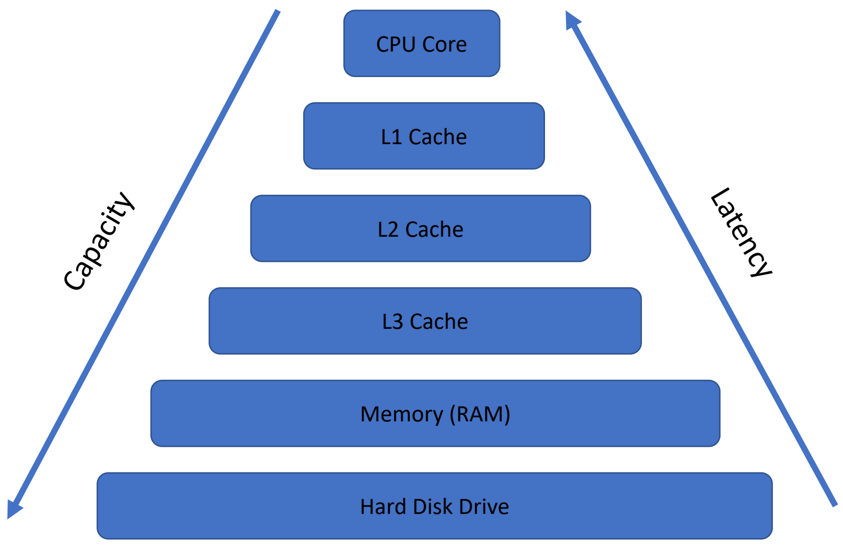

Figure 1.1: Memory hierarchy in a computer.

[*]© 2018 IEEE. Reprinted, with permission, from Kai Zhang and Chao Tian, "Fundamental limits of coded caching: From uncoded prefetching to coded prefetching", *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp.1153-1164, 2018.
© 2019 IEEE. Reprinted, with permission,from Shuo Shao, Jesus Gómez-Vilardebó, Kai Zhang and Chao Tian, "On the fundamental limit of coded caching systems with a single demand type", *2019 IEEE Information Theory Workshop (ITW)*, pp. 1-5, 2019.
© 2018 IEEE. Reprinted, with permission, from Kai Zhang and Chao Tian, "On the symmetry reduction of information inequalities", *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2396-2408, 2016.
© 2017 IEEE. Reprinted, with permission, from Kai Zhang, Chao Tian and Hushengli, "Coded prefetching and efficient delivery in decentralized caching systems", *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1-5, 2017.

Since cache has limited amount of memory size, smart strategies need to be proposed in terms of what data need to be stored in it and how to update its content. Eviction policies such as LRU (Least Recently Used), LFU (Least Frequently Used), MRU (Most Recently Used) and FIFO (First In, First Out) have been widely used [7–16].

1.1 Caching Network

The idea of computer cache has been utilized in the network setting, especially the global CDN (Content Delivery Network, Figure 1.2) [17, 18], which significantly improves the performance of content delivery by reducing the need to repeatedly access the frequently requested files from remote server. When a user requests file contents from the server using CDN, after fulfill the user's request, the CDN stores the content into the local cache to satisfy the future same request. Various CDN strategies including pricing policies are discussed [19–32].

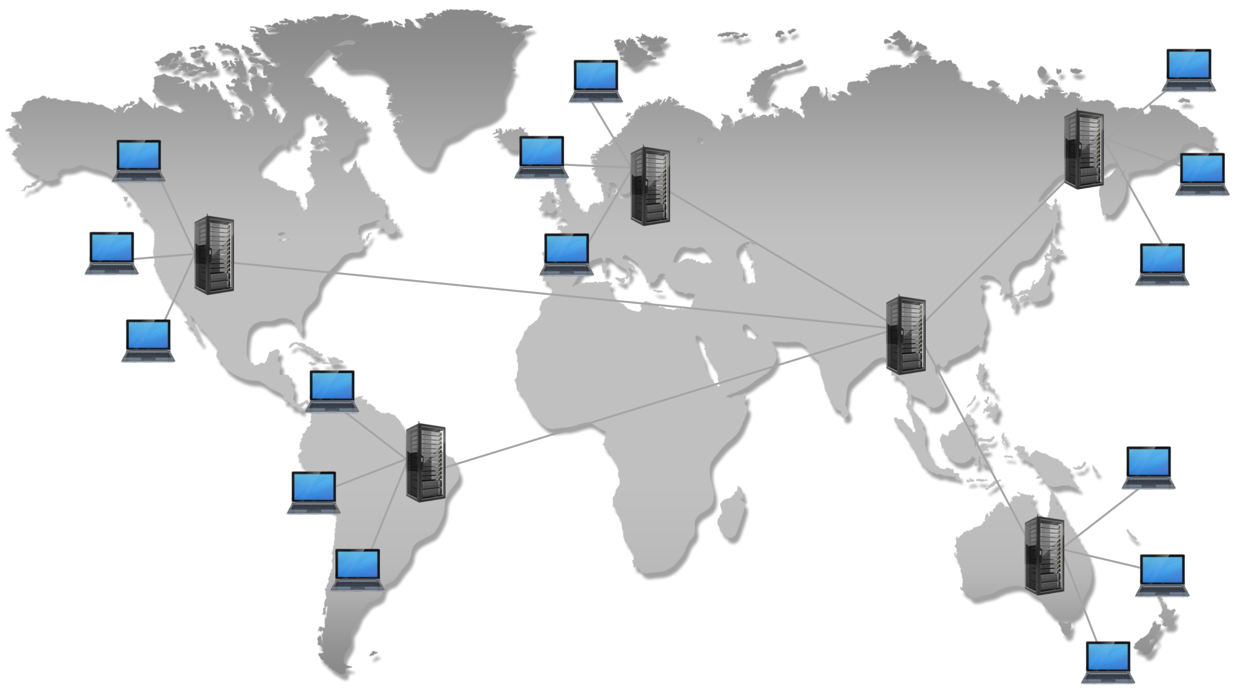

Figure 1.2: Content delivery network (CDN).

Caching has also been used for wireless networks such as mobile network. Wireless network itself is an attractive communication setting where broadcasting strategies have been widely studied. However, a cache-aided wireless network does not receive much attention until recent years. Studies such as [33–39] serve as the seminal works in this field.

Caching is specifically useful in the video-on-demand system (Figure 1.3). This is mainly because video contents' multiplier effect on the internet traffic. Below are several facts about internet traffic according to the Cisco Annual Internet Report (2018–2023) White Paper.

Figure 1.3: Video-on-demand service providers.

By 2023, 66% of the installed flat-panel TV sets will be UHD, up from 33% in 2018 [40]. Nowadays, the HD television are almost used by every household. Moreover, the amount of UHD (Ultra-High Definition), or 4K, video streaming is more and more frequently requested compared to traditional 720P or 1080P video streaming. The bit rate for UHD videos are more than double the bit rate for HD videos, and nine times more than the bit rate of SD (Standard Definition) videos.

The amount of applications related to video are continuing to grow, this puts significant bandwidth demands in the future. Scenarios such as Cloud Gaming, 8K IP video and UHD VR requires significant amount of bandwidth which cannot be globally achieved in current networks.

An important feature of internet video traffic is that it is highly temporal. A research on the Internet traffic [41] studied the average daily internet traffic rate at two large globally Internet Exchange Traffic points, London and Amsterdam, and confirmed this phenomenon. Among all the Internet usage categories, the TV watching has an interesting feature: it remains relatively low for most of the time of the day but significantly grows between 7:00pm-11:00pm. The reason is actually very simple: most people are either working or sleeping during the other time when they do not have time to sit in front of TVs and watch movies or TV shows. But during evening, most people are off-work and a majority of them will spend this time at home with families watching TV.

The video content accounts for most of the internet traffic due to its large size, especially 4K videos, this places a lot of burden on the internet during the evening.

A straightforward solution to this problem is if there is a technique that is able to move some of the traffic during peak traffic time to the off-peak traffic time, then the traffic rate during a day can be smoothed and evenly spreaded during a whole day, which will release the burden on the internet at evening. This burden can actually be solved with the help of cache [20, 42–55].

Modern caching systems are studied in various aspects, from cache deployment to cache dimensioning [17, 56–61], from general caching networks [42, 62–64] to special caching networks such as hierarchical caching network [19, 20, 65, 66]. Figure 1.4 demonstrates how cache can be used to alleviate the burden in the above situation [67]. When a user requests some contents from the server, the router first requests the cache engine to find the content, if it is not in the cache engine. If it is not there, the server will create a separate IP and reroute that request to the remote server, the server sends the content back to the content engine, and the content engine will send it back to the user, at the same time store that content locally. Then future requests for the same content, by the same or other users, can be fulfilled by the local cache engine.

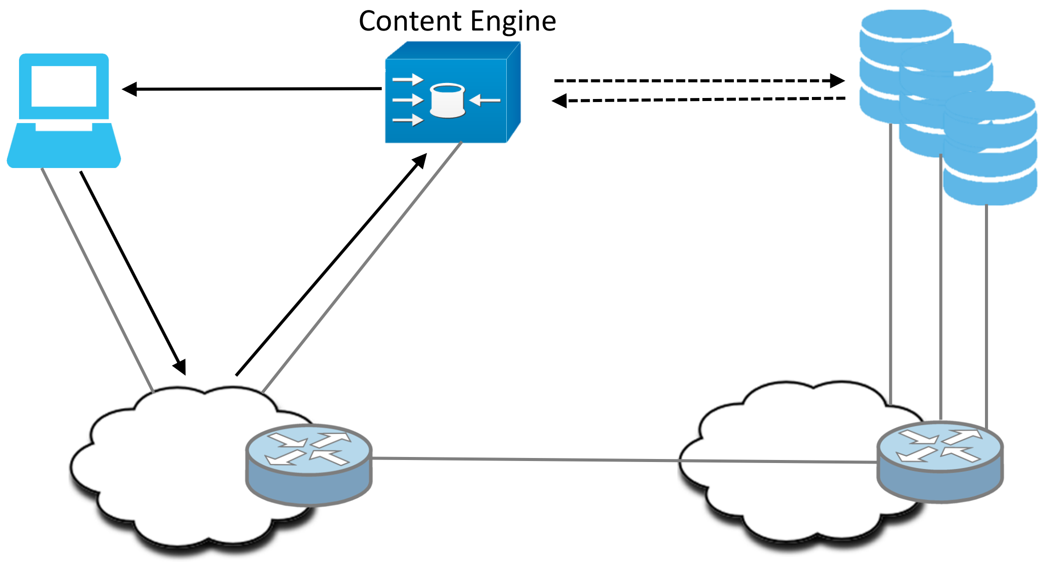

Figure 1.4: Transparent network caching.

1.2 General Cache Use Cases

Cache has been used and successfully deployed in many existing systems. Below are some well-known cache systems.

Google Global Cache (GGC) is deployed by Google that lets ISPs to be able to serve Google contents within their own networks. According to google support, it has the advantage of being 1. Transparent to users; 2. Able to reduce external traffic; 3. Robust; 4. Easy to set up [68].

Facebook is using Facebook Photo CDN [69] to deliver the photos. Facebook has 1.7 billion user photos and take 160TB of photo storage. Facebook uses a fleet of servers to receive photo uploads, and to feed images to CDN partners. Facebook also has its own hierarchical CDN to fulfill the photo storage and requests.

Netflix also has its own CDN called Netflix Open Connect [70, 71]. Netflix lunched open connect in 2011. It is a global network that has the responsibility to delivery Netflix TV shows and movies worldwide.

Other applications such as Amazon AWS, Cadami, and Akamai Intelligent Platform [72, 73] are all such user cases. They together set an example of how useful cache could be in networks in reality.

1.3 Coded Caching

The technique of caching can be used to relieve the congestion on the broadcast channel by prefetching file contents to local memory space close to the user, and thus reducing the amount of data retrieved from the remote data center in peak traffic time. Traditionally, caching has mainly been considered in single user settings such as CPU cache. The hit-ratio is the key measure of performance. As networked systems become more prevalent, caching systems involving multiple users have attracted increasingly more research attention.

The performance of different caching strategies has been widely analyzed from an information theoretic perspective. These works can be categorized into two general categories: the innerbound and the outerbound [2,39,74–86]. In the article [39], Maddah-Ali and Niesen provided a formal information theoretic formulation for the caching problem in multiuser settings. In this formulation, there are N files, each of F bits, and K users. Each user has a local cache memory of capacity MF (thus a normalized capacity of M). In the prefetching (or sometimes referred to as the placement) phase, the users can fill their caches with contents from the central server without the knowledge of the precise requests at the deliver phase. In the delivery phase, each user reveals the request for a single file from the central server, and the central server must multicast certain common (and possibly coded) information to all the users in order to accommodate these requests. Since in the prefetching phase, the requests in the later phase are unknown a-prior, the cached contents must be strategically prepared at all the users. The goal is to minimize the amount of multicast information which has rate RF (or equivalently the normalized rate of R), under the constraint on the normalized cache memory M. There is a natural tradeoff between the amount of cache memory and the delivery transmission rate, which is often referred to simply as the memory-rate tradeoff or the (M, R) tradeoff. It was shown in [39] that in terms of this memory-rate tradeoff, coding can be rather beneficial, while solutions based on uncoded prefetching and delivery will suffer a significant loss. Subsequent works extended it also to decentralized caching placements [87], caching with nonuniform demands [88], online caching placements [89], and hierarchical coded caching [90], and many others.

Recently, there were quite a few works [2, 39, 77, 91–100] aiming to find better codes with improved memory-rate tradeoff and the purpose is to find the optimal codes and thus completely characterize the fundamental limit of this tradeoff. Yu *et al.* [100] proposed a strategy that is optimal when prefetching is restricted to be uncoded, which in fact directly improves on the scheme in [39]. The key insight is that the original delivery strategy in [39] may have redundancy in the transmissions, which can be systematically removed to reduce the delivery rate in some cases. Tian and Chen [77] proposed a coded prefetching and the corresponding delivery strategy, which relies on a combination of rank metric codes and maximum distance separable (MDS) codes in a non-binary finite field. In the regime when the memory size M is relatively small, the scheme in [77] can achieve a better performance than that in [100]. Another code construction using coded prefetching was proposed more recently by Gómez-Vilardebó in [101], which can provide further improvement, over the schemes in [100] and [77], in the low memory regime for a specific range of (N, K). The characterization of the fundamental limit of the memory-rate tradeoff however remains open, which appears to require both improved coding schemes and stronger outer bounding techniques [92, 102–104]. Schemes for both uncoded prefetching and coded prefetching have been proposed and various outer bounds have also been discovered [92, 102–104]. Nevertheless, except a few special cases, the fundamental limit of coded caching systems still remains unknown.

Information theory provides a natural framework and a rich set of tools to determine the fundamental limits of communication systems and information systems. The derivation of such fundamental limits (or outer bounds) in a coding problem, requires a strategic combination of information equalities and inequalities. The most often used set of information inequalities are the Shannon-type inequalities, which were first formally identified by Yeung [105]. Moreover, Yeung identified the minimal set of such inequalities, referred to as elemental inequalities. Although there exist non-Shannon-type inequalities [106], in many cases, Shannon-type inequalities together with problem-specific constraints are sufficient to produce the (true) fundamental limits, particularly for systems with strong symmetries [107–109].

Since Shannon-type inequalities and many problem specific constraints are linear in the joint entropies, the overall derivation of the fundamental limits or outer bounds can be viewed as a series of linear programs where the variables are these joint entropies, and can in principle be solved using any linear program (LP) solver. However, for practical engineering problems, the resultant LPs are usually very large, which make solving them numerically impossible. In a recent work [110], it was shown that symmetry (together with other problem-specific reductions) can be effectively utilized to reduce the scale of the LP, which led to a conclusive solution for the $(4, 3, 3)$ regenerating code problem, and moreover, proved the existence of a fundamental difference between functional repair regenerating codes and exact repair regenerating codes. The intuition behind the reduction is that, the inherent symmetry structure in the problem implies the existence of symmetric optimal solutions, and thus many LP variables can be assumed to have the same values, and as a consequence there is no need to represent them differently. This further induces a significant reduction in terms of the LP constraints, and a computer solver can be used on this much reduced problem. Although the intuition itself is straightforward, our understanding on the amount of reduction through symmetry is limited. In this work, we seek to develop a better understanding on this issue.

Although the computational complexity in many modern solvers is more directly related to the number of nonzero elements in the LP [111], the scale of the LP are also good indicators of the LP computational scale, particularly in the context we are interested in. There is in fact a simple method to estimate these quantities. The group action defined through the permutation group on the power set of the random variables in the problem induces certain equivalence classes, and the maximum number of elements in any class is the number of permutations in the specific problem setting. Therefore, lower bounds to the two quantities can be obtained by dividing the total number of LP variables and that of LP constraints by the number of permutations. However, since some orbits can have fewer elements, these lower bounds are not precise. A method to pinpoint the symmetry structure and count these numbers accurately is thus needed.

Pólya counting theorem is a powerful tool to tackle this type of orbit counting problem which requires finding the cycle index of the group action. Our main task thus reduces to first identifying the group and group action, and then finding the cycle index.

In practical systems where a central coordinating mechanism can be costly, e.g., in more dynamic and mobile environments, decentralized coding becomes necessary. In this setting, rather than letting the server centrally control the placement of cached contents, each user independently determine the prefetching contents [87, 95, 100]. Decentralized coded caching has been studied under other settings, non-uniform demand was studied in [88], random demand in [112], online caching in [89], distinct cache capacities in [113] and various other delivery schemes in [114, 115].

Given the current state of the art, a natural question to ask is whether it is possible to extend the coded prefetching strategy in [77] from the centralized setting to a decentralized setting. Moreover, since the codes in [77] is not binary, it is anticipated that the coding overhead will increase, and thus it is important to understand how to reduce its impact.

1.4 Dissertation Outline

In Chapter 2, we show that the scheme in [77] can be slightly modified, where the MDS code used in the delivery phase can be replaced by a code using only binary additions (XOR). Though the alternative perspective itself does not provide further improvement on the known memory-rate tradeoff, it allows us to make a conceptual connection between the scheme in [77] and that in [100]. It further enables us to view these two schemes as the extremes of a more general scheme. The intermediate operating points of this more general scheme can indeed provide new tradeoff points previously not known in the literature, which we demonstrate using an explicit example for $(N, K) = (3, 4)$. Extending this example, a general explicit code construction is then rigorously presented and analyzed, which provides a new inner bound to the fundamental limit of the memory-rate tradeoff region in the caching problem. The inner bound does not have a simple closed form expression, but can be represented as a linear program to facilitate its computation.

In Chapter 3, we study single-demand type systems, where during the placement phase, the users and server know a priori that the demand vector in the delivery phase must be of a given demand type. Clearly, single-demand type systems have a more relaxed coding requirement than the original setting, however, it is still highly nontrivial since for each single demand type, a large number of different demand vectors are possible, with a rich set of symmetry relations among them [92]. Because of the relaxed coding requirement, any inner bound for the fully mixed demand type system will also be an inner bound for a single demand system, however, an outer bound for the fully mixed demand type systems may not be a valid outer bound for the single demand type systems.

Our first step is to collect the best-known inner bounds and outer bounds for both fully mixed demand type systems and single demand type systems in the literature for the canonical $(N, K) = (3, 3)$ system, some of which are from very recent developments [97, 98] in the area. This exercise reveals important insight and fundamental differences between the different classes of systems. It is shown that the demand type where all files are requested poses the most significant challenged in terms of characterizing the fundamental limit, however, codes designed for this demand type can in fact achieve (M, R) pairs that are strictly impossible for another demand type. This is contrary to popular belief that such a demand type is the “worst case", and also confirms that there is indeed a tension for codes designed for different demand types, and a fully mixed system would need to balance such conflicting interests. Next we focus on the case $N = K$ and for the demand type where all files are requested, and propose a new code construction based on a novel sub-packetization design. The construction is a generalization of the code recently proposed in [98], however, in contrast to the code specific designed for $N = K = 3$ or 4 that yields a single (M, R) pair each, our construction is for general N when $N = K$ which can produce multiple new (M, R) points. Finally, we consider outer bounds for such single demand type systems, where several existing bounds are first verified to be valid for this relaxed setting, and additionally a new outer bound is identified; the outer bounds indeed match the proposed scheme in some cases.

In Chapter 4, we study the outerbound problem of coded caching network, specifically, the symmetry of information inequalities when used to derive the outerbound of caching network.

We propose a generic three-layer decomposition of the group structure in typical coding problems and information systems, which allows us to conveniently apply the Pólya counting theorem. In addition, we apply this approach to two other similar problems: the problem of extremal pairwise cyclically symmetric entropy inequalities in [116, 117] and the regenerating code problem [110, 118, 119]. For all these problems, we provide explicit formulae for or efficient algorithms to compute the two quantities of interest. In fact, two of them lead to cycle indices previously studied in the literature [120–122], while the other appears less well studied before.

The three-layer decomposition not only fits the three problems we study here, but also is well motivated in general. The first layer, referred to as the base layer, can be viewed as reflecting certain physically meaningful permutations of the components in the communication or information systems, while the second layer is a permutation induced by the base layer on the random variables representing more abstract relations among the system components. The third layer is on the power set of the random variables, which directly relates to the joint entropies.

It should also be clarified that the cycle index is a concept associated with a group action, and a group acting on different sets may induce different cycle indices. A more intuitive interpretation of a group action is through its permutation representation, which in fact directly relates to the cycle index. In this chapter, we make a conscientious effort to reduce explicit reliance on the notion of group actions, but rather favor permutation representations because of the explicit physical interpretation. We will occasionally revert to group actions, which sometimes are more concise, and may be more meaningful for readers familiar with the mathematical tools developed using group actions.

The proposed method can be used to calculate the numbers of LP variables and LP constraints after symmetry reduction, while the aforementioned estimates are inaccurate. Numerically we observe that their relative difference becomes negligible as they both grow large, and thus our result is more reassuring than surprising in nature. We only consider the reduction due to the symmetry, which does not include other possible reductions, for example, reductions due to implication relations among the random variables. Such reductions were indeed utilized in [110] and can in fact be rather significant, however, it is difficult to identify the general amount of reduction since it is highly problem dependent; this topic is thus beyond the scope of our current study. Furthermore, our focus is on the reduction of the LP variables and the LP constraints, not the isomorphic relation among asymmetric problem instances, nor the symmetry in the geometry of the constrained entropic polytope; results related to these aspects can be found in [123, 124]. A less obvious but important byproduct of our study is the formalization of the permutation representations of the symmetry in the three problems, which are in fact needed in representing the problem in a computer program, and may be of value to researchers interested in implementing such software.

In Chapter 5, we propose a decentralized coded placement with prefetching in a binary extension field $\mathbb{F}_{2^m}$, and an efficient delivery scheme in the base binary field. We show that this decentralized caching scheme can achieve improvement over the decentralized strategy in [87, 100]. Moreover, we provide methods to reduce or balance the impacts of coding overheads.

Chapter 6 concludes this dissertation.

2. FUNDAMENTAL LIMITS OF CODED CACHING: FROM UNCODED PREFETCHING TO CODED PREFETCHING[*]

Caching can be used to relieve contention on communication resources by prefetching data to a local or fast memory space, and thus avoiding data retrieval from the remote or slower data source during peak traffic time. Traditionally, caching has mainly been considered in single user settings, *e.g.,* on-CPU caches vs. RAM in computers, where the hit-ratio is the key measure of performance. As networked systems become more prevalent, caching systems involving multiple users have attracted increasingly more research attention.

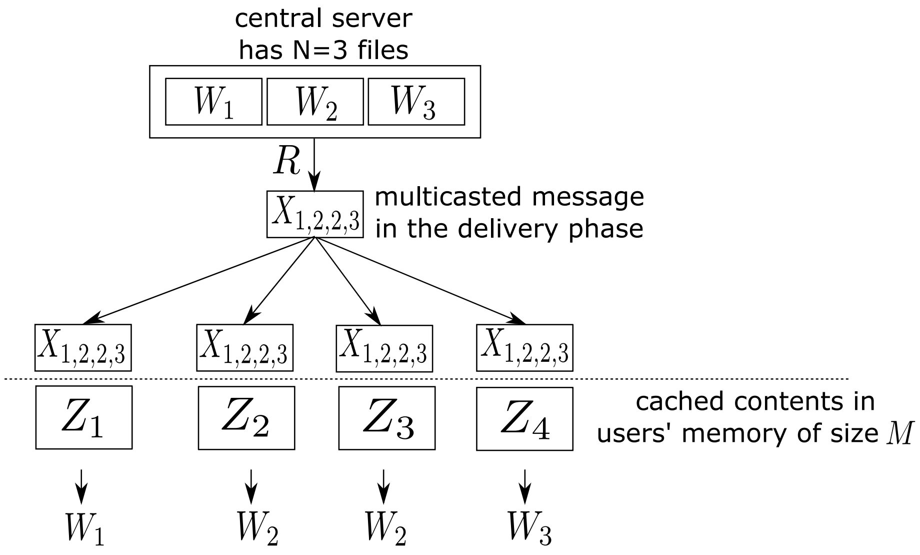

Figure 2.1: An example caching system instance, where there are $N = 3$ files, denoted as (W_1, W_2, W_3), and $K = 4$ users, whose cached contents are (Z_1, Z_2, Z_3, Z_4), respectively. In this instance the users request files (W_1, W_2, W_2, W_3), respectively. Reprinted with permission from [1, 2], © 2018 IEEE.

[*]© 2018 IEEE. Reprinted, with permission, from Kai Zhang and Chao Tian, "From Uncoded Prefetching to Coded Prefetching in Coded Caching Systems", *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 2087-2091, 2018.
© 2018 IEEE. Reprinted, with permission, from Kai Zhang and Chao Tian, "Fundamental limits of coded caching: From uncoded prefetching to coded prefetching", *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp.1153-1164, 2018.

In their award-winning article [39], Maddah-Ali and Niesen provided a formal information theoretic formulation for the caching problem in multiuser settings (Figure 2.1). In this formulation, there are N files, each of F bits, and K users. Each user has a local cache memory of capacity MF (thus a normalized capacity of M). In the prefetching (or sometimes referred to as the placement) phase, the users can fill their caches with contents from the central server without the knowledge of the precise requests at the deliver phase. In the delivery phase, each user reveals the request for a single file from the central server, and the central server must multicast certain common (and possibly coded) information to all the users in order to accommodate these requests. Since in the prefetching phase, the requests at the later phase are unknown a-prior, the cached contents must be strategically prepared at all the users. The goal is to minimize the amount of multicast information which has rate RF (or equivalently the normalized rate of R), under the constraint on the normalized cache memory M. There is a natural tradeoff between the amount of cache memory and the delivery transmission rate, which is often referred to simply as the memory-rate tradeoff or the (M, R) tradeoff. It was shown in [39] that in terms of this memory-rate tradeoff, coding can be rather beneficial, while solutions based on uncoded prefetching and delivery will suffer a significant loss. Subsequent works extended it also to decentralized caching placements [87], caching with nonuniform demands [88], online caching placements [89], and hierarchical coded caching [90], and many others.

There were quite a few recent efforts [77, 93–96, 99–101] aiming to find better codes with improved memory-rate tradeoff, toward the eventual goal of finding the optimal codes and thus completely characterizing the fundamental limit of this tradeoff. In particular, Yu *et al.* [100] proposed a strategy that is optimal when prefetching is restricted to be uncoded, which in fact directly improves on the scheme in [39]. The key insight in [100] appears to be that the original delivery strategy in [39] may have redundancy in the transmissions, which can be systematically removed to reduce the delivery rate in some cases. In another recent work, Tian and Chen [77] proposed a coded prefetching and the corresponding delivery strategy, which relies on a combination of rank metric codes and maximum distance separable (MDS) codes in a non-binary finite field. In the regime when the memory size M is relatively small, the scheme in [77] can achieve a better performance than that in [100]. Another code construction using coded prefetching was proposed more recently by Gómez-Vilardebó in [101], which can provide further improvement, over the schemes in [100] and [77], in the low memory regime for a specific range of (N, K). The characterization of the fundamental limit of the memory-rate tradeoff however remains open, which appears to require both improved coding schemes and stronger outer bounding techniques [92, 102–104].

In this chapter, we show that the scheme in [77] can be slightly modified, where the MDS code used in the delivery phase can be replaced by a code using only binary additions (XOR). Though the alternative perspective itself does not provide further improvement on the known memory-rate tradeoff, it allows us to make a conceptual connection between the scheme in [77] and that in [100]. It further enables us to view these two schemes as the extremes of a more general scheme. The intermediate operating points of this more general scheme can indeed provide new tradeoff points previously not known in the literature, which we demonstrate using an explicit example for $(N, K) = (3, 4)$. Extending this example, a general explicit code construction is then rigorously presented and analyzed, which provides a new inner bound to the fundamental limit of the memory-rate tradeoff region in the caching problem. The inner bound does not have a simple closed form expression, but can be represented as a linear program to facilitate its computation.

The rest of the chapter is organized as follows. Section 2.1 first provides some necessary background on existing results and rank metric codes, and then introduces the notion of transmission type. A critical observation is given in Section 2.2 which connects the two classes of schemes as extreme cases, and then a new memory-rate pair previous unknown in the literature is produced by considering the intermediate cases for $(N, K) = (3, 4)$. The new inner bound is given formally in Section 2.3, and the corresponding coding scheme, its analysis, the proof of the correctness are given in Section 2.4. A technical proof is relegated to the appendix.

2.1 Relevant Results and Preliminaries

In this section, we first briefly review existing results on the coded caching problem, and then provide necessary background on rank metric codes, which serve an instrumental role in the new code construction. A new concept important in our code construction, *i.e.,* the transmission type, is then introduced.

2.1.1 Existing Schemes Using Uncoded Prefetching

The scheme in [39], which uses uncoded prefetching, can achieve the following memory-rate pairs

$$(M, R) = \left(\frac{tN}{K}, \frac{K-t}{1+t}\right), \quad t = 0, 1, \ldots, K, \tag{2.1}$$

and since another trivial point is clearly $(M, R) = (0, N)$, the lower convex hull of them provides an upper bound to the optimal tradeoff, as stated in [39]. More recently, Yu *et al.* [100] gave a scheme which achieves the memory-rate tradeoff points of

$$(M, R) = \left(\frac{tN}{K}, \frac{\binom{K}{t+1} - \binom{K-\min\{K,N\}}{t+1}}{\binom{K}{t}}\right), \quad t = 0, 1, \ldots, K. \tag{2.2}$$

These points strictly improve the rate component R in (2.1) when $K - N \geq t + 1$. Both schemes in [39] and [100] use the same uncoded prefetching strategy, but the delivery strategy in [100] is a direct improvement to that in [39]. It was shown in [100] that in the restricted class of schemes where only uncoded prefetching is allowed, the tradeoff provided in (2.2) is in fact optimal. These two coding schemes can roughly be understood as follows.

Choose a fixed integer t, where $0 \leq t \leq K$, and partition each file into $\binom{K}{t}$ segments of equal size; each segment is thus uniquely associated with a cardinality-t subset $\mathcal{S}$ of the full user set $\{1, 2, \ldots, K\}$, and this segment is placed in the caches of users in $\mathcal{S}$ during the prefetching phase. During the delivery phase, consider each $(t + 1)$ subset $\mathcal{B}$ of users: within this group, each user is requesting a segment that is in all the other users' caches, and the server thus sends the XOR of all such segments of this group. Each user in this group can recover their respectively desired segment, since all other segments involved in this transmission are known to this user. As mentioned earlier, these transmissions as a whole, taken over all the possible choices of $\mathcal{B}$, may in fact have redundancy among themselves (*i.e.*, they are linearly dependent) for certain (N, K) parameters, and eliminating such redundancy results in the scheme in [100].

Let us examine an example with $N = 3$ files, denoted as A, B, C, respectively, and $K = 4$ users. Set the auxiliary variable $t = 2$, then each file is partitioned into $\binom{4}{2} = 6$ segments, for example, file A has segments $A_{1,2}, A_{1,3}, A_{1,4}, A_{2,3}, A_{2,4}, A_{3,4}$, and the segment $A_{1,2}$ is given to users 1 and 2, etc.. Suppose now the users' requests are (A, A, B, C), *i.e.,* the first two users request file A, the third user requests file B, and the fourth user requests file C. Consider the set of users $\mathcal{B} = \{1, 2, 3\}$, associated with which we should transmit $A_{2,3} + A_{1,3} + B_{1,2}$ according to the coding scheme discussed above, where the addition is in the binary field. Clearly the three users in $\mathcal{B}$ can recover their individually desired segments, *i.e.,* $A_{2,3}, A_{1,3}, B_{1,2}$, respectively. For other subsets of users, the transmissions are formed similarly, and the complete set of delivery transmissions is

$$A_{2,3} + A_{1,3} + B_{1,2}, A_{2,4} + A_{1,4} + C_{1,2}, A_{3,4} + B_{1,4} + C_{1,3}, A_{3,4} + B_{2,4} + C_{2,3}. \tag{2.3}$$

In this particular case, the transmissions of (2.3) do not have any redundancy.

The schemes in both [95] and [96] use uncoded prefetching, and since the scheme in [100] is optimal for this class of codes, the two schemes in [95] and [96] do not provide any additional improvement over (2.2).

2.1.2 Existing Schemes Using Coded Prefetching

Even in the pioneering work [39], it was observed that uncoded prefetching schemes are not sufficient to characterize the fundamental limit of the memory-rate tradeoff, and one code example using coded prefetching was given for the case $(N, K) = (2, 2)$ as an illustration. In [93], Chen *et al.* extended this example to the general case $N \leq K$, and showed that the single memory-rate pair $\left(\frac{1}{K}, \frac{N(K-1)}{K}\right)$ is achievable and in fact optimal.

More recently, Tian and Chen [77] proposed a more general scheme with coded prefetching for

$N \leq K$. It was shown that the scheme can achieve the memory-rate tradeoff pairs

$$(M, R) = \left(\frac{t[(N-1)t + K - N]}{K(K-1)}, \frac{N(K-t)}{K} \right), \quad t = 0, 1, \ldots, K. \tag{2.4}$$

With $t = 1$, it produces exactly the memory-rate pair given in [93].

The general scheme in [77] is somewhat involved, but the digest is as follows. Each file is again partitioned into $\binom{K}{t}$ segments of equal size, and given to the relevant users as in [100]; however, instead of directly storing them, each user caches certain linear combinations of these corresponding segments, mixed across all the files. During delivery, each symbol being transmitted is a linear combination of the segments from a single file, that serves two roles: firstly, the segments forming a single linear combination being transmitted are all present at certain user's cache that is not requesting this file, thus this user can use it to help resolve the cached symbols when sufficient such transmissions are collected; secondly, these segments are not present at some users which are requesting that file, thus can also help them to recover the missing segments. In order to guarantee the decodability, the cached contents and the transmitted contents should be made linearly independent, and for this purpose, rank metric codes can be utilized to produce the cached linear combinations, and MDS codes can be used to produce the delivery transmissions.

Let us consider again the example $(N, K) = (3, 4)$ and $t = 2$. In this case, the linear combinations of the segments

$$A_{1,2}, A_{1,3}, A_{1,4}, B_{1,2}, B_{1,3}, B_{1,4}, C_{1,2}, C_{1,3}, C_{1,4} \tag{2.5}$$

are placed at user 1's cache, where each segment is viewed as a symbol in a large finite field. According to the scheme in [77], there should be a total of 5 linear combinations cached; the coefficients of these linear combinations are not critical in this construction, for which either deterministic rank metric codes can be used, or random assignments can be used with a high probability of being a valid choice in a sufficiently large finite field (which implies the existence of a deterministic assignment). Now consider again the requests (A, A, B, C). In this case, the server will send the following 9 symbols

$$A_{3,4}, B_{1,2}, B_{1,4}, B_{2,4}, C_{1,2}, C_{1,3}, C_{2,3}, A_{1,3}+A_{2,3}, A_{1,4}+A_{2,4}, \tag{2.6}$$

where the addition is in the same finite field of the information symbol which is usually not binary. The last two linear combinations can also be viewed as the parity symbols of two MDS codes. Now user 1 collects from (2.6) the symbols $B_{1,2}, B_{1,4}, C_{1,2}, C_{1,3}$, which, together with 5 cached linear combinations, leads to a total of 9 linear combinations of the basis in (2.5). Since the linear combinations are designed to be linearly independent, all of the symbols can be resolved. User 1 then collects $A_{1,3}+A_{2,3}, A_{1,4}+A_{2,4}$ from which $A_{2,3}$ and $A_{2,4}$ can be recovered by eliminating $A_{1,3}$ and $A_{1,4}$, since they have been resolved from the cached content. It can be verified in a similar manner that all other users can also recover the requested files, and for any other demand patterns, transmissions of 9 symbols will always suffice. The memory-rate pair achieved by the scheme in [39] is $(M,R)=(\frac{3}{2},\frac{2}{3})$ while the scheme in [77] gives $(M,R)=(\frac{5}{6},\frac{3}{2})$, which are illustrated in Fig. 2.2.

Amiri and Gunduz [99] showed that the following tradeoff point is achievable when $N \leq K$

$$(M,R)=\left(\frac{N-1}{K},\frac{N(2K-N)}{2K}\right), \tag{2.7}$$

using a coded prefetching scheme. However, it can be verified that the pair (M,R) in (2.7) is precisely on the time-sharing line between (2.2) and (2.4) with $t=1$. More recently, Gómez-Vilardebó [101] showed that the following memory-rate pairs are achievable:

$$(M,R)=\left(\frac{N}{Kg},N-\frac{N(N+1)}{K(g+1)}\right), \quad g=1,...,N, \tag{2.8}$$

which can offer further improvement when $N \leq K \leq (N^2+1)/2$. The lower convex hull of (2.2), (2.4) and (2.8) provides the best known upper bound to the fundamental limit of (M,R) tradeoff known in the literature.

2.1.3 Linearized Polynomial and Rank Metric Codes

Similar as in [77], rank metric codes based on linearized polynomials (see [125]) can be used to facilitate our code constructions. The following lemma is relevant in this regard; see, *e.g.,* [126].

Lemma 1. *A linearized polynomial in finite field $\mathbb{F}_{q^m}$*

$$f(x) = \sum_{i=1}^{P} v_i x^{q^{i-1}}, \; v_i \in \mathbb{F}_{q^m} \tag{2.9}$$

can be uniquely identified from evaluations at any P points $x = \theta_i \in \mathbb{F}_{q^m}$, $i = 1, 2, \ldots, P$, that are linearly independent over $\mathbb{F}_q$.

Another relevant property of linearized polynomials is that they satisfy the following condition

$$f(ax + by) = af(x) + bf(y), \; a, b \in \mathbb{F}_q, \; x, y \in \mathbb{F}_{q^m}, \tag{2.10}$$

which is the reason that they are called "linearized". This property implies the following lemma, the proof of which can be found in [77].

Lemma 2. *Let $f(x)$ be a linearized polynomial in $\mathbb{F}_{q^m}$ as given in (2.9), and let $\theta_i \in \mathbb{F}_{q^m}$, $i = 1, 2, \ldots, P_o$, be linearly independent over $\mathbb{F}_q$. Let G be a $P_o \times P$ full rank (rank P) matrix with entries in $\mathbb{F}_q$, then $f(x)$ can be uniquely identified from*

$$[f(\theta_1), f(\theta_2), \ldots, f(\theta_{P_o})] \cdot G. \tag{2.11}$$

With a fixed set of $\theta_i \in \mathbb{F}_{q^m}$, $i = 1, 2, \ldots, P_o$, which are linearly independent, we can view $(v_1, \ldots, v_P)$ as information symbols to be encoded, and the evaluations $[f(\theta_1), f(\theta_2), \ldots, f(\theta_{P_o})]$ as the coded symbols. This is a (P_o, P) MDS code in terms of rank metric [125], where $P_o \geq P$. More importantly, the above lemma says any full rank (rank P) $\mathbb{F}_q$ linear combinations of the coded symbols are sufficient to decode all the information symbols. This linear-transformation-invariant property had been utilized previously in other coding problems such as network coding with errors and erasures [127], locally repairable codes with regeneration [128], and layered regenerating codes [129].

The codes thus obtained are not systematic, but they can be converted to systematic codes by viewing the information symbols $(w_1, w_2, \ldots, w_P)$ as the first P evaluations $[f(\theta_1), f(\theta_2), \ldots,$ $f(\theta_P)]$, which can be used to find the coefficients of the linearized polynomial $(v_1, v_2, \ldots, v_P)$, and then the additional parity symbols can be generated by evaluating this linearized polynomial at the remaining points $(\theta_{P+1}, \ldots, \theta_{P_o})$. Systematic rank-metric codes are instrumental in our construction.

2.1.4 Demand Vectors and Transmission Types

Denote the N files in the system as $W_1, W_2, \ldots, W_N$, and denote the demands by the users in the delivery phase as $\boldsymbol{d} = (d_1, d_2, \ldots, d_K)$, where $d_k \in \{1, 2, ..., N\}$ is the index of the file that user-k requests. For convenience, denote the set $\{1, 2, \ldots, n\}$ as I_n. Recall that once the auxiliary parameter t is fixed, each file W_n in the scheme of [39] is the collection of all segments $W_{n,\mathcal{S}}$ where $\mathcal{S} \subseteq I_K$ and $|\mathcal{S}| = t$, where $|\mathcal{S}|$ is the cardinality of the set $\mathcal{S}$. For a given demand vector $\boldsymbol{d} = (d_1, d_2, \ldots, d_K)$, denote the set of users requesting file W_n as

$$I^{[n]} \triangleq \{k \in I_K : \text{user } k \text{ requests file } W_n\}, \qquad n = 1, 2, \ldots, N. \tag{2.12}$$

Further define $m_n \triangleq |I^{[n]}|$, $n = 1, 2, \ldots, N$. In the coding scheme we shall present, an arbitrary element (for example, the minimum element) in $I^{[n]}$ will be chosen, denoted as $\ell^{[n]}$, as the leader of $I^{[n]}$. The support of vector $\boldsymbol{m}$ is written as $\mathbb{supp}(\boldsymbol{m})$, *i.e.*, $\mathbb{supp}(\boldsymbol{m}) = \{n | m_n > 0\}$, and its cardinality is denoted as $N^* = |\mathbb{supp}(\boldsymbol{m})|$, which is the number of files being requested in $\boldsymbol{d}$. For convenience, also define $\tilde{N} \triangleq \min(N, K)$.

The notion of the *transmission type* is associated with each transmission in the scheme in [39]. For a set of users $\mathcal{B} \subseteq I_K$ where $|\mathcal{B}| = t + 1$, the associated delivery transmission in the scheme of [39], for a fixed demand vector $(d_1, d_2, \ldots, d_K)$, can be compactly written as the binary field summation

$$\oplus_{k\in\mathcal{B}} W_{d_k,\mathcal{B}\setminus k}. \tag{2.13}$$

Each such transmission, or alternatively the subset $\mathcal{B}$, is thus associated with an N-dimensional vector $\boldsymbol{t}$, whose n-th coordinate t_n specifies the number of users that are demanding file W_n in the set $\mathcal{B}$. We call this vector the transmission type of the subset $\mathcal{B}$. For example, in the $(3,4)$ case discussed above when the demand vector is $(W_1, W_2, W_3, W_4) = (A, A, B, C)$, the transmission type of the user set $\mathcal{B} = \{1,2,3\}$ is $\boldsymbol{t} = (2,1,0)$, and the exact transmission is $A_{2,3} + A_{1,3} + B_{1,2}$ where there are exactly two $W_1 = A$ symbols involved and one $W_2 = B$ symbol involved. Similarly, the transmission types of the user sets $\{2,3,4\}$ and $\{1,3,4\}$ are both $\boldsymbol{t} = (1,1,1)$.

Denote the collection of all valid transmission types for a given demand vector $\boldsymbol{d}$ with the auxiliary parameter being t as $\mathcal{T}_{\boldsymbol{d}}^{(t)}$. It is clear that for any valid transmission type $\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{d}}^{(t)}$, we have $\sum_{n=1}^{N} t_n = t+1$, and thus the auxiliary parameter t can be uniquely determined from any valid $\boldsymbol{t}$. The support of a transmission type $\boldsymbol{t}$ is denoted as $\mathbb{supp}(\boldsymbol{t})$. With a slight abuse of notation, we write the transmission type of a given set $\mathcal{B} \subseteq I_K$, where $|\mathcal{B}| = t+1$, as $\mathcal{T}(\mathcal{B})$.

The notion of transmission type should be contrasted to the notion of *demand type* introduced in [91], which is a length-N vector formed by sorting $(m_1, m_2, \ldots, m_N)$. This notion is also important in our work, because the symmetry in the proposed code implies that only one demand vector per demand type needs to be considered. Denote the collection of the representative demand vectors, one representative demand vector per demand type, as $\mathcal{D}$.

2.2 A Hidden Connection and Partial Decomposition

The two schemes in [77] and [39] (and its improved counterpart [100]) may seem very different at the first sight: one uses coded prefetching and the other uncoded, one is non-binary code while the other is binary, and one relies on sophisticated coding techniques such as rank metric codes and the other only relatively simple combinatorics. Nevertheless, a closer look reveals some curious connections between the two schemes. For example, the tradeoff points in (2.4) lead to the rate values $R = \frac{N(K-t)}{K}$ for $t = 0, 1, \ldots, K$, which are exactly the same set of M values given in (2.2). This connection may or may not be a simple coincidence, however we next describe a much less obvious observation which leads to the main result of this paper.

2.2.1 A Hidden Connection

Consider again the example case for $(N, K) = (3, 4)$ and $t = 2$. Let us decompose the transmissions in (2.3) by separating different files in the same linear combination. For example, the linear combination $A_{2,3}+A_{1,3}+B_{1,2}$ is decomposed into a pair of transmissions $(A_{2,3}+A_{1,3}, B_{1,2})$. It can be verified that decomposing all the linear combinations in (2.3) in fact produces exactly the same set of linear combinations in (2.6), after removing the repeated transmissions. Thus in this example, the delivery transmissions in the scheme [77] can be obtained by fully decomposing the delivery transmissions of the scheme in [39], when the auxiliary parameter t is chosen to be the same in the two schemes.

We note that the addition in (2.6) is not in a binary field, while the addition in (2.3) is in the binary field. However, if a binary extension field $\mathbb{F}_{2^m}$ is used in (2.6), the delivery can indeed be accomplished using only additions of the information symbols in this binary extension field, *i.e.,* the coefficients of the linear combinations are either 0 or 1. Clearly, such additions are equivalent to additions in the base binary field, when the information and coded symbols are represented in their binary vector form.

2.2.2 Partial Decomposition and a New Code Example

The above observation naturally raises the following question: since the delivery strategy in the scheme of [77] can be viewed as being obtained from fully decomposing the delivery transmissions of the scheme in [39], will partial decomposition, with a correspondingly modified prefetching strategy, produce new memory-rate tradeoff pairs? Next we provide an example code, which shows that the answer to this question is indeed positive.

Consider again the case $(N, K) = (3, 4)$ and $t = 2$, but this time each user caches 8 (instead of 9 as in Sec. 2.1.1, or 5 as in Sec. 2.1.2) linear combinations of the information symbols of

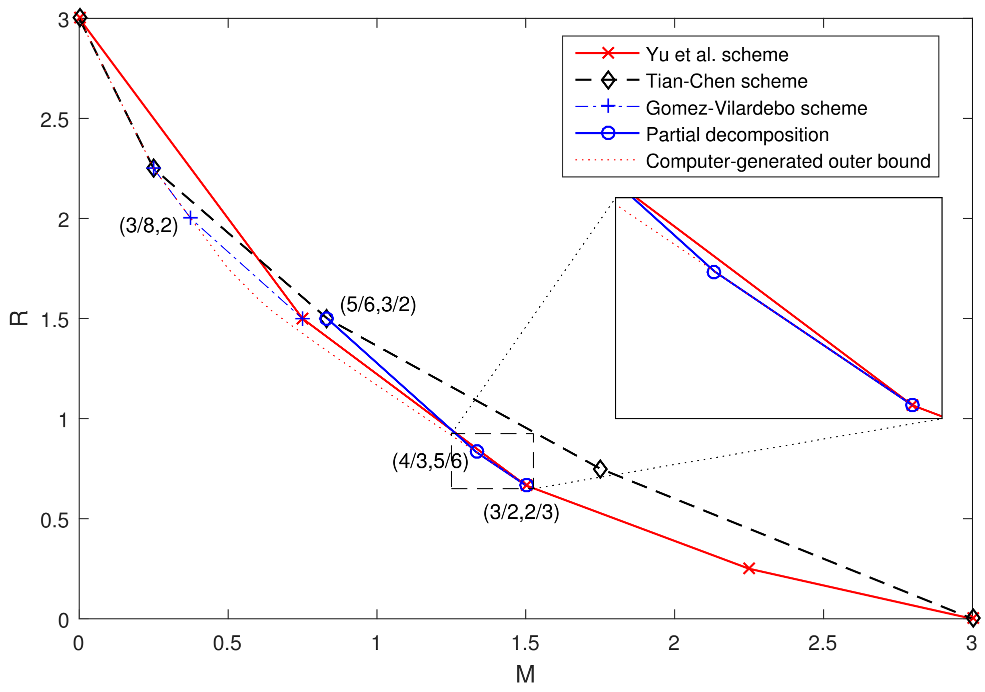

Figure 2.2: A new tradeoff point for $(N, K) = (3, 4)$. Reprinted with permission from [1, 2], © 2018 IEEE.

the corresponding uncoded file segments. The coefficients of the linear combinations can again be either from deterministic rank metric codes (see Section 2.4), or generated randomly in a large finite field.

We next argue that delivering a total of 5 coded symbols is sufficient in this case, which gives an achievable memory-rate pair $(M, R) = (4/3, 5/6)$. The memory-rater pair is strictly better than $(4/3, 23/27)$ achieved by the lower convex hull of the schemes [77, 100, 101], which is currently the best known upper bound in the literature; see Fig. 2.2 for an illustration. For completeness, a computer-generated outer bound is also included in the figure, which was obtained in a separate work [92]. Interestingly, both $(M, R) = (3/8, 2)$ given by the code in [101] and $(M, R) = (4/3, 5/6)$ obtained in this work are in fact on this outer bound, and thus optimal.

Due to the symmetry in the code, we only need to consider the demand vectors (A, A, B, C), (A, A, B, B), (A, A, A, C), and (A, A, A, A).

- For the demand (A, A, B, C), instead of fully decomposing the transmissions in (2.3), we now partially decompose them as

$$A_{2,3}+A_{1,3}+B_{1,2}, A_{2,4}+A_{1,4}+C_{1,2}, B_{1,4}+C_{1,3}, B_{2,4}+C_{2,3}, A_{3,4}. \tag{2.14}$$

User 1 first collects $B_{1,4}+C_{1,3}$, and thus together with the 8 cached linear combinations, can resolve all the symbols in (2.5) since he has a total of 9 linearly-independent linear combinations of the 9 symbols; now user 1 essentially has uncoded cache contents, and thus can recover the needed file segments of A (which are $A_{2,3}, A_{2,4}, A_{3,4}$) using the remaining transmissions. Users 2, 3, and 4 can use a similar strategy.

- For the demand (A, A, B, B), we can transmit the following symbols

$$A_{2,3}+A_{1,3}+B_{1,2}, A_{2,4}+A_{1,4}+B_{1,2}, B_{1,4}+B_{1,3}, B_{2,4}+B_{2,3}, A_{3,4}. \tag{2.15}$$

User 1 can collect $B_{1,4}+B_{1,3}$ in order to resolve the cached symbols, and the decoding is similar to the previous case. It is also obvious user 3 and user 4 can indeed recover file B.

- For the demand (A, A, A, C), we can transmit

$$A_{2,3}+A_{1,3}+A_{1,2}, A_{2,4}+A_{1,4}+C_{1,2}, A_{1,4}+C_{1,3}, A_{2,4}+C_{2,3}, A_{3,4}. \tag{2.16}$$

The decoding strategies of other users are similar to the previous cases, and let us only consider user 3 as an illustration. User 3 can use $A_{3,4}$ to resolve the symbols in the cache, then recover $A_{1,4}$ from $A_{1,4}+C_{1,3}$, $A_{2,4}$ from $A_{2,4}+C_{2,3}$, and $A_{1,2}$ from $A_{2,3}+A_{1,3}+A_{1,2}$.

- For the demand (A, A, A, A), we can transmit

$$A_{2,3}+A_{1,3}+A_{1,2}, A_{2,4}+A_{1,4}+A_{1,2}, A_{1,4}+A_{1,3}, A_{2,4}+A_{2,3}, A_{3,4}. \tag{2.17}$$

Using a similar strategy as above, it is seen that all users can indeed recover file A.

In this example case, the delivery transmissions are obtained by partially decomposing the transmissions in the scheme of [39], and in compensation, the number of cached linear combinations in users' memory is reduced from that of [39]. The number of linear combinations stored in the cache needs to guarantee that the coded symbols can all be resolved to their uncoded form, after a sufficient number of symbols have been collected from the delivery. The rest of the paper is devoted to the task of using this idea to build a general class of codes which yield a new inner bound to the memory-rate tradeoff.

2.3 A New Inner Bound to the Optimal Memory-Rate Tradeoff

We first formally define the partial decomposition patterns, and then present the new inner bound. The prefetching strategy and the delivery strategy behind this new bound are presented and analyzed in the next section.

2.3.1 A Formal Description of Partial Decomposition

Fix the auxiliary parameter $t \in I_K$, and for now also consider a fixed demand vector $\boldsymbol{d}$. A valid partial decomposition pattern on a transmission type $\boldsymbol{t}$ is specified by a partition $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ on $\mathbb{supp}(\boldsymbol{t})$, *i.e.,* the elements of $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ are mutually exclusive and jointly exhaustive subsets of $\mathbb{supp}(\boldsymbol{t})$. For a given transmission type $\boldsymbol{t}$ and its partial decomposition pattern $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$, the decomposed transmissions are formed by keeping the symbols in the same partition in $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ together, but those across partitions separated. More precisely, let $\mathcal{T}(\mathcal{B}) = \boldsymbol{t}$, then the transmission (2.13) can be rewritten and thus decomposed as

$$\begin{aligned} \oplus_{k\in\mathcal{B}} W_{d_k,\mathcal{B}\setminus k} &= \oplus_{\mathcal{P}\in\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}} \left[\oplus_{n\in\mathcal{P}} \left(\oplus_{k\in\mathcal{B}\cap I^{[n]}} W_{n,\mathcal{B}\setminus k}\right)\right] \\ &\Rightarrow \quad \oplus_{n\in\mathcal{P}} \left(\oplus_{k\in\mathcal{B}\cap I^{[n]}} W_{n,\mathcal{B}\setminus k}\right), \ \mathcal{P} \in \mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}, \end{aligned} \tag{2.18}$$

where $\mathcal{P} \subseteq \mathbb{supp}(\boldsymbol{t})$ is used to enumerate over the partitions specified by $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$. Note that $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ is the decomposition pattern for a transmission type $\boldsymbol{t}$ (and a demand vector $\boldsymbol{d}$), which implies that the transmissions of the same transmission type are not allowed to use different decomposition patterns. In order to specify the delivery transmissions for a given demand vector $\boldsymbol{d}$, the decomposition patterns for all transmission types should be given, which are written as a set $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)} \triangleq \{\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} | \boldsymbol{t} \in \mathcal{T}_{\boldsymbol{d}}^{(t)}\}$.

Consider again the example for $(N, K) = (3, 4)$: suppose the demand vector is

$$\boldsymbol{d} = (A, A, B, C) = (1, 1, 2, 3),$$

and for the transmission type $\boldsymbol{t} = (1, 1, 1)$, the decomposition pattern is $\mathcal{P}_{(1,1,1),(1,1,2,3)} = \{\{1\}, \{2, 3\}\}$. With these settings the two transmissions $A_{3,4} + B_{1,4} + C_{1,3}$ and $A_{3,4} + B_{2,4} + C_{2,3}$ in the coding scheme [39] will be decomposed into $\{A_{3,4}, B_{1,4} + C_{1,3}\}$ and $\{A_{3,4}, B_{2,4} + C_{2,3}\}$, respectively.

For any demand vector $\boldsymbol{d}$, a special uncoded transmission pattern, denoted as $\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}$, is also allowed. When $K - t \geq \tilde{N}$, this strategy corresponds to directly transmitting a subset of files in the uncoded form. For general parameters, the transmission strategy will be given more precisely in Section 2.4.2. The introduction of this pattern is motivated by the coding strategy in [77] when $N^* < \tilde{N}$.

2.3.2 A New Inner Bound

Define the following quantity for any transmission pattern $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$ except $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)} = \breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}$

$$R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} \triangleq \sum_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{d}}^{(t)}} \sum_{\mathcal{P} \in \mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}} \left[\left(\prod_{n \in \mathcal{P}} \binom{m_n}{t_n} - \prod_{n \in \mathcal{P}} \binom{m_n - 1}{t_n} \right) \cdot \prod_{n \in \mathrm{supp}(\boldsymbol{t}) \setminus \mathcal{P}} \binom{m_n}{t_n} \right], \tag{2.19}$$

and for $k = 1, 2, \ldots, K$,

$$M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} \triangleq N \binom{K-1}{t-1} - \Delta M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k}, \tag{2.20}$$

where

$$\Delta M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} \triangleq \sum_{\substack{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}:\\ t_{d_k}>0}} \sum_{\substack{\mathcal{P}\in\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}:\\ d_k\notin\mathcal{P}}} \binom{m_{d_k}-1}{t_{d_k}-1} \cdot \left[\left(\prod_{n\in\mathcal{P}}\binom{m_n}{t_n} - \prod_{n\in\mathcal{P}}\binom{m_n-1}{t_n}\right) \cdot \prod_{n\in\mathbb{supp}(\boldsymbol{t})\setminus\{\mathcal{P}\cup\{d_k\}\}}\binom{m_n}{t_n}\right]. \tag{2.21}$$

For the special transmission pattern $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)} = \breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}$, the corresponding quantities are defined as

$$R_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}} \triangleq \min(K-t,\tilde{N})\binom{K}{t}, \tag{2.22}$$

and for $k = 1, 2, \ldots, K$,

$$M_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)},k} \triangleq N\binom{K-1}{t-1} - \Delta M_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)},k}, \tag{2.23}$$

where

$$\Delta M_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)},k} \triangleq \min(K-t,\tilde{N})\binom{K-1}{t-1}. \tag{2.24}$$

In the above, the following convention for the degenerate cases of combinatorics has been used

$$\binom{a}{b} = \begin{cases} 0, & \text{if } a < b \\ 1, & \text{if } a \geq 0 \text{ and } b = 0 \end{cases}. \tag{2.25}$$

Intuitively speaking, the vector $(M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},1}, \ldots, M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},K}, R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}})$ provides the cache memory requirements at the users and the rate requirement on the delivery transmission in the proposed coding scheme, when the demand vector $\boldsymbol{d}$ and the decomposition patterns $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$ are fixed. Thus these numbers roughly provide the memory-rate tradeoff for the specific demand vector for a fixed decomposition pattern.

We first observe that there may still be unbalance among the cache memory requirements at different users $(M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}},1},...,M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}},K})$, meaning that different users may have different cache memory requirements under the decomposition pattern $\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}$. This issue can be mitigated by coding across multiple instances in the prefetching phase and then producing the delivery transmissions using multiple different decomposition patterns on different instances to achieve better balance among users; note that simple space-sharing is not sufficient to achieve such a performance. A second important observation is that regardless the demand vectors or the decomposition patterns, the caching strategy can essentially be kept the same, which is to store a certain number of linear combinations of a fixed set of symbols. The two observations lead to the following definition and the main theorem below, the formal proof of which will be given in the sequel.

Define the region $\mathcal{R}^{(t)}$ to be the collection of the memory-rate pairs (M,R) such that there exists a set of real-valued $\{\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}}\}$ such that

$$\sum_{\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} \alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} = 1, \qquad \forall \boldsymbol{d}\in\mathcal{D} \tag{2.26}$$

$$\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} \geq 0, \qquad \forall \boldsymbol{d}\in\mathcal{D}, \forall \boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}} \tag{2.27}$$

$$\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} \leq 1, \qquad \forall \boldsymbol{d}\in\mathcal{D}, \forall \boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}} \tag{2.28}$$

$$\sum_{\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} \alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} \leq R\binom{K}{t}, \qquad \forall \boldsymbol{d}\in\mathcal{D} \tag{2.29}$$

$$\sum_{\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} \alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}},k} \leq M\binom{K}{t}, \qquad \forall \boldsymbol{d}\in\mathcal{D}, \forall k\in I_K. \tag{2.30}$$

The auxiliary variables $\{\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}}\}$ serve a similar role to the time-sharing variables, however the region cannot be directly obtained by the time-sharing argument, and is instead obtained by a slightly more elaborate coding approach. We need the following technical definition [*] to state the

[*]Here we allow a sequence of codes to achieve the (M,R) pair in an asymptotic manner, *i.e.,* approaches this normalized memory-rate pair as the size of the file grows to infinity. Strictly speaking, this approach of definition is not necessary for us, since the quantities we obtain in (2.19) and (2.20) are always integers, and thus the extreme points of the constrained polytope in (2.26)-(2.30) will always be rational, which can be achieved precisely using the proposed scheme. Then a time-sharing argument can be invoked to argue any irrational-valued memory-rate pairs in the region can be achieved. Definition 1 however avoids this line of argument altogether.

main result, which is a new inner bound to the memory-rate tradeoff region.

Definition 1. *A memory-rate pair (M, R) is called achievable, if for any $\delta > 0$, and for any sufficiently large file size F, there exists a code with a normalized memory size no greater than $M + \delta$ and a normalized transmission rate no greater than $R + \delta$.*

Theorem 1. *For any $t = 0, 1, 2, \ldots, K$, any $(M, R) \in \mathcal{R}^{(t)}$ is achievable. Consequenctly, the convex closure* $\mathbf{cl}\left(\cup_{t=0,\ldots,K}\mathcal{R}^{(t)}\right)$ *is achievable, where* $\mathbf{cl}(\cdot)$ *means the convex closure.*

The proof of this theorem will be given in Section 2.4. We also have the following corollary, whose proof is given in the appendix.

Corollary 1. *The memory-rate pairs in (2.2) and those in (2.4) are in the region* $\mathbf{cl}\left(\cup_{t=0,\ldots,K}\mathcal{R}^{(t)}\right)$.

Since $\mathcal{R}^{(t)}$ is a polytope constrained by the conditions in (2.26)-(2.30), $\mathbf{cl}\left(\cup_{t=0,\ldots,K}\mathcal{R}^{(t)}\right)$ is also a polytope. Using standard technique [130], $\mathbf{cl}\left(\cup_{t=0,\ldots,K}\mathcal{R}^{(t)}\right)$ can be conveniently written as a region constrained by only linear constraints, and thus its boundary can be efficiently computed using linear programming.

To illustrate Theorem 1, we show that for the case $(N, K) = (3, 4)$, the aforementioned new memory-rate pair $(\frac{4}{3}, \frac{5}{6})$ is indeed in the region $\mathcal{R}^{(2)}$. For this purpose, we need to find a set of $\{\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\}$ such that the conditions in (2.26)-(2.30) hold for each $\boldsymbol{d} \in \mathcal{D}$.

- For $\boldsymbol{d} = (A, A, B, C) = (1, 1, 2, 3)$, let $\alpha = 1$ for the decomposition pattern $\boldsymbol{\mathcal{P}}^{(2)}_{(1,1,2,3)}$

$$\begin{aligned}\mathcal{P}_{(2,1,0),(1,1,2,3)} &= \{\{1, 2\}\} = \{\{A, B\}\},\\ \mathcal{P}_{(2,0,1),(1,1,2,3)} &= \{\{1, 3\}\} = \{\{A, C\}\},\\ \mathcal{P}_{(1,1,1),(1,1,2,3)} &= \{\{1\}, \{2, 3\}\} = \{\{A\}, \{B, C\}\},\end{aligned} \tag{2.31}$$

which is exactly the decomposition pattern used for (2.14). It can be verified that here

$$R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 1 + 1 + [(2 - 1) + (1 * 2)] = 5 \tag{2.32}$$

using (2.19), and

$$M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},1} = 9 - 1 = 8, \tag{2.33}$$

where the only nonzero term comes from the transmission type $(1,1,1)$ and partition $\mathcal{P} = \{2,3\} = \{B,C\}$ in (2.21). It can be verified similarly that $M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = 8$ for $k = 2,3,4$.

- For $\boldsymbol{d} = (A,A,B,B) = (1,1,2,2)$, two decomposition patterns are used: the first is the one without any decomposition, and the second is

$$\begin{aligned} \mathcal{P}_{(2,1,0),(1,1,2,2)} &= \{\{1\},\{2\}\} = \{\{A\},\{B\}\}, \\ \mathcal{P}_{(1,2,0),(1,1,2,2)} &= \{\{1\},\{2\}\} = \{\{A\},\{B\}\}. \end{aligned} \tag{2.34}$$

Note that this suggests a different coding approach than that used in the example of Section 2.2: the existence of two decomposition patterns implies that we can achieve this memory-rate pair by coding across two instances, using the two decomposition patterns given above. It is clear that for the first pattern

$$M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = 9,\ k = 1,2,3,4, \quad \text{and } R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 4, \tag{2.35}$$

and it can be verified that for the second pattern

$$M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = 7,\ k = 1,2,3,4, \quad \text{and } R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 6. \tag{2.36}$$

It is clear that choosing $\alpha = 0.5$ for both patterns satisfies the conditions (2.26)-(2.30).

- For $\boldsymbol{d} = (A,A,A,C) = (1,1,1,3)$, again two decomposition patterns are used: the first is the one without any decomposition, and the other is

$$\mathcal{P}_{(2,0,1),(1,1,1,3)} = \{\{1\},\{3\}\} = \{\{A\},\{C\}\},$$
$$\mathcal{P}_{(3,0,0),(1,1,1,3)} = \{\{1\}\} = \{\{A\}\}. \tag{2.37}$$

This case is similar to the previous one, and the parameter α can also be chosen to be 0.5 each.

- For $\boldsymbol{d} = (A, A, A, A) = (1, 1, 1, 1)$, two decomposition patterns are used: the first is the one without any decomposition, and the second is the special uncoded transmission. For the first pattern

$$M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = 9,\ k = 1, 2, 3, 4, \quad \text{and } R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 3, \tag{2.38}$$

and for the second pattern

$$M_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)},k} = 3,\ k = 1, 2, 3, 4, \quad \text{and } R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 12. \tag{2.39}$$

We can choose $\alpha = \frac{5}{6}$ for the first pattern and the conditions (2.26)-(2.30) indeed hold.

2.4 The New Coding Scheme

We first give the prefetching strategy and the delivery strategy. The correctness of the code is then proved, which establishes Theorem 1.

2.4.1 The Prefetching Strategy

The prefetching strategy is in fact rather straightforward, which is to encode the symbols allocated to a user using a rank metric code to produce the linear combinations. However, since we allow coding across multiple instances, a technical issue arises as what are the proportions of different delivery patterns. These values are needed to determine two parameters: the total number of instances to code across, and the total number of coded symbols to cache. To address this technical issue, we consider the following line of argument.

Suppose a memory-rate tradeoff pair $(M, R) \in \mathcal{R}^{(t)}$. The definition of $\mathcal{R}^{(t)}$ implies that there exists a set of $\{\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\}$ for which the conditions in (2.26)-(2.30) hold. Let us assume that a positive integer r is chosen such that there exists a set of non-negative integers $\{r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\}$

$$\left| \frac{r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}}{r} - \alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} \right| \leq \epsilon. \tag{2.40}$$

Clearly ϵ can be arbitrarily small by choosing r sufficiently large. Essentially, during the delivery phase, for each demand type $\boldsymbol{d}$, within the total of r instances that are being coded across, we will use the decomposition pattern $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$ on $r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}$ of them during delivery.

Let us now fix r and $\{r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\}$. For $\boldsymbol{d} \in \mathcal{D}$, define the memory-rate pair

$$(M'_{\boldsymbol{d}}, R'_{\boldsymbol{d}}) \triangleq \frac{1}{r\binom{K}{t}} \left(\max_{k \in I_K} \sum_{\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k}, \sum_{\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} \right), \tag{2.41}$$

Let us also define $M'_r \triangleq \max_{\boldsymbol{d} \in \mathcal{D}} M'_{\boldsymbol{d}}$ and $R'_r \triangleq \max_{\boldsymbol{d} \in \mathcal{D}} R'_{\boldsymbol{d}}$, which will be the effective memory-rate pair of this code.

The key design constraint is that the prefetching strategy needs to be independent of the demand vector, which we describe next. In the proposed code, each file contains $r\binom{K}{t}$ symbols. Each symbol is thus denoted as $W_{n,\mathcal{S}}^{(i)}$, where $i \in I_r$, $n \in I_N$ and $\mathcal{S} \subseteq I_K$ with $|\mathcal{S}| = t$, is assumed to be a symbol in $\mathbb{F}_{2^m}$ for some sufficiently large m to be specified shortly. Each file symbol (segment) will be provided to t users as indicated by $\mathcal{S}$, to be stored as a component of some linear combinations. There are a total of

$$P \triangleq rN\binom{K-1}{t-1}$$

symbols allocated to each user, however, only $P_o - P$ linear combinations of them are stored in the cache, and the parameter P_o is directly related to the normalized memory M'_r as

$$P_o - P = rM'_r \binom{K}{t}. \tag{2.42}$$

Note that P_o is always an integer. A (P_o, P) systematic rank metric code is then used to encode the P symbols at each user, and the $P_o - P$ parities of this code are placed at each user's cache. For such a rank metric code to exist, $m \geq P_o$ suffices.

Our plan next is to show that for each $\boldsymbol{d} \in \mathcal{D}$, a valid delivery strategy exists with a delivery rate $R'_{\boldsymbol{d}}$. Then by making the integer r sufficiently large, and choosing the integers $\{r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\}$ appropriately such that $\epsilon \geq 0$ is made arbitrarily small, we have

$$\begin{aligned}\lim_{r\to\infty}(M'_r, R'_r) = \frac{1}{\binom{K}{t}}\left(\max_{\boldsymbol{d}\in\mathcal{D}}\max_{k\in I_K}\sum_{\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k}, \max_{\boldsymbol{d}\in\mathcal{D}}\sum_{\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\right)\\ \preceq (M, R).\end{aligned} \tag{2.43}$$

This would prove that the targeted (M, R) is indeed achievable.

2.4.2 The Delivery Strategy

Consider any demand vector $\boldsymbol{d} \in \mathcal{D}$, and recall the parameters $\{r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\}$ have been chosen. For convenience, suppose there are a total of q possible decomposition patterns for the demand vector $\boldsymbol{d}$, with the first one as $\boldsymbol{\mathcal{P}}_{\boldsymbol{d},1}^{(t)} = \breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}$ which is the special case associated with the uncoded transmissions, and $\boldsymbol{\mathcal{P}}_{\boldsymbol{d},j}^{(t)}$, for $j = 2, 3, \ldots, q$ other decomposition patterns. For a specific transmission type $\boldsymbol{t}$, the corresponding decomposition in $\boldsymbol{\mathcal{P}}_{\boldsymbol{d},j}^{(t)}$ is written as $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d},j}$. Note that $\sum_{j=1}^{q} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d},j}^{(t)}} = r$. For the demand vector $\boldsymbol{d}$, the transmissions in the proposed scheme are as given in Algorithm 1.

The transmissions on line-6 and line-13 in Algorithm 1 are uncoded, which stem from the special transmission pattern $\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}$. We first need to show that the steps on line-4 and line-10 are valid, *i.e.,* such a set $\mathcal{A}^*$ or $\mathcal{A}$ can always be found. The latter case is immediate by observing that in this case $K - t \geq N^*$, and thus $N^* \leq \min(K - t, \tilde{N}) \leq \tilde{N}$, and we can always find a set $\mathcal{A}$

Algorithm 1: The delivery strategy. Reprinted with permission from [1,2], © 2018 IEEE.

```
Input: $t$, $\boldsymbol{d}$, $\{r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}\}$, and $\{W_{n,\mathcal{S}}^{(i)}\}$
1  Compute $\boldsymbol{m}$, $\mathbb{supp}(\boldsymbol{m})$, and $N^*$ from $\boldsymbol{d}$.
2  if $K - t \leq N^* - 1$ then
3      for $\mathcal{S} \subseteq I_K$: $|\mathcal{S}| = t$ do
4          Find a set $\mathcal{A}^* \subseteq \mathbb{supp}(\boldsymbol{m})$ such that $\cup_{n \in \mathcal{A}^*} I^{[n]} \subseteq \mathcal{S}$ and $|\mathcal{A}^*| = N^* - K + t$
5          for $i = 1$ to $r_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}$ do
6              Transmit $W_{n,\mathcal{S}}^{(i)}$, all $n \in \mathbb{supp}(\boldsymbol{m}) \setminus \mathcal{A}^*$.
7          end
8      end
9  else
10     Choose a set $\mathcal{A} \subseteq I_N$, such that $|\mathcal{A}| = \min(K - t, \tilde{N})$ and $\mathbb{supp}(\boldsymbol{m}) \subseteq \mathcal{A}$
11     for $n \in \mathcal{A}$ do
12         for $i = 1$ to $r_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}$ do
13             Transmit $W_{n,\mathcal{S}}^{(i)}$, all $\mathcal{S} \subseteq I_K$ such that $\mathcal{S} = t$.
14         end
15     end
16 end
17 for $j = 2$ to $q$ do
18     for $\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{d}}^{(t)}$ do
19         for $\mathcal{P} \in \mathcal{P}_{\boldsymbol{t},\boldsymbol{d},j}$ do
20             for $\mathcal{B} : \mathcal{T}(\mathcal{B}) = \boldsymbol{t}$, $\left(\bigcup_{n \in \mathcal{P}} \{\ell^{[n]}\}\right) \cap \mathcal{B} \neq \emptyset$ do
21                 for $i = \sum_{k=1}^{j-1} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} + 1$ to $\sum_{k=1}^{j} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k}$ do
22                     Transmit $\oplus_{n \in \mathcal{P}} \left( \oplus_{k \in \mathcal{B} \cap I^{[n]}} W_{n,\mathcal{B} \setminus k}^{(i)} \right)$;
23                 end
24             end
25         end
26     end
27 end
```

such that $\mathbb{supp}(\boldsymbol{m}) \subseteq \mathcal{A} \subseteq I_N$. To see that the step on line-4 is also valid, first observe that in this case $K - t \leq N^* - 1$, and we need to find a set of files $\mathcal{A}^*$, such that the given set of users $\mathcal{S}$ (where $|\mathcal{S}| = t$) includes all the users that request files in $\mathcal{A}^*$. Suppose we cannot find such a set, this means that there are less than $N^* - K + t$ such files, or more than $N^* - (N^* - K + t) = K - t$ files that are being requests by some users not in $\mathcal{S}$, but this is impossible, since there are only $K - t$ users not in the set $\mathcal{S}$. Thus the supposition is not true, and we can always find such a set $\mathcal{A}^*$. It is straightforward to count the total number of transmissions as $r_{\boldsymbol{d},\check{\boldsymbol{\mathcal{P}}}^{(t)}_{\boldsymbol{d}}} \min(K - r, \tilde{N})\binom{K}{t}$, when Algorithm 1 completes line-16.

The transmissions on line-22 in Algorithm 1 have the following property: at least one of the component $W_{n,\mathcal{B}\setminus k}$ (for any fixed superscript $^{(i)}$) in the transmission must have k that is a leader. It is a simple combinatorial counting task to show that the total number of transmissions in this part of the algorithm is given by $\sum_{j=2}^{q} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}} R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}}$. Essentially we examine all the transmission types, for which the decomposition pattern for the transmission type follows $\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}$ in the corresponding instances (indexed by the superscript $^{(i)}$), then count the transmitted linear combinations associated with each partition in this decomposition. We must eliminate the transmissions where no leader in this partition $\mathcal{P}$ is included, which is indeed accounted for as the term $-\prod_{n\in\mathcal{P}}\binom{m_n-1}{t_n}$ in $R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}}$. Thus after the algorithm runs to completion, a total of $\sum_{j=1}^{q} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}} R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}}$ symbols are transmitted.

The transmissions on line-22 in Algorithm 1 are a subset of the decomposed transmissions given in (2.18) for the patterns $\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}$, since transmissions without any leader are not allowed as mentioned early. This removes the redundancy in the transmissions after a native decomposition. The precise linear independence relations can be captured in a set of lemmas given in the sequel.

2.4.3 Three Auxiliary Lemmas

When stating these lemmas, we omit the superscript $^{(i)}$ which is used to index the code instances that are being coded across, as well as the decomposition pattern index $j = 2, 3, \ldots, q$, since they are irrelevant in these settings. We shall return to this notation later on when it becomes important.

Lemma 3 (Redundancy Reduction Lemma)**.** *Fix a demand vector* $\boldsymbol{d}$ *and a valid transmission type* $\boldsymbol{t}$*. Designate a subset* $\mathcal{P} \subseteq \mathbb{supp}(\boldsymbol{t})$ *as the variable set, and* $\bar{\mathcal{P}} = \mathbb{supp}(\boldsymbol{t}) \setminus \mathcal{P}$ *as the fixed* *set. Further fix an arbitrary subset* $\mathcal{A} \subseteq \cup_{n \in \bar{\mathcal{P}}} I^{[n]}$ *such that* $\mathcal{A} \cap I^{[n]} = t_n$ *for all* $n \in \bar{\mathcal{P}}$*. Let* $\mathcal{L} \triangleq \cup_{n \in \mathcal{P}} \{\ell^{[n]}\}$ *be the leader set. Let* $\mathcal{Q}_n \subseteq I^{[n]} \setminus \ell^{[n]}$ *be any subset such that* $|\mathcal{Q}| = t_n$*, and let* $\mathcal{Q} \triangleq \cup_{n \in \mathcal{P}} \mathcal{Q}_n$*. The following equation holds*

$$\bigoplus_{\substack{\mathcal{V} \subseteq \mathcal{Q} \cup \mathcal{L}: \\ |\mathcal{V} \cap I^{[n]}| = t_n, \\ \forall n \in \mathcal{P}}} \oplus_{k \in \mathcal{V}} W_{d_k, \mathcal{V} \cup \mathcal{A} \setminus \{k\}} = 0. \tag{2.44}$$

Proof. Observe that

$$LHS = \oplus_{\ell \in \mathcal{L}} \bigoplus_{\substack{\mathcal{V} \subseteq \mathcal{Q} \cup \mathcal{L}: \\ |\mathcal{V} \cap I^{[n]}| = t_n, \\ \forall n \in \mathcal{P}}} \oplus_{k \in \mathcal{V} \cap I^{[\ell]}} W_{d_\ell, \mathcal{V} \cup \mathcal{A} \setminus \{k\}} \tag{2.45}$$

Consider any fixed $\ell \in \mathcal{L}$, and enumerate all set $\mathcal{V}$ by parts $\mathcal{V} \triangleq (\hat{\mathcal{V}}, \tilde{\mathcal{V}})$

$$\begin{aligned} &\bigoplus_{\substack{\mathcal{V} \subseteq \mathcal{Q} \cup \mathcal{L}: \\ |\mathcal{V} \cap I^{[n]}| = t_n, \\ \forall n \in \mathcal{P}}} \oplus_{k \in \mathcal{V} \cap I^{[\ell]}} W_{d_\ell, \mathcal{V} \cup \mathcal{A} \setminus \{k\}} \\ &\quad = \oplus_{\substack{\hat{\mathcal{V}} \subseteq (\mathcal{Q} \setminus \mathcal{Q}_{d_\ell}) \cup (\mathcal{L} \setminus \{\ell\}): \\ |\mathcal{V} \cap I^{[n]}| = t_n, \\ \forall n \in \mathcal{P} \setminus \{d_\ell\}}} \left(\oplus_{\substack{\tilde{\mathcal{V}} \subseteq \mathcal{Q}_{d_\ell} \cup \{\ell\}: \\ |\tilde{\mathcal{V}}| = t_{d_\ell}}} \left(\oplus_{k \in \tilde{\mathcal{V}}} W_{d_\ell, (\hat{\mathcal{V}} \cup \mathcal{A}) \cup (\tilde{\mathcal{V}} \setminus \{k\})} \right) \right). \end{aligned} \tag{2.46}$$

Now consider a fixed $\hat{\mathcal{V}}$, and consider the inner summation

$$\oplus_{\substack{\tilde{\mathcal{V}} \subseteq \mathcal{Q}_{d_\ell} \cup \{\ell\}: \\ |\tilde{\mathcal{V}}| = t_{d_\ell}}} \left(\oplus_{k \in \tilde{\mathcal{V}}} W_{d_\ell, (\hat{\mathcal{V}} \cup \mathcal{A}) \cup (\tilde{\mathcal{V}} \setminus \{k\})} \right). \tag{2.47}$$

This is a summation of $(t_{d_\ell} + 1) t_{d_\ell}$ file symbols, each of which is in the form $W_{d_\ell, (\hat{\mathcal{V}} \cup \mathcal{A}) \cup \dot{\mathcal{V}}}$, where $\dot{\mathcal{V}}$ is a subset of $\mathcal{Q}_{d_\ell} \cup \{\ell\}$ such that $|\dot{\mathcal{V}}| = t_{d_\ell} - 1$. Since $|\mathcal{Q}_{d_\ell} \cup \{\ell\}| = t_{d_\ell} + 1$ and the summation form in (2.47) is symmetric, each file symbol appears exactly twice, which cancel out each other in this binary (extension) field. The proof is thus complete. □

The above lemma can be used to show that the decomposed transmissions without any leaders are redundant. To see this, notice that (2.44) can be rewritten as

$$\left(\oplus_{\substack{\mathcal{V}\subseteq\mathcal{Q}\cup\mathcal{L}:\mathcal{V}\neq\mathcal{Q}\\ |\mathcal{V}\cap I^{[n]}|=t_n,\forall n\in\mathcal{P}}} \oplus_{k\in\mathcal{V}} W_{d_k,\mathcal{V}\cup\mathcal{A}\setminus\{k\}} \right) \oplus \left(\oplus_{k\in\mathcal{Q}} W_{d_k,\mathcal{Q}\cup\mathcal{A}\setminus\{k\}}\right) = 0 \tag{2.48}$$

Clearly the summation in the second bracket, which is one of decomposed parts from (2.18) without any leaders, can be expressed as a linear combination of those in the first bracket, which all have some leaders and are indeed in the delivery transmissions given on line-22 in Algorithm 1. Conversely, the transmissions obtained by directly decomposing those in the delivery transmissions of [39] can be reconstructed using the transmissions given on line-22 in Algorithm 1. Lemma 3 is a generalized version of a similar lemma in [100], which was used to remove the redundancy in the coding scheme given in [39].

The next two lemmas essentially state that there is no further linear redundancy in the transmissions in line-22 of Algorithm 1 to be removed. In order to state the lemmas, the following definition is needed. For any fixed $\boldsymbol{d}$, $\boldsymbol{t}$, $\mathcal{P} \in \mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$, and $\mathcal{A} \subseteq \cup_{n\in\mathrm{supp}(\boldsymbol{t})\setminus\mathcal{P}} I^{[n]}$ for which $\mathcal{A} \cap I^{[n]} = t_n$ for all $n \in \mathrm{supp}(\boldsymbol{t}) \setminus \mathcal{P}$, let

$$\mathcal{W}_{\boldsymbol{d},\boldsymbol{t},\mathcal{P},\mathcal{A}} \triangleq \bigcup_{\substack{\mathcal{B}\subseteq\cup_{n\in\mathcal{P}}I^{[n]}:\\ \mathcal{B}\cap I^{[n]}=t_n,\\ \forall n\in\mathcal{P}}} \left\{W_{d_k,\mathcal{A}\cup\mathcal{B}\setminus k} : d_k \in \mathcal{P}\right\}. \tag{2.49}$$

The next lemma states that the decomposed transmissions can in fact be separated naturally into mutually exclusive groups.

Lemma 4. *For any $\boldsymbol{d}$ and $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$, and any $(\boldsymbol{t'}, \mathcal{P}', \mathcal{A}') \neq (\boldsymbol{t''}, \mathcal{P}'', \mathcal{A}'')$, where $\mathcal{P}' \in \mathcal{P}_{\boldsymbol{t'},\boldsymbol{d}}$ and $\mathcal{P}'' \in \mathcal{P}_{\boldsymbol{t''},\boldsymbol{d}}$, we have $\mathcal{W}_{\boldsymbol{d},\boldsymbol{t'},\mathcal{P}',\mathcal{A}'} \cap \mathcal{W}_{\boldsymbol{d},\boldsymbol{t''},\mathcal{P}'',\mathcal{A}''} = \emptyset$.*

Proof. Suppose that the two sets have a common element $W_{n,\mathcal{C}}$. Then this implies that,

$$\begin{aligned} t'_n &= t''_n = |\{\hat{k} \in \mathcal{C} : d_{\hat{k}} = n\}| + 1, \\ t'_i &= t''_i = |\{\hat{k} \in \mathcal{C} : d_{\hat{k}} = i\}|, \quad i \in I_N, \quad i \neq n. \end{aligned} \tag{2.50}$$

i.e., $\boldsymbol{t}' = \boldsymbol{t}''$; let us write this transmission type as $\boldsymbol{t}$. It also follows that $\mathcal{P}' = \mathcal{P}''$, since $n \in \mathcal{P}'$ and $n \in \mathcal{P}''$, but $\mathcal{P}' \cap \mathcal{P}'' = \emptyset$ if $\mathcal{P}'$ and $\mathcal{P}''$ are distinct; we can thus denote this partition as $\mathcal{P}$. It further follows that $\mathcal{A}' = \mathcal{A}'' = \mathcal{C} \cap \cup_{n \in \mathbb{supp}(\boldsymbol{t}) \setminus \mathcal{P}} I^{[n]}$. This is a contradiction, and thus there is no common element between the two sets. The proof is thus complete. □

Lemma 5. *Each transmission on line-22 in Algorithm 1 is a linear combination of the elements in a single set $\mathcal{W}_{\boldsymbol{d},\boldsymbol{t},\mathcal{P},\mathcal{A}}$. All the linear combinations in the transmissions on line-22 of Algorithm 1 using symbols in a single set $\mathcal{W}_{\boldsymbol{d},\boldsymbol{t},\mathcal{P},\mathcal{A}}$ are linearly independent.*

Proof. The first statement is through direct inspection. We can prove the second statement by analyzing the rank of the corresponding coding matrix, which is however rather long and tedious. We instead prove it through a shortcut, directly utilizing the optimality result established in [100].

Fix a demand vector $\boldsymbol{d} \in \mathcal{D}$, a transmission type $\boldsymbol{t}$, and a partition $\mathcal{P} \in \mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$. We only need to prove that for a fixed $\mathcal{A}$, the transmissions

$$\oplus_{n \in \mathcal{P}} \left(\oplus_{k \in \mathcal{B} \cap I^{[n]}} W_{n,(\mathcal{B} \setminus k) \cup \mathcal{A}} \right), \tag{2.51}$$

when $\mathcal{B}$ ranges over all subsets of $\cup_{n \in \mathcal{P}} I^{[n]}$ that satisfy the condition

$$\mathcal{B} \subseteq \cup_{n \in \mathcal{P}} I^{[n]} : \mathcal{B} \cap I^{[n]} = t_n, \forall n \in \mathcal{P} \tag{2.52}$$

are indeed linearly independent. For this purpose, the exact choice of $\mathcal{A}$ is not relevant, and thus we might as well simply drop it by defining

$$\hat{W}_{n,\mathcal{B} \setminus k} \triangleq W_{n,(\mathcal{B} \setminus k) \cup \mathcal{A}}, \tag{2.53}$$

which lead to the representation

$$\oplus_{n \in \mathcal{P}} \left(\oplus_{k \in \mathcal{B} \cap I^{[n]}} \hat{W}_{n,\mathcal{B} \setminus k} \right), \tag{2.54}$$

where $\mathcal{B}$ has the same range as (2.52). Now consider a caching system that has files $\{\hat{W}_n : n \in \mathcal{P}\}$, the users $\cup_{n\in\mathcal{P}} I^{[n]}$, and the demand vector formed by taking the demand vector $\boldsymbol{d}$ at the coordinates $\cup_{n\in\mathcal{P}} I^{[n]}$. The transmissions (2.54) are in fact part of the transmissions in the scheme in [100] for this system when choosing $t = |\mathcal{P}| - 1$. These transmissions cannot possibly be linearly dependent, because if so, the dependence could have been removed to further improve the delivery transmission rate, but it was shown in [100] that this transmission scheme is in fact optimal for each demand vector. The proof is thus complete. □

2.4.4 The Correctness of the Coding Scheme

The next proposition shows that the code is indeed valid for any $\boldsymbol{d} \in \mathcal{D}$.

Proposition 1. *Each user can use the delivery transmissions in Algorithm 1 and the cached content to recover the requested file for any $\boldsymbol{d} \in \mathcal{D}$.*

Proof. Consider an arbitrary user k_o, whose demands is d_{k_o}. From the transmissions in line-1 to line-16, the user can clearly collect for each $\mathcal{S}$, where $k_o \in \mathcal{S}$, a total of $r_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}^{(t)}_{\boldsymbol{d}}} \min(K - r, \tilde{N})$ uncoded symbols, in the form of $W^i_{n,\mathcal{S}}$ for $n \in \mathbb{supp}(\boldsymbol{m}) \setminus \mathcal{A}^*$ (or $n \in \mathcal{A}$), and there are clearly $\binom{K-1}{t-1}$ possible ways to choose such a $\mathcal{S}$. Thus a total of $r_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}^{(t)}_{\boldsymbol{d}}} \Delta M_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}^{(t)}_{\boldsymbol{d}},k}$ symbols are collected.

Then consider the transmissions on line-22 in the algorithm. For each transmission type $\boldsymbol{t}$ such that $d_{k_o} \in \mathbb{supp}(\boldsymbol{t})$, consider every $\mathcal{B}$ such that $\mathcal{T}(\mathcal{B}) = \boldsymbol{t}$, $|\mathcal{B}| = t + 1$, $k_o \in \mathcal{B}$, and $\left(\bigcup_{n\in\mathcal{P}}\{\ell^{[n]}\}\right) \cap \mathcal{B} \neq \emptyset$. User-$k$ collects the following transmissions in the corresponding instances:

$$\oplus_{n\in\mathcal{P}} \left(\oplus_{k\in\mathcal{B}\cap I^{[n]}} W^{(i)}_{n,\mathcal{B}\setminus k}\right), \mathcal{P} \text{ such that } \mathcal{P} \in \mathcal{P}_{\boldsymbol{t},\boldsymbol{d},j} \text{ and } d_{k_o} \notin \mathcal{P}; \tag{2.55}$$

First note that since $d_{k_o} \notin \mathcal{P}$ but $n \in \mathcal{P}$, and $k \in I^{[n]}$, we have $k_o \neq k$ in the inner enumeration. Thus $k_o \in \mathcal{B} \setminus k$. This implies that all the collected transmissions are linear combinations of the symbols of the form $W^{(i)}_{n,\mathcal{S}}$ where $k_o \in \mathcal{S}$, which are the components of the linear combinations stored in the cache of user-k_o. It is straightforward to count that user-k collects a total of $\Delta M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j},k_o}$ for the decomposition pattern $\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d},j}$ such transmissions in (2.55) for each fixed i value.

Together with the cached contents, user-k_o has a total of

$$M' + \sum_{j=1}^{q} r_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d},j}^{(t)}} \Delta M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d},j}^{(t)},k_o} \geq rN\binom{K-1}{t-1}, \tag{2.56}$$

linear combinations of the $rN\binom{K-1}{t-1}$ symbols $W_{n,\mathcal{S}}^{(i)}$ where $k_o \in \mathcal{S}$. The collected linear combinations from the delivery transmissions are clearly linearly independent due to Lemma 4 and Lemma 5. Thus these linear combinations that user-k_o has gathered can be viewed as a full rank transformation of the corresponding rank metric code symbols, which were produced at the prefetching stage by user-k_o. By Lemma 2, user k_o can recover all these symbols to their uncoded form. At this point, user-k_o essentially has all the symbols as if the uncoded prefetching strategy in [39] was used. It remains to argue that user-k_o can also recover the file symbols of file d_{k_o}, which is in the form $W_{d_{k_o},\mathcal{S}}^{(i)}$ such that $k_o \notin \mathcal{S}$. This is straightforward to check for $i = 1, \ldots, r_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}$, because all the needed symbols are transmitted in the uncoded form. For the other cases, since by Lemma 3, the original transmissions in the delivery scheme in [39] can be completely reconstructed using the transmissions in Algorithm 1, and with these transmissions and the uncoded prefetched symbols in the cache, user-k_o can indeed recover the missing file symbols through the decoding strategy in [39]. The proof is thus complete. $\square$

2.5 Conclusion

In this chapter, we proposed a connection between the caching strategy in [77] and that in [100], that is decomposing the delivery transmissions in [100] yields those in [77] in some cases. This allows us to view the coding strategy in [77] and that in [100] as the two extremes of a more general scheme. The general scheme can achieve some memory-rate pairs previously unknown in the literature, and can be computed using a linear programming approach.

We note that although the new scheme unifies the codes in [77] and [100], it does not appear to include the codes in [101]. We suspect that an improved code can be found by analyzing the transmission types more carefully to optimize explicitly the decomposition patterns, and then incorporate certain coding components in [101]; this is part of our ongoing work. Our work reported here is information theoretic in nature, and little attention has been paid to the complexity of the codes. Particularly, the proposed code has a large alphabet size, a large subpacketization factor, and needs to code across a large number of instances. Such a code is challenging to use directly in practical systems, and effort toward simplifying it appears worthwhile.

3. ON THE FUNDAMENTAL LIMIT OF CODED CACHING SYSTEMS WITH A SINGLE DEMAND TYPE[*]

Caching is a technique to alleviate communication load during peak hours by prefetching certain contents to the memory of the end users during off-peak hours. Recently, Maddah-Ali and Niesen [39] proposed a framework for caching, and showed that coded caching can achieve significant improvement over uncoded caching. This caching system, with N files and K users, operates in two stages: during the prefetching stage, each user fills the cache memory of size M with information on the files, and during the delivery stage, the users reveal their requests, and the central server broadcasts common information of size R to all the users, which can be used jointly with the cached contents to fulfill the requests.

The optimal tradeoff between M and R is of fundamental importance in this setting. Despite significant efforts, the problem of characterizing this fundamental limit still remains open. Schemes for both uncoded prefetching and coded prefetching have been proposed [2,39,77,93,94, 97–100], and various outer bounds have also been discovered [92, 102–104]. Nevertheless, except a few special cases, the fundamental limit of coded caching systems still remains unknown.

In the placement phase, each user has no prior knowledge of the demands in the delivery phase, and the prefetched contents need to be properly designed to accommodate all possible demand vectors. In a recent work [91] (see also [92]), the notion of demand type was introduced to classify the demand vectors, which lead to simplifications in a computer-aided investigation of the outer bounds. From this perspective, the original setting [39] in fact allows fully mixed demand types, and it appears that one reason for the afore-mentioned difficult is the tension among the coding requirements to accommodate different demand types. Thus a natural question is how different demand types impact this optimal (M, R) tradeoff.

To develop better understanding of this issue, in this section we consider single-demand type

[*]© 2019 IEEE. Reprinted, with permission, from Shuo Shao, Jesus Gómez-Vilardebó, Kai Zhang and Chao Tian, "On the Fundamental Limit of Coded Caching Systems with a Single Demand Type", *2019 IEEE Information Theory Workshop (ITW)*, pp. 1-5, 2019.

systems, where during the placement phase, the users and server know a priori that the demand vector in the delivery phase must be of a given demand type. Clearly, single-demand type systems have a more relaxed coding requirement than the original setting, however, it is still highly non-trivial since for each single demand type, a large number of different demand vectors are possible, with a rich set of symmetry relations among them [92]. Because of the relaxed coding requirement, any inner bound for the fully mixed demand type system will also be an inner bound for a single demand system, however, an outer bound for the fully mixed demand type systems may not be a valid outer bound for the single demand type systems.

Our first step is to collect the best known inner bounds and outer bounds for both fully mixed demand type systems and single demand type systems in the literature for the canonical $(N, K) = (3, 3)$ system, some of which are from very recent developments [97, 98] in the area. This exercise reveals important insight and fundamental differences between the different classes of systems. It is shown that the demand type where all files are requested poses the most significant challenged in terms of characterizing the fundamental limit, however, codes designed for this demand type can in fact achieve (M, R) pairs that are strictly impossible for another demand type. This is contrary to popular belief that such a demand type is the “worst case", and also confirms that there is indeed a tension for codes designed for different demand types, and a fully mixed system would need to balance such conflicting interests. Next we focus on the case $N = K$ and for the demand type where all files are requested, and propose a new code construction based on a novel sub-packetization design. The construction is a generalization of the code recently proposed in [98], however, in contrast to the code specific designed for $N = K = 3$ or 4 that yields a single (M, R) pair each, our construction is for general N when $N = K$ which can produce multiple new (M, R) points. Finally, we consider outer bounds for such single demand type systems, where several existing bounds are first verified to be valid for this relaxed setting, and additionally a new outer bound is identified; the outer bounds indeed match the proposed scheme in some cases.

3.1 Single Demand Type Systems

In an (N, K) coded caching system, there are N mutually independent uniformly distributed files $(W_1, W_2, \dots, W_N)$, each of F bits. There are K users, each with a cache memory of capacity MF bits. In the placement phase, each user stores some content, denoted as Z_k for user-k's content, in its cache memory. In the delivery phase, user-k requests a file $d(k)$, and a central server broadcasts a message $X_{d(1),d(2),\dots,d(K)}$ of rate RF bits to every user, such that each user can decode the requested file, together with the cached contents. The optimal tradeoff between M and R is the fundamental mathematical object of interest.

The notion of demand type was first introduced in [91] (see also [92]), which is restated below.

Definition 2. *For a demand vector $(d(1), d(2), \dots, d(K))$ in an (N, K) coded caching system, denote the number of users requesting file W_n as m_n, where $n \in [1 : N]$. We call the vector obtained by sorting the values $(m_1, m_2, \dots, m_N)$ in a decreasing order as the demand type of $(d(1), d(2), \dots, d(K))$.*

For example, in an $(N, K) = (3, 3)$ system, the demand vector $(d(1), d(2), d(3)) = (1, 2, 1)$ belongs to the demand type $(2, 1, 0)$. As mentioned earlier, characterizing the fundamental tradeoff between M and R for fully mixed demand type systems appears rather difficult, despite considerable efforts. In a single demand type system, however, it is known a priori that the demand must have the form of a given type, and thus the prefetching can be designed accordingly. Our interest is thus to characterize the achievable region of all (M, R) pairs for such single demand type systems. For example, for the $(N, K) = (3, 3)$ system, the coded caching system for the single demand type $(2, 1, 0)$ only needs to accommodate the following demand vectors

$$\begin{aligned}
&(d(1), d(2), d(3)) = \\
&(1,1,2),(1,2,1),(2,1,1),(2,2,1),(2,1,2),(1,2,2) \\
&(1,1,3),(1,3,1),(3,1,1),(3,3,1),(3,1,3),(1,3,3) \\
&(2,2,3),(2,3,2),(3,2,2),(3,3,2),(3,2,3), \text{ or } (2,3,3).
\end{aligned}$$

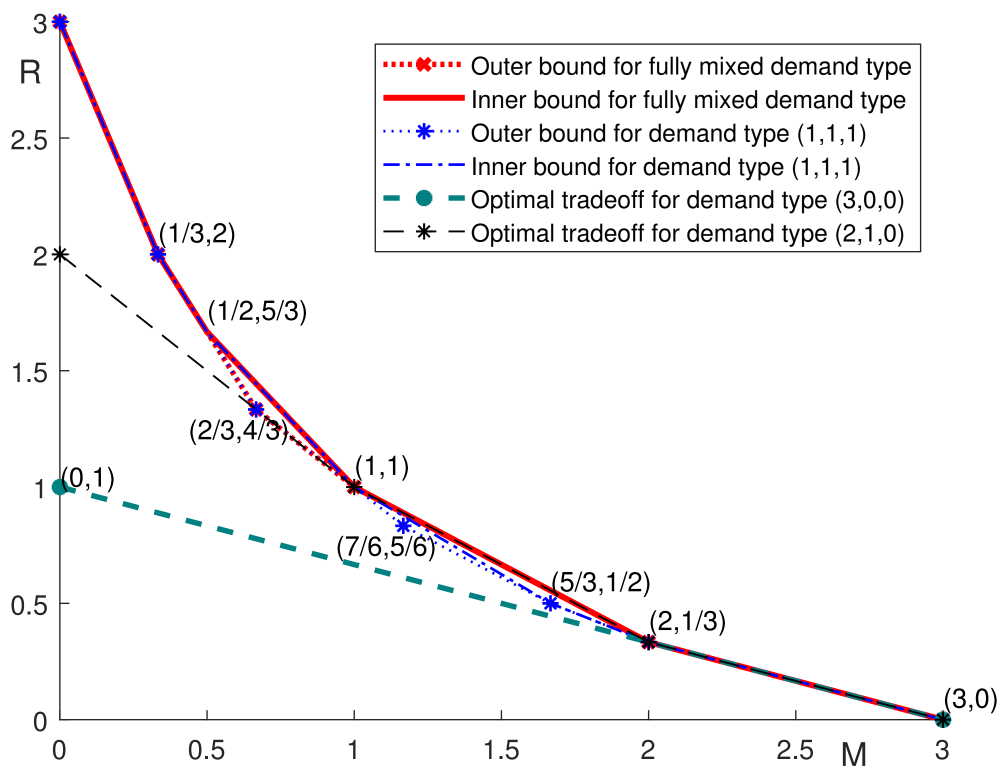

Figure 3.1: Inner bounds and outer bounds for $(3,3)$ systems: fully mixed and single demand types. Reprinted with permission from [3], © 2019 IEEE.

3.2 Fully Mixed and Single Demand Type Systems: The $(3,3)$ Case

In this section, we consider the canonical $(N,K)=(3,3)$ system, and collect the best known inner bounds and outer bounds for both fully mixed demand type systems and single demand type systems in the literature. This exercise reveals important insight and fundamental differences between the two classes of systems.

3.2.1 The Fully Mixed Demand Type System

The best known outer bound for this system can be found in [92], which are all the non-negative pairs of (M,R) satisfying the constraints

$$3M+R\geq 3, 6M+3R\geq 8, M+R\geq 2,$$

$$2M+3R\geq 5, M+3R\geq 3.$$

The lower convex hull of the best known inner bound, on the other hand, is given by the lower convex hull of the points

$$(0,3),(1/3,2),(1/2,5/3),(1,1),(2,1/3),(3,0),$$

where the second and third points are achieved by the scheme in [97], while the others can be achieved by that in [39].

3.2.2 Single Demand Type Systems

Next we provide the best known results for the three single demand type systems for $(N,K) = (3,3)$.

1. For the system with the demand type $(3,0,0)$, the achievable region is precisely all the non-negative (M,R) such that

   $$M + 3R \geq 3, \tag{3.1}$$

   i.e., in this case, the inner bound and the outer bound match. The outer bound can be obtained by a simple cut-set argument [39], while the inner bound is trivial through a memory-sharing argument.

2. For the system with the demand type $(2,1,0)$, the achievable region is precisely all the non-negative (M,R) such that

   $$M + R \geq 2, \quad 2M + 3R \geq 5, \quad M + 3R \geq 3.$$

   In this case, the corner points $(1,1)$ and $(2,1/3)$ can be achieved using the scheme in [39], and $(0,2)$ and $(3,0)$ are trivial. The outer bound was established in [92].

3. For the system with the demand type $(1, 1, 1)$, the best known outer bound is given as [92]

$$3M + R \geq 3, 6M + 3R \geq 8, M + R \geq 2,$$
$$12M + 18R \geq 29, 3M + 6R \geq 8, M + 3R \geq 3.$$

The lower convex hull of the best known inner bound is given by the lower convex hull of the points

$$(0, 3), (1/3, 2), (1/2, 5/3), (1, 1),$$
$$(5/3, 1/2), (2, 1/3), (3, 0).$$

The second and the third points can be achieved by the scheme in [97], the point $(5/3, 1/2)$ by the scheme in [98], and the others by that in [39].

3.2.3 Fully Mixed vs. Single Demand

By comparing the rate region of different demand type systems in Fig.3.1, we make the following observations:

1. The point $(5/3, 1/2)$, which is achievable for the system with the single demand type $(1, 1, 1)$ as shown in [98], is in fact not achievable for the $(2, 1, 0)$ demand type, thus also not achievable for the fully mixed demand type system.

2. Between fully mixed and single demand type systems, single demand type systems can indeed achieve lower rates than the fully mixed demand system.

3. Different single demand type systems provide different out bounds for the fully mixed system, with the one with fewer files demanded produce better bounds at high memory regimes, while those with more files being better at low memory regimes.

The first observation implies that the case when all files are requested is not necessarily the "worst case", contrary to popular belief. Thus designing codes for this demand type alone is not sufficient to yield the optimal scheme for the fully mixed demand type systems, however the optimal codes constructed for such a demand type are likely to play a role in the optimal codes for the fully mixed system. Motivated by the observations above and the new code construction in [98], which only provided a single point $N = K = 3$ or $N = K = 4$, in the sequel we focus on the case $N = K$ for general N, and the demand type when all files are requested, i.e., when the demand type is $(1, 1, \dots, 1)$. From now on, we shall also refer to this special case as the fully demanded coded caching system, or fully demanded system for short.

3.3 Inner Bounds for $(N, K = N)$ Fully Demanded Coded Caching

We propose a novel code construction $(N, K = N)$ for general N values, whose performance is given in the following theorem.

Theorem 2. *For an (N, K) caching system with the single demand type $(1, 1, \dots, 1)$ and $N = K$, the following rate-memory pairs are achievable.*

$$(M, R) \quad = \quad \left(r + \frac{r+1}{K}, \frac{K}{r+1} - 1\right), \tag{3.2}$$

for $r = \{0, ..., K-1\}$.

Note that for $N = 3$ and $N = 4$, by setting $r = K - 1$, we recover the operating points given in [98]; on the other hand, setting $r = 0$ gives the point in [93] (see also [77]). Operating points for other values of r are previously unknown to be achievable in this system. In the sequel we provide the code construction, which is also illustrated with an example for $N = K = 4$, and $r = 2$.

3.3.1 Prefetching

Let us define the set of all user indexes as $\mathcal{K} = \{1, ..., K\}$ and the set of all file indexes as $\mathcal{F} = \{1, ..., N\}$, and in this case we have $N = K$. We partition each file W_f, for all $f \in \mathcal{F}$ into a number of non-overlapping subfiles of equal size. One subfile $W_{f,\mathcal{R},s}$, for each pair of integer s and set $\mathcal{R}$, with $s \notin \mathcal{R}$, and $s = 1, 2, \dots, K$ and $|\mathcal{R}| = r$, where $r \in \{0, ...K-1\}$. The total number of subfiles is thus

$$\binom{K}{r}\binom{K-r}{1} = K\binom{K-1}{r}. \tag{3.3}$$

We will prefetch these subfiles in three different ways, called *type* I to *type* III respectively.

3.3.1.1 Type *I Subfiles*

Given a user k, consider the subfiles $W_{f,\mathcal{R},s}$ satisfying $k \in \mathcal{R}$ and thus $k \neq s$ for every file $f \in \mathcal{F}$. The total number of these subfiles is

$$m_I = N\binom{K-1}{r-1}(K-r).$$

3.3.1.2 Type *II Subfiles*

Next, consider the subfiles $W_{f,\mathcal{R},s}$ for each file $f \in \mathcal{F}$. We define the coded subfiles

$$Z_{\mathcal{R},s} = \bigoplus_{f\in\mathcal{F}} W_{f,\mathcal{R},s}$$

for each set $\mathcal{R}$ and s satisfying $s \notin \mathcal{R}$. Observe that there are $m_{II} = \binom{K-1}{r}$ of these coded subfiles.

3.3.1.3 Type *III Subfiles*

In addition, we define the coded subfiles

$$Z_{f,\mathcal{R}^-,s} = \bigoplus_{u\in\mathcal{K}\backslash\{\mathcal{R}^-,s\}} W_{f,u\cup\mathcal{R}^-,s}$$

for each file $f \in \mathcal{F}$ and each set $\mathcal{R}^-$ satisfying $|\mathcal{R}^-| = r-1$ and $\mathcal{R}^- \subset \mathcal{K}\backslash s$. Each user k caches the above coded subfiles for $K-1$ subfiles (all except one), and except for the set tuples $\{\mathcal{R}^-, s\}$ including one arbitrary user different from k. The total number of these coded subfiles is $m_{III} = (K-1)\binom{K-2}{r-1}$.

Our coded caching scheme caches the m_I uncoded subfiles, the m_{II} coded subfiles $Z_{\mathcal{R},s}$ and the m_{III} coded subfiles $Z_{f,\mathcal{R}^-,s}$. Because each subfile has $\frac{F}{K\binom{K-1}{r}}$ bits, the required cache load at users equals MF with

$$M = \frac{m_I + m_{II} + m_{III}}{K\binom{K-1}{r}} = r + \frac{r+1}{K}.$$

3.3.2 Delivery

Next, we describe the broadcasted coded subfiles for a fixed s. For every set $\mathcal{R}^+ \subset \mathcal{K}\backslash s$ with $r+1$ users, i.e. $|\mathcal{R}^+| = r{+}1$, the server broadcasts

$$Y_{\mathcal{R}^+,s} \;\; = \;\; \bigoplus_{u\in\mathcal{R}^+} W_{\mathbf{d}(u),\mathcal{R}^+\backslash u,s}.$$

Since $s \notin \mathcal{R}^+$, the total number of transmissions associated to a given s is $\binom{K-1}{r+1}$, and the total number of transmission is

$$T \;\; = \;\; \sum_{s\in\mathcal{K}} \binom{K-1}{r+1} = K\binom{K-1}{r+1},$$

and over the number of each file's subfiles $\frac{F}{K\binom{K-1}{r}}$, the communication load RF is

$$R \;\; = \;\; \frac{\binom{K}{1}\binom{K-1}{r+1}}{K\binom{K-1}{r}} = \frac{K}{r+1} - 1. \tag{3.4}$$

3.3.3 Decoding Subfiles Uncoded at Users Different from u

Firstly we consider the decoding at user u of subifiles $W_{\mathbf{d}(u),\mathcal{R},s}$ with $u \neq s$. If $u \in \mathcal{R}$, then file $W_{\mathbf{d}(u),\mathcal{R},s}$ can be found uncoded at the cache of user u. Instead, if $u \notin \mathcal{R}$ then user u computes

$$\begin{aligned}
&W_{\mathbf{d}(u),\mathcal{R},s} \\
=&W_{\mathbf{d}(u),\mathcal{R},s} \oplus \bigoplus_{j\in\mathcal{R}} W_{\mathbf{d}(j),\{\mathcal{R}\cup u\}\backslash j,s} \oplus \bigoplus_{i\in\mathcal{R}} W_{\mathbf{d}(i),\{\mathcal{R}\cup u\}\backslash i,s} \\
=&Y_{\mathcal{R}\cup u,s} \oplus \bigoplus_{i\in\mathcal{R}} W_{\mathbf{d}(i),\{\mathcal{R}\cup u\}\backslash i,s}
\end{aligned}$$

using $Y_{\mathcal{R}\cup u,s}$ and the subfiles $W_{\mathbf{d}(i),\{\mathcal{R}\setminus i\}\cup u,s}$ for all $i \in \mathcal{R}$ that are uncoded in the cache of user u.

3.3.4 Decoding Subfiles Coded at the Cache of User u

Next, we show how user u obtains $W_{\mathbf{d}(u),\mathcal{R},u}$ for all $\mathcal{R}$. Recall that these subfiles are all coded in coded subfiles of its own cache. First user s computes

$$\begin{aligned}\bigoplus_{v\notin\mathcal{R}\cup u} Y_{\mathcal{R}\cup v,u} &= \bigoplus_{v\notin\mathcal{R}\cup u}\bigoplus_{t\in\mathcal{R}\cup v} W_{\mathbf{d}(t),\{\mathcal{R}\cup v\}\setminus t,u} \\ &= \bigoplus_{v\notin\mathcal{R}\cup u} W_{\mathbf{d}(v),\mathcal{R},s} \\ &\quad \oplus \bigoplus_{t\in\mathcal{R}} W_{\mathbf{d}(t),\mathcal{R},s} \bigoplus_{v\notin\mathcal{R}\setminus t} W_{\mathbf{d}(t),\{\mathcal{R}\setminus t\}\cup v,s}.\end{aligned}$$

Recall that user u caches

$$Z_{\mathbf{d}(t),\mathcal{R}\setminus t} = \bigoplus_{v\notin\mathcal{R}\setminus t} W_{\mathbf{d}(t),\{\mathcal{R}\setminus t\}\cup v,u}.$$

Thus, user u can compute

$$\begin{aligned}\Omega_{\mathcal{R},u} &= \bigoplus_{v\notin\mathcal{R}\cup u} Y_{\mathcal{R}\cup v} \oplus \bigoplus_{t\in\mathcal{R}} Z_{\mathbf{d}(t),\mathcal{R}\setminus t} \\ &= \bigoplus_{v\notin\mathcal{R}\cup u} W_{\mathbf{d}(v),\mathcal{R},u} \oplus \bigoplus_{t\in\mathcal{R}} W_{\mathbf{d}(t),\mathcal{R},u} \\ &= W_{\mathbf{d}(u),\mathcal{R},u} \oplus \bigoplus_{f\in\mathcal{F}} W_{f,\mathcal{R},u},\end{aligned}$$

where $\bigoplus_{f\in\mathcal{F}} W_{f,\mathcal{R},u}$ is cached in user u as well, hence $W_{\mathbf{d}(u),\mathcal{R},u}$ can be decoded by the user u.

3.3.5 An Example: $(N, K) = (4, 4)$ and $r = 2$

Let the parameter $r = 2$, and thus there are $\binom{K}{r}\binom{K-r}{1} = 12$ subfiles per file. This corresponds to the point $(M, R) = (11/4, 1/3)$. This point can be achieved by the scheme in [98], and now we present a new approach using the proposed scheme. The prefetching strategy is shown in Table 3.1, which contains all cached symbols in User 1. Consider demand (W_1, W_2, W_3, W_4), our delivery strategy yields a transmission of four coded symbols, as speci-

Table 3.1: Prefetched symbols at user 1 for caching system $(N = 4, K = 4), r = 2$. Reprinted with permission from [3], © 2019 IEEE.

	Cached Symbols at User 1
Type I	$W_{1,12,3}, W_{1,12,4}, W_{1,13,2}, W_{1,13,4}, W_{1,14,2}, W_{1,14,3}$ $W_{2,12,3}, W_{2,12,4}, W_{2,13,2}, W_{2,13,4}, W_{2,14,2}, W_{2,14,3}$ $W_{3,12,3}, W_{3,12,4}, W_{3,13,2}, W_{3,13,4}, W_{3,14,2}, W_{3,14,3}$ $W_{4,12,3}, W_{4,12,4}, W_{4,13,2}, W_{4,13,4}, W_{4,14,2}, W_{4,14,3}$
Type II	(1) $W_{1,23,1} \oplus W_{2,23,1} \oplus W_{3,23,1} \oplus W_{4,23,1}$ (2) $W_{1,24,1} \oplus W_{2,24,1} \oplus W_{3,24,1} \oplus W_{4,24,1}$ (3) $W_{1,34,1} \oplus W_{2,34,1} \oplus W_{3,34,1} \oplus W_{4,34,1}$
Type III	(4) $W_{2,23,1} \oplus W_{2,24,1}$, (5) $W_{3,23,1} \oplus W_{3,24,1}$ (6) $W_{4,23,1} \oplus W_{4,24,1}$, (7) $W_{2,23,1} \oplus W_{2,34,1}$ (8) $W_{3,23,1} \oplus W_{3,34,1}$, (9) $W_{4,23,1} \oplus W_{4,34,1}$

Table 3.2: Delivery for demand $\mathbf{d} = (W_1, W_2, W_3, W_4)$. Reprinted with permission from [3], © 2019 IEEE.

Delivery
(10) $W_{2,34,1} \oplus W_{3,24,1} \oplus W_{4,23,1}$
(11) $W_{1,34,2} \oplus W_{3,14,2} \oplus W_{4,13,2}$
(12) $W_{1,24,3} \oplus W_{2,14,3} \oplus W_{4,12,3}$
(13) $W_{1,23,4} \oplus W_{2,13,4} \oplus W_{3,12,4}$

fied in Table 3.2. The decoding process is as follows. First, observe that, the 6 requested subfiles $W_{1,13,2}, W_{1,14,2}, W_{1,12,3}, W_{1,14,3}, W_{1,12,4}, W_{1,13,4}$ are stored uncoded by User 1. Second, the 3 subfiles $W_{1,34,2}, W_{1,34,2}, W_{1,34,2}$ can be directly decoded from the three transmissions (11-13), respectively, since in each transmission all other subfiles are stored uncoded by User 1. Third, we can use the transmission (10), to recover the remaining 3 subfiles of W_1, which are coded at the cache of User 1, as

$$W_{1,23,1} = (1) \oplus (5) \oplus (7) \oplus (10),$$

$$W_{1,24,1} = (2) \oplus (4) \oplus (6) \oplus (7) \oplus (10),$$

$$W_{1,34,1} = (3) \oplus (5) \oplus (8) \oplus (9) \oplus (10).$$

Hence all the 12 subfiles of W_1 are successfully recovered by User 1. The decoding steps for the other users follows the same pattern.

3.4 General Outer Bounds and Evaluation

In this section, we consider the outer bounds of the fully demanded coded caching system, i.e. the demand type is $(1, 1, \ldots, 1)$. We first verify the validity of several existing outer bounds derived for the fully mixed demand type system, in the context of this single demand type system setting. Then, a new outer bound is provided, which is only applicable to the fully demanded coded caching system.

3.4.1 General Outer Bounds for Fully Demanded System

In this subsection we give out two outer bounds in Theorem 3 and Theorem 4. Theorem 3 is induced from the outer bounds of the original fully mixed demand type system in [104], and Theorem 4 is identified merely for the fully demanded coded caching system.

Two different sets of outer bounds were given in [104]. One is obtained by the intersection of outer bounds derived using single demand types, hence only one particular outer bound remains valid for our setting which however becomes trivial, another set of outer bounds still holds, as its core inequality

$$RF \geq \sum_{k=1}^{K} H(W_k | Z_{[1:k]}, W_{[1:k-1]})$$

holds exactly for the fully demanded coded caching system. Therefore, we have the following theorem.

Theorem 3 (Yu et al. [104])**.** *In the $(N, K = N)$ coded caching system with demand type $(1, 1, \ldots, 1)$, all achievable (M, R) rate pairs are lower-bounded by the lower convex envelop of the points*

$$\left(\frac{K - \ell + 1}{s}, \frac{s-1}{2} + \frac{\ell(\ell-1)}{2s}\right),$$

where $s \in \{1, 2, \dots, K\}$ *and* $\ell \in 0, 1, \dots, K$.

Similary, the outer bound of the worst case rate in [103] is also valid for this setting. However it is weaker than Theorem 3 (see Fig. 3.2), hence the details are omitted here. We can also prove the following outer bound.

Theorem 4. *In the* $(N, K = N)$ *coded caching system with demand type* $(1, 1, \dots, 1)$*, all achievable* (M, R) *rate pairs must satisfy:*

$$KM + K(K-1)R \geq K^2 - 1. \tag{3.5}$$

The proof of this bound is omitted due to space constraint. We note that [98], a similar outer bound was established for $N = 3, 4$, and Theorem 4 generalizes those bounds. When setting $r = K - 1$ in Theorem 2, the achievable rate pair $(M, R) = (K - 2 + \frac{K-1}{K}, \frac{1}{K-1})$ indeed matches this outer bound, hence optimal for the fully demanded system.

3.4.2 Example Evaluation: $(N, K) = (5, 5)$

When $(N, K) = (5, 5)$, Theorem 3 reduces to

$$\begin{aligned} 5M + R \geq 5, &\quad 20M + 5R \geq 24, \quad 36M + 10R \geq 47, \\ 6M + 2R \geq 9, &\quad\ \ 2M + R \geq 4, \qquad 5M + 4R \geq 13, \\ 5M + 6R \geq 16, &\quad\ 5M + 9R \geq 19, \quad\ 5M + 12R \geq 21, \\ 5M + 16R \geq 23, &\quad 5M + 20R \geq 24, \quad\ M + 5R \geq 5. \end{aligned}$$

and Theorem 4 reduces to

$$5M + 20R \geq 24.$$

The inner bounds and outer bounds are shown in Fig. 3.2. It can be seen that three new operating points $(7/5, 3/2)$, $(13/5, 2/3)$ and $(19/5, 1/4)$ are obtained by the proposed code construction.

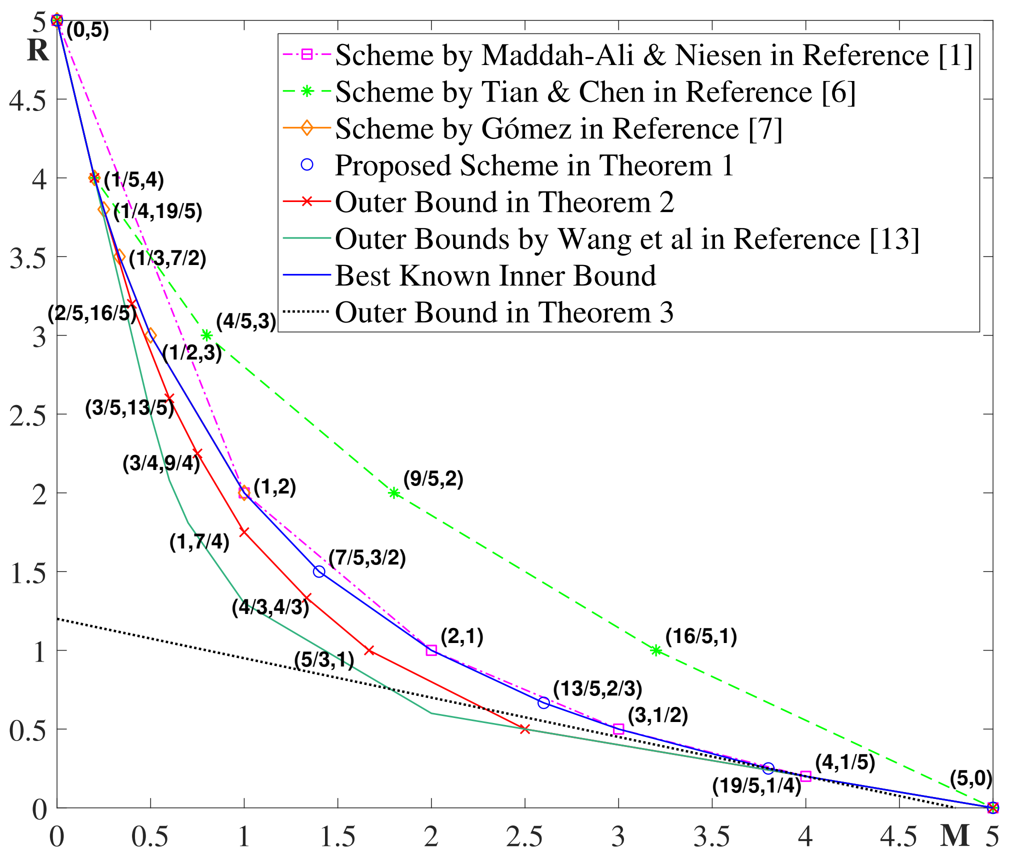

Figure 3.2: Inner bounds and outer bounds for the $(5, 5)$ system with demand type $(1, 1, 1, 1, 1)$. Reprinted with permission from [3], © 2019 IEEE.

3.5 Conclusion

We considered the single demand type coded caching systems in this work. For the canonical $(3, 3)$ system, the single demand type systems are compared thoroughly with fully mixed demand type systems. Even in this case, we see that codes designed for the demand type $(1, 1, 1)$ in fact operates strictly outside of the achievable region of the $(2, 1, 0)$. This is contrary to the popular belief that the full demand is the worst case demand type. We then proposed a new scheme for the fully demanded system, which can achieve rate pair $(M, R) = (r + \frac{r+1}{K}, \frac{K}{1+r} - 1)$ with $r \in [0 : K - 1]$. Lastly, we adapted several outer bounds original obtained for the fully mixed demand type systems, and also identified a new outer bound specific for the single demand type (full demand) system. The proposed code construction provide operating point that indeed match the outer bounds in some cases.

4. ON THE SYMMETRY REDUCTION OF INFORMATION INEQUALITIES[*]

Information theory provides a natural framework and a rich set of tools to determine the fundamental limits of communication systems and information systems. The derivation of such fundamental limits (or outer bounds) in a coding problem, requires a strategic combination of information equalities and inequalities. The most often used set of information inequalities are the Shannon-type inequalities, which were first formally identified by Yeung [105]. Moreover, Yeung identified the minimal set of such inequalities, referred to as elemental inequalities. Although there exist non-Shannon-type inequalities [106], in many cases, Shannon-type inequalities together with problem-specific constraints are sufficient to produce the (true) fundamental limits, particularly for systems with strong symmetries [107–109].

Since Shannon-type inequalities and many problem specific constraints are linear in the joint entropies, the overall derivation of the fundamental limits or outer bounds can be viewed as a series of linear programs where the variables are these joint entropies, and can in principle be solved using any linear program (LP) solver. However, for practical engineering problems, the resultant LPs are usually very large, which make solving them numerically impossible. In a recent work [110], it was shown that symmetry (together with other problem-specific reductions) can be effectively utilized to reduce the scale of the LP, which led to a conclusive solution for the $(4, 3, 3)$ regenerating code problem, and moreover, proved the existence of a fundamental difference between functional repair regenerating codes and exact repair regenerating codes. The intuition behind the reduction is that, the inherent symmetry structure in the problem implies the existence of symmetric optimal solutions, and thus many LP variables can be assumed to have the same values, and as a consequence there is no need to represent them differently. This further induces a significant reduction in terms of the LP constraints, and a computer solver can be used on this much reduced

[*]© 2018 IEEE. Reprinted, with permission, from Kai Zhang and Chao Tian, "On the symmetry reduction of information inequalities", *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2396-2408, 2016.
© 2016 IEEE. Reprinted, with permission, from Kai Zhang and Chao Tian, "Symmetry reduction of information inequalities", *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1-5, 2016.

problem. Although the intuition itself is straightforward, our understanding on the amount of reduction through symmetry is limited. In this work, we seek to develop a better understanding on this issue.

There are two useful measures of the scale of the LP: the number of LP variables, and the number of LP constraints. Although the computational complexity in many modern solvers is more directly related to the number of nonzero elements in the LP [111], these two measures are usually good indicators of the computational scale, particularly in the context we are interested in. The key questions we ask are thus the following:

- After the symmetry reduction, how many unique joint entropy terms will still remain?
- After the symmetry reduction, how many unique elemental information inequalities will still remain?

There is in fact a simple method to estimate (lower bound) these quantities. The group action defined through the permutation group on the power set of the random variables in the problem induces certain equivalence classes (often referred to as orbits of a group action), and the maximum number of elements in any class is the number of permutations in the specific problem setting. Therefore, lower bounds to the two quantities can be obtained by dividing the total number of LP variables and that of LP constraints by the number of permutations. However, since some orbits can have fewer elements, these lower bounds are not precise. A method to pinpoint the symmetry structure and count these numbers accurately is thus needed.

A powerful tool to tackle this type of orbit counting problem is the Pólya counting theorem, which requires finding the cycle index of the group action. Our main task thus reduces to first identifying the group and group action, and then finding the cycle index. In this study, we propose a generic three-layer decomposition of the group structure in typical coding problems and information systems, which allows us to conveniently apply the Pólya counting theorem. In particular we apply this approach on three problems: the problem of extremal pairwise cyclically symmetric entropy inequalities in [116, 117], the regenerating code problem [110, 118, 119], and the caching problem [39, 77, 91, 131, 132]. For all these problems, we provide explicit formulae for or efficient algorithms to compute the two quantities of interest. In fact, two of them lead to cycle indices previously studied in the literature [120–122, 133], while the other appears less well studied before.

The three-layer decomposition not only fits the three problems we study here, but also is well motivated in general. The first layer, referred to as the base layer, can be viewed as reflecting certain physically meaningful permutations of the components in the communication or information systems, while the second layer is a permutation induced by the base layer on the random variables representing more abstract relations among the system components. The third layer is on the power set of the random variables, which directly relates to the joint entropies.

It should also be clarified that the cycle index is a concept associated with a group action, and a group acting on different sets may induce different cycle indices. A more intuitive interpretation of a group action is through its permutation representation, which in fact directly relates to the cycle index. In this work, we make a conscientious effort to reduce explicit reliance on the notion of group actions, but rather favor permutation representations because of the explicit physical interpretation. In the rest of the chapter, we will occasionally revert to group actions, which sometimes are more concise, and may be more meaningful for readers familiar with the mathematical tools developed using group actions.

The proposed method can be used to calculate the numbers of LP variables and LP constraints after symmetry reduction, while the aforementioned estimates are inaccurate. Numerically we observe that their relative difference becomes negligible as they both grow large, and thus our result is more reassuring than surprising in nature. We only consider the reduction due to the symmetry, which does not include other possible reductions, for example, reductions due to implication relations among the random variables. Such reductions were indeed utilized in [110] and can in fact be rather significant, however, it is difficult to identify the general amount of reduction since it is highly problem dependent; this topic is thus beyond the scope of our current study. Furthermore, our focus is on the reduction of the LP variables and the LP constraints, not the isomorphic relation among asymmetric problem instances, nor the symmetry in the geometry of the constrained entropic polytope; results related to these aspects can be found in [123, 124]. A less obvious but important byproduct of our study is the formalization of the permutation representations of the symmetry in the three problems, which are in fact needed in representing the problem in a computer program, and may be of value to researchers interested in implementing such software.

The rest of this chapter is organized as follows, in section 4.1 we provide a generic three-layer decomposition to investigate the problem. Section 4.2 gives an introduction to the critical mathematical tool. In sections 4.3, 4.4, 4.5, we provide the results on three different problems. Technical proofs are given in the appendices.

4.1 Permutation Group, Cycle Index and the Pólya Counting Theorem

In this section we provide some necessary backgrounds on groups, group actions, and the Pólya counting theorem.

4.1.1 Permutation Group

A *group* is a set G with a binary operation $\circ : G \times G \to G$, satisfying the four axioms:

- (Closure) $\forall a, b \in G, a \circ b \in G$;
- (Identity) $\exists e \in G, \forall a \in G, e \circ a = a \circ e = a$;
- (Associativity) $\forall a, b, c \in G, (a \circ b) \circ c = a \circ (b \circ c)$;
- (Inverse) $\forall a \in G, \exists b \in G, a \circ b = b \circ a = e$.

Therefore, a group is formally denoted as $(G, \circ)$. In the sequel, we will omit $\circ$ and just write G as a group for simplicity, and may also omit $\circ$ in $a \circ b$. A *permutation* π defined on the finite set $\mathcal{Y}$ is a bijective mapping from $\mathcal{Y}$ to itself, and a *permutation group* G is a group with its elements being distinct permutations (of set $\mathcal{Y}$), and the operation $\circ$ is the composition of permutations. Two specific permutation groups are particularly important in this study. The *symmetric group* Sym($\mathcal{Y}$) defined on the set $\mathcal{Y}$ is a permutation group which contains all possible permutations of $\mathcal{Y}$. Especially, when $\mathcal{Y}$ contains n elements, the symmetric group can be written as S_n without loss of any generality. The symmetric group plays an important role in the theory of group action.

Cayley's theorem states that every group is isomorphic to a subgroup of some symmetric group [134]. The *cyclic group* C_n defined on an n-element set is a permutation group where there exists a $\pi \in C_n$ such that $C_n = \{\pi^k : k \in \mathbb{Z}\}$.

A *group action* of a group G on a set $\mathcal{Y}$ is a function $* : G \times \mathcal{Y} \to \mathcal{Y}$, which satisfies the two axioms:

- (Identity) $\forall y \in \mathcal{Y}, e * y = y$;
- (Associativity) $\forall g, h \in G$ and $\forall y \in \mathcal{Y}, g * (h * y) = (gh) * y$.

In the rest of the chapter, we shall use $g(y)$ to represent a group action, which is particularly meaningful for permutation groups, i.e., g is a permutation that maps an element y to $g(y)$.

The *permutation representation* of a group action $(G, \mathcal{Y})$ is a group homomorphism $\phi : G \to \text{Sym}(\mathcal{Y})$ [135]. Especially, when G is a permutation group, each permutation in G will permute $\mathcal{Y}$ in a canonical way. Group actions and permutation representations are simply different viewpoints of the same mathematical concept. This claim is made formal by the following theorem which can be found in [135].

Theorem 5. *(Equivalence of group actions and permutation representations [135]) For a fixed group G and a set $\mathcal{Y}$, let A be the set of all group actions of G on $\mathcal{Y}$, and P be the set of all permutation representations of G on $\mathcal{Y}$, there exist mutually inverse bijections $F : A \to P$ and $I : P \to A$, given by*

$$F(*) : \phi(g) = (y \mapsto g * y : y \in \mathcal{Y}), \tag{4.1}$$

$$I(\phi) : g * y = \phi(g)(y) \quad (g \in G, y \in \mathcal{Y}), \tag{4.2}$$

where $* : G \times \mathcal{Y} \to \mathcal{Y}$ is a group action and $\phi : G \to Sym(\mathcal{Y})$ is a permutation representation.

A special group action (in its permutation representation) related to our study is the *power group*[†], which is a group action produced by two group actions [120]. Let G be a group acting

[†]The permutation representation of the power group is in fact isomorphic to the direct product of the two groups [120].

on the set $\mathcal{Y}$, and H be a group acting on the set $\mathcal{V}$, then the *power group* H^G is defined to be acting on the set $\mathcal{V}^{\mathcal{Y}} = \{f : \mathcal{Y} \to \mathcal{V}\}$, i.e., all mappings from $\mathcal{Y}$ to $\mathcal{V}$. The cardinality of this set is clearly $|\mathcal{V}^{\mathcal{Y}}| = |\mathcal{V}|^{|\mathcal{Y}|}$, where $|\cdot|$ denotes the cardinality of a set. Let us assign an arbitrary but fixed order on the distinct elements of $\mathcal{Y}$ and write them in order as $y_1, y_2, \ldots, y_{|\mathcal{Y}|}$. Any such function $f : \mathcal{Y} \to \mathcal{V}$ can be represented by the vector, $[a_{y_1}, a_{y_2}, \ldots, a_{y_{|\mathcal{Y}|}}] \triangleq [f(y_1), f(y_2), \ldots, f(y_{|\mathcal{Y}|})]$. The case particularly relevant to our study is when G and H are permutation groups. For this case, let $g \in G, h \in H$, then $h^g \in H^G$ acts on a vector $[a_{y_1}, a_{y_2}, \ldots, a_{y_{|\mathcal{Y}|}}]$ as

$$\begin{aligned}
&h^g\big([a_{y_1}, a_{y_2}, \ldots, a_{y_{|\mathcal{Y}|}}]\big)\\
=&h^g\big([f(y_1), f(y_2), \ldots, f(y_{|\mathcal{Y}|})]\big)\\
=&[h(f(g(y_1))), h(f(g(y_2))), \ldots, h(f(g(y_{|\mathcal{Y}|})))]\\
=&[h(a_{g(y_1)}), h(a_{g(y_2)}), \ldots, h(a_{g(y_{|\mathcal{Y}|})})].
\end{aligned} \tag{4.3}$$

4.1.2 The Cycle Index

A permutation π acting on a finite set $\mathcal{Y}$ can be represented by its *cycle notation*, which is a product of disjoint cycles, with each cycle being a subset of $\mathcal{Y}$ in which the elements are cyclically permuted by π. The order of the cycles does not matter, and moreover, the rotations of elements in a cycle yield the same cycle. The *cycle index of a permutation* π is a monomial $x_1^{c_1} x_2^{c_2} \ldots x_n^{c_n}$, which indicates that there are c_i length-i cycles in this permutation. For example, permutation $\pi = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 1 & 2 \end{smallmatrix}\right)$ defined on set $\mathcal{Y} = \{1, 2, 3, 4, 5\}$ can be represented by the cycle notation $\pi = (2\ 5)(1\ 3\ 4)$, and its cycle index is $x_2^1 x_3^1$. Notice that a permutation of n elements can have a cycle length of at most n, and some of the exponents in its cycle index may be zero. Let G be a group whose elements are permutations of $\mathcal{Y}$, then the *cycle index of group G* is the summation of the cycle indices of all permutations in G divided by the number of permutations in G,

$$P_G(x_1, x_2, \ldots, x_n) = \frac{1}{|G|} \sum_{\pi \in G} x_1^{c_1} x_2^{c_2} \ldots x_n^{c_n}. \tag{4.4}$$

Some permutations in G may have the same cycle index, thus they together yield a monomial in P_G with the coefficient being greater than one, in fact, $P_G(x_1, x_2, \ldots, x_n)$ is a posynomial function in terms of variables $x_1, x_2, \ldots, x_n$. Since there is no special meaning associated with the variable x, we omit it and just write P_G instead. In the sequal, we use $\mathcal{I}_n$ to denote the set $\{1, 2, \ldots, n\}$. The cycle indices of several permutation groups have been well studied [136].

- The cycle index of the symmetric group S_n acting on $\mathcal{I}_n$ is

$$P_{S_n} = \frac{1}{n!} \sum_{(i)} h(i) \prod_{k=1}^{n} x_k^{i_k}, \tag{4.5}$$

  where $(i) = (i_1, i_2, \ldots, i_n)$ is a partition of n, i_k is the number of parts equal to k; the summation is over all partitions (i), and $h(i)$ is given as $h(i) = n!/\prod_{k=1}^{n} k^{i_k} i_k!$. For example, $S_3 = \{I, (12)(3), (13)(2), (23)(1), (123), (132)\}$ has a cycle index of $P_{S_3} = \frac{1}{6}(x_1^3 + 3x_1x_2 + 2x_3)$.

- The cycle index of the cyclic group C_n acting on $\mathcal{I}_n$ is

$$P_{C_n} = \frac{1}{n} \sum_{k|n} \phi(k) x_k^{n/k}, \tag{4.6}$$

  where the summation is over all positive divisors k of n, including 1 and n; $\phi(k)$ is the Euler's totient function, giving the number of natural numbers which are less than and relatively prime to k. For example, $C_3 = \{I, (123), (132)\}$ has a cycle index of $P_{C_3} = \frac{1}{3}(x_1^3 + 2x_3)$.

- The cycle index of a power group H^G acting on the set $\mathcal{V}^{\mathcal{Y}}$ is

$$P_{H^G} = \frac{1}{|G||H|} \sum_{h^g \in H^G} \prod_{k=1}^{|\mathcal{V}|^{|\mathcal{Y}|}} x_k^{j_k(g;h)}, \tag{4.7}$$

  where for $k > 1$,

$$j_k(g;h) = \frac{1}{k} \sum_{s|k} \mu\Big(\frac{k}{s}\Big) j_1(g^s; h^s), \tag{4.8}$$

and

$$j_1(g;h) = \prod_{k=1}^{|\mathcal{Y}|} \Big(\sum_{s|k} s j_s(h) \Big)^{j_k(g)}, \tag{4.9}$$

where function $j_k(\pi)$ gives the number of length-k cycles in a permutation π; $\mu(n)$ is the Möbius function, taking a value from $\{-1, 0, 1\}$ depending on the factorization of n into prime numbers.

4.1.3 The Pólya Counting Theorem

The Pólya counting theorem is a useful tool in combinatorics to count the number of orbits of a group acting on a set, which plays an instrumental role in our study. It is a generalized version of the Burnside's lemma.

The coloring problem is a standard route to introduce the Pólya counting theorem, which can be described as follows. There are a finite number of objects (vertices / beads / etc.) forming a set $\mathcal{D}$, and a finite number of colors ($r$ed, $g$reen, $b$lue, etc.) forming another set $\mathcal{R} = \{r, g, b, \ldots\}$ that can be used to color the objects. Different ways of coloring may be viewed as identical under some group actions: this can be specified as a permutation group G acting on $\mathcal{D}$, reflecting different operations (rotations, reflections, etc.) that can be applied on these objects. The coloring problem asks the number of unique ways of performing such coloring. In the sequel, we write the set $\mathcal{D}$ as a subscript in $G_{\mathcal{D}}$, to make explicit what set the permutation group G is acting on.

To state the Pólya counting theorem, several more concepts are necessary [137]. Let z be a *coloring* of all the objects from $\mathcal{D}$, then the set of all colorings is $\mathcal{D}^{\mathcal{R}} = \{z : \mathcal{D} \to \mathcal{R}\}$. For a group action (written in its permutation representation $G_{\mathcal{D}}$), two colorings $z_1, z_2 \in \mathcal{D}^{\mathcal{R}}$ are *equivalent*, denoted as $z_1 \sim z_2$, if and only if there exists a $\pi \in G_{\mathcal{D}}$ such that

$$z_1(\pi(d)) = z_2(d), \forall d \in \mathcal{D}. \tag{4.10}$$

The equivalence relation partitions all possible colorings $\mathcal{D}^{\mathcal{R}}$ into disjoint subsets, and each subset is denoted as an *equivalence class* (often referred to as an *orbit*), denoted as $\mathcal{B}$.

The *weight w of an element* in a set $\mathcal{R}$ is a (any) symbol assigned to this element. The symbol itself means nothing but a placeholder to denote the existence of an element in a formula. For example, we can simply assign $w(r) = r, w(g) = g$ and so on. A weight is assigned to each coloring, and the *weight of a coloring z* is

$$W(z) = \prod_{d \in \mathcal{D}} w[z(d)]. \tag{4.11}$$

Note that if $z_1 \sim z_2$, then $W(z_1) = W(z_2)$. The *orbit weight* $W(\mathcal{B})$ is this (common) weight value of the colorings in this orbit. The *orbit inventory* is the sum of all orbit weights,

$$\sum_{\mathcal{B} \subseteq \mathcal{D}^{\mathcal{R}}} W(\mathcal{B}), \tag{4.12}$$

where each orbit weight $W(\mathcal{B})$ can be meaningfully interpreted as a way of coloring using a specific number of each color. The coloring problem posed above is thus reduced to finding the orbit inventory, which is well solved by the Pólya counting theorem.

Theorem 6. *(The Pólya Counting Theorem) Let $\mathcal{D}$ be a finite set of objects and $\mathcal{R}$ be a finite set of colors, and $G_{\mathcal{D}}$ be a permutation group acting on $\mathcal{D}$. The orbit inventory from domain $\mathcal{D}$ to range $\mathcal{R}$ is*

$$\sum_{\mathcal{B} \subseteq \mathcal{D}^{\mathcal{R}}} W(\mathcal{B}) = P_{G_{\mathcal{D}}}\left(\sum_{r \in \mathcal{R}} w(r), \sum_{r \in \mathcal{R}} (w(r))^2, \ldots, \sum_{r \in \mathcal{R}} (w(r))^n \right), \tag{4.13}$$

that is, the orbit inventory is obtained by substituting $\sum_{r \in \mathcal{R}} w(r)$ into x_1, $\sum_{r \in \mathcal{R}} (w(r))^2$ into x_2, $\ldots$,$\sum_{r \in \mathcal{R}} (w(r))^n$ into x_n in the cycle index $P_{G_{\mathcal{D}}}$.

Example 1. *(The necklace problem) Consider a necklace made of four beads, i.e., $\mathcal{D} = \mathcal{I}_4$, acted on $\mathcal{D}$ by a cyclic group $G_{\mathcal{I}} = C_4 = \{I, (1234), (13)(24), (1432)\}$, corresponding to the $0^{\mathrm{o}}, 90^{\mathrm{o}}, 180^{\mathrm{o}}$ and 270^{o} rotations, respectively. Different ways to color the beads using, e.g., three colors $\mathcal{R} = \{r, g, b\}$, can be found using the theorem above. The cycle index is $P_{G_{\mathcal{I}}} = \frac{1}{4}(x_1^4 + x_2^2 + 2x_4)$. Assigning the weights $w(r) = r, w(g) = g, w(b) = b$ and substi-*

tuting $x_1 = r + g + b, x_2 = r^2 + g^2 + b^2, x_4 = r^4 + g^4 + b^4$ *give* $P_{G_{\mathcal{I}}} = b^4 + b^3g + 2b^2g^2 + bg^3 + g^4 + b^3r + 3b^2gr + 3bg^2r + g^3r + 2b^2r^2 + 3bgr^2 + 2g^2r^2 + br^3 + gr^3 + r^4$. *Thus there is one way to color the necklace into three blue balls and one green ball, two ways to color it into two blue balls and two green balls, one way to color it into one blue ball and three green balls, and so on.*

In the example above, we wrote $G_{\mathcal{I}}$ and $P_{G_{\mathcal{I}}}$ instead of $G_{\mathcal{I}_4}$ and $P_{G_{\mathcal{I}_4}}$ for notational convenience. In the sequel, we will use the same simplification on $\mathcal{I}_n$ when the value of n is clear from the context.

4.2 A Generic Three-Layer Decomposition

4.2.1 Linear Programming on the Entropy Space

The overall derivation of the fundamental limits or outer bounds of information systems and communication systems can be viewed as a series of linear programs. For a set of m random variables, $\mathcal{X} = \{X_1, X_2, \dots, X_m\}$, the corresponding LP variables in the LP of interest will represent the following quantities:

$$H(X_{\mathcal{A}}),\ \mathcal{A} \subseteq \mathcal{I}_m, \tag{4.14}$$

where $H(X_{\mathcal{A}}) \overset{\text{def}}{=} H(X_i : i \in \mathcal{A})$ and we take the convention that $H(X_\emptyset) = 0$. The LP constraints are the following elemental Shannon-type inequalities [105]:

$$H(X_i|\{X_k, k \neq i\}) \geq 0, \quad i, k \in \mathcal{I}_m, \tag{4.15}$$

$$I(X_i; X_j|X_{\mathcal{K}}) \geq 0,\ \text{where } \mathcal{K} \subseteq \mathcal{I}_m \backslash \{i, j\}, \quad i \neq j. \tag{4.16}$$

As aforementioned, there may be also problem-specific constraints in each problem, however, they are usually simple to represent, and we will only mention them when necessary in the three problems we treat.

4.2.2 A Generic Three-Layer Decomposition

The above LP has a set of random variables $\mathcal{X}$ indexed by the base set $\mathcal{I}_m$. However, in many coding problems the set of random variables cannot be simply indexed by $\mathcal{I}_m$. Moreover, the LP variables (4.14) are defined on all subsets of $\mathcal{X}$, i.e., the *power set* $\mathcal{P}(\mathcal{X})$, which implies that the set $\mathcal{P}(\mathcal{X})$ is the set we should be interested in.

The symmetry embedded in a problem induces certain group structure on the power set $\mathcal{P}(\mathcal{X})$, but the overall symmetry can be decomposed into different layers. The core symmetry in an engineering problem can usually be defined on one or more simple base set $\mathcal{I}_m$, formally introduced as a permutation group $G_{\mathcal{I}_m}$ defined on the set $\mathcal{I}_m$. Then the group $G_{\mathcal{I}_m}$ acting on the set of random variables $\mathcal{X}$ and set $\mathcal{P}(\mathcal{X})$ can be defined through their permutation representations, thus as two induced permutation groups $G_{\mathcal{X}}$ and $G_{\mathcal{P}(\mathcal{X})}$. The orbits of LP variables and LP constraints are directly related to the permutation group $G_{\mathcal{P}(\mathcal{X})}$. To paraphrase, the following three layers are usually present in a typical coding problem:

- Layer 1: The base index set(s) $\mathcal{I}_{m_i}$ and simple permutation group(s) $G_{\mathcal{I}_{m_i}}$, $i = 1, 2, \dots, k$;
- Layer 2: The induced random variable set $\mathcal{X}$ and the induced permutation group $G_{\mathcal{X}}$;
- Layer 3: The power set $\mathcal{P}(\mathcal{X})$ and the induced permutation group $G_{\mathcal{P}(\mathcal{X})}$.

The following example comes from the regenerating code problem with $n = 4$ nodes.

Example 2. *The nodes can be indexed by the index set $\mathcal{I}_n = \mathcal{I}_4 = \{1, 2, 3, 4\}$. In Layer 1, there is only one base index set $\mathcal{I}_4$ with a symmetric group $G_{\mathcal{I}} = S_4$. In Layer 2, the set of random variables is $\mathcal{X} = \{X_{ij} : i, j \in \mathcal{I}_4\}$, thus there are $m = n^2 = 16$ random variables. Each permutation $\pi \in S_4$ acting on set $\mathcal{I}_4$ produces a permutation $\pi_{\mathcal{X}}^{ind}$ acting on set $\mathcal{X}$,*

$$\pi_{\mathcal{X}}^{ind} : \mathcal{X} \to \mathcal{X} \quad as \quad \pi_{\mathcal{X}}^{ind}(X_{ij}) \overset{\text{def}}{=} X_{\pi(i)\pi(j)},$$

and all the permutations $\pi_{\mathcal{X}}^{ind}$ form an induced permutation group[‡] *$G_{\mathcal{X}}$. Permutation $\pi_{\mathcal{X}}^{ind}$ permutes X_{ij} in the way that it is isomorphic to the permutation π permuting i, j. For example, the*

[‡]We use the permutation representation $G_{\mathcal{X}}$ to replace the group action of the group $G_{\mathcal{I}}$ acting on the induced set $\mathcal{X}$. The fact that $G_{\mathcal{X}}$ is a group directly follows from the fact that it is homomorphic to the symmetric group. The same interpretation applies for all induced permutation groups defined in this study.

permutation $\pi = (2)(143) \in S_4$ *induces a permutation in Layer 2 as*

$$\pi_{\mathcal{X}}^{ind} = (X_{22})(X_{11}X_{44}X_{33})(X_{12}X_{42}X_{32})(X_{13}X_{41}X_{34})(X_{14}X_{43}X_{31})(X_{21}X_{24}X_{23}).$$

In Layer 3, the power set is $\mathcal{P}(\mathcal{X}) = \{X_{\emptyset}, X_{\mathcal{A}_1}, X_{\mathcal{A}_2}, \ldots, X_{\mathcal{A}_{2^{16}-1}}\}$*, where*[§] $X_{\mathcal{A}_1} = \{X_{44}\}, X_{\mathcal{A}_2} = \{X_{43}\}, X_{\mathcal{A}_3} = \{X_{43}, X_{44}\}, \ldots, X_{\mathcal{A}_{2^{15}}} = \{X_{11}\}, \ldots, X_{\mathcal{A}_{2^{16}-1}} = \mathcal{X}$*. Each permutation* $\pi_{\mathcal{X}}^{ind} \in G_{\mathcal{X}}$ *acting on set* $\mathcal{X}$ *produces a permutation* $\pi_{\mathcal{P}(\mathcal{X})}^{ind}$ *acting on set* $\mathcal{P}(\mathcal{X})$*,*

$$\pi_{\mathcal{P}(\mathcal{X})}^{ind} : \mathcal{P}(\mathcal{X}) \to \mathcal{P}(\mathcal{X}) \quad as \quad \pi_{\mathcal{P}(\mathcal{X})}^{ind}(X_{\mathcal{A}}) \overset{\text{def}}{=} \{\pi_{\mathcal{X}}^{ind}(X_{ij}) : X_{ij} \in X_{\mathcal{A}}\},$$

and all the $\pi_{\mathcal{P}(\mathcal{X})}^{ind}$ *permutations form an induced permutation group* $G_{\mathcal{P}(\mathcal{X})}$*. Permutation* $\pi_{\mathcal{P}(\mathcal{X})}^{ind}$ *permutes* $X_{\mathcal{A}}$ *in the way that it is isomorphic to the permutation* $\pi_{\mathcal{X}}^{ind}$ *permuting* $X_{ij} \in X_{\mathcal{A}}$*, which is further isomorphic to the permutation* π *permuting* i *and* j*. In particular, the above* $\pi_{\mathcal{X}}^{ind}$ *induces a permutation in Layer 3 as*

$$\pi_{\mathcal{P}(\mathcal{X})}^{ind} = (X_{\emptyset})(X_{\mathcal{A}_{1024}})(X_{\mathcal{A}_{32768}}X_{\mathcal{A}_1}X_{\mathcal{A}_{32}}) \ldots\ldots (X_{\mathcal{A}_{3072}}X_{\mathcal{A}_{1280}}X_{\mathcal{A}_{1536}}) \ldots\ldots (X_{\mathcal{A}_{2^{16}-1}}).$$

In order to answer the two questions we posed earlier, there is in fact no need to directly study the structure of $G_{\mathcal{P}(\mathcal{X})}$ in the third layer, because through a proper coloring problem formulation, the Pólya counting theorem can be invoked, and only the cycle index of the group $G_{\mathcal{X}}$ in the second layer is needed. With this in mind, a meta algorithm that decomposes the computation of the Pólya counting theorem is given below. As we shall show, all three problems considered in this work can be solved this way.

[§]This notation comes from that we see $\mathcal{P}(\mathcal{X})$ as isomorphic (denoted as $\simeq$ for short) to the binary representation of all integers from 0 to $2^{16} - 1$. To be specific, each element (subset) in $\mathcal{P}(\mathcal{X})$ corresponds to a 16-digit binary sequence, where "1" indicates the corresponding random variable being present in this subset, whereas "0" indicates nonexistence. Therefore, $\{X_{44}\} \simeq 0\ldots001$ (Binary) $\simeq X_{\mathcal{A}_1}$ (Decimal), $\{X_{11}, X_{12}, X_{21}\} \simeq 1100100\ldots0 \simeq X_{\mathcal{A}_{51200}}$ ($2^{15} + 2^{14} + 2^{11} = 51200$).

Meta Algorithm 1.

Step 1 (Layer 1): Find the cycle index $P_{G_{\mathcal{I}}}$;

Step 2 (Layer 2): Derive the cycle index $P_{G_{\mathcal{X}}}$, after confirming $G_{\mathcal{X}}$ is a group;

Step 3 (Layer 3): (Number of LP variables) For $P_{G_{\mathcal{X}}}$, apply (4.13) using a two-color set $\mathcal{R} = \{r, g\}$ and assign $w(r) = w(g) = 1$; output the obtained number subtracting one;

Step 4 (Layer 3): (Number of LP constraints of form (4.16)): For $P_{G_{\mathcal{X}}}$, apply (4.13) using a three-color set $\mathcal{R} = \{r, g, b\}$, then expand the formula and output the summation of all the coefficients before the terms including r^2.

Step 1 of the meta algorithm may be omitted, if one can directly identify $P_{G_{\mathcal{X}}}$. However, our experience suggests that Step 3 and 4 can be significantly simplified if $P_{G_{\mathcal{I}}}$ is identified. In Step 2, confirming that $G_{\mathcal{X}}$ is a group is necessary but usually trivial, as we will discuss in each problem. Step 3 of the algorithm turns calculating the number of patterns into a coloring problem with two colors, one for the random variables in the set $X_{\mathcal{A}}$ in $H(X_{\mathcal{A}})$ and the other for those in the set $\mathcal{X} \backslash X_{\mathcal{A}}$. The number of patterns is the summation of all coefficients of the orbit inventory (4.13), and by assigning $w(r) = w(g) = 1$, we can directly obtain it. To eliminate the empty set $H(X_{\emptyset}) = 0$, one is subtracted in the end. Step 4 of the algorithm provides the number of constraints of form (4.16) as the summation of all coefficients before terms $r^2 g^p b^q$. To see this indeed gives the number of orbits, consider using red for X_i and X_j, green for the random variables in $X_{\mathcal{K}}$, and blue for all other random variables in $\mathcal{X} \backslash (X_i \cup X_j \cup X_{\mathcal{K}})$. The same (red) color can be used for X_i and X_j because $I(X_i; X_j | \bullet) = I(X_j; X_i | \bullet)$, thus X_i, X_j can be colored indistinguishably as one group, and we have used p, q to denote the cardinality of $X_{\mathcal{K}}$ and $\mathcal{X} \backslash (X_i \cup X_j \cup X_{\mathcal{K}})$, respectively. For constraints of form (4.15), the number of orbits is usually trivial to find, and thus we have omitted it in the meta algorithm.

From now on, we slightly abuse the notation for simplicity: in the second layer, the subscript $\mathcal{X}$ in each $\pi^{ind}_{\mathcal{X}}$ will be omitted, since we do not need to study $\pi^{ind}_{\mathcal{P}(\mathcal{X})}$ and $G_{\mathcal{P}(\mathcal{X})}$ in the third layer directly, writing π^{ind} will cause no confusion.

4.3 Extremal Pairwise Cyclically Symmetric Entropy Inequalities

The extremal pairwise cyclically symmetric entropy inequalities were proposed in [116, 117], which generalize the celebrated Han's inequalities [138]. There are a total of n random variables in this problem, thus $m = n$, which are denoted as $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, and the corresponding index set is $\mathcal{I}_n$. The group $G_{\mathcal{I}}$ is a cyclic group defined on $\mathcal{I}_n$, i.e., $G_{\mathcal{I}} = C_n$. The orbit $\mathcal{O}$ of a nonempty subset $\mathcal{A} \subseteq \mathcal{I}_n$ is a collection of distinct subsets resulted by performing C_n on $\mathcal{A}$,

$$\mathcal{O}(\mathcal{A}) = \{\mathcal{A}, \pi(\mathcal{A}), \ldots, \pi^{n-1}(\mathcal{A})\}. \tag{4.17}$$

In an orbit $\mathcal{O}$ there are $|\mathcal{O}|$ elements and each element is a subset of $\mathcal{I}_n$ with cardinality $\ell_{\mathcal{O}}$. For any given $\mathcal{A} \subseteq \mathcal{I}_n$, the joint entropy of a subset of random variables with probability distribution $P(X_1, X_2, \ldots, X_n)$ is written as

$$H_P(X_{\mathcal{A}}) = H_P(\{X_i : i \in \mathcal{A}\}). \tag{4.18}$$

The *cyclic orbit entropy* $H_P(\mathcal{O})$ is the average entropy defined on a non-empty orbit $\mathcal{O}$,

$$H_P(\mathcal{O}) = \frac{1}{|\mathcal{O}|} \sum_{\mathcal{A} \in \mathcal{O}} H_P(X_{\mathcal{A}}). \tag{4.19}$$

The average orbit entropy for orbit $\mathcal{O}$ is defined as

$$h_P(\mathcal{O}) = \frac{1}{|\mathcal{O}|\ell_o} \sum_{\mathcal{A} \in \mathcal{O}} H_P(X_{\mathcal{A}}). \tag{4.20}$$

The following extremal value of $c \in \mathbb{R}^+$ is of particular interest [117],

$$c_{\mathcal{O},\mathcal{O}'} = \min\{c : c h_P(\mathcal{O}) \geq h_P(\mathcal{O}') \text{ for any distribution } P\}, \tag{4.21}$$

when the minimum exists. A moment of thought reveals that this definition can be rewritten as

$$c_{\mathcal{O},\mathcal{O}'} = \sup_{P:h_P(\mathcal{O})>0} \frac{h_P(\mathcal{O}')}{h_P(\mathcal{O})}. \tag{4.22}$$

The constraint $h_P(\mathcal{O}) > 0$ can be safely ignored in this context, since for any orbit, $h_P(\mathcal{O}) \leq 0$ if and only if $H_P(X_i) = 0$ for $i = 0, \ldots, n-1$.

Analytically characterizing the upper bound of $c_{\mathcal{O},\mathcal{O}'}$ turns out to be difficult and thus a computer-aided approach was used in [117]. An upper bound of $c_{\mathcal{O},\mathcal{O}'}$ is the optimal value of the following LP problem[¶],

$$\text{maximize: } \frac{\ell_{\mathcal{O}}}{|\mathcal{O}'|\ell_{\mathcal{O}'}} H_P(\mathcal{O}') \tag{4.23}$$

$$\text{subject to: } H_P(\mathcal{O}_n) - H_P(\mathcal{O}_{n-1}) \geq 0, \tag{4.24}$$

$$\begin{aligned} &H_P(\mathcal{O}(\{i\} \cup \mathcal{Q})) + H_P(\mathcal{O}(\{j\} \cup \mathcal{Q})) - H_P(\mathcal{O}(\mathcal{Q})) \\ &- H_P(\mathcal{O}(\{i\} \cup \{j\} \cup \mathcal{Q})) \geq 0, i \neq j,\ i,j \in \mathcal{I}_n,\ \mathcal{Q} \subseteq \mathcal{I}_n \backslash \{i,j\}, \end{aligned} \tag{4.25}$$

$$H_P(\mathcal{O}) = 1, \tag{4.26}$$

where $\mathcal{O}_l, l \in \{0, 1, n-1, n\}$ denotes the unique orbit $\mathcal{O}$ of $\mathcal{I}_n$ with $\ell_{\mathcal{O}} = \ell$.

The constraint (4.26) is a problem-specific equality, and there is only one such equality for a specific pair of orbits, and thus for the given LP. We are interested in the number of distinct constraints (4.24) and (4.25), which are already symmetry-reduced from (4.15) and (4.16), respectively. It is straightforward to see that there is only one inequality of form (4.24).

In this problem, the first layer is on $\mathcal{I}_n$ with cyclic group C_n. In the second layer, the induced

[¶]There are two distinct arguments to reach this symmetric version of the LP. The first argument is that we can show that without loss of generality, only symmetric distributions need to be considered. As such the joint entropies in each orbit are equal for such probability distributions, which lead to a symmetric optimization problem. In this specific problem, this optimization problem has a non-LP objective, and we can further show that it can be replaced with a linear objective without loss of generality. The second approach is to start with an optimization problem without any symmetry on the solution, and directly show that it is equivalent to a symmetric LP in the problem setting. Generally speaking, the first argument is usually more straightforward and intuitive, thus is what we follow for the other two problems. To be complete, in Appendix A we provide a proof following the second argument for this problem as an illustration.

Table 4.1: The numbers of LP variables and LP constraints in the extremal pairwise cyclically symmetric entropy inequalities problem, problem parameter n from 5 to 10. Reprinted with permission from [4, 5], © 2017,2018 IEEE.

	n	5	6	7	8	9	10
Number of LP Variables	Original	31	63	127	255	511	1023
	Estimated	6.2	10.5	18.1429	31.875	56.7778	102.3
	After Symmetry Reduction	7	13	19	35	59	107
Number of LP Constraints	Original	85	246	679	1800	4617	11530
	Estimated	17	41	97	225	513	1153
	After Symmetry Reduction	17	43	97	229	513	1161

set is the set of random variables $\mathcal{X}$, and the induced permutation π^{ind} (by a permutation $\pi \in C_n$) is

$$\pi^{ind} : \mathcal{X} \to \mathcal{X} \quad \text{as} \quad \pi^{ind}(X_i) \overset{\text{def}}{=} X_{\pi(i)}, \tag{4.27}$$

and all such π^{ind} permutations form a permutation group,

$$G_{\mathcal{X}} = \{\pi^{ind} : \pi^{ind} \text{ induced by } \pi \in C_n\}, \tag{4.28}$$

which is clearly a cyclic group as well. As a consequence, the cycle index $P_{G_{\mathcal{X}}}$ is simply (4.6), and Meta Algorithm 1 can now be invoked straightforwardly.

Example 3. *For $n = 4$, in the first layer, $\mathcal{I}_4 = \{1, 2, 3, 4\}$ and $G_{\mathcal{I}}$ is a cyclic group $C_4 = \{e, (1234), (13)(24), (1432)\}$. The second layer is $\mathcal{X} = \{X_1, X_2, X_3, X_4\}$ and $G_{\mathcal{X}} = C_4$, which is isomorphic to the first layer. The numbers of LP variables and constraints can be estimated as follows. Since there are $2^n - 1 = 15$ variables and $n + \binom{n}{2}2^{n-2} = 28$ elemental Shannon-type constraints originally, we can lower bound the number of LP variables as $15/|C_4| = 3.75$, and lower bound the number of LP constraints as $28/|C_4| = 7$.*

The precise number of orbits (LP variables) can be calculated using (4.6), which in this example is $P_{G_{\mathcal{X}}} = \frac{1}{4}\big(x_4 \cdot \phi(4) + x_2^2 \cdot \phi(2) + x_1^4 \cdot \phi(1)\big)$. Step 3 of Meta Algorithm 1 gives

$$P_{G_{\mathcal{X}}} = \frac{1}{4}\Big((r^4 + g^4) \cdot \phi(4) + (r^2 + g^2)^2 \cdot \phi(2) + (r + g)^4 \cdot \phi(1)\Big) = r^4 + r^3 g + 2r^2 g^2 + r g^3 + g^4.$$

The sum of all coefficients is 6*, which is the number of orbits. These orbits can be listed as* $\mathcal{O}_0 = \emptyset, \mathcal{O}_1 = \{\{1\},\{2\},\{3\},\{4\}\}, \mathcal{O}_2 = \{\{1,2\},\{2,3\},\{3,4\},\{1,4\}\}, \mathcal{O}_3 = \{\{1,3\},\{2,4\}\}$, $\mathcal{O}_4 = \{\{1,2,3\},\{2,3,4\},\{1,3,4\},\{1,2,4\}\}, \mathcal{O}_5 = \{\{1,2,3,4\}\}$*. Removing the empty orbit* $\mathcal{O}_0$ *gives the number of LP variables as* 5*.*

For the number of constraints (4.25), following the Step 4 of the Meta Algorithm 1, we have

$$\begin{aligned} P_{G_\mathcal{X}} &= \frac{1}{4}\Big((r^4+g^4+b^4)\cdot\phi(4) + (r^2+g^2+b^2)^2\cdot\phi(2) + (r+g+b)^4\cdot\phi(1)\Big) \\ &= \ldots\ldots + 2r^2g^2 + 3r^2gb + 2r^2b^2 + \ldots\ldots, \end{aligned}$$

the sum of the coefficients before the terms including r^2 *is* 7*. Thus there are 8 distinct elemental Shannon-type constraints in total (7 of form (4.25) and 1 of form (4.24)).*

The results for several other parameters can be found in Table 4.1. The numbers after symmetry reduction are provided using our approach, and the estimates can be quite close or even match in some cases.

4.4 The Regenerating Code Problem

Regenerating codes are a class of erasure codes that are useful in distributed storage systems which can provide more efficient repair [118, 119]. In a system consisting of n nodes, a message can be encoded using regenerating code and then stored separately in the n nodes, such that a user can recover the entire message by connecting to k nodes. If a node fails, a repaired node can join in the system by connecting to any d of the $n-1$ remaining nodes and downloading messages from them (Fig. 4.1). In this setting, one key issue is to study the trade-off between the amount of data stored in the nodes (storage) and the amount of data downloaded during the regeneration (repair bandwidth). We shall focus on the case $d = n-1$ since this is the most practically important case, and it is also the simplest notationally and conceptually.

The random variables form the set $\mathcal{X} = \{X_{ij} : i, j \in \mathcal{I}_n\}$, when $i = j$, X_{ii} represents the information stored at each node i; when $i \neq j$, X_{ij} represents the information transmitted from

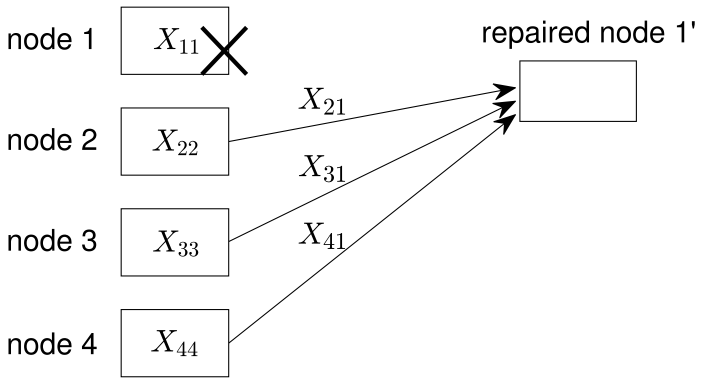

Figure 4.1: (4,3,3) regenerating code, node 1 fails. Reprinted with permission from [4, 5], © 2017,2018 IEEE.

node i to help a failed node j during repair[‖]. Therefore, there are $m = n^2$ random variables in this problem, implying that before symmetry reduction, there are $(2^{n^2} - 1)$ LP variables and $n^2 + \binom{n^2}{2} 2^{n^2-2}$ Shannon-type inequalities as constraints. It is clear that a permutation of the storage nodes would yield another valid code, and without loss of generality, only symmetric codes need to be considered. We will not write down the precise form of the LP for this problem, in terms of the objective function and the constraints here, since this is rather straightforward and has been given before [110].

The three-layer decomposition is as follows. The first layer is on the index set of the storage nodes, i.e., $\mathcal{I}_n$, with the symmetric group $G_{\mathcal{I}} = S_n$, which directly reflects the permutations of the storage nodes. The second layer is on the random variable set $\mathcal{X}$, and the induced permutation π^{ind} by a $\pi \in S_n$ is

$$\pi^{ind} : \mathcal{X} \to \mathcal{X} \quad \text{as} \quad \pi^{ind}(X_{ij}) \overset{\text{def}}{=} X_{\pi(i)\pi(j)}, \tag{4.29}$$

and all the π^{ind} permutations form a group

$$G_{\mathcal{X}} = \{\pi^{ind} : \pi^{ind} \text{ induced by } \pi \in S_n\}. \tag{4.30}$$

[‖]In previous works, e.g., [110], $\mathcal{W} = \{W_i : i \in \mathcal{I}_n\}$ were used to denote the contents stored in each node, and $\mathcal{S} = \{S_{ij} : i, j \in \mathcal{I}_n, i \neq j\}$ were used to denote the transmitted contents during repair. The reason that we adopt X_{ij} will be clear in the sequel.

The fact that $G_{\mathcal{X}}$ is a group[**] can be checked by its definition, since each π^{ind} trivially corresponds to a permutation $\pi \in S_n$. The cycle index of $G_{\mathcal{X}}$ is given by the following theorem.

Theorem 7. *In the $(n, k, n-1)$ regenerating code problem, if the cycle index of $G_{\mathcal{I}}$ can be represented as the following posynomial*

$$P_{G_{\mathcal{I}}} = \sum_{q=1,\dots,p(n)} a_q \cdot x_1^{c_{q1}} x_2^{c_{q2}} \dots x_n^{c_{qn}}, \tag{4.31}$$

where $p(n)$ is the partition function giving the number of possible partitions of a natural number n, and a_q is a coefficient denoting the number of permutations in $G_{\mathcal{I}}$ that correspond to a specific partition, then the cycle index of $G_{\mathcal{X}}$ is

$$P_{G_{\mathcal{X}}} = \sum_{q=1,\dots,p(n)} \left(a_q \cdot \prod_{\substack{l_1=1,2,\dots,n,\\ l_2=l_1+1,\dots,n}} x_{l_1}^{l_1 c_{ql_1}^2} x_{\mathrm{lcm}(l_1,l_2)}^{c_{ql_1 l_2}} \right), \tag{4.32}$$

*where $c_{ql_1l_2} = \frac{2l_1l_2c_{ql_1}c_{ql_2}}{\mathrm{lcm}(l_1,l_2)}$, and "*lcm*" stands for "least common multiple".*

The proof of this theorem can be found in Appendix B. Intuitively, the random variable set $\mathcal{X}$ can be viewed as a matrix with the element X_{ij} being on the ith row and jth column. Each dimension is indexed by $\mathcal{I}_n$, acted on by the same symmetric group S_n. Any permutation $\pi \in S_n$ produces one of the two cases: if i and j belong to cycles of the same length, say a length-l_1 cycle, then X_{ij} will also belong to a length-l_1 cycle; otherwise, if i belongs to a length-l_1 cycle and j belongs to a length-l_2 cycle ($l_2 \neq l_1$), then X_{ij} will belong to a length-$\mathrm{lcm}(l_1, l_2)$ cycle.

The only remaining issue is the number of constraints of form (4.15) after symmetry reduction, which is always 2 in this problem regardless of n, because the random variables in Layer 2 form two orbits, one contains all random variables located on the diagonal of the matrix, the other one contains those not on the diagonal.

[**]From the perspective of group action, this fact is straightforward to see in this case. Alternatively, one can also check the four group axioms. To be specific, the closure is implied by its definition; the associativity also comes with the definition since it is reduced to checking the associativity of π; the identity is the induced permutation of the identity permutation $e \in S_n$; the inverse of each permutation $\pi_{\mathcal{X}}^{ind}$ is the permutation π taking the inverse π^{-1}, and then be induced, which is $(\pi_{\mathcal{X}}^{-1})^{ind} \in G_{\mathcal{X}}$.

The cycle index in (4.32) is similar to the cycle index of the direct product of two symmetric groups acting on two independent sets, which can be found in [136]. The difference from the case that we have at hand is the following: here there is only one symmetric group acting simultaneously on the two sets (both indices in X's subscript).

Table 4.2: The numbers of LP variables and LP constraints in the $(n, k, n-1)$ regenerating code problem, problem parameter n from 3 to 8. Reprinted with permission from [4, 5], © 2017,2018 IEEE.

	n	3	5	6	7	8
Number of LP Variables	Original	511	3.3554×10^7	6.8719×10^{10}	5.6295×10^{14}	1.8447×10^{19}
	Estimated	85.6667	279620.2583	9.5444×10^7	1.1170×10^{11}	4.5751×10^{14}
	After Symmetry Reduction	103	291967	9.6929×10^7	1.1228×10^{11}	4.5820×10^{14}
Number of LP Constraints	Original	4617	2.5166×10^9	1.0823×10^{13}	1.6551×10^{17}	9.2972×10^{21}
	Estimated	769.5	2.0972×10^7	1.5032×10^{10}	3.2839×10^{13}	2.3058×10^{17}
	After Symmetry Reduction	802	21120194	1.5082×10^{10}	3.2885×10^{13}	2.3071×10^{17}

Example 4. *Consider the* $(4, 3, 3)$ *regenerating code problem, since there are* $n = 4$ *nodes, the random variables are* $\mathcal{X} = \{X_{11}, X_{12}, \ldots, X_{44}\}$*. Estimates for the number of LP variables can be found as* $(2^{16} - 1)/|S_4| = 2730.63$*, and that of LP constraints as* $(16 + \binom{16}{2}2^{14})/|S_4| = 81920.70$ *after symmetry reduction (neither of the estimates is an integer).*

The group $G_{\mathcal{I}}$ *in the problem is a symmetric group* S_4*, thus having a cycle index of*

$$P_{G_{\mathcal{I}}} = \frac{1}{24}(x_1^4 + 3x_2^2 + 6x_4 + 6x_1^2x_2 + 8x_1x_3).$$

Theorem 7 gives the cycle index of the induced permutation group $G_{\mathcal{X}}$ *as*

$$P_{G_{\mathcal{X}}} = \frac{1}{24}(x_1^{16} + 3x_2^8 + 6x_4^4 + 6x_1^4x_2^6 + 8x_1x_3^5).$$

Using Step 3 of Meta Algorithm 1, the number of LP variables can be calculated as 3043*, which is about* 4.6433% *of the original number of LP variables* $2^{16} - 1 = 65535$*. The number of LP constraints of type (4.16) can be calculated using Step 4 of Meta Algorithm 1, which gives the summation of coefficients before terms including* r^2 *as* 83200*. Therefore, the total number of LP constraints is* 83202*, which reduces to about* 4.2317% *of the original number of LP constraints* $16 + \binom{16}{2}2^{14} = 1966096$*.*

The results for problems with several other parameters can be found in Table 4.2. For the $(4, 3, 3)$ case, it was found through a computer program [110] that the number of unique LP variables, when taking the problem implication relations also into account, is in fact 176, which is a reduction of approximately 372-fold from 65535. Using the calculation given in the example above, it is seen that approximately 22-fold of this reduction is due to symmetry, while the remaining is due to the problem-specific implication relations in this case.

4.5 The Caching Problem

Caching is a technique used in communication systems to reduce peak-hour transmission by prefetching part of a file in the user's local cache during off-peak hours. In the caching problem

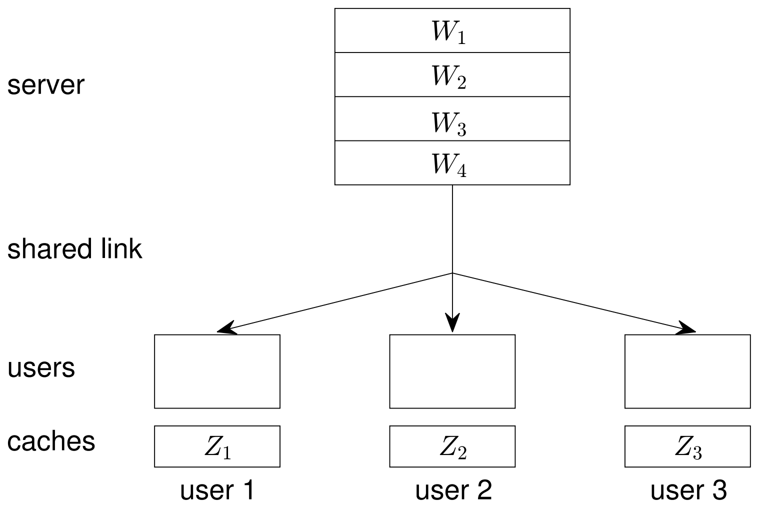

Figure 4.2: An $(N, K) = (4, 3)$ cache network. Reprinted with permission from [4, 5], © 2017,2018 IEEE.

formulated in [39], there are N files, each file has a size of F bits. There are K users, each user has a cache memory of MF. In the placement phase, the users prefetch contents to fill up their local caches. In the delivery phase, each user will reveal the request to the server, and then the server should multicast information to fulfill all users' requests at transmission size RF. The task is to strategically fill up the caches in the placement phase, thus minimizing the amount of transmission in the delivery phase.

Let the set of files be $\mathcal{W} = \{W_1, W_2, \dots, W_N\}$, indexed by $\mathcal{I}_N$, and the set of the cached contents at the users be $\mathcal{Z} = \{Z_1, Z_2, \dots, Z_K\}$, indexed by $\mathcal{I}_K$. The transmitted information also form a set

$$\mathcal{X}_N^K \overset{\text{def}}{=} \{X_{a_1 a_2 \dots a_K} : a_k \in \mathcal{I}_N\}, \tag{4.33}$$

where $X_{a_1 a_2 \dots a_K}$ denotes the transmitted information when user k requests file a_k, $k = 1, 2, ..., K$. The random variables are thus the set $\mathcal{X} = \mathcal{W} \cup \mathcal{Z} \cup \mathcal{X}_N^K$, therefore $m = N + K + N^K$. The symmetry embedded in this problem can be described as follows: suppose an efficient coding scheme has been found for the system, it is clear that if the file indices are permuted, the indices in the coding scheme can be permuted in a similar manner which gives an equivalent code. Similarly, the user indices can also be permuted and an equivalent code can be found as well. The file index permutation and user index permutation can be combined together to operate on the set $\mathcal{X}_N^K$. It can be shown that without loss of generality only symmetric solution needs to be considered [91]. We omit the precise LP again here, and next focus on applying Meta Algorithm 1 on this problem.

The three-layer decomposition can thus be done as follows. In the first layer, there are two base sets $\mathcal{I}_N$ and $\mathcal{I}_K$ with two symmetric groups $G_{\mathcal{I}_N} = S_N$ and $G_{\mathcal{I}_K} = S_K$. Let us use $\hat{\pi}, \bar{\pi}$ to represent any permutation from $G_{\mathcal{I}_N}$ and $G_{\mathcal{I}_K}$, respectively, and additionally write $\bar{\pi}^{-1}$ as $\bar{\pi}'$. In the second layer, we first study the subset $\mathcal{X}_N^K$ of $\mathcal{X}$. Each pair of permutations $\hat{\pi}, \bar{\pi}'$ induces a permutation $\hat{\pi}^{\bar{\pi}'}$, defined as

$$\hat{\pi}^{\bar{\pi}'} : \mathcal{X}_N^K \to \mathcal{X}_N^K \quad \text{as} \quad \hat{\pi}^{\bar{\pi}'}(X_{a_1 a_2 \ldots a_K}) \overset{\text{def}}{=} X_{\hat{\pi}(a_{\bar{\pi}'(1)})\hat{\pi}(a_{\bar{\pi}'(2)})\ldots\hat{\pi}(a_{\bar{\pi}'(K)})}, \tag{4.34}$$

and all the $\hat{\pi}^{\bar{\pi}'}$ permutations form a permutation group

$$G_{\mathcal{X}_N^K} = \{\hat{\pi}^{\bar{\pi}'} : \hat{\pi}^{\bar{\pi}'} \text{ induced by } \hat{\pi} \in G_{\mathcal{I}_N} \text{ and } \bar{\pi}' \in G_{\mathcal{I}_K}\}. \tag{4.35}$$

Finally, the permutation acting on the set $\mathcal{X}$ can be defined as

$$\pi^{ind} : \mathcal{X} \to \mathcal{X} \quad \text{as} \quad \pi^{ind}(x) \overset{\text{def}}{=} \begin{cases} W_{\hat{\pi}(i)}, & \text{if } x = W_i \in \mathcal{W}, \\ Z_{\bar{\pi}(j)}, & \text{if } x = Z_j \in \mathcal{Z}, \\ \hat{\pi}^{\bar{\pi}'}(x), & \text{if } x \in \mathcal{X}_N^K, \end{cases} \tag{4.36}$$

and all the π^{ind} permutations form a permutation group

$$G_{\mathcal{X}} = \{\pi^{ind} : \pi^{ind} \text{ induced by } \hat{\pi} \in G_{\mathcal{I}_N} \text{ and } \bar{\pi}' \in G_{\mathcal{I}_K}\}. \tag{4.37}$$

In other words, each pair of $\hat{\pi}, \bar{\pi}'$ permute two disjoint subsets $\mathcal{W}, \mathcal{Z}$ of the set $\mathcal{X}$, and their induced permutation $\hat{\pi}^{\bar{\pi}'}$ permutes the set $\mathcal{X}_N^K$. Obviously, there are $|G_{\mathcal{X}}| = |G_{\mathcal{I}_N}| \cdot |G_{\mathcal{I}_K}| = N!K!$ permutations in $G_{\mathcal{X}}$. For example, in the $(N, K) = (4, 3)$ caching problem, $\mathcal{I}_N = \{1, 2, 3, 4\}, G_{\mathcal{I}_N} = S_4$

and $\mathcal{I}_K = \{1,2,3\}, G_{\mathcal{I}_K} = S_3$, then $\mathcal{X}_N^K = \{X_{a_1a_2a_3} : a_k \in \{1,2,3,4\}, k \in \{1,2,3\}\}$, and permutation $\hat{\pi} = (1)(234)$ and $\bar{\pi}' = (123)$ would induce a permutation π^{ind} mapping each $x \in \mathcal{X}$,

$$\pi^{ind}(W_2) = W_{\hat{\pi}(2)} = W_3,$$
$$\pi^{ind}(Z_1) = Z_{\bar{\pi}(1)} = Z_3, \text{ (since } \bar{\pi} = (132))$$
$$\pi^{ind}(X_{124}) = \hat{\pi}^{\bar{\pi}'}(X_{124}) = X_{321}.$$

To derive the cycle index of the permutation group $G_{\mathcal{X}}$, we first consider the group $G_{\mathcal{X}_N^K}$. Recall the definition of a power group, and specifically let $\mathcal{Y} = \mathcal{I}_K, G = S_K$ and $\mathcal{V} = \mathcal{I}_N, H = S_N$. It is clear that the mapping (4.3) induces exactly the same mapping as in (4.34) with $h = \hat{\pi}$ and $g = \bar{\pi}^{-1} = \bar{\pi}'$. Thus the permutation group $G_{\mathcal{X}_N^K}$ is isomorphic to the permutation representation of the power group $S_N^{S_K}$ acting on $\mathcal{I}_N^{\mathcal{I}_K}$. The cycle index of $G_{\mathcal{X}_N^K}$ can be directly given by (4.7), and the cycle index of $G_{\mathcal{X}}$ can now be given by the following theorem.

Theorem 8. *In the (N, K) caching problem, the group $G_{\mathcal{I}_N}$ acting on $\mathcal{I}_N$ and the group $G_{\mathcal{I}_K}$ acting on $\mathcal{I}_K$ are both symmetric groups, and $\hat{\pi}$ and $\bar{\pi}'$ are two permutations from them, respectively. The random variable set is $\mathcal{X} = \mathcal{W} \cup \mathcal{Z} \cup \mathcal{X}_N^K$. The cycle index of the group $G_{\mathcal{X}}$ is*

$$P_{G_{\mathcal{X}}} = \frac{1}{N!K!} \sum_{\pi^{ind} \in G_{\mathcal{X}}} \Big(\prod_{k=1}^{N^K} x_k^{j_k(\hat{\pi};\bar{\pi}')} \cdot \prod_{k=1}^{N} x_k^{j_k(\hat{\pi})} \cdot \prod_{k=1}^{K} x_k^{j_k(\bar{\pi}')} \Big). \tag{4.38}$$

Proof. Each permutation $\hat{\pi}^{\bar{\pi}'}$ permutes the subset $\mathcal{X}_N^K$, and its components $\hat{\pi}, \bar{\pi}$ permute two disjoint subsets that are also disjoint to $\mathcal{X}_N^K$. This implies that the overall cycle index of a permutation is the product of the cycle indices of the three disjoint components. Finally, the $P_{G_{\mathcal{X}}}$ is the summation over all permutations in $G_{\mathcal{X}}$, divided by the number of permutations in it. □

Directly evaluating the cycle index in Theorem 8 can be time consuming, since a total of $N!K!$ permutations exist. The following corollary can be used to simplify this computation.

Corollary 2. *If two pairs of permutations $(\hat{\pi}_1, \bar{\pi}_1')$ and $(\hat{\pi}_2, \bar{\pi}_2')$ have the same cycle indices, that* *is, $\hat{\pi}_1$ has the same cycle index as $\hat{\pi}_2$ and $\bar{\pi}'_1$ has the same cycle index as $\bar{\pi}'_2$, then the permutations π_1^{ind} and π_2^{ind} produced by them, respectively, will have the same cycle index.*

The corollary can be proved simply by observing that the cycle index of a permutation π^{ind} in the parenthesis of (4.38) is a function of only $j_k(\hat{\pi})$ and $j_k(\bar{\pi}')$ (since $j_k(\hat{\pi};\bar{\pi}')$ is a function of only them as well), and the function $j_k(\pi)$ is only determined by the cycle index of a permutation π, but not by what precise elements are in each cycle or how they are permuted. This corollary implies that $P_{G_\mathcal{X}}$ needs only be computed from each type of cycle indices of $(\hat{\pi}, \bar{\pi}')$, thus the computational burden is significantly reduced from computing all $N!K!$ permutations to computing only $p(N)p(K)$ of them.

Example 5. *Consider the case of $N = 4$ files $\mathcal{W} = \{W_1, W_2, W_3, W_4\}$ and $K = 3$ users $\mathcal{Z} = \{Z_1, Z_2, Z_3\}$; see Figure 4.2. The estimates are given as $(2^{4+3+64} - 1)/(|S_3| \cdot |S_4|) = 16397105843297370000 \approx 1.6397 \times 10^{19}$ LP variables and $(71 + \binom{71}{2} \times 2^{69})/(|S_3| \cdot |S_4|) = 1018670200514849000000 0\approx 1.0187 \times 10^{22}$ LP constraints after symmetry reduction.*

For Step 1 of Meta Algorithm 1, we have

$$\begin{aligned} P_{G_{\mathcal{I}_N}} &= \frac{1}{24}(x_1^4 + 3x_2^2 + 6x_4 + 6x_1^2x_2 + 8x_1x_3), \\ P_{G_{\mathcal{I}_K}} &= \frac{1}{6}(x_1^3 + 2x_3 + 3x_1x_2). \end{aligned}$$

Using (4.38), Step 2 of Meta Algorithm 1 gives

$$\begin{aligned} P_{G_\mathcal{X}} = x_1^{71} &+ 3x_1^3x_2^{34} + 6x_1^3x_4^{17} + 6x_1^{13}x_2^{29} + 8x_1^5x_3^{22} + 2x_1^8x_3^{21} + 6x_2^4x_3x_6^{10} + 12x_3x_4^2x_{12}^5 \\ &+ 12x_1^4x_2^2x_3^3x_6^9 + 16x_1^5x_3^{22} + 3x_1^{21}x_2^{25} + 9x_1x_2^{35} + 18x_1x_2x_4^{17} + 18x_1^{11}x_2^{30} + 24x_1^3x_2x_3^6x_6^8. \end{aligned}$$

Step 3 of Meta Algorithm 1 gives the number of LP variables as $16397107774631008960 - 1 \approx 1.6397 \times 10^{19}$. The total number of LP variables reduces to about 0.6944% of the original size $2^{4+3+4^3} - 1 \approx 2.3612 \times 10^{21}$. The number of type (4.15) constraints is easily seen as 5. Step 4 of Meta Algorithm 1 gives the number of type (4.16) constraints as $10186702114698904297472 \approx$

1.0187×10^{22}*, the result of summation is* 0.6944% *of the original scale* $71 + \binom{71}{2} 2^{69} \approx 1.4669 \times 10^{24}$.

The results for several other parameters are given in Table 4.3.

Table 4.3: The numbers of LP variables and LP constraints in the (N, K) caching problem. Reprinted with permission from [4, 5], © 2017,2018 IEEE.

	(N, K)	$(2, 3)$	$(2, 4)$	$(2, 5)$	$(2, 6)$	$(3, 3)$
Number of LP Variables	Original	8191	4194304	5.4976×10^{11}	4.7224×10^{21}	8.5899×10^{9}
	Estimated	682.5833	87381.3333	2.2906×10^{9}	3.2794×10^{18}	2.3861×10^{8}
	After Symmetry Reduction	1016	106783	2337846175	3.2798×10^{18}	2.3949×10^{8}
Number of LP Constraints	Original	159757	242221078	1.0184×10^{14}	3.0176×10^{24}	1.1339×10^{12}
	Estimated	13313.0833	5046272.4583	4.2434×10^{11}	2.0956×10^{21}	3.1496×10^{10}
	After Symmetry Reduction (form (4.15))	4	5	5	6	5
	After Symmetry Reduction (form (4.16))	15240	5376072	426941056160	2.0956×10^{21}	3.1515×10^{10}

4.6 Conclusion

We studied the symmetry structure in several problems and proposed a generic decomposition of the group structure when applying the Pólya counting theorem, in order to count the exact amount of reduction of variables and constraints in the LP problems through symmetry. In practice, we would also like to directly generate isomorphic-free LP variables and constraints. The works [139–141] indeed provide important insights in this direction, however, the task itself still appears difficult. It should be emphasized again that there are indeed other possible reductions which are not necessarily related to symmetry, that can be used to significantly reduce the scale of the LPs, and thus there is considerable potential to further reduce the computational scale in the resultant LP. This, however, is beyond the scope of the current work, but may be of interest in its own right.

5. CODED PREFETCHING AND EFFICIENT DELIVERY IN DECENTRALIZED CACHING SYSTEMS*

In content delivery systems, data traffic can be bursty and usually peaks at a particular time period of a day. This variation in the traffic can induce significant stress on the content server and the delivery network during this peak time. Caching techniques have been introduced to relieve the stress on such networks, by means of letting the users prefetch certain file contents to store locally during off-peak traffic hours. Recently, Maddah-Ali and Niesen [39] introduced an information theoretic model to study the fundamental limit of caching in such settings.

In the model proposed in [39], there are a server and multiple users who are connected to the server using a shared link, which may be a wireless broadcast channel. The overall system operates in two phases, namely a prefetching (placement) phase when the server places file contents into user's cache, and a delivery phase when the server multicast certain content to fulfill each user's request. Users' requests are not known during the placement phase, and thus the difficulty is to design a caching strategy to enable, in the delivery phase, the least amount of data transmission for all possible combination of demands. Existing caching strategies in this centralized setting can be categorized according to whether the prefetching is uncoded or coded. The first class includes those in [39, 96, 100], while the latter class includes those in [77, 94, 95, 99, 101, 142]. In particular, the strategy in [77] is unique in the sense that codes in more general finite fields are used, which appears to provide further improvement over codes using only binary operations [39, 94–96, 99–101, 142].

In practical systems where a central coordinating mechanism can be costly, e.g., in more dynamic and mobile environments, decentralized coding becomes necessary. In this setting, rather than letting the server centrally control the placement of cached contents, each user independently determine the prefetching contents (see [87, 95, 100]). Decentralized coded caching has been stud-

*© 2017 IEEE. Reprinted, with permission, from Kai Zhang, Chao Tian and Husheng Li, "Coded prefetching and efficient delivery in decentralized caching systems", *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1-5, 2017.

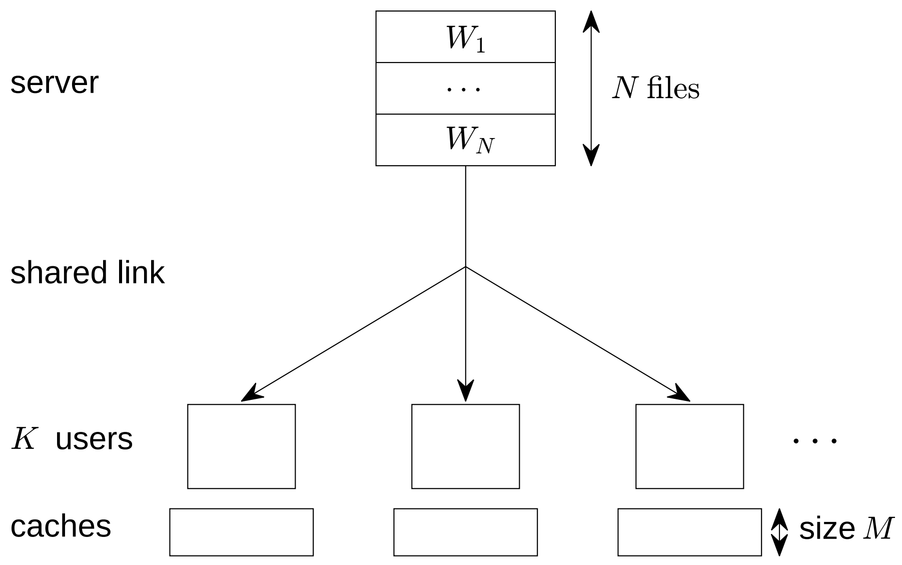

Figure 5.1: A cache network with N files and K users with a cache size M. Reprinted with permission from [6], © 2017 IEEE.

ied under other settings, non-uniform demand was studied in [88], random demand in [112], online caching in [89], distinct cache capacities in [113] and various other delivery schemes in [114, 115].

Given the current state of the art, a natural question to ask is whether it is possible to extend the coded prefetching strategy in [77] from the centralized setting to a decentralized setting. Moreover, since the codes in [77] is not binary, it is anticipated that the coding overhead will increase, and thus it is important to understand how to reduce its impact. In this work, we propose a decentralized coded placement with prefetching in a binary extension field $\mathbb{F}_{2^m}$, and an efficient delivery scheme in the base binary field. We show that this decentralized caching scheme can achieve improvement over the decentralized strategy in [87, 100]. Moreover, we provide methods to reduce or balance the impacts of coding overheads.

5.1 Preliminary and Motivation

In this section we will give the system model of the caching system and introduce the caching problem, and briefly review several caching strategies and their performance.

5.1.1 System Model

A centralized caching system contains a server with N files, which are all of the same size of F bits, and K users, each has a local cache memory M (normalized by the file size F); see Figure 5.1. We denote the files as $W_1, W_2, \ldots, W_N$. The system operates in two phases, a placement phase when the central server coordinates the prefetching of contents at all users. During the delivery phase, each user requests one file, i.e., user k requests file W_{d_k}. The server multicasts certain information at a transmission rate R (also normalized by the file size) to the users, together with the prefetched content at each user, to satisfy all the user demands.

When the central server does not know accurately the number of active users K and their identities during the prefetching phase, centralized prefetching strategies are not applicable and decentralized prefetching strategies are required. For both centralized and decentralized settings, we wish to find efficient coding strategies to minimize the transmission rate R for a given memory constraint M. Existing caching strategies can be categorized into strategies with uncoded prefetching and those with coded prefetching.

5.1.2 Uncoded Prefetching: Centralized vs. Decentralized

The strategy proposed in [39] used uncoded prefetching, which was further improved in [100] by removing the redundant transmissions in the delivery phase. This improvement results in the memory-transmission-rate pair

$$(M_C(r), R_C(r)) = \left(\frac{rN}{k}, \frac{\binom{K}{r+1} - \binom{K-\min\{K,N\}}{r+1}}{\binom{K}{r}} \right), \quad r = 0, \ldots, K.$$

It should be emphasized that the delivery strategy in [39] and [100] requires only operations in the binary field.

Following their initial work in [39] for the centralized setting, Maddah-Ali and Niesen proposed a decentralized coded caching strategy in [87]. In this strategy, each user, independently from each other, determines whether each bit in each file is cached in the local memory with a given probability. In the delivery phase, bits cached by the same number of users are grouped together, then a similar delivery strategy as the centralized caching scheme in [39] can be applied (with appropriate zero-padding). Similarly as in the centralized version, the performance can be further improved by reducing the transmission [100], which gives the pair

$$(M_D, R_D) = \left(M, \frac{N-M}{M}\Big(1 - \Big(\frac{N-M}{N}\Big)^{\min\{N,K\}}\Big)\right),$$

$M \in [0, N]$. It should be noted that for the given strategy to achieve this performance, the file size F needs to be sufficiently large, since otherwise the coding overhead becomes rather significant. We will revisit the issue of coding overheads in the later part of this chapter.

5.1.3 Coded Prefetching: Centralized vs. Decentralized?

Tian and Chen proposed a centralized coding strategy in [77], which uses coded prefetching and coded delivery in a more general finite field $\mathbb{F}_q$. This strategy achieves the following (M, R) pair

$$(M_C(t), R_C(t)) = \left(\frac{t[(N-1)t+K-N]}{K(K-1)}, \frac{N(K-t)}{K}\right),$$
$$t = 0, \dots, K.$$

The strategy is somewhat involved, but we provide an example to facilitate later discussions. Consider an $(N, K) = (2, 4)$ case, and set $t = 2$. Each file is partitioned into $\binom{4}{2}$ symbols in $\mathbb{F}_{2^4}$, for example, file W_1 is partitioned into $W_{1,12}, W_{1,13}, W_{1,14}, W_{1,23}, W_{1,24}, W_{1,34}$, where symbol $W_{1,12}$ is present in the cache of user 1 and user 2, etc.. The prefetching is coded, for example, user 1 can prefetch

$$W_{1,12} + W_{2,12}, W_{1,13} + W_{2,13}, W_{1,14} + W_{2,14}$$
$$W_{1,12} + W_{1,13} + W_{1,14} + \alpha(W_{2,12} + W_{2,13} + W_{2,14}),$$

for some $\alpha \in \mathbb{F}_{2^4}$ to be specified. The other users can prefetch in a similar manner. Consider the demand (W_1, W_1, W_1, W_2), then the following symbols are transmitted:

$$W_{2,12}, W_{2,13}, W_{2,23}.W_{1,12} + W_{1,13} + W_{1,23}$$
$$W_{1,14} + \alpha W_{1,24} + \beta W_{1,34}, W_{1,14} + \beta W_{1,24} + \gamma W_{1,34},$$

for some $\beta, \gamma \in \mathbb{F}_{2^4}$ to be specified. It can be verified that once receiving $W_{2,12}, W_{2,13}$, user 1 can resolve all the symbols present in the local cache (if α is chosen appropriately). Then with the other transmissions, the missing symbols $W_{1,24}, W_{1,34}, W_{1,23}$ can also be successfully recovered by user 1 (if β, γ are chosen appropriately).

A natural question is whether this strategy can be extended to the decentralized setting, and whether improvement over existing decentralized caching strategies can be obtained. Before giving such a decentralized prefetching strategy, we discuss a variation of the Tian-Chen scheme where the delivery uses only binary operations.

5.1.4 A Variation of Tian-Chen Strategy

In a follow-up work [1], a variation of the delivery strategy of that given in [77] was discussed, which achieves the same performance as the original strategy. This variation uses only binary operation in the delivery, and this feature leads to the particular advantage of reduced the coding overhead in the decentralized setting. We next provide an example to illustrate this variation, but omit the details.

Let us consider the previous example of $N = 2$ files and $K = 4$ users. We observe that in the second step the transmissions can instead be,

$$W_{1,14} + W_{1,24}, W_{1,14} + W_{1,34}.$$

The addition is performed in $\mathbb{F}_{2^4}$, which clearly can be viewed as a binary field addition on the individual bits.

5.2 A Decentralized Coded Caching Scheme with Coded Prefetching

The decentralized coded caching strategy we propose includes a random coded prefetching phase and a coded delivery phase. From here on, we use $\mathcal{N}$ to denote the file index set and $\mathcal{K}$ to denote the user index set.

5.2.1 Prefetching Strategy

In the prefetching phase, each file is partitioned into F file symbols, each symbol is of size m bits, which is in the finite field $\mathbb{F}_{2^m}$. Each user decides independently at random for each file symbol that whether it will be part of his cached content, with a probability $p \in [0, 1]$. Each user maintains $M_D F$ linear combinations of file symbols to fill his cache. Thus the random prefetching can be accomplished in an on-the-fly fashion: if a user k decides to use a specific file symbol, he will randomly pick $M_D F$ coefficients from $\mathbb{F}_{2^m}$, multiply with this symbol, respectively, then add them to the existing $M_D F$ linear combinations in his cache.

At the delivery phase, the server will have the knowledge regarding whether a user has a file symbol as a component in the linear combinations, but not necessarily the precise linear combination compositions. For simplicity of further discussions, we will group the file symbols as follows. We use $\mathcal{W}_{i,\mathcal{S}}, i \in \mathcal{N}, \mathcal{S} \subseteq \mathcal{K}$ to denote the collection of symbols from file W_i which are present (as components of linear combinations) at the caches of the users in the set $\mathcal{S}$, and $W_{i,\mathcal{S}}^{(j)}$ is the jth symbol in this set, $j = 1, \ldots, c_{i,\mathcal{S}}$, where $c_{i,\mathcal{S}} = |\mathcal{W}_{i,\mathcal{S}}|$ and $|\cdot|$ is the cardinality of a set.

We use $\mathcal{K}_t$ to denote the collection of all subsets of $\mathcal{K}$ which has cardinality t. Thus at the end of the prefetching phase, each file W_i is partitioned into sets of symbols $\cup_{t=0,1,\ldots,K}\mathcal{W}_{i,\mathcal{K}_t}$, where

$$\mathcal{W}_{i,\mathcal{K}_t} = \cup_{\mathcal{S}\in\mathcal{K}_t}\mathcal{W}_{i,\mathcal{S}}$$

is the collection of all symbols of W_i that are cached by any t number of users. Further denote

$$\mathcal{W}_{\mathcal{K}_t} = \cup_{i\in\mathcal{N}}\mathcal{W}_{i,\mathcal{K}_t}$$

as the t-*homogeneous group* of file symbols, which contains all the symbols from all files that are cached by t users.

Inside a t-homogeneous group, due to random prefetching, the cardinality of $\mathcal{W}_{i,\mathcal{S}}$ may be different for different i and $\mathcal{S}$, i.e., $c_{i,\mathcal{S}} \neq c_{i',\mathcal{S}'}$ when $\{i, \mathcal{S}\} \neq \{i', \mathcal{S}'\}$, we define

$$c_t^{max} \triangleq \max_{i \in \mathcal{N}, \mathcal{S} \in \mathcal{K}_t} c_{i,\mathcal{S}}.$$

In other words, c_t^{max} is the maximum of j in a t-homogeneous group such that there is at least one symbol $W_{i,\mathcal{S}}^{(j)}, i \in \mathcal{N}, \mathcal{S} \in \mathcal{K}_t$ exists. For a fix j, all the jth symbols from the t-homogeneous group can be grouped together as a set

$$\mathcal{W}_{\mathcal{K}_t}^{(j)} = \{W_{i,\mathcal{S}}^{(j)}, i \in \mathcal{N}, \mathcal{S} \in \mathcal{K}_t\}, j = 1, \ldots, c_t^{max}.$$

Hence, the t-homogeneous group can be partitioned into different sets according to j,

$$\mathcal{W}_{\mathcal{K}_t} = \cup_{j=1}^{c_t^{max}} \mathcal{W}_{\mathcal{K}_t}^{(j)}.$$

We further define $\bar{c}_t$ to be the smallest integer that is not less than the expectation of all $c_{i,\mathcal{S}}$ inside a t-homogeneous group,

$$\bar{c}_t = \lceil \mathbb{E}(c_{i,\mathcal{S}}) \rceil, i \in \mathcal{N}, \mathcal{S} \in \mathcal{K}_t.$$

We introduce two other notations to represent the linear combinations in each user's cache. First, we use $\mathcal{W}_{k \in \mathcal{K}_t}$ to denote the set of those symbols that the subscript includes index k, i.e.,

$$\mathcal{W}_{k \in \mathcal{K}_t} = \{W_{i,\mathcal{S}} : i \in \mathcal{N}, k \in \mathcal{S} \in \mathcal{K}_t\},$$

and similar notation applies to the other sets defined above. Second, bold symbol $\mathbf{W}$ is used to represent the column vector form of each set of symbols we defined, for example, $\mathbf{W}_{k \in \mathcal{K}_t}$ denotes a column vector containing all symbols in $\mathcal{W}_{k \in \mathcal{K}_t}$.

Table 5.1: A random instance of the file symbols after prefetching. Reprinted with permission from [6], © 2017 IEEE.

t	W_1	W_2	$\bar{c}_t$	c_t^{max}
0		$W_{2,\emptyset}^{(1)}$, $W_{2,\emptyset}^{(2)}$	1	2
1	$W_{1,1}^{(1)}$, $W_{1,1}^{(2)}$ $W_{1,2}^{(1)}$, $W_{1,2}^{(2)}$ $W_{1,3}^{(1)}$	$W_{2,1}^{(1)}$, $W_{2,1}^{(2)}$, $W_{2,1}^{(3)}$ $W_{2,2}^{(1)}$ $W_{2,3}^{(1)}$	2	3
2	$W_{1,12}^{(1)}$, $W_{1,12}^{(2)}$ $W_{1,13}^{(1)}$ $W_{1,23}^{(1)}$	$W_{2,12}^{(1)}$ $W_{2,13}^{(1)}$ $W_{2,23}^{(1)}$	2	2
3	$W_{1,123}^{(1)}$, $W_{1,123}^{(2)}$	$W_{2,123}^{(1)}$	2	2

Using the above notations, the $M_D F$ linear combinations in user k's cache can be represented as

$$\begin{bmatrix} \boldsymbol{\alpha}^1_{k\in\mathcal{K}_1} & \cdots & \boldsymbol{\alpha}^1_{k\in\mathcal{K}_t} & \cdots & \boldsymbol{\alpha}^1_{k\in\mathcal{K}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{\alpha}^l_{k\in\mathcal{K}_1} & \cdots & \boldsymbol{\alpha}^l_{k\in\mathcal{K}_t} & \cdots & \boldsymbol{\alpha}^l_{k\in\mathcal{K}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{\alpha}^{M_D F}_{k\in\mathcal{K}_1} & \cdots & \boldsymbol{\alpha}^{M_D F}_{k\in\mathcal{K}_t} & \cdots & \boldsymbol{\alpha}^{M_D F}_{k\in\mathcal{K}} \end{bmatrix} \begin{bmatrix} \mathbf{W}_{k\in\mathcal{K}_1} \\ \vdots \\ \mathbf{W}_{k\in\mathcal{K}_t} \\ \vdots \\ \mathbf{W}_{k\in\mathcal{K}} \end{bmatrix},$$

where each $\boldsymbol{\alpha}^l_{k\in\mathcal{K}_t}$ is a row vector of the coefficients before the vector $\mathbf{W}_{k\in\mathcal{K}_t}$.

Example 6. *Let us consider an example of the $(N, K) = (2, 3)$ caching system. Set $p = \frac{1}{3}$, suppose each file contains $F = 11$ file symbols in the finite field $\mathbb{F}_{2^3}$, thus each symbol is a 3-bit vector. Depending on which users cache a file symbol after the random prefetching phase, all file symbols are indexed as Table 5.1.*

Suppose the linear combinations in user 1's cache are

$$\begin{bmatrix} 3 & 2 & 5 & 5 & 1 & 1 & 2 & 1 & 6 & 0 & 4 & 2 & 7 \\ 3 & 5 & 6 & 1 & 5 & 4 & 7 & 7 & 3 & 1 & 7 & 4 & 7 \\ 4 & 4 & 5 & 0 & 5 & 5 & 2 & 3 & 7 & 0 & 1 & 6 & 0 \\ 2 & 5 & 4 & 7 & 0 & 2 & 7 & 4 & 4 & 6 & 0 & 2 & 2 \\ 3 & 6 & 0 & 1 & 1 & 0 & 3 & 2 & 6 & 4 & 3 & 0 & 5 \\ 0 & 6 & 1 & 7 & 0 & 4 & 0 & 7 & 2 & 3 & 4 & 3 & 6 \\ 1 & 0 & 2 & 4 & 6 & 1 & 2 & 0 & 3 & 0 & 7 & 0 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{W}^{(1)}_{1\in\mathcal{K}_1} \\ \mathbf{W}^{(2)}_{1\in\mathcal{K}_1} \\ \mathbf{W}^{(3)}_{1\in\mathcal{K}_1} \\ \mathbf{W}^{(1)}_{1\in\mathcal{K}_2} \\ \mathbf{W}^{(2)}_{1\in\mathcal{K}_2} \\ \mathbf{W}^{(1)}_{1\in\mathcal{K}_3} \\ \mathbf{W}^{(2)}_{1\in\mathcal{K}_3} \end{bmatrix},$$

where

$$[\mathbf{W}^{(1)^T}_{1\in\mathcal{K}_1}, \mathbf{W}^{(2)^T}_{1\in\mathcal{K}_1}, \mathbf{W}^{(3)^T}_{1\in\mathcal{K}_1}]^T = [W^{(1)}_{1,1}, W^{(1)}_{2,1}, W^{(2)}_{1,1}, W^{(2)}_{2,1}, W^{(3)}_{2,1}]^T,$$
$$[\mathbf{W}^{(1)^T}_{1\in\mathcal{K}_2}, \mathbf{W}^{(2)^T}_{1\in\mathcal{K}_2}]^T = [W^{(1)}_{1,12}, W^{(1)}_{1,13}, W^{(1)}_{2,12}, W^{(1)}_{2,13}, W^{(2)}_{1,12}]^T,$$
$$[\mathbf{W}^{(1)^T}_{1\in\mathcal{K}_3}, \mathbf{W}^{(2)^T}_{1\in\mathcal{K}_3}]^T = [W^{(1)}_{1,123}, W^{(1)}_{2,123}, W^{(2)}_{1,123}]^T,$$

and all coefficients are chosen uniformly at random from $\mathbb{F}_{2^3}$ generated by the primitive polynomial x^3+x+1. Each coefficient is a 3-bit unsigned binary integer which is a vector representation of the finite field element, for example, '7' corresponds to the vector representation '$[1\ 1\ 1]$'.

5.2.2 Delivery Strategy

Denote the group of users requesting file n as $I^{[n]}$, and fix an arbitrary user $\ell^{[n]}$ in it as the leader. Due to space constraint, here we only present in Algorithm 1 the delivery strategy when all files are being requested, i.e., $|I^{[n]}| \geq 1$ for $n = 1, 2, \dots, N$. For other type of demands, the delivery strategy can be similarly adapted from the centralized strategy for such cases given in [1].

In the delivery strategy, we need to consider all t-homogeneous groups for $t = 0, \dots, K$. For each group, we compute the value of $\bar{c}_t$ and deal with the set $\cup_{j=1}^{\bar{c}_t} \mathcal{W}^{(j)}_{\mathcal{K}_t}$ in phase I and $\cup_{j=\bar{c}_t+1}^{c_t^{max}} \mathcal{W}^{(j)}_{\mathcal{K}_t}$ in phase II, separately. For each set $\mathcal{W}^{(j)}_{\mathcal{K}_t}$ in phase I, its cardinality should be $N\binom{K}{t}$. Due to random prefetching, there might be some symbols $W^{(j)}_{i,\mathcal{S}}$ in this set that do not actually exist (see Table 5.2 for an example). In this case, the server will simply assign these symbols with all zeros.

Algorithm 2: Coded delivery algorithm. Reprinted with permission from [6], © 2017 IEEE.

```
Delivery scheme (phase I):
Transmit all symbols in the set $\cup_{j=1}^{\bar{c}_t}\mathcal{W}_{\emptyset}^{(j)}$ uncoded;
for $t = \{1, \dots, K-1\}$ do
    for $j = \{1, \dots, \bar{c}_t\}$ do
        for $\mathcal{A} \subseteq \cup_{i' \neq i} I^{[i']}$ where $t \geq |\mathcal{A}| > 0$ do
            Send the following summations in $\mathbb{F}_{2^m}$:
                $$\bigoplus_{k \in \mathcal{B}} W_{i,\mathcal{A}\cup\mathcal{B}\setminus\{k\}}^{(j)},$$
                $$|\mathcal{B}| = t+1-|\mathcal{A}|, \ell^{[i]} \in \mathcal{B} \subseteq I^{[i]}.$$
        end
    end
end
Delivery scheme (phase II):
for $t = \{0, \dots, K\}$ do
    Transmit all symbols in the set $\cup_{j=\bar{c}_t+1}^{c_t^{max}}\mathcal{W}_{\mathcal{K}_t}^{(j)}$ uncoded.
end
```

Example 7. *(continuing example 1.) Suppose the demands of all three users are* $\mathbf{d} = (W_1, W_1, W_2)$. *The value of each* $\bar{c}_t$ *is calculated in Table 5.1. For example, in the case* $t = 2$, $\mathcal{W}_{\mathcal{K}_t}^{(1)}$ *and* $\mathcal{W}_{\mathcal{K}_t}^{(2)}$ *are both delivered in phase I, thus the transmissions should be exactly the same except the superscript* j. *However, the symbols* $W_{1,13}^{(2)}$ *and* $W_{1,23}^{(2)}$ *do not exist in the latter, thus they are all-zero, which are omitted in the transmission.*

Consider the decoding steps of user 1, first he can collect the following 6 symbols from all the above transmissions

$$W_{2,1}^{(1)}, W_{2,1}^{(2)}, W_{2,1}^{(3)}, W_{2,12}^{(1)}, W_{1,12}^{(2)}, W_{1,123}^{(2)},$$

together with the 7 linear combinations in his cache, he can resolve all 13 present file symbols in his cache (assuming the 13 linear combinations are all linear independent). Second, he uses the

Table 5.2: The transmissions in each phase of the delivery. Reprinted with permission from [6], © 2017 IEEE.

t	Phase I	Phase II
0	$(j=1): W_{2,\emptyset}^{(1)}$	$(j=2): W_{2,\emptyset}^{(2)}$
1	$(j=1): W_{1,3}^{(1)}, W_{2,1}^{(1)}, W_{2,2}^{(1)}, W_{1,1}^{(1)}+W_{1,2}^{(1)}$ $(j=2): W_{2,1}^{(2)}, W_{1,1}^{(2)}+W_{1,2}^{(2)}$	$(j=3): W_{2,1}^{(3)}$
2	$(j=1): W_{1,13}^{(1)}+W_{1,23}^{(1)}, W_{2,12}^{(1)}$ $(j=2): W_{1,12}^{(2)}$	
3	$(j=1):$ –	$(j=2): W_{1,123}^{(2)}$

following transmissions

$$W_{1,3}^{(1)}, W_{1,1}^{(1)}+W_{1,2}^{(1)}, W_{1,1}^{(2)}+W_{1,2}^{(2)}, W_{1,13}^{(1)}+W_{1,23}^{(1)},$$

to he can decode $W_{1,3}^{(1)}, W_{1,2}^{(1)}, W_{1,2}^{(2)}, W_{1,23}^{(1)}$, till now he recovers all the 11 symbols from file W_1. The memory-rate pair in this example is $(M_D, R_D) = (\frac{7}{11}, \frac{13}{11})$.

It can be shown that user 1 can decode for all possible file demands, and similarly for user 2 through K. Although the coding matrix may have different number of columns and different coding coefficients at different users, the probability of successful decoding is guaranteed for all possible demands, and this probability indeed approaches one as the size of the finite field grows.

5.3 Performance Analysis

The performance is summarized below.

Theorem 9. *For N files and K users each with a cache of size M, where $N \leq K$, the following memory-rate tradeoff pair*

$$(M_D, R_D) = \Big((N-1)p^2 + p, N(1-p)\Big), p \in [0,1]$$

is achievable.

An illustration is given in Figure 5.2. The proof of this theorem has two main parts. Firstly we need to show that with the delivery strategy, each user with high probability can accumulate enough linear combinations, such that all the symbols present in the cache can be resolved. Intuitively, this is precisely the same problem as in [77], and the only difference is that there are more than one t-homogeneous group here, and they are randomly mixed together during prefetching. However, it can indeed be shown that this does not cause any essential difference, when the number of symbols in a file is sufficiently large such that the law of large numbers can be properly invoked. After the cached symbols are resolved, we essentially reduce it to an uncoded prefetching situation, and the other transmissions in the delivery phase can provide any missing file segments. Secondly, we have to also identify the performance of the given strategy, which though somewhat lengthy, is also relatively straightforward again thanks to the law of large numbers. It should be further noted that due to the random coding nature of the prefetching strategy, the alphabet $\mathbb{F}_{2^m}$ needs to be chosen sufficiently large to drive the probability of decoding failure to zero. We omitted the technical details here due to space constraint.

5.4 Coding Overheads

In the discussion until this point, we have largely ignored the coding overheads in the systems. A close examination reveals that there are overheads associated with both the prefetching and delivery, which may considerably increase the memory M and the transmission rate R, respectively. Next we discuss these two aspects, and focus on the difference from existing uncoded prefetching strategies.

5.4.1 Coding Overhead in Prefetching

First consider the existing uncoded prefetching strategies in the decentralized setting (e.g., [87]), where each user needs to store the identities of the symbols in the cache. This overhead can be rather overwhelming, if the bit-based strategy in [87] is applied directly. Since the identity of each bit in a file is associated with the length of the file, prefetching one information bit in fact requires multiple overhead bits, and thus the majority of the cache memory will be consumed

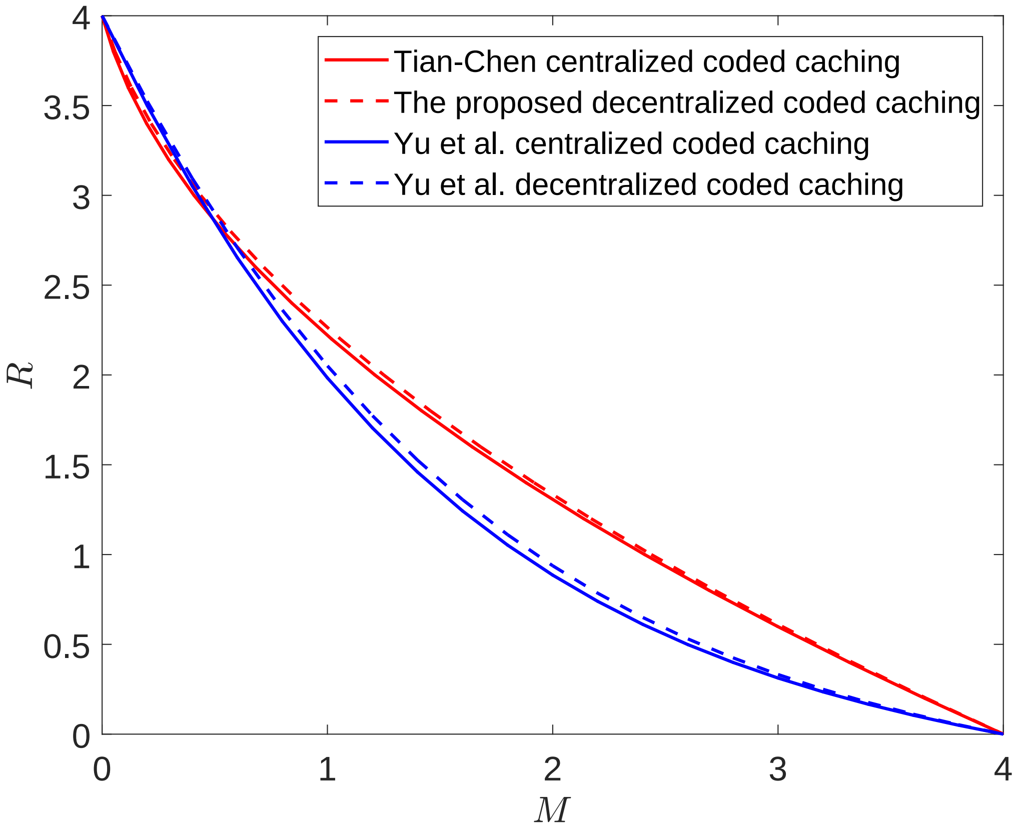

Figure 5.2: The memory-rate tradeoff $R(M)$ achieved by the proposed decentralized coded caching scheme for an $N = 4, K = 20$ caching system. Reprinted with permission from [6], © 2017 IEEE.

by the overhead. In order to mitigate this effect, a subfile division method can be adopted (see also [88]): before prefetching, first divide each file into n subfiles, each of sufficiently large size. Now we can randomly prefetch bits in the first subfiles of all the files as in [87], but for all the subsequent subfiles, use this fixed prefetching pattern for each file. Thus, the coding overhead is only associated with the first subfile, thus significantly less. There is however a tradeoff: larger n reduces the relative overhead, however smaller n makes each subfile large and thus less zero-padding is required in the delivery phase.

In the coded prefetching strategies we propose in this work, in addition to the identities of the information symbols present in the cache, the coefficients used to form this linear combination also need to be stored. This can be more significant since each stored symbol is in fact associated with multiple coefficients. Using the same subfile division approach, this impact can be mitigated, however, it is clear that the same tradeoff effect still persists on the choice of n. We leave a quantitative analysis of such effect to a future work, due to the space constraint. One may also wonder whether a pseudo-random generator can be used to eliminate the necessity of storing coding coefficients, by storing only the random seed. This approach is however futile, since in order to recover the information symbols, the coding coefficients are generally required during decoding.

5.4.2 Coding Overhead in Delivery

In the delivery strategy of [87] for the decentralized setting proposed by Maddah-Ali and Niesen, the server needs to include with each transmission the identity of the file bits that form this eventual transmitted bit. This overhead can again be rather overwhelming, but the subfile division strategy can also mitigate its impact. The delivery strategy we proposed in this work essentially has the same overhead properties as that in [87], and thus the same technique can be used. It should be noted that in the original centralized Tian-Chen strategy [77], the delivery is also not binary, and thus the coding coefficients indeed need to be transmitted in this stage. The alternative binary delivery strategy in [1] eliminates this burden altogether.

5.5 Conclusion

In this chapter, we proposed a coded prefetching coded delivery strategy for cache networks in the decentralized setting, which provides performance improvement upon existing strategies. Special attention is given to the coding overheads caused by the coded prefetching and the non-binary nature of the code, and methods to mitigate the impact are discussed.

6. CONCLUSION

The technique of caching plays an important role in the Internet, and this importance has been overlooked before. In recent years, as this importance has been rediscovered, a lot of researchers have performed tremendous studies from different perspective of the caching network. A thorough overview of the techniques can be found in this paper [143]. This dissertation, however, is mainly focusing on the study of innerbound and outerbound of the memory-rate tradeoff in the basic caching network setting from an information theoretic perspective, and a specific application of our proposed coding scheme on the decentralized caching network setting.

The dissertation proposed two coding schemes that both improves the previous innerbound of the memory-rate tradeoff. A symmetry reduction technique is also proposed to solve the problem of high computational complexity when using information inequalities to derive the outerbound. The proposed coding technique is applied onto the decentralized caching network which are more common to see in reality networks and has its own problems. The proposed caching strategy are proved to work with good performance under such setting.

In reality caching networks, there are some issues needs to be addressed. The fundamental limits of memory-rate tradeoff are mostly derived in a simple setting, for example, the file size are assumed to be the same, and the files need to keep fixed that cannot be changed, so the cached contents need to be updated in a smart way otherwise will render the coded caching strategy unusable. The users cannot be changed, such as user leaving the system or new user join in the system, which also breaks the coding scheme. In reality, this often happens in a cellular network where users are joining and leaving a certain network constantly since users are in a moving state. Some studies have been conducted to tackle these problems, but these issues are still far away from being completely solved. Some other interesting directions include hierarchical caching and online caching, where the former considers a network with multiple servers in different levels. The later tries to solve the problem when files in the server can be changed. Other topics such as device-to-device caching network is also considered, as it enables the communication between users and thus will further alleviate the burden on the bottleneck traffic.

In summary, caching is a useful technique and an interesting topic. This dissertation tries to push the study of it a little bit further. During the writing of this dissertation, advanced techniques such as machine learning are noticed that can be applied to the study of coded caching, for instance, to predict which file are more popular compared to others, thus could put more of that file to the user's cache in the prefetching phase in the online caching problem. Indeed, non-deterministic ways such as machine learning and statistical methods can play an important role in the coded caching problem, especially when tackling the real caching systems as neither the files nor the users are in a deterministic state, they are constantly changing during any time. Further studies of these and other techniques will hopefully help the study of this topic.

REFERENCES

[1] K. Zhang and C. Tian, "From uncoded prefetching to coded prefetching in coded caching systems," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 2087–2091, IEEE, 2018.

[2] K. Zhang and C. Tian, "Fundamental limits of coded caching: From uncoded prefetching to coded prefetching," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1153–1164, 2018.

[3] S. Shao, J. Gómez-Vilardebó, K. Zhang, and C. Tian, "On the fundamental limit of coded caching systems with a single demand type," in *2019 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2019.

[4] K. Zhang and C. Tian, "Symmetry reduction of information inequalities," *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1–5, 2016.

[5] K. Zhang and C. Tian, "On the symmetry reduction of information inequalities," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2396–2408, 2017.

[6] K. Zhang, C. Tian, and H. Li, "Coded prefetching and efficient delivery in decentralized caching systems," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, IEEE, 2017.

[7] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," *IBM Systems journal*, vol. 5, no. 2, pp. 78–101, 1966.

[8] W. King, "Analysis of demand paging algorithms," in *Proceedings of IFII'Congress (Information Processing 71)*, pp. 485–490, 1973.

[9] R. Fagin, "Asymptotic miss ratios over independent references," *Journal of Computer and System Sciences*, vol. 14, no. 2, pp. 222–250, 1977.

[10] J. Li, S. Shakkottai, J. C. Lui, and V. Subramanian, "Accurate learning or fast mixing? dynamic adaptability of caching algorithms," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1314–1330, 2018.

[11] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," *SIAM Journal on discrete mathematics*, vol. 17, no. 1, pp. 134–160, 2003.

[12] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1331–1344, 2018.

[13] Z. Zhao, L. Guardalben, M. Karimzadeh, J. Silva, T. Braun, and S. Sargento, "Mobility prediction-assisted over-the-top edge prefetching for hierarchical vanets," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1786–1801, 2018.

[14] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1300–1313, 2018.

[15] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Caching-aware recommendations: Nudging user preferences towards better caching performance," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, 2017.

[16] E. Leonardi and G. Neglia, "Implicit coordination of caches in small cell networks under unknown popularity profiles," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1276–1285, 2018.

[17] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3, pp. 1587–1596, IEEE, 2001.

[18] T. Kelly and D. Reeves, "Optimal web cache sizing: Scalable methods for exact solutions," *Computer Communications*, vol. 24, no. 2, pp. 163–173, 2001.

[19] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Placement algorithms for hierarchical cooperative caching," *Journal of Algorithms*, vol. 38, no. 1, pp. 260–302, 2001.

[20] J. Dai, F. Liu, B. Li, B. Li, and J. Liu, "Collaborative caching in wireless video streaming through resource auctions," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 2, pp. 458–466, 2012.

[21] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *2010 Proceedings IEEE INFOCOM*, pp. 1–9, IEEE, 2010.

[22] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz, "Selfish caching in distributed systems: a game-theoretic analysis," in *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pp. 21–30, 2004.

[23] L. Wang, G. Tyson, J. Kangasharju, and J. Crowcroft, "Milking the cache cow with fairness in mind," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2686–2700, 2017.

[24] M. Taghizadeh, K. Micinski, S. Biswas, C. Ofria, and E. Torng, "Distributed cooperative caching in social wireless networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1037–1053, 2012.

[25] K. Poularakis, G. Iosifidis, I. Pefkianakis, L. Tassiulas, and M. May, "Mobile data offloading through caching in residential 802.11 wireless networks," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 71–84, 2016.

[26] J. Krolikowski, A. Giovanidis, and M. Di Renzo, "A decomposition framework for optimal edge-cache leasing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1345–1359, 2018.

[27] E. Gourdin, P. Maillé, G. Simon, and B. Tuffin, "The economics of cdns and their impact on service fairness," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 22–33, 2017.

[28] R. T. Ma and D. Towsley, “Cashing in on caching: On-demand contract design with linear pricing,” in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pp. 1–6, 2015.

[29] K. Hosanagar, R. Krishnan, J. Chuang, and V. Choudhary, “Pricing and resource allocation in caching services with multiple levels of quality of service,” *Management Science*, vol. 51, no. 12, pp. 1844–1859, 2005.

[30] N. Economides and B. E. Hermalin, “The strategic use of download limits by a monopoly platform,” *The RAND Journal of Economics*, vol. 46, no. 2, pp. 297–327, 2015.

[31] J. Kwak, G. Paschos, and G. Iosifidis, “Dynamic cache rental and content caching in elastic wireless cdns,” in *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 1–8, IEEE, 2018.

[32] J. Tadrous, A. Eryilmaz, and H. El Gamal, “Joint smart pricing and proactive content caching for mobile services,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2357–2371, 2015.

[33] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, “Wireless caching: Technical misconceptions and business barriers,” *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, 2016.

[34] C. V. N. Index, *Global mobile data traffic forecast update, 2016–2021. [Online]. Available: https://www.ramonmillan.com/documentos/bibliografia/VisualNetworkingIndexGlobal MobileDataTrafficForecastUpdate2016_Cisco.pdf*, 2017.

[35] Ericsson, *Ericsson mobility report November 2019. [Online]. Available: https://www.ericsson.com/en/mobility-report/reports/november-2019*, 2019.

[36] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, “Comparison of caching strategies in modern cellular backhaul networks,” in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pp. 319–332, 2013.

[37] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[38] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, 2013.

[39] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[40] Cisco, *Cisco Annual Internet Report (2018–2023) White Paper. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html*, 2020.

[41] J. Morley, K. Widdicks, and M. Hazas, "Digitalisation, energy and data demand: The impact of internet traffic on overall and peak electricity consumption," *Energy Research & Social Science*, vol. 38, pp. 128–137, 2018.

[42] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 1078–1086, IEEE, 2014.

[43] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying flows and buffer state for youtube's http adaptive streaming service in mobile networks," in *Proceedings of the 9th ACM Multimedia Systems Conference*, pp. 138–149, 2018.

[44] Z.-L. Zhang, J. Kurose, J. D. Salehi, and D. Towsley, "Smoothing, statistical multiplexing, and call admission control for stored video," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, pp. 1148–1166, 1997.

[45] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-peer video-on-demand system: Design and evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1706–1716, 2007.

[46] F. Hartanto, J. Kangasharju, M. Reisslein, and K. Ross, "Caching video objects: layers vs versions?," *Multimedia Tools and Applications*, vol. 31, no. 2, pp. 221–245, 2006.

[47] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on circuits and systems for video technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[48] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.

[49] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale vod system," in *Proceedings of the 6th International COnference*, pp. 1–12, 2010.

[50] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Louédec, "Efficient http-based streaming using scalable video coding," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 329–342, 2012.

[51] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2275–2284, 2016.

[52] P. Ostovari, A. Khreishah, and J. Wu, "Multi-layer video streaming with helper nodes using network coding," in *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 524–532, IEEE, 2013.

[53] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1751–1767, 2018.

[54] M. Choi, J. Kim, and J. Moon, "Wireless video caching and dynamic streaming under differentiated quality requirements," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1245–1257, 2018.

[55] C. Ge, N. Wang, W. K. Chai, and H. Hellwagner, "Qoe-assured 4k http live streaming via transient segment holding at mobile edge," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1816–1830, 2018.

[56] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on replica server placement algorithms for content delivery networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1002–1026, 2016.

[57] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM transactions on networking*, vol. 8, no. 5, pp. 568–582, 2000.

[58] M. Bateni and M. Hajiaghayi, "Assignment problem in content distribution networks: unsplittable hard-capacitated facility location," *ACM Transactions on Algorithms (TALG)*, vol. 8, no. 3, pp. 1–19, 2012.

[59] E. Cronin, S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1369–1382, 2002.

[60] A. Benoit, V. Rehn-Sonigo, and Y. Robert, "Replica placement and access policies in tree networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 12, pp. 1614–1627, 2008.

[61] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 3, pp. 1282–1290, IEEE, 1999.

[62] T. Bektaş, J.-F. Cordeau, E. Erkut, and G. Laporte, "Exact algorithms for the joint object placement and request routing problem in content distribution networks," *Computers & Operations Research*, vol. 35, no. 12, pp. 3860–3884, 2008.

[63] G. Carofiglio, L. Mekinda, and L. Muscariello, "Joint forwarding and caching with latency awareness in information-centric networking," *Computer Networks*, vol. 110, pp. 133–153, 2016.

[64] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1635–1648, 2016.

[65] V. Pacifici, S. Jošilo, and G. Dán, "Distributed algorithms for content caching in mobile backhaul networks," in *2016 28th International Teletraffic Congress (ITC 28)*, vol. 1, pp. 313–321, IEEE, 2016.

[66] K. Poularakis and L. Tassiulas, "On the complexity of optimal content placement in hierarchical caching networks," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 2092–2103, 2016.

[67] Cisco, *Cisco Network Caching White Paper. [Online]. Available: https://www.cisco.com/c/dam/global/de_at/assets/docs/Net_Caching.pdf*, 2000.

[68] J. Leguay, G. S. Paschos, E. A. Quaglia, and B. Smyth, "Cryptocache: Network caching with confidentiality," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2017.

[69] Q. Huang, K. Birman, R. Van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of facebook photo caching," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pp. 167–181, 2013.

[70] Netflix, *How netflix works with isps around the globe to deliver a great viewing experience. [Online]. Available: https://media.netflix.com/en/company-blog/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience*, 2016.

[71] Netflix, *Netflix Open Connect Overview. [Online]. Available: https://openconnect.netflix.com/Open-Connect-Overview.pdf*, 2019.

[72] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[73] O. Katz, R. Perets, and G. Matzliach, "Digging deeper—an in-depth analysis of a fast flux network," *Akamai, Cambridge, MA, USA, White Paper*, 2017.

[74] A. Sengupta, R. Tandon, and O. Simeone, "Fog-aided wireless networks for content delivery: Fundamental latency tradeoffs," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6650–6678, 2017.

[75] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1176–1188, 2018.

[76] A. Malik, S. H. Lim, and W.-Y. Shin, "On the effects of subpacketization in content-centric mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1721–1736, 2018.

[77] C. Tian and J. Chen, "Caching and delivery via interference elimination," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1548–1560, 2018.

[78] J. Gómez-Vilardebó, "A novel centralized coded caching scheme with coded prefetching," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1165–1175, 2018.

[79] Y. Lu, W. Chen, and H. V. Poor, "Coded joint pushing and caching with asynchronous user requests," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1843–1856, 2018.

[80] S. M. Azimi, O. Simeone, A. Sengupta, and R. Tandon, “Online edge caching and wireless delivery in fog-aided networks with dynamic content popularity,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1189–1202, 2018.

[81] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, “On coding for cache-aided delivery of dynamic correlated content,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1666–1681, 2018.

[82] J. Hachem, N. Karamchandani, S. Moharir, and S. N. Diggavi, “Caching with partial adaptive matching,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1831–1842, 2018.

[83] M. M. Amiri and D. Gündüz, “Caching and coded delivery over gaussian broadcast channels for energy efficiency,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1706–1720, 2018.

[84] M. Mahdian, N. Prakash, M. Médard, and E. Yeh, “Updating content in cache-aided coded multicast,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1203–1216, 2018.

[85] Y.-P. Wei, K. Banawan, and S. Ulukus, “Cache-aided private information retrieval with partially known uncoded prefetching: Fundamental limits,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1126–1139, 2018.

[86] A. A. Zewail and A. Yener, “Combination networks with or without secrecy constraints: The impact of caching relays,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1140–1152, 2018.

[87] M. A. Maddah-Ali and U. Niesen, “Decentralized coded caching attains order-optimal memory-rate tradeoff,” *IEEE/ACM Transactions On Networking*, vol. 23, no. 4, pp. 1029–1040, 2014.

[88] U. Niesen and M. A. Maddah-Ali, “Coded caching with nonuniform demands,” *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146–1158, 2016.

[89] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 836–845, 2015.

[90] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3212–3229, 2016.

[91] C. Tian, "Symmetry, demand types and outer bounds in caching systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 825–829, IEEE, 2016.

[92] C. Tian, "Symmetry, outer bounds, and code constructions: A computer-aided investigation on the fundamental limits of caching," *Entropy*, vol. 20, no. 8, p. 603, 2018.

[93] Z. Chen, P. Fan, and K. B. Letaief, "Fundamental limits of caching: Improved bounds for users with small buffers," *IET Communications*, vol. 10, no. 17, pp. 2315–2318, 2016.

[94] S. Sahraei and M. Gastpar, "K users caching two files: An improved achievable rate," in *2016 Annual Conference on Information Science and Systems (CISS)*, pp. 620–624, IEEE, 2016.

[95] M. M. Amiri, Q. Yang, and D. Gündüz, "Coded caching for a large number of users," in *2016 IEEE Information Theory Workshop (ITW)*, pp. 171–175, IEEE, 2016.

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

[97] J. Gómez-Vilardebó, "Fundamental limits of caching: Improved rate-memory tradeoff with coded prefetching," *IEEE Transactions on Communications*, vol. 66, no. 10, pp. 4488–4497, 2018.

[98] K. K. Vijith, B. K. Rai, and T. Jacob, "Towards the exact rate memory tradeoff in coded caching," in *2019 National Conference on Communications (NCC)*, pp. 1–6, IEEE, 2019.

[99] M. M. Amiri and D. Gündüz, "Fundamental limits of coded caching: Improved delivery rate-cache capacity tradeoff," *IEEE Transactions on Communications*, vol. 65, no. 2, pp. 806–815, 2016.

[100] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “The exact rate-memory tradeoff for caching with uncoded prefetching,” *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, 2017.

[101] J. Gómez-Vilardebó, “Fundamental limits of caching: Improved bounds with coded prefetching,” *arXiv preprint arXiv:1612.09071*, 2016.

[102] H. Ghasemi and A. Ramamoorthy, “Improved lower bounds for coded caching,” *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4388–4413, 2017.

[103] C.-Y. Wang, S. S. Bidokhti, and M. Wigger, “Improved converses and gap results for coded caching,” *IEEE Transactions on Information Theory*, vol. 64, no. 11, pp. 7051–7062, 2018.

[104] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Characterizing the rate-memory tradeoff in cache networks within a factor of 2,” *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 647–663, 2018.

[105] R. W. Yeung, *Information theory and network coding*. Springer Science & Business Media, 2008.

[106] Z. Zhang and R. W. Yeung, “On characterization of entropy function via information inequalities,” *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1440–1452, 1998.

[107] R. W. Yeung and Z. Zhang, “On symmetrical multilevel diversity coding,” *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 609–621, 1999.

[108] C. Tian, “Latent capacity region: A case study on symmetric broadcast with common messages,” *IEEE transactions on information theory*, vol. 57, no. 6, pp. 3273–3285, 2011.

[109] J. Jiang, N. Marukala, and T. Liu, “Symmetrical multilevel diversity coding and subset entropy inequalities,” *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 84–103, 2013.

[110] C. Tian, “Characterizing the rate region of the (4, 3, 3) exact-repair regenerating codes,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 967–975, 2014.

[111] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear programming and network flows*. John Wiley & Sons, 2011.

[112] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3923–3949, 2017.

[113] M. M. Amiri, Q. Yang, and D. Gündüz, "Decentralized coded caching with distinct cache capacities," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, pp. 734–738, IEEE, 2016.

[114] A. Ramakrishnan, C. Westphal, and A. Markopoulou, "An efficient delivery scheme for coded caching," in *2015 27th International Teletraffic Congress*, pp. 46–54, IEEE, 2015.

[115] K. Wan, D. Tuninetti, and P. Piantanida, "Novel delivery schemes for decentralized coded caching in the finite file size regime," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1183–1188, IEEE, 2017.

[116] J. Chen, A. Salimi, T. Liu, and C. Tian, "Orbit-entropy cones and extremal pairwise orbit-entropy inequalities," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 2614–2618, IEEE, 2016.

[117] J. Chen, H. Ye, C. Tian, T. Liu, and Z. Xiao, "Cyclically symmetric entropy inequalities," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 2299–2303, IEEE, 2016.

[118] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE transactions on information theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[119] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Transactions on Information Theory*, vol. 58, no. 3, pp. 1837–1852, 2011.

[120] F. Harary and E. Palmer, "The power group enumeration theorem," *Journal of Combinatorial Theory*, vol. 1, no. 2, pp. 157–173, 1966.

[121] M. A. Harrison and R. G. High, "On the cycle index of a product of permutation groups," *Journal of Combinatorial Theory*, vol. 4, no. 3, pp. 277–299, 1968.

[122] W.-D. Wei and J.-Y. Xu, "Cycle index of direct product of permutation groups and number of equivalence classes of subsets of zv," *Discrete Mathematics*, vol. 123, no. 1-3, pp. 179–188, 1993.

[123] C. Li, S. Weber, and J. M. Walsh, "Multilevel diversity coding systems: Rate regions, codes, computation, & forbidden minors," *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 230–251, 2016.

[124] J. Apte and J. M. Walsh, "Exploiting symmetry in computing polyhedral bounds on network coding rate regions," in *2015 International Symposium on Network Coding (NetCod)*, pp. 76–80, IEEE, 2015.

[125] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problemy Peredachi Informatsii*, vol. 21, no. 1, pp. 3–16, 1985.

[126] R. Lidl, H. Niederreiter, and P. Cohn, *Encyclopedia of mathematics and its applications vol 20*. Cambridge, Cambridge university press, 1983.

[127] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Transactions on Information theory*, vol. 54, no. 8, pp. 3579–3591, 2008.

[128] N. Silberstein, A. S. Rawat, and S. Vishwanath, "Error-correcting regenerating and locally repairable codes via rank-metric codes," *IEEE Transactions on Information Theory*, vol. 61, no. 11, pp. 5765–5778, 2015.

[129] C. Tian, B. Sasidharan, V. Aggarwal, V. A. Vaishampayan, and P. V. Kumar, "Layered exact-repair regenerating codes via embedded error correction and block designs," *IEEE Transactions on Information Theory*, vol. 61, no. 4, pp. 1933–1947, 2015.

[130] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[131] C. Tian, “A note on the fundamental limits of coded caching,” *arXiv preprint arXiv:1503.00010*, 2015.

[132] A. Sengupta, R. Tandon, and T. C. Clancy, “Improved approximation of storage-rate tradeoff for caching via new outer bounds,” in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 1691–1695, IEEE, 2015.

[133] D. Younger, “Graph theory (frank harary),” *SIAM Review*, vol. 14, no. 2, p. 350, 1972.

[134] N. Jacobson, *Basic algebra I.* Courier Corporation, 2012.

[135] N. Loehr, *Bijective combinatorics.* Chapman and Hall/CRC, 2011.

[136] J. A. Barnes and F. Harary, *Graph theory in network analysis.* Elsevier, 1983.

[137] C. L. Liu, *Introduction to combinatorial mathematics.* McGraw-Hill, 1968.

[138] H. Te Sun, “Nonnegative entropy measures of multivariate symmetric correlations,” *Information and Control*, vol. 36, pp. 133–156, 1978.

[139] A. Betten, “Classifying discrete objects with orbiter,” *ACM Communications in Computer Algebra*, vol. 47, no. 3/4, pp. 183–186, 2014.

[140] A. Betten, *Orbiter – A program to classify discrete objects. [Online]. Available: https://github.com/abetten/orbiter*, 2018.

[141] H. Brown, L. Hjelmeland, and L. Masinter, “Constructive graph labeling using double cosets,” *Discrete Mathematics*, vol. 7, no. 1-2, pp. 1–30, 1974.

[142] Z. Chen, P. Fan, and K. B. Letaief, “Fundamental limits of caching: Improved bounds for small buffer users,” *arXiv preprint arXiv:1407.1935*, 2014.

[143] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, “The role of caching in future communication systems and networks,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1111–1125, 2018.

APPENDIX A

PROOF OF LEMMAS, THEOREMS AND COROLLARIES

This appendix includes the proofs of several lemmas and theorems in chapter 4.

A.1 Proof of Corollary 1

We first show that (2.2), *i.e.*, the memory-rate tradeoff points given in [100], can be obtained by specializing (2.19), (2.20), and (2.26)-(2.30). For this case, the decomposition patterns are given by $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} = \{\mathbb{supp}(\boldsymbol{t})\}$, *i.e.,* there is no decomposition. As such, (2.19) reduces to

$$R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} = \sum_{\boldsymbol{t}\in\mathcal{T}^{(t)}_{\boldsymbol{d}}} \prod_{n\in\mathbb{supp}(\boldsymbol{t})} \binom{m_n}{t_n} - \sum_{\boldsymbol{t}\in\mathcal{T}^{(t)}_{\boldsymbol{d}}} \prod_{n\in\mathbb{supp}(\boldsymbol{t})} \binom{m_n - 1}{t_n}. \tag{A.1}$$

The first summation is clearly $\binom{K}{t+1}$, since it is simply the number of ways to choose $(t+1)$ users from the K users, however counted one transmission type at a time. To simplify the second term in (A.1), let us consider any demand vector $\boldsymbol{d} \in \mathcal{D}$. For any transmission type $\boldsymbol{t}$ where there exists $n \in \mathbb{supp}(\boldsymbol{t})$ such that $t_n = m_n$, the product $\prod_{n\in\mathbb{supp}(\boldsymbol{t})} \binom{m_n-1}{t_n}$ is clearly zero. The second summation can thus be viewed as counting the number of ways to choose $(t+1)$ users, however, with the leaders $\{\ell^{[n]}, m_n \neq 0\}$ not being chosen; there is clearly $\binom{K-N^*}{t+1}$ ways to do so. This implies that

$$R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}}} = \binom{K}{t+1} - \binom{K-N^*}{t+1} \leq \binom{K}{t+1} - \binom{K-\tilde{N}}{t+1}, \quad \forall \boldsymbol{d} \in \mathcal{D}. \tag{A.2}$$

Similarly (2.20) can be simplified for any $\boldsymbol{d} \in \mathcal{D}$ as

$$M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}^{(t)}_{\boldsymbol{d}},k} = N\binom{K-1}{t-1}, \tag{A.3}$$

since the other term disappears with the choice $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} = \{\mathbb{supp}(\boldsymbol{t})\}$. The quantities (A.2) and (A.3) are independent of $\boldsymbol{d}$. It is clear that (2.2) is identical to (A.2) and (A.3) after normalization with the file size $F = \binom{K}{t}$. It is easy to verify that they are indeed in the region $\mathcal{R}^{(t)}$, thus it is clearly inside **cl** $\left(\cup_{t=0,\dots,K}\mathcal{R}^{(t)}\right)$.

Next we show that (2.4), *i.e.*, the memory-rate tradeoff points given in [77], can also be obtained by specializing (2.19), (2.20), and (2.26)-(2.30). In this case, the decomposition patterns are given by $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} = \{\{n\} : n \in \mathbb{supp}(\boldsymbol{t})\}$, *i.e.,* $\mathbb{supp}(\boldsymbol{t})$ is partitioned into sets, where each set is a singleton. It follows that

$$\begin{aligned} R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} &= \sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}} \sum_{n\in\mathbb{supp}(\boldsymbol{t})} \left[\left(\binom{m_n}{t_n} - \binom{m_n-1}{t_n}\right) \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n\}} \binom{m_{n'}}{t_{n'}}\right] \\ &= \sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}} \sum_{n\in\mathbb{supp}(\boldsymbol{t})} \left[\binom{m_n-1}{t_n-1} \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n\}} \binom{m_{n'}}{t_{n'}}\right]. \end{aligned} \tag{A.4}$$

Define $\mathbb{1}_c$ to be the indicator function which is equal to 1 when the condition c holds, and is equal to 0 otherwise. We can now rewrite the summation as

$$\begin{aligned} R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} &= \sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}} \sum_{n\in\mathbb{supp}(\boldsymbol{m})} \mathbb{1}_{n\in\mathbb{supp}(\boldsymbol{t})} \left[\binom{m_n-1}{t_n-1} \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n\}} \binom{m_{n'}}{t_{n'}}\right] \\ &= \sum_{n\in\mathbb{supp}(\boldsymbol{m})} \sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}} \mathbb{1}_{n\in\mathbb{supp}(\boldsymbol{t})} \left[\binom{m_n-1}{t_n-1} \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n\}} \binom{m_{n'}}{t_{n'}}\right]. \end{aligned} \tag{A.5}$$

Notice that the equality

$$\sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}} \mathbb{1}_{n\in\mathbb{supp}(\boldsymbol{t})} \left[\binom{m_n-1}{t_n-1} \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n\}} \binom{m_{n'}}{t_{n'}}\right] = \binom{K-1}{t}, \tag{A.6}$$

since the left hand side is the number of ways to choose $t+1$ users among the K users, with $\ell^{[n]}$

already chosen, counted one transmission type at a time. It follows that for any $\boldsymbol{d} \in \mathcal{D}$,

$$R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = N^* \binom{K-1}{t}. \tag{A.7}$$

Let us turn to (2.20), the second term of which in this case can be simplified as

$$\begin{aligned}
\Delta M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} &= \sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}:t_{d_k}>0} \sum_{n\in\mathbb{supp}(\boldsymbol{t})\setminus\{d_k\}} \left(\binom{m_{d_k}-1}{t_{d_k}-1}\binom{m_n-1}{t_n-1} \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n,d_k\}} \binom{m_{n'}}{t_{n'}} \right) \\
&= \sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}:t_{d_k}>0} \sum_{n\in\mathbb{supp}(\boldsymbol{m})\setminus\{d_k\}} \mathbb{1}_{n\in\mathbb{supp}(\boldsymbol{t})} \left[\binom{m_{d_k}-1}{t_{d_k}-1}\binom{m_n-1}{t_n-1} \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n,d_k\}} \binom{m_{n'}}{t_{n'}} \right] \\
&= \sum_{n\in\mathbb{supp}(\boldsymbol{m})\setminus\{d_k\}} \sum_{\boldsymbol{t}\in\mathcal{T}_{\boldsymbol{d}}^{(t)}} \mathbb{1}_{\{n,d_k\}\subseteq\mathbb{supp}(\boldsymbol{t})} \left[\binom{m_{d_k}-1}{t_{d_k}-1} \cdot \binom{m_n-1}{t_n-1} \cdot \prod_{n'\in\mathbb{supp}(\boldsymbol{t})\setminus\{n,d_k\}} \binom{m_{n'}}{t_{n'}} \right] \\
&= (N^*-1)\binom{K-2}{t-1},
\end{aligned} \tag{A.8}$$

where the last equality is because for each fixed $n \in \mathbb{supp}(\boldsymbol{m}) \setminus \{d_k\}$, the inner summation is simply the number of ways to choose $t+1$ users in the K users, with $\ell^{[d_k]}$ and $\ell^{[n]}$ already chosen. Thus we arrive at

$$M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = N\binom{K-1}{t-1} - (N^*-1)\binom{K-2}{t-1}. \tag{A.9}$$

Note that neither (A.7) nor (A.9) depends on $\boldsymbol{d}$ or k.

For $N^* = \tilde{N}$, normalizing both of them by $\binom{K}{t}$ already gives exactly the memory-rate trade-off pairs in (2.4). This leaves us only the case when $N^* \neq \tilde{N}$ to consider. We shall use two decomposition patterns for this case. Define

$$\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}} = \begin{cases} \frac{\tilde{N}-N^*}{K-N^*} & K-t \le \tilde{N} \\ \frac{(K-t)(\tilde{N}-N^*)}{K(\tilde{N}-N^*)+tN^*} & \text{otherwise} \end{cases} \tag{A.10}$$

which is clearly non-negative and is associated with the uncoded transmission pattern, and $1-\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}$

which is also non-negative and is associated with the transmission pattern whose rate and memory are given in (A.7) and (A.9). It is easy to check that when $K-t \le \tilde{N}$,

$$\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)},k} + (1-\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}) M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = N\binom{K-1}{t-1} - (\tilde{N}-1)\binom{K-2}{t-1}. \tag{A.11}$$

and

$$\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)},k} + (1-\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}) R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = \tilde{N}\binom{K-1}{t}. \tag{A.12}$$

On the other hand, when $K-t > \tilde{N}$, (A.12) still holds, but

$$\begin{aligned}
&\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)},k} + (1-\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}) M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} \\
&= N\binom{K-1}{t-1} - \binom{K-2}{t-1}\left(\tilde{N} - \frac{\tilde{N}(\tilde{N}-N^*)+t\tilde{N}}{K(\tilde{N}-N^*)+tN^*}\right) \\
&< N\binom{K-1}{t-1} - \binom{K-2}{t-1}(\tilde{N}-1),
\end{aligned} \tag{A.13}$$

where the last inequality is because

$$\tilde{N}(\tilde{N}-N^*)+t\tilde{N} < (K-t)(\tilde{N}-N^*)+t\tilde{N} = K(\tilde{N}-N^*)+tN^* \tag{A.14}$$

by the condition $K-t > \tilde{N}$. Thus for $N^* \neq \tilde{N}$, we have found the correct $\alpha_{\breve{\boldsymbol{\mathcal{P}}}_{\boldsymbol{d}}^{(t)}}$ to satisfy the conditions in (2.26)-(2.30), and indeed the memory-rate pair (2.4) is in the region $\mathcal{R}^{(t)}$. The proof is thus complete.

A.2 The Original Optimization Problem and Its Equivalence to the LP Problem

Without assuming any symmetry, the elemental inequalities of n random variables are

$$H_P(X_{\mathcal{I}_n}) - H_P(X_{\mathcal{I}_n\backslash\{j\}}) \geq 0, \quad \text{any } j \in \mathcal{I}_n,$$
$$H_P(X_{\{i\}\cup\mathcal{Q}}) + H_P(X_{\{j\}\cup\mathcal{Q}}) - H_P(X_{\mathcal{Q}}) - H_P(X_{\{i\}\cup\{j\}\cup\mathcal{Q}}) \geq 0, \quad \text{any } i \neq j,\ \mathcal{Q} \subseteq \mathcal{I}_n\backslash\{i,j\},$$
$$H_P(X_{\mathcal{A}}) \geq 0, \quad \text{any } \mathcal{A} \subseteq \mathcal{I}_n.$$

For simplicity, in the following we use $Z_{\mathcal{A}}$ to represent entropy $H_P(X_{\mathcal{A}})$, and the last inequality can be safely omitted by the same reason that has been mentioned in section IV. Thus the following optimization problem provides an upper bound on $c_{\mathcal{O},\mathcal{O}'}$, we refer it as $P_1(\mathcal{O},\mathcal{O}')$,

$$\begin{aligned} P_1(\mathcal{O},\mathcal{O}') : \quad & \text{maximize: } \frac{|\mathcal{O}|\ell_{\mathcal{O}}}{|\mathcal{O}'|\ell_{\mathcal{O}'}} \frac{\sum_{\mathcal{A}\in\mathcal{O}'} Z_{\mathcal{A}}}{\sum_{\mathcal{A}\in\mathcal{O}} Z_{\mathcal{A}}} \\ & \text{subject to: } Z_{\mathcal{I}_n} - Z_{\mathcal{I}_n\backslash\{j\}} \geq 0, \quad \text{any } j \in \mathcal{I}_n, \\ & \qquad Z_{\{i\}\cup\mathcal{Q}} + Z_{\{j\}\cup\mathcal{Q}} - Z_{\mathcal{Q}} - Z_{\{i\}\cup\{j\}\cup\mathcal{Q}} \geq 0, \quad \text{any } i \neq j, \mathcal{Q} \subseteq \mathcal{I}_n\backslash\{i,j\}. \end{aligned}$$

This optimization problem does not have a linear objective, however it can be transformed into an equivalent LP problem, which is referred to as $P_2(\mathcal{O},\mathcal{O}')$,

$$P_2(\mathcal{O},\mathcal{O}') : \quad \text{maximize: } \frac{\ell_{\mathcal{O}}}{|\mathcal{O}'|\ell_{\mathcal{O}'}} \sum_{\mathcal{A}\in\mathcal{O}'} Z_{\mathcal{A}} \tag{A.15}$$

$$\text{subject to: } Z_{\mathcal{I}_n} - Z_{\mathcal{I}_n\backslash\{j\}} \geq 0, \quad \text{any } j \in \mathcal{I}_n, \tag{A.16}$$

$$Z_{\{i\}\cup\mathcal{Q}} + Z_{\{j\}\cup\mathcal{Q}} - Z_{\mathcal{Q}} - Z_{\{i\}\cup\{j\}\cup\mathcal{Q}} \geq 0,$$
$$\text{any } i \neq j,\ \mathcal{Q} \subseteq \mathcal{I}_n\backslash\{i,j\}, \tag{A.17}$$

$$\frac{1}{|\mathcal{O}|} \sum_{\mathcal{A}\in\mathcal{O}} Z_{\mathcal{A}} = 1. \tag{A.18}$$

This LP problem can be further reduced to a symmetric LP, where all entropies of subsets inside the same orbit are equal, this is referred to as $P_3(\mathcal{O},\mathcal{O}')$,

$$P_3(\mathcal{O},\mathcal{O}'): \quad \text{maximize: } \frac{\ell_{\mathcal{O}}}{|\mathcal{O}'|\ell_{\mathcal{O}'}} \sum_{\mathcal{A}\in\mathcal{O}'} Z_{\mathcal{A}} \tag{A.19}$$

$$\text{subject to: } Z_{\mathcal{I}_n} - Z_{\mathcal{I}_n\backslash\{j\}} \geq 0, \quad \text{any } j \in \mathcal{I}_n, \tag{A.20}$$

$$Z_{\{i\}\cup\mathcal{Q}} + Z_{\{j\}\cup\mathcal{Q}} - Z_{\mathcal{Q}} - Z_{\{i\}\cup\{j\}\cup\mathcal{Q}} \geq 0,$$

$$\text{any } i \neq j,\ \mathcal{Q} \subseteq \mathcal{I}_n\backslash\{i,j\}, \tag{A.21}$$

$$Z_{\mathcal{A}} = 1, \quad \text{any } \mathcal{A} \in \mathcal{O}, \tag{A.22}$$

$$Z_{\mathcal{A}} = Z_{\mathcal{B}}, \quad \text{any } \mathcal{A},\mathcal{B},\mathcal{O}'' \text{ such that } \mathcal{A},\mathcal{B} \in \mathcal{O}''. \tag{A.23}$$

Next we prove that they are all equivalent. The optimal values of the objective functions in three problems are denoted as $(P_1), (P_2)$ and (P_3), respectively.

Theorem 10. $(P_1) = (P_2)$.

Proof. First notice that the objective function of $P_1(\mathcal{O},\mathcal{O}')$ becomes exactly that of $P_2(\mathcal{O},\mathcal{O}')$ by letting $\frac{1}{|\mathcal{O}|}\sum_{\mathcal{A}\in\mathcal{O}} Z_{\mathcal{A}} = 1$, this implies that $(P_1) \geq (P_2)$ because $P_2(\mathcal{O},\mathcal{O}')$ is more constrained than $P_1(\mathcal{O},\mathcal{O}')$. Thus we can focus on the other direction $(P_1) \leq (P_2)$.

Denote the optimal vector for $P_1(\mathcal{O},\mathcal{O}')$ as $\{Z^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$, and suppose it yields $\frac{1}{|\mathcal{O}|}\sum_{\mathcal{A}\in\mathcal{O}} Z^*_{\mathcal{A}} = c$. We only need to show that there is a vector $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$, which satisfies all the constraints of $P_2(\mathcal{O},\mathcal{O}')$ and has the same objective function value as $P_1(\mathcal{O},\mathcal{O}')$, because this would imply $(P_2) \geq (P_1)$. This vector can be constructed as

$$\hat{Z}^*_{\mathcal{A}} = \frac{1}{c} Z^*_{\mathcal{A}}, \mathcal{A} \subseteq \mathcal{I}_n. \tag{A.24}$$

Thus when substituting the vector $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ in, it is obvious that the objective function $P_2(\mathcal{O},\mathcal{O}')$ achieves the same value as $P_1(\mathcal{O},\mathcal{O}')$, and this vector satisfies all the constraints in $P_2(\mathcal{O},\mathcal{O}')$, by observing that the linear scaling factor $\frac{1}{c}$ can be canceled out on both sides of the inequalities.

The two conditions $(P_1) \geq (P_2)$ and $(P_1) \leq (P_2)$ yield $(P_1) = (P_2)$. □

Theorem 11. $(P_2) = (P_3)$.

Proof. First notice that $(P_2) \geq (P_3)$ because both problems have the same objective function, but $P_3(\mathcal{O}, \mathcal{O}')$ is more constrained than $P_2(\mathcal{O}, \mathcal{O}')$. Thus we can focus on the other direction $(P_2) \leq (P_3)$.

Denote the optimal vector for $P_2(\mathcal{O}, \mathcal{O}')$ as $\{Z^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$, we only need to show that there is a vector $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$, which satisfies all the constraints of $P_3(\mathcal{O}, \mathcal{O}')$ and has the same objective function value as $P_2(\mathcal{O}, \mathcal{O}')$, because this would imply

$$(P_3) \geq \frac{\ell_{\mathcal{O}}}{|\mathcal{O}'|\ell_{\mathcal{O}'}} \sum_{\mathcal{A}\in\mathcal{O}'} Z_{\mathcal{A}} = (P_2).$$

For this purpose, consider cyclic shift g_k, $k = 0, 1, \ldots, n-1$ of the indices $\mathcal{I}_n$, and the induced operations on $\{Z^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$,

$$g_k(Z^*_{\mathcal{A}}) \triangleq Z^*_{g_k(\mathcal{A})}, \quad k = 0, 1, \ldots, n-1.$$

Consider the following vector,

$$\hat{Z}^*_{\mathcal{A}} = \frac{1}{n}\sum_{k=0}^{n-1} Z^*_{g_k(\mathcal{A})} \triangleq \hat{Z}^*_{\tilde{\mathcal{O}}}, \quad \mathcal{A} \in \tilde{\mathcal{O}},$$

which implies that it is not a function of $\mathcal{A}$, but only a function of the orbit it belongs, and thus constraint (A.23) is satisfied. Furthermore, for any $\tilde{\mathcal{O}}$, suppose $d|\tilde{\mathcal{O}}| = n$, where d is an integer and $d \geq 1$, then

$$\sum_{\mathcal{A}\in\tilde{\mathcal{O}}} \hat{Z}^*_{\mathcal{A}} = \sum_{\mathcal{A}\in\tilde{\mathcal{O}}} \frac{1}{n}\sum_{k=0}^{n-1} Z^*_{g_k(\mathcal{A})} = \sum_{\mathcal{A}\in\tilde{\mathcal{O}}} \frac{1}{d|\tilde{\mathcal{O}}|} d \sum_{k=0}^{|\tilde{\mathcal{O}}|-1} Z^*_{g_k(\mathcal{A})} = \sum_{\mathcal{A}\in\tilde{\mathcal{O}}} \frac{1}{|\tilde{\mathcal{O}}|} \sum_{\mathcal{A}\in\tilde{\mathcal{O}}} Z^*_{\mathcal{A}} = \sum_{\mathcal{A}\in\tilde{\mathcal{O}}} Z^*_{\mathcal{A}}.$$

Thus when substituting the vector $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ in, the objective function achieves the same value with $P_2(\mathcal{O}, \mathcal{O}')$. It remains to show that $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ satisfies the constraints (A.20-A.22) in $P_3(\mathcal{O}, \mathcal{O}')$.

First notice that by setting $\tilde{\mathcal{O}} = \mathcal{O}$, we have $\hat{Z}^*_{\mathcal{A}} = \frac{1}{d} \cdot d \cdot \frac{1}{|\mathcal{O}|} \sum_{k=0}^{|\mathcal{O}|-1} Z^*_{g_k(\mathcal{A})} = 1$ by (A.18), thus constraint (A.22) is satisfied. Now consider an arbitrary constraint of the form (A.20),

$$Z_{\mathcal{I}_n} - Z_{\mathcal{I}_n \backslash \{j\}} \geq 0,$$

in $P_3(\mathcal{O}, \mathcal{O}')$, substituting $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ into its left hand side gives

$$\hat{Z}^*_{\mathcal{I}_n} - \hat{Z}^*_{\mathcal{I}_n \backslash \{j\}} = Z^*_{\mathcal{I}_n} - \frac{\sum_{j\in\mathcal{I}_n} Z^*_{\mathcal{I}_n \backslash \{j\}}}{n} = \frac{\sum_{j=1}^{n} \left(Z^*_{\mathcal{I}_n} - Z^*_{\mathcal{I}_n \backslash \{j\}} \right)}{n} \geq 0,$$

where the last inequality is true because $\{Z^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ is a valid solution for $P_2(\mathcal{O}, \mathcal{O}')$, and the quantity in the parenthesis is nonnegative as one of the constraints.

Similarly, for the second type of constraint (A.21),

$$Z_{\{i\}\cup\mathcal{Q}} + Z_{\{j\}\cup\mathcal{Q}} - Z_{\mathcal{Q}} - Z_{\{i\}\cup\{j\}\cup\mathcal{Q}} \geq 0,$$

substitute $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ into the left hand side and multiply it by n, we have

$$\begin{aligned}
&\left(\hat{Z}^*_{\{i\}\cup\mathcal{Q}} + \hat{Z}^*_{\{j\}\cup\mathcal{Q}} - \hat{Z}^*_{\mathcal{Q}} - \hat{Z}^*_{\{i\}\cup\{j\}\cup\mathcal{Q}} \right) n \\
=& \sum_{k=0}^{n-1} Z^*_{g_k(\{i\}\cup\mathcal{Q})} + \sum_{k=0}^{n-1} Z^*_{g_k(\{j\}\cup\mathcal{Q})} - \sum_{k=0}^{n-1} Z^*_{g_k(\mathcal{Q})} - \sum_{k=0}^{n-1} Z^*_{g_k(\{i\}\cup\{j\}\cup\mathcal{Q})} \\
=& \sum_{k=0}^{n-1} \left(g_k(Z^*_{\{i\}\cup\mathcal{Q}}) + g_k(Z^*_{\{j\}\cup\mathcal{Q}}) - g_k(Z^*_{\mathcal{Q}}) - g_k(Z^*_{\{i\}\cup\{j\}\cup\mathcal{Q}}) \right) \geq 0,
\end{aligned}$$

where in the last step we again use the fact that $\{Z^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ is a valid solution for $P_2(\mathcal{O}, \mathcal{O}')$. Thus the construction $\{\hat{Z}^*_{\mathcal{A}:\mathcal{A}\subseteq\mathcal{I}_n}\}$ indeed satisfies all the constraints in $P_3(\mathcal{O}, \mathcal{O}')$ and this completes the proof. □

It is straightforward to see that in the symmetric LP $P_3(\mathcal{O}, \mathcal{O}')$, the last constraint (A.23) forces all entropies of subsets in the same orbit to be equal, thus the LP variables are in fact all cyclic orbit entropies $Z_{\mathcal{O}}$ instead of all $Z_{\mathcal{A}}$. Hence, $P_3(\mathcal{O}, \mathcal{O}')$ can be rewritten as exactly the LP in section IV.

A.3 Proof of Theorem 7

Proof. In the regenerating code problem, the permutation group $G_{\mathcal{I}}$ is a symmetric group S_n, thus its cycle index is given as (4.5). Alternatively, it can be written in the form of (4.31), and the number of monomials in it is the number of partitions $p(n)$. This is implied by the fact that the permutations associated with the same partition have the same cycle index.

Without loss of generality, only one permutation $\pi \in P_{G_{\mathcal{I}}} = P_{S_n}$ needs to be examined, suppose that it has a cycle index of $x_1^{c_{q1}} x_2^{c_{q2}} \dots x_n^{c_{qn}}$, and we will prove that it induces a permutation π^{ind} having a cycle index of $\prod_{\substack{l_1=1,2,\dots,n;\\ l_2=l_1+1,\dots,n}} x_{l_1}^{l_1 c_{ql_1}^2} x_{\mathrm{lcm}(l_1,l_2)}^{c_{ql_1 l_2}}, c_{ql_1 l_2} = \frac{2 l_1 l_2 c_{ql_1} c_{ql_2}}{\mathrm{lcm}(l_1,l_2)}$. Since only one π is under study, the notation 'q' in the subscript of each exponent c is omitted.

The proof is done by examining the mapping of each element in $\mathcal{X}$ under permutation π^{ind}. The set $\mathcal{X}$ can be illustrated as a matrix where each $X_{ij} \in \mathcal{X}$ is denoted as an element located on the ith row and jth column. Let us assume that in permutation π, i belongs to a length-l_1 cycle and j belongs to a length-l_2 cycle, $i, j \in \mathcal{I}_n$, then this implies that $\pi^{l_1}(i) = i$ and $\pi^{l_2}(j) = j$. The elements in the matrix can be ordered according to the cycle notation, just as Figure A.1 shows. For the sake of simplicity but without loss of generality, we suppose that each cycle in π contains consecutive increasing numbers, and the operation π^{ind} will map X_{ij} to $X_{i'j'}$, where $i' = [(i - a_i + 1) \mod l_1] + a_i$ and $j' = [(j - a_j + 1) \mod l_2] + a_j$, and a_i (a_j) is the starting value of the length-l_1 (length-l_2) cycle containing i (j). That is, the induced permutation π^{ind} moves X_{ij} along the diagonal direction by one position within the same sub-matrix, and cycles back in the sub-matrix when reaching to the boundary.

- For those X_{ij}'s whose subscripts i and j belong to the same length cycle (not necessarily the same cycle) in π, i.e., $l_1 = l_2$, X_{ij} will also belong to a length-l_1 cycle under the mapping by π^{ind}, since

$$X_{ij} \to \pi^{ind}(X_{ij}) \to (\pi^{ind})^2(X_{ij}) \to \dots \to (\pi^{ind})^{l_1-1}(X_{ij}) \to (\pi^{ind})^{l_1}(X_{ij}), \qquad \text{(A.25)}$$

  where $(\pi^{ind})^{l_1}(X_{ij}) = X_{\pi^{l_1}(i)\pi^{l_1}(j)} = X_{ij}$.

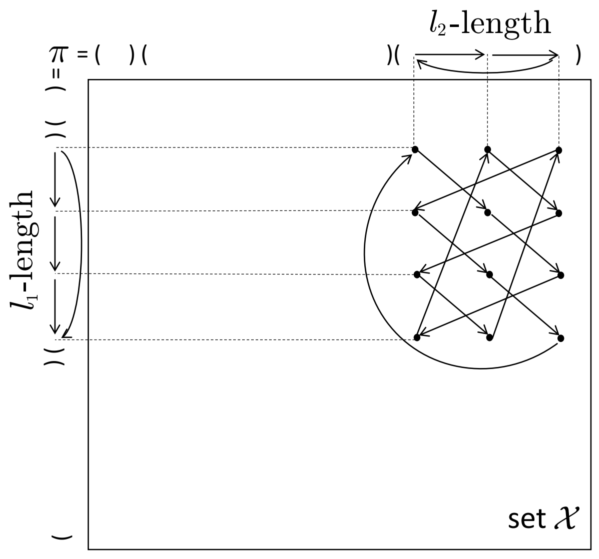

Figure A.1: The elements X_{ij} where i belongs to a length-l_1 cycle and j belongs to a length-l_2 cycle, in this figure, $l_1 = 4, l_2 = 3$. Reprinted with permission from [4,5], © 2017,2018 IEEE.

Since there are c_{l_1} such length-l_1 cycles in π, i and j would both have $l_1 c_{l_1}$ different choices, therefore the number of distinct X_{ij} is $(l_1 c_{l_1})^2$. Based on (A.25), the $l_1^2 c_{l_1}^2$ distinct X_{ij}'s will form $\frac{l_1^2 c_{l_1}^2}{l_1} = l_1 c_{l_1}^2$ cycles in permutation π^{ind}, each of length l_1. That is, the term $x_{l_1}^{c_{l_1}}, l_1 = 1, 2, \ldots, n$ in the cycle index of π indicates the presence of a term $x_{l_1}^{l_1 c_{l_1}^2}$ in the cycle index of π^{ind}.

- For those X_{ij}'s whose subscripts i and j belong to different length cycles in π, i.e., $l_1 \neq l_2$, clearly, it will take exactly lcm(l_1, l_2) steps for π^{ind} to map such an X_{ij} to itself,

$$X_{ij} \to \pi^{ind}(X_{ij}) \to (\pi^{ind})^2(X_{ij}) \to \ldots \to (\pi^{ind})^{\mathrm{lcm}(l_1,l_2)-1}(X_{ij}) \to (\pi^{ind})^{\mathrm{lcm}(l_1,l_2)}(X_{ij}),$$

where $(\pi^{ind})^{\mathrm{lcm}(l_1,l_2)}(X_{ij}) = X_{\pi^{\mathrm{lcm}(l_1,l_2)}(i)\pi^{\mathrm{lcm}(l_1,l_2)}(j)} = X_{ij}$.

For any fixed l_1, l_2, the number of distinct X_{ij}'s is $l_1 c_{l_1} l_2 c_{l_2}$, they form $\frac{l_1 l_2 c_{l_1} c_{l_2}}{\mathrm{lcm}(l_1,l_2)}$ cycles in the induced permutation group π^{ind}, each of length $\mathrm{lcm}(l_1, l_2)$. That is, every possible paired-term $x_{l_1}^{c_{l_1}} x_{l_2}^{c_{l_2}}, l_1 \neq l_2, l_1, l_2 = 1, 2, \ldots, n,$ in the cycle index of π indicates the presence of a term $x_{\mathrm{lcm}(l_1,l_2)}^{\frac{l_1 l_2 c_{l_1} c_{l_2}}{\mathrm{lcm}(l_1,l_2)}}$ in the cycle index of π^{ind}.

In fact, in (b) if we just consider the X_{ij} where i belongs to a cycle shorter than the cycle j belongs to, i.e., $l_1 < l_2$, in other words, for all $l_1 \neq l_2$ sub-matrices we only study exactly half of them, then the other half is symmetric to the diagonal of the matrix and the same argument holds true, which leads to $\prod_{\substack{l_1=1,2,\ldots,n;\\ l_2=l_1+1,\ldots,n}} x_{\mathrm{lcm}(l_1,l_2)}^{c_{l_1 l_2}}, c_{l_1 l_2} = \frac{2 l_1 l_2 c_{l_1} c_{l_2}}{\mathrm{lcm}(l_1,l_2)}$. Hence, the final result is $\prod_{\substack{l_1=1,2,\ldots,n;\\ l_2=l_1+1,\ldots,n}} x_{l_1}^{l_1 c_{l_1}^2} x_{\mathrm{lcm}(l_1,l_2)}^{c_{l_1 l_2}}, c_{l_1 l_2} = \frac{2 l_1 l_2 c_{l_1} c_{l_2}}{\mathrm{lcm}(l_1,l_2)}$. This completes the proof. $\square$