Title

Content-Type Coding

Author

Song, Linqi

Publication Date

2017

UNIVERSITY OF CALIFORNIA

Los Angeles

Content-Type Coding

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Linqi Song

2017

ABSTRACT OF THE DISSERTATION

Content-Type Coding

by

Linqi Song
Doctor of Philosophy in Electrical Engineering
University of California, Los Angeles, 2017
Professor Christina Panagio Fragouli, Chair

Traditionally, we use networks to securely and efficiently convey specific information messages to one or more receivers. However, communication networks today are increasingly used to serve a fundamentally different traffic, that delivers types of content rather than specific messages. For instance, when we want to find photos of an event, we may not care which specific photos we receive - we only care about the type of content, namely, that they are photos of the correct event. Content-type traffic pervades a host of applications today, e.g., search engines, recommender systems, and advertising networks. Research on content-type networks is very popular. Most of the existing work looks at how to classify content into types; what to replicate, where and how to store, and from where to retrieve specific data. In contrast, we investigate a totally different question: are there benefits in designing information transmission schemes specifically tailored to content-type traffic?

Our research indicates that in some cases, these benefits can be significant. We design a polynomial-time algorithm for pliable index coding that requires at most $O(\log^2(n))$ broadcast transmissions to serve n clients, as compared to $O(n)$ broadcast transmissions for conventional index coding. This indicates that the exponential benefits of pliable index coding can be effectively realized. Moreover, we explore two applications: recommender systems and distributed computing. In recommender systems, we ask: how much we can gain in terms of bandwidth and user satisfaction, if recommender systems took into account not only the user preferences, but also the fact that they may need to serve these users under bandwidth constraints, as is the case over wireless networks. In other words, what if the recommender systems became bandwidth aware? In this setup, the user is satisfied to receive any message she does not already have, with a satisfaction proportional to her preference for that message. We show, through a number of scenaria, that although the optimization problems are in general NP-hard, polynomial time algorithms with constant approximation ratio can be designed to achieve more than 80% of the satisfaction and to save 90% of bandwidth. In distributed computing, to improve the communication efficiency in the data shuffling phase, we examine the pliable index coding problem under data shuffling constraints, where each of the m messages can satisfy at most c out of n clients. We show that the constrained pliable index coding can achieve up to $O(n)$ (best case) benefits over index coding. We prove that the problem is NP-hard and the optimal broadcast transmissions for random instances is almost surely upper bounded by $O(\min\{\frac{n}{c\log(n)}, \frac{n}{\log(m)}\})$ for $c = o(\frac{n^{1/7}}{\log^2(n)})$ and $O(\min\{\frac{n}{c} + \log(c), \frac{n}{\log(m)}\})$ for $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$. Building upon constrained pliable index coding, we design a hierarchical data shuffling scheme for distributed computing. By leveraging the many possible shuffling choices, our proposed shuffling scheme is able to reduce the communication cost and achieve benefits up to $O(ns/m)$ over the index coding method, where ns/m is the average number of workers caching a message, and m, n, and s are the numbers of messages, workers, and cache size, respectively. In addition, we study the beneficial cases of content-type coding over large scale networks and over erasure channels. Compared with message-specific coding, we show that the benefits can be up to the number of messages in the content type for the former case and up to 19.5% for the latter case with symmetric setting.

The dissertation of Linqi Song is approved.

Richard D Wesel

Suhas N. Diggavi

Gregory J Pottie

Christina Panagio Fragouli, Committee Chair

University of California, Los Angeles

2017

To TianTian

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. Christina Fragouli, who had kindly accepted me in the ARNI lab and since then has invariably been the best advisor a student could wish for. I have benefited and learned tremendously from her active involvement, passion, and enthusiasm for high-quality research. She has always provided a clear route for the research direction, which helped me to avoid detours and to only focus on important and valuable things. She has deep knowledge in our research area and has always given me excellent suggestions and insightful ideas that inspire me to generate novelty, when I face obstacles or feel frustrated. I especially thank her for reading my proofs and fortunately she loved them. During my Ph.D. process, she was always there to help me with the writing, ranging from conference papers to important emails. She has an open mind and has encouraged and supported me to attend conferences, such as NetCod, ITA, and ISIT, and to meet with people in the information theory community. I would also like to express my gratitude to her for her care in every aspects of life. She is an extremely kind, respectful and honest person. Her face has been always with warm smile whenever I got any exciting result or nothing at all. Her encourage and help makes our ARNI lab a helpful and cooperative community, and also a family-friendly community that has helped me a lot in balancing the academic study and family life.

I would also like to thank my other committee members, Prof. Gregory Pottie, Prof. Suhas N. Diggavi, and Prof. Richard Wesel for their instructions, insight and help in various aspects. It was my great honor to collaborate with Prof. Pottie for a paper about activity classification and to be his TA for an online course. I was also lucky to be Prof. Diggavi's TA and to take his two courses, especially the network information theory course, which was tough but helped me a lot during research. I am also grateful to Prof. Wesel for helping me several times, from passing over a TAship to me to giving insightful comments, and I was fascinated to listen to his outlook on future coding techniques in 5G networks.

I would also like to thank Prof. Mihaela van der Schaar for offering me a chance to UCLA and collaborating several papers. I would also like to thank Prof. Raghu Meka for serving on my doctoral committee during my Qualification exam, thank Prof. Lieven Vandenberghe for his accommodation in several group seminars, and thank Prof. William Hsu for collaboration.

I also thank my colleagues and good friends at ARNI lab: Dr. Martina Cardone, Dr. Ayan Sengupta, Dr. Iris Safaka, Mohammed Karmoose, Yahya Ezzeldin, Gaurav Kumar Agarwal, and Sundara Rajan Srinivasavaradhan; and my other good friends at UCLA: Dr. Yu Zhang, Dr. Yuanzhang Xiao, Dr. Jie Xu, Dr. Cem Tekin, Siming Song, Haobo Wang, Wenlong Jiang, Jinxi Guo, Dr. Bokai Yan, Dr. Erkao Bao, Dr. Yajing Liu. They helped me in many aspects in my research and my life, and made my PhD life much more colorful.

Finally, I would like to thank my lovely daughter, TianTian, my intelligent wife, Xinyi, my parents, my parents-in-law, and my brother. They helped taking care of TianTian during the past few years. Their love, support, and encouragement get me through my PhD career. No word in the world can express my gratitude for them.

VITA

2002–2006	B.E. (Electronic Engineering), Tsinghua University, Beijing, China
2006–2009	M.E. (Electronic Engineering), Tsinghua University, Beijing, China
2009–2012	Project Manager, China Mobile Communications Corporation, Beijing, China
2012–present	Research Assistant, Electrical Engineering Department, UCLA, Los Angeles, California, USA

PUBLICATIONS

L. Song, and C. Fragouli, “A pliable index coding approach to data shuffling,” submitted to *IEEE Transactions on Information Theory.* (Available: *arXiv preprint, arXiv:1701.05540*, conference version: accepted and to appear in *ISIT17.*)

L. Song, and C. Fragouli, “Making recommendations bandwidth aware,” submitted to *IEEE Transactions on Information Theory.* (Available: *arXiv preprint, arXiv:1607.03948*, conference version: accepted and to appear in *ISIT17.*)

M. Karmoose, L. Song, M. Cardone, and C. Fragouli, “Private broadcasting: an index coding approach,” accepted and to appear in *ISIT17.*

L. Song, and C. Fragouli, “A polynomial-time algorithm for pliable index coding,” submitted to *IEEE Transactions on Information Theory.* (Available: *arXiv preprint, arXiv:1610.06845*, conference version: in *ISIT16.*)

L. Song, and C. Fragouli, "Content-type coding", in *IEEE International Symposium onNetwork Coding (NetCod)*, Sydney, Australia, 2015.

J. Xu, L. Song, J. Y. Xu, G. J. Pottie, and M. van der Schaar, "Personalized active learning for activity classification using wireless wearable sensors," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 5, pp. 865-876, 2016. (Conference version: in *GLOBECOM14*.)

L. Song, C. Tekin, and M. van der Schaar, "Online learning in large-scale contextual recommender systems," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 433-445, 2016. (Conference version: in *ICASSP14*.)

L. Song, W. Hsu, J. Xu, and M. van der Schaar, "Using contextual learning to improve diagnostic accuracy: application in breast cancer screening," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 3, pp. 902-914, 2016.

L. Song, Y. Xiao, and M. van der Schaar, "Demand side management in smart grids using a repeated game framework," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 7, pp. 1412-1424, 2014. (Conference version: in *ICASSP14*.)

CHAPTER 1

Introduction

1.1 Motivation

In communication networks, conventional transmission and coding aim to securely and efficiently convey *specific* information messages to one or more receivers. This broad aim encompasses most of the work in the field, from the channel coding theorem of Shannon [Sha48], to recent breakthroughs in channel coding [Ari09, KRU11], network coding [ACL00], and index coding [BK98]. However, communication networks today are increasingly used to serve a fundamentally different traffic, that delivers *type of content* rather than specific messages. As a simple example, when we use the Internet to access our bank account, we ask and want to see very specific information. However, if we search for a photo of a hummingbird, for instance, to fill a blank region of our blog, and we find millions of results. Sometimes, we do not care which specific hummingbird photos we receive - we do not even know what hummingbird photos are available - we only care about the content type, that it is a hummingbird photo and not an owl photo. Another example is the coupon distribution system in a shopping mall that delivers coupons for users. A user may have some coupons in her wireless device and may not know exactly the specific coupons that exist, but would be happy to receive any new coupon that she does not have.

Content-type traffic pervades a host of applications today, especially those relying on Big Data. For instance, content-delivery networks, such as the Akamai network, in many cases do not need to satisfy message-specific requests, but instead content-type requests (e.g., latest news on Greece, popular sites on CNN.com, etc.); search engines and recommender systems (e.g., Google, Pandora, Amazon) generate in the majority content-type traffic; advertising networks (e.g., placing ads on hotels or cars), and newsfeeds on social networks (e.g., cultural trends, following celebrities) also fall in the content-type category. The fact that content forms a significant percentage of the Internet traffic has been well recognized in many research fields, such as communications, networking, machine learning, and information retrieval. However, most of the existing work looks at how to classify content into types; what to replicate, where and how to store and from where to retrieve specific data.

Current communication network structure and transmission schemes are mainly designed for specific message communications between servers and clients or among clients. When serving as a content-type communication network, current network structure and coding schemes are not efficient enough to serve a number of these content-traffic based applications. Our *content-type coding* is specifically designed to increase the efficiency of such applications. Recall the previous message-specific communication example of checking bank account information. The specific information messages are transmitted as packets that contain IP address information to indicate the specific transmitters and receivers. While in content-type applications, such as searching for a hummingbird photo, in order to satisfy the client's request, it is not necessary to build up a specific message transmission scheme at first. Instead, the network can first determine the requested content type and wisely choose a specific message from the content type, and then transmit this specific message.

When dealing with content-type coding, we may face the following challenges.

• By leveraging the new degree of freedom for content-type applications, how can we quantitatively evaluate the benefits we get using content-type coding compared with traditional message-specific coding schemes? These comparisons include both worst case performance and average case performance and sometimes it may be hard to find an associating message-specific counterpart as a benchmark. In addition, we cannot be so optimistic, because in certain instances this new degree of freedom can introduce benefits but in others it may not.

• Even if for these beneficial cases, the content-type coding problem may be NP-hard. Can we realize these benefits by designing low complexity polynomial-time approximation algorithms?

- Can we design content-type codes for proof-of-concept applications? Unlike classical message-specific coding schemes that treat applications as black boxes and focus only on transmissions, our content-type coding may need domain knowledge of applications. The design of content-type codes may rely on different application workflow and models. Can content-type coding help in applications, such as recommender systems and distributed computing systems?

1.2 Related Work

1.2.1 Index Coding and Pliable Index Coding

Index coding was first introduced by Birk [BK98]. The conventional index coding problem considers a server with m messages and n clients [BK98, LS09, BKL10, CS08]. Each client has as side-information a subset of the messages and requires a specific message she does not have. The aim is to find an efficient way of broadcasting the messages over a noiseless channel such that all clients can be satisfied with the minimum number of transmissions. The reason we emphasize index coding is that this is a general coding framework that is shown to be equivalent to the network coding problem [ESG08, EEL15].

The content-type coding problem in the index coding framework is termed pliable index coding that was first introduced in [BF12]. Pliable index coding still considers a server and n clients with side information. However, we now assume that the clients are pliable, and are happy to receive any new message they do not already have.

For conventional index coding, it has been shown that the problem is NP-hard and in the worst case may require $\Omega(n)$ transmissions [Pee96, BBJ11]. The optimal linear code length is shown as the minimum rank of a family of matrices and has a sandwich property. Namely, the optimal code length is lower bounded and upper bounded by the clique number and the chromatic number of some specifically defined graphs [BBJ11]. Various other techniques, e.g., linear programming [BKL10], interference alignment [MCJ14], information theory [ABK13], network coding and matroid theory [ESG10], have also been used to analyze the index coding problem. In [LS09], the insufficiency of linear codes to achieve the optimum is shown by some special examples. The equivalence of index coding and network coding is studied in [ESG10, EEL15]. In [ABK13, DSC14], the capacity of index coding is studied through information theoretical analysis. In addition, several aspects of index coding problem are also investigated in the literature, such as the complementary index coding problem [CAS11], security of index coding [DSC12], efficient algorithms [CS08], and index coding with outerplanar side information [BL11]. The analysis of index coding over random graphs characterizes "typical" or "average" performance of index coding problem. One can refer to [Bol13] to get more details about random graphs. The work in [HL12] shows that the minimum length of index code for a random graph is almost surely $\Omega(\sqrt{n})$. A recent work improves this bound by showing that the minrank achieves $\Theta(n/\log(n))$ almost surely [GRW16].

In contrast, pliable index coding is also NP-hard, but requires an exponentially smaller number of transmissions (over index coding), in the worst case $O(\log^2(n))$ [BF12]. For pliable index coding with t-requests, the work in [BF13] has shown that an upper bound is $O(t\log(n) + \log^3(n))$ for the code length. These results show that pliable index coding may have exponential benefits over conventional index coding in terms of the number of transmissions.

1.2.2 Recommender Systems

Recommender systems decide which item (message) to offer to users (clients) so as to maximize a benefit (for instance, in advertising networks the benefit could be the profit gained from the ad placement) [RV97, LCL14], given a number of m items (messages) and n users (clients) in the system. The benefits are aggregated as sums of individual benefit (score) $s_{i,j}$ if message j is received by client i. For example, this score can be interpreted as the Click Through Rate (CTR) in advertising or news recommendations [RDR07]; this score can also be calculated based on user's preference rankings of messages [Bor81,DG77,Kem59,DKN01].

One feature of current recommender systems is that they are oblivious to the cost of distributing the content from the server to the points of consumption, which, however, forms many times the point of failure: unsatisfactory delivery is identified as a core threat to the user experience and has already caused loss of billions of revenue dollars [Con15]. Wireless consumption in particular, that is increasingly gaining momentum, is inherently subject to bandwidth constraints. How to realize as many benefits as possible in the recommender system is a main challenge provided that the bandwidth is limited.

1.2.3 Distributed Computing

A promising research area that has recently emerged, is on how to use coding techniques to improve the communication efficiency in distributed computing systems [LLP15, AT16, LMA15]. In particular, index coding has been used to increase the efficiency of data shuffling, that can form a major communication bottleneck for big data applications [LLP15, LMA15, CZM11].

A commonly known distributed system model is the "master-workers" model [LLP15, AT16], where a master node has m messages and is connected through a broadcast channel to n worker nodes. Each worker i is equipped with a cache that can store s_i messages. Data shuffling occurs in iterations, where in each iteration we need to refresh the data the workers have, with a random selection of s_i out of m messages from the master node. Application examples include distributed machine learning, where data shuffling updates the training data in workers [LLP15], and mobile cloud gaming systems where each iteration equips the users with new attributes, e.g., new maps [SHN11].

1.3 Notation

Throughout the thesis, we will use $[y]$ ($y \in \mathbf{Z}^+$ is a positive integer) to denote the set $\{1, 2, \ldots, y\}$ and use $|Y|$ to denote the cardinality of the set Y. We will denote by $\mathbf{F}_q$ the finite field with order q. Unless otherwise specified, we will denote the number of messages by m and a specific message by $j \in [m]$ or $b_j \in \mathbf{F}_q$, denote the number of clients by n and a specific client by $i \in [n]$, and denote the number of transmissions by K and a specific transmission by $k \in [K]$. We will interchangeably use notions "message b_j" to highlight a symbol in a finite field $\mathbf{F}_q$ and "message j" to highlight the index. Sometimes, we will say a client c_i instead of i for clearness purposes.

1.4 Main Contributions and Organization

In this thesis, we have explored several content-type coding problems, provided theoretical and algorithmic understanding of these problems, and demonstrated proof-of-concept applications. The main contributions of this thesis are as follows.

1) *Contributions to pliable index coding.*

• We propose a decoding criterion for pliable index coding, which can determine whether a given coding matrix $\boldsymbol{A}$ can satisfy a pliable index coding problem instance.

• Leveraging the decoding criterion, we propose a polynomial-time approximation algorithm for pliable index coding, termed BinGreedy, that requires in the worst case $O(\log^2(n))$ broadcast transmissions.

• We extend the proposed BinGreedy algorithm to multiple requests case where each client requires t unknown messages that she does not have. We show that our algorithm requires at most $O(t\log(n) + \log^2(n))$ broadcast transmissions.

• We construct lower bound instances that require at least $\Omega(\log(n))$ transmissions for linear pliable index coding and at least $\Omega(t + \log(n))$ transmissions for the t-requests case, indicating that the ratio of upper and lower bounds is within a factor of $O(\log(n))$.

• We provide a probabilistic analysis and show that the required number of transmissions is almost surely $\Theta(\log(n))$ for random graph instances, as compared to $\Theta(n/\log(n))$ for index coding.

2) *Contributions to bandwidth aware recommender systems.*

We model the bandwidth aware recommender system in the pliable index coding framework and examine the trade-off between benefit and bandwidth across three scenaria: no side information, side information with equal size, side information with arbitrary size.

• For the scenario without side information, we consider the Borda count ranking model

[Bor81] and show that the problem is NP-hard. However, we can design a simple greedy polynomial time algorithm that achieves an approximation ratio of 1.58. We provide upper and lower bounds of optimal benefits, as well as an average case analysis, both indicating diminishing returns: the benefits increase with the number of transmissions K only by a multiplicative factor of $1 - 1/K$.

• For the scenario with equal size side information, we consider that each client has a partial ranking over the desired messages. We prove lower bounds on the optimal benefits, and design a polynomial-time algorithm with an $O(1)$ approximation ratio.

• For the scenario with arbitrary size side information, we consider that each client has a score associated with the desired messages. We prove that this problem is hard to approximate within a ratio of $n^{1-\epsilon}$ for any $\epsilon > 0$. We also establish a connection with the maximum weighted independent set problem, based on which we design and evaluate a heuristic coded algorithm.

3) *Contributions to data shuffling in distributed computing systems.*

• We propose a constrained pliable index coding framework under the data-shuffling constraint, where each message can satisfy at most c clients. We show that the constrained pliable index coding can achieve up to $O(n)$ benefits over index coding. We prove that the problem is NP-hard and the optimal code length for random instances is almost surely upper bounded by $O(\min\{\frac{n}{c\log(n)}, \frac{n}{\log(m)}\})$ for $c = o(\frac{n^{1/7}}{\log^2(n)})$ and $O(\min\{\frac{n}{c} + \log(c), \frac{n}{\log(m)}\})$ for $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$.

• We design a hierarchical transmission scheme to achieve a "random-like" data shuffling that utilizes pliable index coding as a component. We show that our scheme can achieve benefits $O(ns/m)$, in terms of broadcast transmissions over index coding, where s is the cache size and ns/m is the average number of workers that cache each message.

4) *Content-type coding over large scale networks and lossy networks.*

• We consider a combination-like network and show that the benefits in a content-type setup can be as large as the size of the content-type (i.e., number of messages in a content-type).

- We propose a capacity-achieving transmission scheme for the broadcast erasure channel with feedbacks, where a source wants to send content-type messages to two receivers. We show that the capacity of the erasure channel in content-type setup outperforms the corresponding capacity in a message-specific setup by up to 19.5% for the symmetric setting.

The rest of the thesis is organized into four chapters. Chapter 2 presents the algorithms and performance bounds for pliable index coding. Chapter 3 and 4 present the applications to bandwidth aware recommender systems and the distributed computing in a pliable index coding framework. Chapter 5 presents beneficial results of content-type coding on large scale networks and lossy networks.

CHAPTER 2

Algorithms and Performance Bounds for Pliable Index Coding

2.1 Introduction

In this chapter, we formally present the pliable index coding problem that we talked about in Chapter 1.2.1. Compared with classical index coding [BK98, LS09, BKL10, CS08], pliable index coding assumes that clients are pliable and are happy to receive any new message they do not already have. The aim is to find an efficient way of broadcasting the messages over a noiseless channel such that all clients can be satisfied with the minimum number of transmissions.

Classical index coding requires in the worst case $\Omega(n)$ transmissions [Pee96, BBJ11] and requires almost surely $\Theta(n/\log(n))$ transmissions for random graph instances [GRW16]. In contrast, pliable index coding requires in the worst case $O(\log^2(n))$ [BF12, BF13] transmissions. The results imply that, if we realize that we need to solve a pliable index coding problem as opposed to the conventional index coding problem, we can be exponentially more efficient in terms of the number of transmissions. However, the pliable index coding problem is NP-hard [BF12], and thus a natural question is, whether we can efficiently realize these benefits through a polynomial-time algorithm.

In this chapter, we first derive an algebraic decoding criterion for linear pliable index coding, which can be used to determine the validity of a specific linear code for a problem instance. Leveraging this criterion, we design a deterministic polynomial-time algorithm, BinGreedy, to solve the pliable index coding problem. This algorithm achieves an upper bound of $O(\log^2(n))$ in terms of the number of transmissions, which matches the upper bound in [BF12].

Secondly, we extend the above algorithm to the multiple requests case where each client would like to recover t unkown messages instead of one. We analytically show that the new algorithm achieves an upper bound of $O(t\log(n) + \log^2(n))$ for the code length, which is tighter than the upper bound $O(t\log(n) + \log^3(n))$ in [BF13].

Thirdly, we construct specific instances to provide lower bounds on the required number of transmissions. We construct instances that require $\Omega(\log(n))$ transmissions and $\Omega(t+\log(n))$ transmissions for pliable index coding and the t-requests case, respectively. These lower bounds are within a $O(\log n)$ factor of the upper bounds.

We proceed to provide a probabilistic analysis over random graphs, where the side information sets are populated independently and randomly by messages for each client with a certain probability. We show that the required number of transmissions is almost surely $\Theta(\log(n))$, which again is exponentially better than the $\Theta(n/\log(n))$ transmissions required for index coding [GRW16].

Finally, we evaluate the deterministic algorithm performance through numerical experiments. We show that in some cases we can achieve up to 50% savings of transmissions over the previously proposed algorithm in [BF12].

The work presented in this chapter was published in [SF15, SF16b, SF16c].

2.2 Problem Formulation

We consider a system with one server and n clients. The server has m messages, represented by symbols in a finite field $b_1, b_2, \ldots, b_m \in \mathbf{F}_q$. Each client i has as side information a subset of the messages, indexed by $S_i \subseteq [m]$, and requires any new message (or t new messages for t-requests case) from the remaining unknown messages, termed request set and indexed by $R_i = [m] \backslash S_i$, where $|R_i| > 0$ (or $|R_i| \geq t$ for t-requests case).

The server first encodes the m original messages into K encoded messages $x_1, x_2, \ldots, x_K \in$

$\mathbf{F}_q$ and then makes broadcast transmissions of the encoded messages over a noiseless broadcast channel. Each client receives the broadcasted messages and then decodes them using her side information. We say that a client is *satisfied* if she can successfully recover one new message that she does not already have, or t unknown messages for t-requests case, referred to as *t-satisfied* or simply *satisfied* when it is clear from the context. Our goal of pliable index coding (or with t-requests) is to minimize the total number of transmissions K by designing the encoding and decoding scheme, such that all clients can be satisfied. For ease of exposition, we denote such a problem instance by $(m, n, \{R_i\}_{i\in[n]})$, or $(m, n, \{R_i\}_{i\in[n]}, t)$ for the t-requests case.

2.2.1 Encoding and Decoding

Formally, we can express the encoding and decoding processes as follows.

- *Encoding* is represented by an encoding function $f_{enc} : \mathbf{F}_q^m \to \mathbf{F}_q^K$, where K is the total number of transmissions or code length. The output of the encoding function $(x_1, x_2, \ldots, x_K) = f_{enc}(b_1, b_2, \ldots, b_m)$ are the K transmitted messages. We assume that the server has full knowledge of the side information sets for all clients, namely, the server knows R_i for all $i \in [n]$.

- *Decoding*, for client $i \in [n]$, is represented by a decoding function $\phi_{i,dec} : \mathbf{F}_q^K \times \mathbf{F}_q^{|S_i|} \to \mathbf{F}_q \times [m]$. The output $\phi_{i,dec}(\{x_k\}_{k\in[K]}, \{b_j\}_{j\in S_i})$ consists of a message in the request set R_i and its index. For the t-requests case, the decoding function is $\phi^t_{i,dec} : \mathbf{F}_q^K \times \mathbf{F}_q^{|S_i|} \to \mathbf{F}_q^t \times [m]^t$. The output $\phi^t_{i,dec}(\{x_k\}_{k\in[K]}, \{b_j\}_{j\in S_i})$ consists of t messages in the request set R_i and their indices.

We restrict the encoding and decoding schemes to be linear in the thesis. In this case, we can further express the encoding and decoding processes as follows.

- *Linear Encoding:* The k-th broadcast transmission x_k is a linear combination of $b_1, \ldots, b_m$, namely, $x_k = a_{k1}b_1 + a_{k2}b_2 + \ldots + a_{km}b_m$, where $a_{kj} \in \mathbf{F}_q$, $j \in [m]$, is the encoding coefficient. Therefore, we can interpret the number of transmissions, K, as the *code length* and the $K \times m$ coefficient matrix $\boldsymbol{A}$ with entries a_{kj} as the *coding matrix.* In matrix form, we can write

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{b}, \tag{2.1}$$

where $\boldsymbol{b}$ and $\boldsymbol{x}$ collect the original messages and encoded transmissions, respectively.

- *Linear Decoding:* Given $\boldsymbol{A}$, $\boldsymbol{x}$, and $\{b_j | j \in S_i\}$, the decoding process for client i is to solve the linear equation (2.1) to get a unique solution of b_j for some $j \in R_i$, or unique solutions $b_{j_1}, b_{j_2}, \dots, b_{j_t}$ for some $j_1, j_2, \dots, j_t \in R_i$ for the t-requests case. Clearly, client i can remove her side information messages, i.e., can create $x_k^{(i)} = x_k - \sum_{j \in S_i} a_{kj} b_j$ from the k-th encoded transmission. As a result, client i needs to solve the equations

$$\boldsymbol{A}_{R_i} \boldsymbol{b}_{R_i} = \boldsymbol{x}^{(i)}, \tag{2.2}$$

to retrieve any one message (or t messages) she does not have, where $\boldsymbol{A}_{R_i}$ is the sub-matrix of $\boldsymbol{A}$ with columns indexed by R_i; $\boldsymbol{b}_{R_i}$ is the message vector with elements indexed by R_i; and $\boldsymbol{x}^{(i)}$ is a K-dimensional column vector with the element $x_k^{(i)}$.

Our goal is to construct the coding matrix $\boldsymbol{A}$, so that the code length K is minimized.

2.2.2 Bipartite Graph Representation

We can represent a pliable index coding problem or its t-requests case using an undirected bipartite graph. On one side, a vertex corresponds to a message and on the other side a vertex corresponds to a client. We connect with edges clients to the messages they *do not* have [BF12], i.e., client i connects to the messages indexed by R_i. For instance, in the example in Fig. 2.1, $R_1 = \{1\}$ and $S_1 = \{2, 3\}$ for client 1; client 4 does not have (and would be happy to receive any of) b_1 and b_2. In this example, if the server transmits $x_1 = b_1 + b_2 + b_3$, $x_2 = b_2 + b_3$, and $x_3 = b_1 + b_2$, then we can write

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

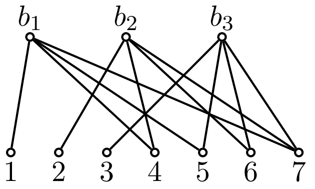


Figure 2.1: Pliable index coding instance with $m = 3, n = 7$.

and the decoding process for client 4 is to solve

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} x_1 - b_3 \\ x_2 - b_3 \\ x_3 \end{bmatrix}.$$

2.3 An Algebraic Decoding Criterion for Linear Pliable Index Coding

We here derive an algebraic criterion that determines whether a client can successfully decode some message given a coding matrix $\boldsymbol{A}$.

Recall that client i needs to solve the linear equations (2.2) in order to recover a new message. In traditional linear encoding and decoding processes, e.g., network coding, we often solve linear equations to get a unique solution for all elements of the message vector. A key difference in pliable index coding is that, we do not need to identify all the elements of the vector $\boldsymbol{b}_{R_i}$, but only require that any one variable b_j, $j \in R_i$ is recovered for client i to be satisfied. Thus we need to achieve a unique solution for one element of the message vector.

We use $\boldsymbol{a}_j$ to denote the j-th column of the matrix $\boldsymbol{A}$ and $\boldsymbol{A}_{R_i \setminus \{j\}}$ to denote a submatrix of $\boldsymbol{A}$ whose columns are indexed by R_i other than j. We also use $\text{span}\{\boldsymbol{A}_{R_i \setminus \{j\}}\}$ to denote the linear space spanned by columns of $\boldsymbol{A}$ indexed by R_i other than j, i.e., $\{\sum_{l \in R_i \setminus \{j\}} \lambda_l \boldsymbol{a}_l | \lambda_l \in \mathbf{F}_q\}$.

Lemma 1 (Decoding Criterion)**.** *In a pliable index coding problem* $(m, n, \{R_i\}_{i \in [n]})$, *given a* *coding matrix* $\boldsymbol{A}$*, client* i *can uniquely decode message* $j \in R_i$*, if and only if*

$$\boldsymbol{a}_j \notin \mathit{span}\{\boldsymbol{A}_{R_i \backslash \{j\}}\}. \tag{2.3}$$

Moreover, in the t-requests case $(m, n, \{R_i\}_{i\in[n]}, t)$*, given a coding matrix* $\boldsymbol{A}$*, client* i *can uniquely decode messages* $j_1, j_2, \ldots, j_t \in R_i$*, if and only if*

$$\boldsymbol{a}_{j_\tau} \notin \mathit{span}\{\boldsymbol{A}_{R_i \backslash \{j_\tau\}}\}, \ \mathit{for\ all}\ \tau \in [t]. \tag{2.4}$$

Proof. From linear algebra, we know that the set of solutions for (2.2) can be expressed as a specific solution plus a vector in the null space of $\boldsymbol{A}_{R_i}$, $\mathcal{N}(\boldsymbol{A}_{R_i})$:

$$\boldsymbol{b}^*_{R_i} + \boldsymbol{b}'_{R_i}, \tag{2.5}$$

where $\boldsymbol{b}^*_{R_i}$ is a specific solution for (2.2) and $\boldsymbol{b}'_{R_i}$ is an arbitrary vector in the null space $\mathcal{N}(\boldsymbol{A}_{R_i})$, i.e., $\boldsymbol{b}^*_{R_i}$ is any fixed vector that satisfies $\boldsymbol{A}_{R_i}\boldsymbol{b}^*_{R_i} = \boldsymbol{x}^{(i)}$ and $\boldsymbol{b}'_{R_i}$ is an arbitrary vector that satisfies $\boldsymbol{A}_{R_i}\boldsymbol{b}'_{R_i} = 0$. The requirement that client i can decode message b_j is equivalent to that $b'_j = 0$ for all $\boldsymbol{b}'_{R_i} \in \mathcal{N}(\boldsymbol{A}_{R_i})$, implying that client i can decode message b_j as the unique solution b^*_j.

We next argue that $b'_j = 0$ is equivalent to the proposed decoding criterion (2.3) or (2.4). We first note that $\boldsymbol{A}_{R_i}\boldsymbol{b}'_{R_i} = b'_j\boldsymbol{a}_j + \sum_{l\in R_i\backslash\{j\}} b_l\boldsymbol{a}_l = 0$ for any vector $\boldsymbol{b}'_{R_i} \in \mathcal{N}(\boldsymbol{A}_{R_i})$.

- *Necessity.* If $\boldsymbol{a}_j \notin \text{span}\{\boldsymbol{A}_{R_i\backslash\{j\}}\}$ is satisfied, then $b'_j = 0$ always holds; otherwise $b'_j\boldsymbol{a}_j + \sum_{l\in R_i\backslash\{j\}} b_l\boldsymbol{a}_l = 0$ for some nonzero b'_j implies that $\boldsymbol{a}_j$ can be expressed as a linear combination of $\boldsymbol{a}_l$, $l \in R_i\backslash\{j\}$, which contradicts the decoding criterion.

- *Sufficiency.* If $b'_j = 0$ always holds, then $\boldsymbol{a}_j \notin \text{span}\{\boldsymbol{A}_{R_i\backslash\{j\}}\}$ is satisfied; otherwise $\boldsymbol{a}_j$ can be expressed as a linear combination of $\boldsymbol{a}_l$, $l \in R_i\backslash\{j\}$ implying that $b'_j = 1$ is also possible, which contradicts the fact that $b'_j = 0$ always holds.

Therefore, we can get a unique solution for b_j if and only if any vector $\boldsymbol{b}'_{R_i}$ in $\mathcal{N}(A_{R_i})$ has a zero value in the element corresponding to j. We can then retrieve b_j by any linear equation solving methods for (2.2). □

For example, considering the instance and coding matrix in Fig. 2.1, we have $R_4 = \{1, 2\}$, $\boldsymbol{a}_1 \notin \text{span}\{\boldsymbol{a}_2\}$, and $\boldsymbol{a}_2 \notin \text{span}\{\boldsymbol{a}_1\}$, so client 4 can decode b_1 and b_2. Client 4 can decode b_2 by $b_2 = x_2 - b_3$ and b_1 by $b_1 = x_3 - b_2$.

2.4 Binary Field Greedy Algorithm for Pliable Index Coding

In this chapter, by leveraging the decoding criterion, we design a polynomial-time deterministic algorithm for pliable index coding that achieves a performance guarantee of $O(\log^2(n))$ in terms of code length. Our algorithm uses operations over the binary field[1] and follows a greedy approach. We first describe our algorithm, which we term BinGreedy, and then show the upper bound performance.

2.4.1 Algorithm Description

Our BinGreedy algorithm is described in Alg. 1. Intuitively, we decide which message we will try to serve to each client: we call *effective clients*, the clients that a specific message aims to satisfy (as we will define more formally in the following), and *effective degree*, the number of such clients each message has. We then create groups of messages that have approximately the same effective degree, and show that because of the regularity of the degree, by coding across only the messages within the group, we can satisfy at least a constant fraction of the effective clients in the group.

The algorithm operates in rounds. Each round has three phases: the sorting phase, the grouping phase and the greedy transmission phase. In the sorting phase, the algorithm sorts the message vertices in a decreasing order in terms of their effective degrees. In the grouping phase, we divide the messages into at most $\log n$ groups based on their effective degrees such that messages in the same group have similar effective degrees. In the transmission phase, to satisfy as many effective clients as possible, the algorithm encodes messages inside a group

[1] Here, we consider a message b_j to be an element in the finite field $\mathbf{F}_{2^\rho}$.

and makes two transmissions per group, thus in total we use at most $2\log n$ transmissions.

Before giving a detailed description of the algorithm, we first formally introduce the definition of *effective degree* of a message and its *effective clients* using the bipartite graph representation.

• *Effective degree* and *effective clients*: given a particular order of the message vertices $\pi = (j_1, j_2, \ldots, j_m)$, the effective degree of message b_{j_l} is defined as the number of b_{j_l}'s neighbors who do not connect with message $b_{j'}$, for any $j' = j_1, j_2, \ldots, j_{l-1}$. The neighbors that contribute to b_{j_l}'s effective degree are called effective clients of b_{j_l}. Let us denote by $N[j]$ the set of neighbors of message b_j and by $N[j_1, j_2, \ldots, j_{l-1}]$ the set $N[j_1] \cup N[j_2] \cup \ldots N[j_{l-1}]$. Formally, the effective clients of message b_{j_l} are defined as $N_\pi^\dagger[j_l] = N[j_l] \backslash N[j_1, j_2, \ldots, j_{l-1}]$ with respect to the order π. Correspondingly, the effective degree of message b_{j_l} is defined as $d_\pi^\dagger[j_l] = |N_\pi^\dagger[j_l]|$ with respect to π.

Note that the effective degree and effective clients for a message b_j may vary when we change the order of the message vertices. We will omit the subscript π when it is clear from the context. In our example in Fig. 2.1, given a message order b_1, b_2, b_3, the effective degrees and clients are $d^\dagger[1] = 4, N^\dagger[1] = \{1, 4, 5, 7\}$, $d^\dagger[2] = 2, N^\dagger[2] = \{2, 6\}$, and $d^\dagger[3] = 1, N^\dagger[3] = \{3\}$. Given a different order b_2, b_3, b_1, the effective degrees and clients are $d^\dagger[2] = 4, N^\dagger[2] = \{2, 4, 6, 7\}$, $d^\dagger[3] = 2, N^\dagger[3] = \{3, 5\}$, and $d^\dagger[1] = 1, N^\dagger[1] = \{1\}$.

In the following, we describe the detailed operations for the three phases in a round.

1) Sorting Phase. We sort the messages into a desired order so that the effective degrees of messages are non-increasing. To describe the procedure, we use the bipartite graph representation of pliable index coding (see Chapter 2.2). We denote by G the original bipartite graph representation of the pliable index coding instance, by $V(G)$ the vertex set of G, and by $V(G')$ the vertex set of any subgraph G' of G. For a vertex $j \in V(G)$, the set of neighbors of j is denoted by $N[j]$. For a vertex j in an induced subgraph G' of G, we define the neighbors of j restricted on subgraph G' as $N_{G'}[j] \triangleq N[j] \cap V(G')$ for all $j \in V(G')$.

• Step 1: We start from the original bipartite graph $G_1 = G$. Find a message vertex j_1 with the maximum degree (number of neighbors) in G_1, with ties broken arbitrarily. Thus we have $|N_{G_1}[j_1]| \geq |N_{G_1}[j]|$ for all $j \in [m] \backslash \{j_1\}$, where $N_{G_1}[j] = N[j]$.

- Step 2: Consider the subgraph G_2 induced by message vertices $[m]\backslash\{j_1\}$ and client vertices $[n]\backslash N[j_1]$. Find a message vertex j_2 with maximum degree in the subgraph G_2, with ties broken arbitrarily. That is, we have $|N_{G_2}[j_2]| \geq |N_{G_2}[j]|$ for all $j \in [m]\backslash\{j_1, j_2\}$, where $N_{G_2}[j] = N[j]\backslash N[j_1]$.

- Step l ($l = 3, \ldots, m$): Consider the subgraph G_l induced by messages $[m]\backslash\{j_1, j_2, \ldots, j_{l-1}\}$ and clients $[n]\backslash N[j_1, j_2, \ldots, j_{l-1}]$. Find a message vertex j_l with maximum degree in the subgraph G_l, with ties broken arbitrarily. That is, we have $|N_{G_l}[j_l]| \geq |N_{G_l}[j]|$ for all $j \in [m]\backslash\{j_1, j_2, \ldots, j_l\}$, where $N_{G_l}[j] = N[j]\backslash N[j_1, j_2, \ldots, j_{l-1}]$.

From the above sorting process, we notice that the effective degrees are $|N[j_1]|$ for message j_1, $|N[j_2]\backslash N[j_1]|$ for message j_2, ..., $|N[j_l]\backslash N[j_1, j_2, \ldots, j_{l-1}]|$ for j_l, etc. It is easy to see that the effective degrees of messages are in a non-increasing order.

2) Grouping Phase. We divide the message vertices into $\log(n)$ groups, namely $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{\log(n)}$, based on their effective degrees, such that for message vertex $j \in \mathcal{M}_g$, the effective degree satisfies $n/2^{g-1} \geq d^{\dagger}[j] > n/2^g$, for $g = 1, 2, \ldots, \log(n)$.

Given the sorting and grouping processes, we have the following property for any message j in group $\mathcal{M}_g$:

$$d^{\dagger}[j] > n/2^g \triangleq d^{(g)}/2, \text{ and } |N[j] \cap \mathcal{N}_g| \leq n/2^{g-1} \triangleq d^{(g)}, \tag{2.6}$$

where $\mathcal{N}_g$ is the set of all effective clients of the messages in $\mathcal{M}_g$, namely, $\mathcal{N}_g = \cup_{j' \in \mathcal{M}_g} N^{\dagger}[j']$. The second part holds because if $|N[j] \cap \mathcal{N}_g| > d^{(g)}$, message j, during the sorting and grouping phases, would have effective degree greater than $d^{(g)}$ and would have been assigned in an earlier group (with smaller g).

One possible sorting order and grouping for the example in Fig. 2.1 are: b_1, b_2, b_3 and $\mathcal{M}_1 = \{1\}$, $\mathcal{M}_2 = \{2\}$, $\mathcal{M}_3 = \{3\}$.

3) Transmission Phase. We make two transmissions for each message group $\mathcal{M}_g$, using a coding submatrix with 2 rows (one for each transmission). Initially, this submatrix is empty.

We sequentially visit each message vertex in $\mathcal{M}_g$ according to the sorting order and create a corresponding column of the coding submatrix, referred to as the coding vector. Hence, a total of $|\mathcal{M}_g|$ steps are carried out for group $\mathcal{M}_g$. At any step, we record the clients that can be satisfied when some message vertices are visited and associated coding vectors are added to the coding submatrix. When a new message vertex is visited, the associated coding vector is selected from $\{(1,0)^T, (0,1)^T, (1,1)^T\}$ such that the maximum number of clients in $\mathcal{N}_g$ can still be satisfied up to current step. In our example in Fig. 2.1, we can construct a coding matrix:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}, A_1 = \begin{bmatrix} 1\,0\,0 \\ 0\,0\,0 \end{bmatrix}, A_2 = \begin{bmatrix} 0\,1\,0 \\ 0\,0\,0 \end{bmatrix}, A_3 = \begin{bmatrix} 0\,0\,1 \\ 0\,0\,0 \end{bmatrix},$$

where every two rows represent the transmissions for a group.

Note that the coding matrix constructed by our algorithm may not be full rank. As such, it suffices to only select a row basis as the coding matrix.

2.4.2 An Illustrative Example

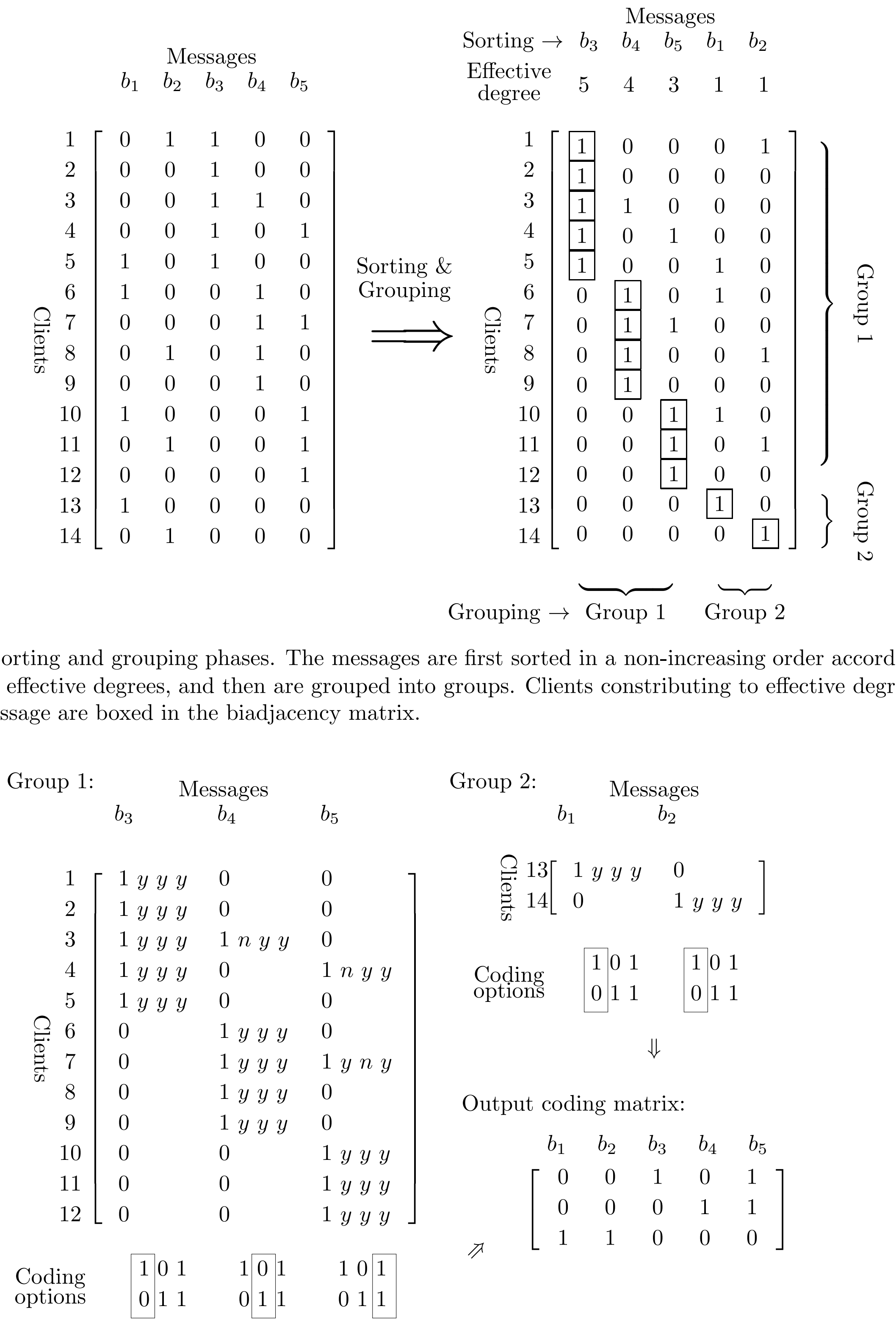
We now show how the algorithm works through an example. We consider the following problem instance represented by the biadjacency matrix[2] on the left hand side in Fig. 2.2 (a). In this biadjacency matrix, we have the number of messages $m = 5$, each represented by a column of the matrix, and the number of clients $n = 14$, each represented by a row of the matrix. The request sets are shown as adjacency relationship in the biadjacency matrix, i.e., the (i, j) entry is 1 if and only if client i does not have message j.

The sorting and grouping phases are shown in Fig. 2.2 (a). In the sorting phase, 5 messages are sorted in a non-increasing order according to their effective degrees, as shown on top of the matrix on the right hand side of Fig. 2.2 (a). We categorize these messages and their associated effective clients into 2 groups, such that the maximum effective degree

[2] For a bipartite graph $G(V_1 \cup V_2, E)$, the biadjacency matrix is a $(0, 1)$ matrix of size $|V_1| \times |V_2|$, whose (i, j) element equals 1 if and only if i connects j.

Algorithm 1 Binary Field Greedy Algorithm (BinGreedy)

```
Initialization: Set $\mathcal{N} = [n]$.
while $\mathcal{N} \neq \emptyset$ do
    Sorting and grouping of message vertices:
    Set $\mathcal{N}_{temp} = \mathcal{N}, \mathcal{M}_{temp} = [m]$.
    for $j = 1 : m$ do
        Find the message $j' \in \mathcal{M}_{temp}$ having the maximum number of neighbors in $\mathcal{N}_{temp}$, with
        ties broken arbitrarily.
        Put message $j'$ in the $j$-th position.
        Remove $j'$ from $\mathcal{M}_{temp}$ and all its neighbors from $\mathcal{N}_{temp}$.
    end for
    Group messages into $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_{\log(n)}$ message groups based on their effective degrees.
    Greedy transmission:
    for $g = 1 : \log(n)$ do
        Initialization: Set $\mathcal{N}_g = \cup_{j \in \mathcal{M}_g} N^{\dagger}[j]$ (effective clients neighboring to $\mathcal{M}_g$), $SAT = \emptyset$
        and $UNSAT = \emptyset$.
        for $j = 1 : |\mathcal{M}_g|$ do
            Assign a coding vector from $\{(1,0)^T, (0,1)^T, (1,1)^T\}$ to the $j$-th message in $\mathcal{M}_g$, denoted
            by $j^{(g)}$, such that the maximum number of clients in $\{i \in SAT | i \text{ is connected with } j^{(g)}\}$
            can still be satisfied, with ties broken arbitrarily.
            Move from $SAT$ to $UNSAT$ these unsatisfied clients in $\{i \in$
            $SAT | i \text{ is connected with } j^{(g)}\}$.
            Add clients in $N^{\dagger}[j^{(g)}]$ to $SAT$.
        end for
        Set coding vectors to be $(0,0)^T$ corresponding to messages in groups other than $g$.
        Remove clients in $SAT$ from $\mathcal{N}$ and their associated edges.
    end for
end while
```



(a) Sorting and grouping phases. The messages are first sorted in a non-increasing order according to their effective degrees, and then are grouped into groups. Clients contributing to effective degree of a message are boxed in the biadjacency matrix.

(b) Greedy transmission phase. For each group, coding vectors of length 2 are sequentially assigned to each message, so as to satisfy as many clients as possible at each step. At each step, the coding options are listed to check if a client can be satisfied (y) or not (n) so far, and selections are boxed.

Figure 2.2: Example of running BinGreedy algorithm in 1 round.

in a group is not more than twice the minimum effective degree in the group.

Fig. 2.2 (b) shows the greedy transmission phase for each group. In a group, we sequentially assign coding sub-vectors of length 2 to each message, such that a maximum number of clients can still be satisfied so far in the group. For example, in group 1, when we assign coding sub-vectors for message b_4, we find that $(1,0)^T$ can satisfy 4 clients and $(0,1)^T$ or $(1,1)^T$ can satisfy 5 clients, so we select $(0,1)^T$ (or $(1,1)^T$) as the coding sub-vector. The final coding matrix achieved by our BinGreedy algorithm is shown at the bottom on the right hand side in Fig. 2.2 (b).

Note that for this instance, one round of encoding is enough to satisfy all clients.

2.4.3 Algorithm Performance

To evaluate the worst case performance of our proposed algorithm in terms of the number of transmissions, we first prove the following lemma.

Lemma 2. *In Alg. 1, the greedy coding scheme can satisfy at least* $1/3$ *of the effective clients* $\mathcal{N}_g$ *in one round.*

Proof. Consider the bipartite subgraph induced by vertices $\mathcal{M}_g \cup \mathcal{N}_g$, i.e., the messages in the group $\mathcal{M}_g$ and their effective clients in $\mathcal{N}_g$. To construct the coding submatrix, at each step we sequentially visit a message vertex j in $\mathcal{M}_g$, following the sorted order, denoted by $1^{(g)}, 2^{(g)}, \ldots, m_g^{(g)} = |\mathcal{M}_g|$, and greedily decide which coding vector will become the $j-th$ column of the coding matrix. We say a message $b_j \in \mathcal{M}_g$ is *untouched* if it is not visited yet up to current step; we say a client $i \in \mathcal{N}_g$ is *untouched* if it is not connected with the visited messages up to current step. Up to a certain step, a client is either untouched or satisfied/unsatisfied by the current assignment. To capture the dynamic changes of satisfied/unsatisfied clients, we define two sets, SAT and $UNSAT$. Assume that up to some step, the algorithm has visited some messages and assigned the corresponding coding vectors. The first set, SAT, collects the clients connecting to messages that have already been visited, and are satisfied by the current assignment of coding vectors according to the criterion in Lemma 1, i.e., for each of these clients, i, given the r coding vectors assigned to messages connecting with i and visited by the algorithm so far, $\alpha_1, \alpha_2, \ldots, \alpha_r$, there exists one coding vector $\alpha_{j'}$ $(1 \leq j' \leq r)$ not in the span of the remaining coding vectors: $\alpha_{j'} \notin \text{span}\{\alpha_1, \ldots, \alpha_{j'-1}, \alpha_{j'+1}, \ldots, \alpha_r\}$. The second set, $UNSAT$, collects clients that are associated with messages already visited by the algorithm and cannot be satisfied by current coding vector assignments. Note that there may exist untouched clients in neither of these groups.

Initially, both SAT and $UNSAT$ are empty. We gradually add clients from $\mathcal{N}_g$ into these two sets as we go through the messages and assign coding vectors. Our first step is to add all $N^\dagger[1^{(g)}]$ (effective clients of the first message $1^{(g)}$ in $\mathcal{M}_g$) to SAT, since any non-zero vector satisfies the decoding criterion for only one message. At each step, some untouched clients may become satisfied, but some satisfied clients may also become unsatisfied. For example, assume a client is connected with 3 messages, 2 of which are visited and assigned coding vectors $(1,0)^T, (0,1)^T$, so the client is satisfied at this point. When the algorithm visits the third message and assigns to it a coding vector $(1,1)^T$, this client becomes unsatisfied as the decoding criterion no longer holds.

We will show that at each step, the number of clients who are moved from SAT to $UNSAT$ is at most $d^{(g)}/3$. Consider the step to assign a vector to message j in $\mathcal{M}_g$. Notice that when we assign a coding vector $(1,0)^T, (0,1)^T$, or $(1,1)^T$ to message j, only clients connecting with message j can be affected. We list possibilities for all the t_0 clients connected with j and satisfied (in SAT) at the beginning of this step:

• Case 1: Assume there are t_1 clients who connect with previously visited messages that are assigned one coding vector $(1,0)^T$ and some (perhaps none) coding vectors $(0,1)^T$. In this case, these clients can decode a new message corresponding to the coding vector $(1,0)^T$ since $(1,0)^T$ does not belong in the span of $(0,1)^T$ according to the decoding criterion. Similarly,

• Case 2: t_2 clients are satisfied by a $(1,0)^T$, several $(1,1)^T$.

• Case 3: t_3 clients are satisfied by a $(0,1)^T$, several $(1,0)^T$.

• Case 4: t_4 clients are satisfied by a $(0,1)^T$, several $(1,1)^T$.

• Case 5: t_5 clients are satisfied by a $(1,1)^T$, several $(0,1)^T$.

• Case 6: t_6 clients are satisfied by a $(1,1)^T$, several $(1,0)^T$.

If we assign a coding vector $(1,0)^T$ to message j, the t_3+t_6 clients can still be satisfied according to Lemma 1. Similarly, if we assign a coding vector $(0,1)^T$ or $(1,1)^T$ to message j, then the t_1+t_5 or t_2+t_4 clients can still be satisfied.

Note that $t_1+t_2+t_3+t_4+t_5+t_6 \geq t_0$ as there may be overlap among the 6 different cases (e.g., a client is satisfied by one $(1,0)^T$ and one $(0,1)^T$, so she is counted twice in both Case 1 and Case 3). Hence, at least one of t_3+t_6, t_1+t_5, t_2+t_4 should be no less than $t_0/3$; our greedy algorithm will move at most $2t_0/3$ clients from SAT to $UNSAT$. According to the property of our sorting and grouping in eq. (2.6), the number of j's neighbors who are connected with previously visited messages is at most $d^{(g)} - d^{\dagger}[j] < d^{(g)}/2$, and furthermore the number of j's neighbors in set SAT is a subset of these neighbors, resulting in $t_0 < d^{(g)}/2$. So at most $d^{(g)}/3$ clients will be moved from SAT to $UNSAT$ in each step.

On the other hand, we observe that for message j's effective clients (j's neighbors who are not connected with previously visited messages), any assignment of vectors $(1,0)^T, (0,1)^T$, or $(1,1)^T$ can satisfy them according to the decoding criterion. So, at least $d^{\dagger}[j] > d^{(g)}/2$ untouched clients are added to SAT. Completing the assignment steps, we can see that at most 2/3 clients in $\mathcal{N}_g$ cannot be satisfied by this scheme. □

We can now prove the following theorem.

Theorem 1. *For the BinGreedy algorithm in Alg. 1, the number of required transmissions is at most* $\frac{2}{log(1.5)}\log^2(n)$.

Proof. From Lemma 2, in each round, we have at most $\log(n)$ groups and $2\log(n)$ transmissions such that at least 1/3 clients are satisfied. This can be repeated for at most $\log(n)/\log(1.5)$ times, where the theorem follows. □

From the construction of our greedy algorithm, we can easily see that the algorithm runs in polynomial time $O(nm^2\log(n))$: there are at most $O(\log(n))$ rounds; for each round, the sorting and grouping phases take time $O(nm^2)$; and the greedy transmission phase in each round takes time $O(mn)$.

2.5 Binary Field Greedy Algorithm for t-requests Case

A straightforward method to solve the t-requests case is by repeatedly solving pliable index coding instances t times, resulting in an upper bound $O(t\log^2(n))$ of the number of broadcast transmissions. In [BF13] an upper bound of code length $O(t\log(n)+\log^3(n))$ is proved achievable. In this chapter, we modify our algorithm to adapt it to the t-requests case and prove that this modified algorithm, which we term BinGreedyT, can achieve a tighter upper bound $O(t\log(n)+\log^2(n))$.

2.5.1 Algorithm Description

The key difference of the BinGreedyT algorithm from the BinGreedy algorithm is the introduction of weights for clients and messages. The main idea behind this is that we would like all clients to receive approximately a similar number of new messages as the transmission proceeds, aiming to avoid that some clients receive too many new messages while others receive too few during the transmission process. For this purpose, we originally assign the same weights for all clients and exponentially reduce the weight of a client each time she can recover a new message. As a result, the algorithm operates in an efficient way which we show to achieve an upper bound $O(t\log(n)+\log^2(n))$.

We first introduce some new definitions. As the algorithm runs, we say that at some point a client is τ-satisfied if she can decode τ unknown messages in R_i. The ultimate goal of our algorithm is to let all clients to be t-satisfied. We again use the bipartite graph representation and denote by $N[j]$ the set of neighbors of message vertex j.

- *Weights of clients, w_i:* we associate a weight $0\le w_i\le 1$ with each of the n clients. Initially, we set $w_i=1$ for all client $i\in[n]$. This weight will be updated over time as the algorithm is being carried out.

- *Weights of messages, $w[j]$:* the weight corresponding to a message vertex b_j is the summation of the weights of b_j's neighbors $N[j]$, i.e., $w[j]=\sum_{i\in N[j]} w_i$.

- *Weights of messages restricted on a subgraph or a client subset, $w_{G'}[j]$ or $w_{N'}[j]$:* given a subgraph G' of G (or a subset of clients $\mathcal{N}' \subseteq [n]$), the weight of a message j restricted on the subgraph G' (or on the client subset $\mathcal{N}'$) is the summation of the weights of b_j's neighbors in the subgraph G' (or in the client subset $\mathcal{N}'$), denoted by $w_{G'}[j] = \sum_{i \in N[j] \cap V(G')} w_i$ (or $w_{\mathcal{N}'}[j] = \sum_{i \in N[j] \cap \mathcal{N}'} w_i$).

• *Effective weights and effective neighbors of messages, $w^\dagger[j]$ and $N^\dagger[j]$:* given a particular order of the message vertices $\pi = (j_1, j_2, \ldots, j_m)$, the effective weight of message b_{j_l} is defined as the sum of weights of b_{j_l}'s neighbors who do not connect with message $b_{j'}$, for any $j' = j_1, j_2, \ldots, j_{l-1}$. These neighbors that contribute to b_{j_l}'s effective weights are called effective clients of b_{j_l}. Let us denote by $N[j_1, j_2, \ldots, j_{l-1}]$ the set of neighbors $N[j_1] \cup N[j_2] \cup \ldots N[j_{l-1}]$. Formally, the effective clients of message b_{j_l} are defined as $N^\dagger_\pi[j_l] = N[j_l] \backslash N[j_1, j_2, \ldots, j_{l-1}]$ with respect to the order π. Correspondingly, the effective weights of message b_{j_l} is defined as $w^\dagger_\pi[j_l] = \sum_{i \in N^\dagger_\pi[j_l]} w_i$ with respect to π. Again, whenever the order π is clear from the context, we will omit the subscripts of the effective weights and effective neighbors.

We next describe the algorithm in Alg. 2. The algorithm operates again in rounds and each round has the same three phases. There are two differences here: one is that the sorting, grouping, and transmissions are based on the effective weights of messages, instead of effective degrees; the other is that we have messages categorized into $\log(n) + 1$ groups, with an additional group to gather messages with "small" weights and we do not encode messages within this group when making transmissions. In every $2\log(n)$ transmissions, we want to make sure that clients worth a certain fraction of weight can decode a new message, such that the total weight of clients in the system is decreasing at a fixed ratio during $2\log(n)$ transmissions. This will result in the claimed performance.

1) Sorting Phase. In the sorting phase, we use a similar technique as in Chapter 2.4 to sort the messages into a non-increasing order according to their effective weights instead of their effective degree.

2) Grouping Phase. Let us denote by $W = w[1]$ the maximum weight of the messages. We divide the message vertices into $\log(n)+1$ groups, namely $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{\log(n)}, \bar{\mathcal{M}}$, based on their effective weights with respect to the above order. For the first $\log(n)$ groups, a message vertex j is in groups $\mathcal{M}_g$, if and only if the effective weights satisfies $W/2^{g-1} \geq w^{\dagger}[j] > W/2^g$. For the remaining messages with "small" weights, i.e., no more than W/n, we put them into the last group $\bar{\mathcal{M}}$. We say a client i in group g if it contributes to the effective weight of a message vertex in group g. The set of clients in group g is denoted by $\mathcal{N}_g$.

According to the above sorting and grouping processes, we have the following property for the message j in group $\mathcal{M}_g$ $(g = 1, 2, \ldots, \log(n))$:

$$w^{\dagger}[j] > W/2^g, \text{ and } \sum_{i \in N[j] \cap \mathcal{N}_g} w_i \leq W/2^{g-1}, \tag{2.7}$$

where $\mathcal{N}_g = \cup_{j' \in \mathcal{M}_g} N^{\dagger}[j']$. The summation term $\sum_{i \in N[j] \cap \mathcal{N}_g} w_i$ in the second part can be seen as the "weight of message j restricted on client set $\mathcal{N}_g$". This holds because otherwise the message j will be assigned in a group less than g in the sorting and grouping phases.

3) Transmission Phase. In the transmission phase, we ignore the last group $\bar{\mathcal{M}}$ and make two transmissions for each message group $\mathcal{M}_g$ $(g = 1, 2, \ldots, \log(n))$, using a coding submatrix with 2 rows (one for each transmission). We sequentially create this submatrix by visiting each of the messages in group $\mathcal{M}_g$, according to the sorting order, and adding for each message one column to the coding submatrix (we refer to this column as the coding vector associated with this message). So in total we have $|\mathcal{M}_g|$ steps for group $\mathcal{M}_g$. At each step, we select each coding vector to be one in the set $\{(1,0)^T, (0,1)^T, (1,1)^T\}$, such that it can satisfy the maximum weight of clients in $\mathcal{N}_g$ up to the current step.

After a round of at most $2\log(n)$ transmissions, if a client i can decode one new message, we reduce the weight by a half: $w_i \rightarrow \frac{w_i}{2}$, and add one of her decoded messages in the side-information set S_i. If this weight equals to $1/2^t$, i.e., client i is t-satisfied, we remove this vertex and its associated edges from the graph. We repeat the process until all clients are t-satisfied.

From the described procedure, it follows that the BinGreedyT algorithm reduces to the BinGreedy algorithm for the $t = 1$ case.

Algorithm 2 Binary Field Greedy Algorithm for t-requests (BinGreedyT)

```
Initialization: Set $\mathcal{N} = [n]$, $w_i = 1$ for all $i \in [n]$.
while $\mathcal{N} \neq \emptyset$ do
    Sorting:
    Set $\mathcal{N}_{temp} = \mathcal{N}, \mathcal{M}_{temp} = [m]$.
    for $j = 1 : m$ do
        Find the message $j' \in \mathcal{M}_{temp}$ having the maximum weight of neighbors in $\mathcal{N}_{temp}$, i.e.,
        $j' = \arg\max_{j'' \in \mathcal{M}_{temp}} \sum_{i \in \mathcal{N}_{temp} \cup N[j'']} w_i$, with ties broken arbitrarily.
        Put message $j'$ in the $j$-th position.
        Remove $j'$ from $\mathcal{M}_{temp}$ and all its neighbors from $\mathcal{N}_{temp}$.
    end for
    Grouping: Group messages into $\log(n) + 1$ groups based on their effective weights.
    Set $W = w[1]$.
    For the first $\log(n)$ groups, put messages whose effective weights are between $W/2^g$ and
    $W/2^{(g-1)}$ into group $g$. Put the remaining messages into the last group. Then we have the
    groups: $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{\log(n)}, \bar{\mathcal{M}}$
    Greedy transmission:
    for $g = 1 : \log(n)$ do
        Initialization: Set $\mathcal{N}_g = \cup_{j \in \mathcal{M}_g} N^{\dagger}[j]$ (effective clients neighboring to $\mathcal{M}_g$), $SAT = \emptyset$
        and $UNSAT = \emptyset$.
        for $j = 1 : |\mathcal{M}_g|$ do
            Assign a coding vector from $\{(1,0)^T, (0,1)^T, (1,1)^T\}$ to the $j$-th message in $\mathcal{M}_g$, denoted
            by $j^{(g)}$, such that the maximum weight of clients in $\{i \in SAT | i \text{ is connected with } j^{(g)}\}$
            can still be satisfied, with ties broken arbitrarily.
            Move from $SAT$ to $UNSAT$ these unsatisfied clients in $\{i \in$
            $SAT | i \text{ is connected with } j^{(g)}\}$.
            Add clients in $N^{\dagger}[j^{(g)}]$ to $SAT$.
        end for
        Set coding vectors to be $(0,0)^T$ corresponding to messages in groups other than $g$.
        Update clients' weights in $SAT$: $w_i = w_i/2$ for all $i \in SAT$; add one message $b_{j'}$ that $i$
        can decode to her side information set $S_i$ and remove this edge between $i$ and $b_{j'}$.
        For all $i \in SAT$, check if $i$ is $t$-satisfied:
        if $w_i = 1/2^t$ then
            Remove client $i$ from $\mathcal{N}$ and the associated edges.
        end if
    end for
end while
```

2.5.2 Algorithm Performance

We aim to show that the above described algorithm has a performance guarantee upper bounded by $O(t\log(n) + \log^2(n))$. We first show that in each round, after the $O(\log(n))$ transmissions, the total weight of clients is at most $\frac{11}{12}$ of that before the $O(\log(n))$ transmissions, denoted by W_T. This implies that the weight is exponentially decreasing, hence, as shown later, we can argue that at most $O(\log(n) + t)$ rounds are needed.

Lemma 3. *The sum of clients' weights in the first* $\log(n)$ *groups is at least* $1/2$ *of the total weight* W_T*, i.e.,* $\sum_{g=1}^{\log(n)} \sum_{i\in\mathcal{N}_g} w_i \geq W_T/2$.

Proof. To see this, recall that the maximum weight of a message is W. According to our sorting and grouping phases in Alg. 2, after $\log(n)$ groups, the maximum weight of a message is at most $W/2^{\log(n)} = W/n$. Then the total weight of clients in group $\bar{\mathcal{M}}$ is at most $\frac{W}{n}n = W$. This means that the sum of clients' weights in the first $\log(n)$ groups is at least $W_T/2$. □

Consider the subgraph induced by vertices $\mathcal{M}_g \cup \mathcal{N}_g$ corresponding to the transmissions of group $\mathcal{M}_g$ ($g = 1, 2, \log(n)$) in a certain round. Similar to Alg. 1, at each step, we sequentially assign to a message a coding vector. We say a client $i \in \mathcal{N}_g$ is *untouched* if it is not connected with the visited messages up to current step. We introduce two sets SAT and $UNSAT$ to dynamically evaluate whether each client is satisfied or not up to the current step (by only considering messages visited up to now and disregarding all unvisited messages), so as to satisfy the maximum weight of clients up to now. Assume the effective weight of a message j in $\mathcal{M}_g$ is between $w^{(g)}/2$ and $w^{(g)}$. Using the property described in eq. (2.7), with the same technique as in Chapter 2.4, we can show that at each step the weight of clients who are moved from SAT to $UNSAT$ is at most $w^{(g)}/3$. For completeness, we show the proof as follows.

Initially, both SAT and $UNSAT$ are empty. We gradually add clients from $\mathcal{N}_g$ into these two sets as we go through the messages and assign coding vectors. Our first step is to add all $N^{\dagger}[1^{(g)}]$ (the effective neighbors of the first message in $\mathcal{M}_g$) to SAT, since any non-zero vector satisfies the decoding criterion for only one message. At each step, some additional clients may become satisfied, but some satisfied clients may also become unsatisfied.

Assume the effective weight of a message j in $\mathcal{M}_g$ is between $w^{(g)}/2$ and $w^{(g)}$. We will show that at each step, the weight of clients who are moved from SAT to $UNSAT$ is at most $w^{(g)}/3$. Consider the step for assigning coding vector to message j. Notice that when we assign a coding vector $(1,0)^T$, $(0,1)^T$, or $(1,1)^T$ to message j, only clients connecting with message j can be affected. We list possibilities for all the clients, with total weight h, that are connected with j and satisfied (in SAT) at the beginning of this step:

- Case 1: Assume there are weight h_1 worth of clients who connect with previously visited messages that are assigned one coding vector $(1,0)^T$ and some (perhaps none) coding vectors $(0,1)^T$. In this case, these clients can decode a new message corresponding to the coding vector $(1,0)^T$ since $(1,0)^T$ does not belong in the span of $(0,1)^T$. Similarly,
- Case 2: weight h_2 worth of clients are satisfied by a $(1,0)^T$, several $(1,1)^T$.
- Case 3: weight h_3 worth of clients are satisfied by a $(0,1)^T$, several $(1,0)^T$.
- Case 4: weight h_4 worth of clients are satisfied by a $(0,1)^T$, several $(1,1)^T$.
- Case 5: weight h_5 worth of clients are satisfied by a $(1,1)^T$, several $(0,1)^T$.
- Case 6: weight h_6 worth of clients are satisfied by a $(1,1)^T$, several $(1,0)^T$.

When we assign a coding vector $(1,0)^T$ to message j, the h_3+h_6 worth of clients can still be satisfied according to the decoding criterion. Similarly, if we assign a coding vector $(0,1)^T$ or $(1,1)^T$ to message j, then the weight h_1+h_5 or h_2+h_4 of clients can still be satisfied.

Note that $h_1+h_2+h_3+h_4+h_5+h_6 \geq h$ as there may be overlap among the 6 different cases (e.g., a client is satisfied by one $(1,0)^T$ and one $(0,1)^T$, so she is counted twice in both Case 1 and Case 3). Hence, at least one of h_3+h_6, h_1+h_5, h_2+h_4 should be no less than $h/3$; our greedy algorithm will move at most $2h/3$ worth of clients from SAT to $UNSAT$.

According to the property of our sorting and grouping phases in eq. (2.7), the weight of j's neighbors who are connected with previously visited messages is at most $w^{(g)} - w^{\dagger}[j] < w^{(g)}/2$; otherwise, j will be grouped into another group with index smaller than g, since j's effective weight would be larger than $w^{\dagger}[j] + w^{(g)}/2 > w^{(g)}$ when performing the sorting process. Furthermore, the number of j's neighbors in set SAT is a subset of these neighbors, resulting in $h < w^{(g)}/2$. So at most $w^{(g)}/3$ clients will be moved from SAT to $UNSAT$ in each step.

On the other hand, we observe that for message j's effective clients (j's neighbors who are not connected with previously visited messages), any assignment of vectors $(1,0)^T, (0,1)^T$, or $(1,1)^T$ can satisfy them once according to the decoding criterion. Hence, at least $w^{\dagger}[j] > w^{(g)}/2$ worth of untouched new clients are added to SAT.

Completing the assignment steps, we can see that clients worth at most $2/3$ weight in $\mathcal{N}_g$ cannot be satisfied by this coding scheme. Therefore, in a round, clients who can decode one new message count for at least $\frac{1}{3}\frac{1}{2} = \frac{1}{6}$ the total weight W_T. According to our weight updating rule that the weights of these clients will be reduced by at least a half: $w_i \rightarrow \frac{w_i}{2}$ (or $w_i \rightarrow 0$ if t-satisfied), resulting in a $\frac{1}{12}$ weight decreasing in total. Or equivalently, the total weight after one round is at most $\frac{11W_T}{12}$.

Therefore, we have the following theorem.

Theorem 2. *For the BinGreedyT algorithm in Alg. 2, the number of required transmissions is at most $O(t\log(n) + \log^2(n))$.*

Proof. From the above argument, after each round, we have at most $2\log(n)$ transmissions such that the remaining weight becomes at most $11/12$ of that before this round. Initially, the total weight is n. Hence, after $O(t + \log(n))$ rounds, the total weight is no more than $n(\frac{11}{12})^{O(t+\log(n))} \leq 1/2^t$. Since the weight for a client who is not t-satisfied is at least $1/2^{t-1} > 1/2^t$, all clients are t-satisfied after $O(t+\log(n))$ rounds of transmissions. The upper bound is proved. □

Note that the time complexity of this algorithm is bounded by $O((t + \log(n))nm^2)$: we need at most $O(t + \log(n))$ rounds in the algorithm, and in each round, the algorithm takes $O(nm^2)$ time to perform sorting and grouping and takes $O(nm)$ time to perform greedy transmission.

2.6 Lower Bounds

In this chapter, we provide instances for pliable index coding and t-requests case that require at least $\Omega(\log(n))$ and $\Omega(t+\log(n))$ transmissions, respectively.

2.6.1 A Lower Bound for Pliable Index Coding

To show a lower bound, we consider the following pliable index coding instances that we term *complete instances*, and define as follows. In a complete instance, we have $n = 2^m - 1$. The request set R_i is the i-th element of the set $2^{[m]}\backslash\emptyset$, where $2^{[m]}$ is the power set of $[m]$. An example of the complete instance with $m = 3$ is shown in Fig. 2.1.

Theorem 3. *In a complete instance $(m, n, \{R_i\}_{i\in[n]})$, the optimal number of transmissions is $\Omega(\log(n))$.*

Proof. Obviously, we can trivially satisfy all clients with $m = \log(n)$ transmissions, where each b_j is sequentially transmitted once. We argue that we cannot do better by using induction. We will prove that the rank of the coding matrix $\boldsymbol{A}$ needs to be at least m for the clients to be satisfied according to Lemma 1. Let J denote a subset of message indices; for the complete instance, Lemma 1 needs to hold for any subset $J \subseteq [m]$.

- For $|J| = 1$, to satisfy the clients who miss only one message, no column of the coding matrix $\boldsymbol{A}$ can be zero. Otherwise, if for example, column j_1 is zero, then the client who only requests message b_{j_1} cannot be satisfied. So $\text{rank}(\boldsymbol{A}_J) = 1$ for $|J| = 1$.

- Similarly, for $|J| = 2$, any two columns of the coding matrix must be linearly independent. Otherwise, if for example, columns j_1 and j_2 are linearly dependent, then $\boldsymbol{a}_{j_1} \in \text{span}\{\boldsymbol{a}_{j_2}\}$ and $\boldsymbol{a}_{j_2} \in \text{span}\{\boldsymbol{a}_{j_1}\}$, and the clients who only miss messages b_{j_1} and b_{j_2} cannot be satisfied. So $\text{rank}(\boldsymbol{A}_J) = 2$.

- Suppose we have $\text{rank}(\boldsymbol{A}_J) = l$ for $|J| = l$. For $|J| = l+1$, we can see that if all clients who only miss $l+1$ messages can be satisfied, then for some $j \in J$, we have $\boldsymbol{a}_j \notin \text{span}\{\boldsymbol{A}_{J\backslash\{j\}}\}$ according to Lemma 1. Therefore, $\text{rank}(\boldsymbol{A}_J) = \text{rank}(\boldsymbol{a}_j) + \text{rank}(\boldsymbol{A}_{J\backslash\{j\}}) = 1 + l$.

Therefore, to satisfy all the clients, the rank of the coding matrix $\boldsymbol{A}$ is m, resulting in $K \geq m$, from which the result follows. □

From Theorem 3, we have two observations: 1) We note that the upper bound is $O(\log^2(n))$ and the lower bound is $\Omega(\log(n))$, which shows that the upper bound is almost tight (i.e., in the order of $\mathit{polynomial}(\log(n))$); 2) If we apply our BinGreedy algorithm for the complete instance, we achieve a code length of $\log(n)$ as well, since we can divide the messages into $\log(n)$ groups, each consisting of one message.

2.6.2 A Lower Bound for t-requests Case

We again use complete instances to derive a lower bound for the t-requests case. Note that the complete instance for t-requests case needs to satisfy $|R_i| \geq t$ for all $i \in [n]$, so we add $t-1$ dummy messages to the complete instance for $t = 1$ case. Using the bipartite graph representation, the complete instance for t-requests is as follows. There are m messages and $n = 2^{m-t+1} - 1$ clients. We divide the messages into 2 types. The first $\log(n+1) \triangleq m_1$ messages are the Type-1 messages and the remaining $t - 1 \triangleq m_2$ messages are the Type-2 messages. All n clients are connected with all the Type-2 messages. We denote by $J_i \subseteq [m_1]$ the set of Type-1 messages a client i is connected to, i.e., J_i is the set of indices of Type-1 messages that i requires. Each J_i of the n clients is a unique subset of $[m_1]$ except the empty set. Note that there are in total $2^{m_1} - 1 = n$ such unique subsets.

Theorem 4. *In a complete instance $(m, n, \{R_i\}_{i\in[n]}, t)$, the optimal number of transmissions is $\Omega(t + \log(n))$.*

Proof. Clearly, $m = \log(n+1) + t - 1$ transmissions are enough to satisfy all clients. So we only show at least $\Omega(t + \log(n))$ transmissions are needed.

By abuse of notation, let us denote by $1^{(1)}, 2^{(1)}, \ldots, m_1^{(1)}$ and $1^{(2)}, 2^{(2)}, \ldots, m_2^{(2)}$ the indices of Type-1 and Type-2 messages and by $[1^{(1)} : m_1^{(1)}]$ and $[1^{(2)} : m_2^{(2)}]$ the sets of these two types of messages.

Suppose the coding matrix for a t-requests case problem is $\mathbf{A}$. We denote by $\mathbf{A}_{J\cup[1^{(2)}:m_2^{(2)}]}$

the submatrix of $\mathbf{A}$ consisting of columns indexed by $J \cup [1^{(2)} : m_2^{(2)}]$, where $J \subseteq [1^{(1)} : m_1^{(1)}]$ is a subset of indices of Type-1 messages. We will use induction to prove that the rank of the coding matrix $\mathbf{A}$ needs to be at least m for all the clients to be t-satisfied according to the decoding criterion. In the complete instance, the decoding criterion needs to hold for all clients, or for all $|J| = 1, 2, \ldots, m_1$.

For $J \subseteq [1^{(1)} : m_1^{(1)}]$ and $|J| = 1$, i.e., to satisfy the clients who miss only one Type-1 message, we need $\text{rank}(\boldsymbol{A}_{J\cup[1^{(2)}:m_2^{(2)}]}) = t$. Since otherwise, for example if the only column $j_1 \in [1^{(1)} : m_1^{(1)}]$ and all $t-1$ columns in $[1^{(2)} : m_2^{(2)}]$ are not linearly independent, then the clients who requests messages $\{j_1\} \cup [1^{(2)} : m_2^{(2)}]$ cannot be t-satisfied according to the decoding criterion. So $\text{rank}(\boldsymbol{A}_{J\cup[1^{(2)}:m_2^{(2)}]}) = t$ for all $|J| = 1$.

Assume we have $\text{rank}(\boldsymbol{A}_{J\cup[1^{(2)}:m_2^{(2)}]}) = l + t - 1$ for all $J \subseteq [1^{(1)} : m_1^{(1)}]$ with $|J| = l$. For $J \subseteq [1^{(1)} : m_1^{(1)}]$ and $|J| = l + 1$, we can see that according to the induction hypothesis, $\text{rank}(\boldsymbol{A}_{J\cup[1^{(2)}:m_2^{(2)}]}) \geq l + t - 1$. If $\text{rank}(\boldsymbol{A}_{J\cup[1^{(2)}:m_2^{(2)}]}) = l + t - 1$, then for any column $j \in J$, $\boldsymbol{a}_j \in \text{span}\{\boldsymbol{A}_{J\cup[1^{(2)}:m_2^{(2)}]\backslash\{j\}}\}$, since columns in $J \cup [1^{(2)} : m_2^{(2)}]\backslash\{j\}$ consist of a basis for this submatrix from the induction hypothesis. Hence, b_j (for any $j \in J$) cannot be decoded by the client who is only connected with $J \cup [1^{(2)} : m_2^{(2)}]$. This client can decode at most $t - 1$ messages and cannot be t-satisfied. As a result, $\text{rank}(\boldsymbol{A}_{J\cup[1^{(2)}:m_2^{(2)}]}) = l + t$, from which the result follows. □

2.7 Pliable Index Coding Over Random Graphs

We use a bipartite graph described in Chapter 2.2 to represent a problem instance. Here, we consider the random problem instance represented by a random bipartite graph $B(m, n, p)$ [Bol13], where there are m messages and n clients, and there is an edge between client i and message j with probability p, i.e., $\Pr\{j \in R_i\} = p$. We aim to calculate the "average" code length, or with high probability what is the required number K of transmissions. We say that a random problem instance $B(m, n, p)$ *almost surely* needs a code length of $K(m, n, p)$ if the probability that the code length is $K(m, n, p)$ tends to 1 as m and n tend to infinity. Next, we show that a random graph $B(m, n, p)$ *almost surely* requires a code length of $\Theta(\log(n))$.

2.7.1 A Lower Bound

To prove a lower bound on K, we introduce the concept of a *coding structure*, which is a collection of $K \times m$ matrices $\boldsymbol{A}$ with elements in a finite field $\mathbf{F}_q$ that satisfy a set of properties. Formally, a *coding structure* $S(J^{(1)}, J^{(2)}, J^{(3)})$, or shortly S, is defined as $S(J^{(1)}, J^{(2)}, J^{(3)}) \triangleq \{\boldsymbol{A} \in \mathbf{F}_q^{K\times m} | \boldsymbol{A}$ satisfies Properties (1) (2) (3)$\}$, where $J^{(1)}, J^{(2)}, J^{(3)} \subseteq [m]$ are disjoint subsets of message indices, $|J^{(1)}| + |J^{(2)}| = K$, $|J^{(2)}| = |J^{(3)}|$, and the properties are listed as follows.

Property.

(1) Column vectors indexed by $J^{(1)}$ and $J^{(2)}$ contain a column basis of matrix $\boldsymbol{A}$.

(2) For any column $j' \in J^{(1)}$, the corresponding column vector is not in the linear space spanned by other column vectors of matrix $\boldsymbol{A}$, i.e., $\boldsymbol{a}_{j'} \notin span\{\boldsymbol{a}_j | j \in [m]\backslash\{j'\}\}$.

(3) For any column $j'' \in J^{(2)} \cup J^{(3)}$, the corresponding column vector is in the linear space spanned by other column vectors indexed by $J^{(2)} \cup J^{(3)}$, i.e., $\boldsymbol{a}_{j''} \in span\{\boldsymbol{a}_j | j \in J^{(2)} \cup J^{(3)}\backslash\{j''\}\}$.

Consider a specific $K \times m$ coding matrix $\boldsymbol{A}$. We next describe a procedure that maps the matrix $\boldsymbol{A}$ (in a non-unique way) to some coding structure $S(J^{(1)}, J^{(2)}, J^{(3)})$. Conversely, in a coding structure, it is easy to construct some matrix as the coding matrix. Thus, if we denote the set of all $K \times m$ coding matrices by $\mathcal{A}$ and the union of all coding structures by $\mathcal{S} = \cup S(J^{(1)}, J^{(2)}, J^{(3)})$, it is easy to see that $\mathcal{S} = \mathcal{A}$.

Mapping Procedure In the following, we will call the columns in $J^{(1)}$, $J^{(2)}$ and $J^{(3)}$ Type-1, Type-2 and Type-3 columns, respectively. We will use the notation $K_1 = |J^{(1)}|$, $K_2 = |J^{(2)}|$, $K_3 = |J^{(3)}|$. We will show that we can select $J^{(1)}$, $J^{(2)}$ and $J^{(3)}$ so that $K_1 + K_2 = K$, $K_2 = K_3$ and properties (1)-(3) are satisfied. Note that a matrix $\boldsymbol{A}$ could be mapped to multiple structures, since there may exist different choices for selecting the columns in $J^{(1)}$, $J^{(2)}$ and $J^{(3)}$.

- In the coding matrix $\boldsymbol{A}$, find an arbitrary column basis, i.e., a maximum number of linearly independent column vectors. There are at most K such columns and without loss of generality, we assume these columns are indexed by $1, 2, \ldots, K'$, where $K' \leq K$.

• We categorize all m column vectors into 3 groups: 2 groups for these K' basis column vectors and a third group for the remaining $m - K'$ column vectors.

– Group 1: $\boldsymbol{A}^{(1)} = \{\boldsymbol{a}_{j_1} | j_1 \in [K'], \boldsymbol{a}_{j_1} \notin \text{span}\{\boldsymbol{a}_j | j \in [m] \backslash \{j_1\}\}\}$. Group 1 consists of column vectors that are not in the linear space spanned by all other column vectors of matrix $\boldsymbol{A}$. We assume $K_1 \leq K'$ such vectors, and without loss of generality, we assume these vectors in Group 1 are indexed by $1, 2, \ldots, K_1$. These are the Type-1 columns.

– Group 2: $\boldsymbol{A}^{(2)} = \{\boldsymbol{a}_{j_2} | j_2 \in [K'], \boldsymbol{a}_{j_2} \in \text{span}\{\boldsymbol{a}_j | j \in [m] \backslash \{j_2\}\}\}$. Group 2 consists of column vectors that are in the linear space spanned by all other column vectors of matrix $\boldsymbol{A}$. We assume $K_2 = K' - K_1$ such vectors, and without loss of generality, we assume these vectors in Group 2 are indexed by $K_1 + 1, K_1 + 2, \ldots, K_1 + K_2 = K'$. These are the Type-2 columns.

– Group 3: $\boldsymbol{A}^{(3)} = \{\boldsymbol{a}_{j_3} | j_3 \notin [K']\}$. Group 3 consists of the remaining $m - K'$ column vectors.

• We select and label K_3 columns in Group 3 as Type-3 columns as follows. We consider the submatrix of $\boldsymbol{A}$ from removing all K_1 columns in Group 1. Initially, we mark all K_2 columns in Group 2 as *active* and we will repeatedly *deactivate* them in the following steps.

1) We pick an arbitrary non-zero vector $\boldsymbol{a}_j$ from Group 3.

2) Label vectors or discard them according to the following rule. We observe that after removing the first K_1 columns, the K_2 column vectors in Group 2 are a basis for the remaining $K \times (m - K_1)$ submatrix. Then the vector $\boldsymbol{a}_j$ that is picked up in Step 1) can be uniquely represented by a linear combination of these basis vectors in Group 2, i.e., $\boldsymbol{a}_j = \lambda_{K_1+1}\boldsymbol{a}_{K_1+1} + \lambda_{K_1+2}\boldsymbol{a}_{K_1+2} + \ldots + \lambda_{K_1+K_2}\boldsymbol{a}_{K_1+K_2}$. Here we can consider $(\lambda_{K_1+1}, \lambda_{K_1+2}, \ldots, \lambda_{K_1+K_2})$ as coordinates under this basis.

Using this linear expansion for $\boldsymbol{a}_j$, we consider the basis vectors in Group 2 that correspond to the non-zero coordinates, i.e., $\boldsymbol{A}^{*(2)} = \{\boldsymbol{a}_{j_2} \in \boldsymbol{A}^{(2)} | \lambda_{j_2} \neq 0\}$. If no vectors in $\boldsymbol{A}^{*(2)}$ are marked *active*, then remove column j without labeling it. If any of these basis vectors is marked *active*, then label the column j as Type-3 column, remove it, and mark all column vectors in $\boldsymbol{A}^{*(2)}$ as *inactive* if they are still *active*.

3) Repeat Steps 1) and 2) until all vectors in Group 2 are marked *inactive*. This can always be achieved. Indeed, according to the definition of Group 2, any column vector $\boldsymbol{a}_{j_2} \in \boldsymbol{A}^{(2)}$ can be represented as a linear combination of the other column vectors of matrix $\boldsymbol{A}$. So, $\boldsymbol{a}_{j_2}$ always appears as a non-zero term in the linear expansion for some vector in Group 3; otherwise it belongs to Group 1.

We observe that after the above process, there are K_1 Type-1 columns, K_2 Type-2 columns, and at most K_2 Type-3 columns. This is because when we label each Type-3 column, we always set *inactive* at least 1 vector in Group 2.

To deal with the case that $\boldsymbol{A}$'s rank K' is less than K, we arbitrarily label $K - K'$ unlabeled column vectors in Group 3 as Type-2 columns to make $K_1 + K_2 = K$; we can also arbitrarily mark another $K_2 - K_3$ unlabeled column vectors in Group 3 as Type-3 columns to make $K_2 = K_3$. It is easy to see that after this padding, the selected Type-1, Type-2, and Type-3 columns satisfy the desired properties.

Given the fact that $\mathcal{S} = \mathcal{A}$, we focus on $\mathcal{S}$ and prove the following two lemmas.

Lemma 4. *There are in total no more than* $\sum_{K_1+K_2=K} \binom{m}{K_1}\binom{m-K_1}{K_2}\binom{m-K_1-K_2}{K_2} \leq 2m^{2K}$ *coding structures corresponding to all* $K \times m$ *coding matrices.*

Proof. We can see that we have at most $\binom{m}{K_1}$ ways to choose the K_1 Type-1 columns, $\binom{m-K_1}{K_2}$ ways to choose the K_2 Type-2 columns among the remaining $m-K_1$ columns, and $\binom{m-K_1-K_2}{K_2}$ ways to choose the $K_3 = K_2$ Type-3 columns among the remaining $m - K_1 - K_2$ columns. Hence, the total number of coding structures is no more than

$$\begin{aligned} \textstyle\sum_{K_2=0}^{K} m^K (m-K)^{K_2} &\leq \textstyle\sum_{K_2=0}^{K-1} m^K (m-K)^{K_2} + m^{2K} \\ &\leq \tfrac{m^K((m-K)^{K+1}-1)}{m-K-1} + m^{2K} \leq 2m^{2K}. \end{aligned} \tag{2.8}$$

□

Lemma 5. *The probability that all* n *clients are satisfied by a coding structure* $S(J^{(1)}, J^{(2)}, J^{(3)})$

can be upper bounded by

$$\Pr\{S(J^{(1)}, J^{(2)}, J^{(3)}) \textit{ can satisfy all } n \textit{ clients}\} \leq \begin{cases} [1-p^{2K}]^n, & p \leq \frac{\sqrt{5}-1}{2}, \\ [1-(1-p)^K]^n, & p > \frac{\sqrt{5}-1}{2}. \end{cases} \tag{2.9}$$

Proof. We denote the coding structure $S(J^{(1)}, J^{(2)}, J^{(3)})$ by S for short. We first notice that if a client i has the following connection pattern: $j' \notin R_i$ for any column $j' \in J^{(1)}$ and $j'' \in R_i$ for any column $j'' \in J^{(2)} \cup J^{(3)}$, then client i cannot be satisfied by coding matrices in coding structure S. Indeed, if a client i has the above connection pattern, then clearly:

• Client i has all messages indexed by $J^{(1)}$ as side information and cannot be satisfied by messages in $J^{(1)}$.

• Client i cannot decode any message in $J^{(2)} \cup J^{(3)}$ according to the decoding criterion, since any column vector indexed by $J^{(2)} \cup J^{(3)}$ is in the linear space spanned by all other vectors in $J^{(2)} \cup J^{(3)}$ from the definitions of $J^{(2)}$ and $J^{(3)}$.

• Client i cannot decode a message not indexed by $J^{(1)}$, $J^{(2)}$, and $J^{(3)}$, because column vectors indexed by $J^{(2)}$ contains a basis for the submatrix that is obtained from removing columns of $J^{(1)}$, and this implies that the messages not indexed by $J^{(1)}$, $J^{(2)}$, and $J^{(3)}$ are in the space spanned by vectors indexed by $J^{(2)}$.

Next, we can lower bound the probability that client i is not satisfied by S by calculating the probability that event $\{j' \notin R_i, \forall j' \in J^{(1)}, \text{ and } j'' \in R_i, \forall j'' \in J^{(2)} \cup J^{(3)}\}$ happens.

$$\begin{aligned} &\Pr\{\text{client } i \text{ is not satisfied by } S\} \\ &\geq \Pr\{j' \notin R_i, \forall j' \in J^{(1)}, \text{ and } j'' \in R_i, \forall j'' \in J^{(2)} \cup J^{(3)}\} \\ &\geq (1-p)^{K_1} p^{2K_2}. \end{aligned} \tag{2.10}$$

Therefore, we can upper bound the probability that all n clients are satisfied by structure S as follows.

$$\Pr\{\text{all } n \text{ clients are satisfied by } S\} \leq [1-(1-p)^{K_1} p^{2K_2}]^n. \tag{2.11}$$

Note that for $p \leq (\sqrt{5}-1)/2$ we have $1-p \geq p^2$, and for $p > (\sqrt{5}-1)/2$ we have $1-p < p^2$. So that the result follows from eq. (2.11) and the fact that $K_1 + K_2 = K$. □

A lower bound is shown in the following theorem.

Theorem 5. *For pliable index coding over random graph $B(m,n,p)$ ($m = O(n^\delta)$ for some constant δ), with probability at least $1 - O(1/n^2)$, the linear pliable index code length can be lower bounded as follows:*

$$K \geq \begin{cases} \frac{\log(n)}{4\log(1/p)}, & p \leq \frac{\sqrt{5}-1}{2}, \\ \frac{\log(n)}{2\log[1/(1-p)]}, & p > \frac{\sqrt{5}-1}{2}. \end{cases} \tag{2.12}$$

Proof. According to Lemmas 4 and 5, we can see that the probability a random graph $B(m,n,p)$ can be satisfied by a pliable index code of length $K = c(p)\log(n)$ (the parameter $c(p) = \frac{1}{4\log(1/p)}$ for $p \leq \frac{\sqrt{5}-1}{2}$ and $c(p) = \frac{1}{2\log[1/(1-p)]}$ for $p > \frac{\sqrt{5}-1}{2}$) is at most

$$\begin{aligned} &2m^{2K}[1 - \frac{1}{2^{\frac{K}{2c(p)}}}]^n = 2m^{2c(p)\log(n)}[1 - \frac{1}{2^{\frac{\log(n)}{2}}}]^n \\ &\leq [2^{2c(p)\log(n)\log(m)+1}]/e^{\sqrt{n}} \leq O(1/n^2). \end{aligned} \tag{2.13}$$

□

From Theorem 5, we distinguish the following special cases for the lower bound depending on p.

• $p \leq O(1/n^\alpha)$ or $1-p \leq O(1/n^\alpha)$ for some constant α: this is the sparse or dense case and we get $\Omega(1)$ lower bound from Theorem 5. To explain this, consider two extreme cases. The first one is a sparse case where each client requires exactly one different message and has all others as side information. Thus, we only need to transmit a linear combination of all the messages, such that each client can decode her required message. The other one is a dense case where each client has only one side-information message and requires any new one from the remaining messages. In this case, we can use 2 arbitrary uncoded transmissions to satisfy all clients.

• Constant p: in this case we achieve $K \geq \Omega(\log(n))$ from Theorem 5, namely, the random instance $B(m, n, p)$ almost surely needs linear code length of $\Omega(\log(n))$. In particular, when $p = (\sqrt{5} - 1)/2 \approx 0.618$, *the Golden Ratio*, this lower bound achieves maximum $0.36 \log(n)$ among all p.

2.7.2 An Upper Bound

To prove an upper bound, we propose a simple coding scheme that achieves code length of $O(\log(n))$ with high probability.

Given a constant p and $m = O(n^\delta)$ for some constant δ, we construct the coding matrix $\boldsymbol{A}$ as follows:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{2.14}$$

The matrix has $\frac{3}{\log(e/(e-1))} \log(n)$ rows. In each row, we have a constant weight: $1/p$ 1s and 0s for other elements[3]. In any two rows, the 1s are non-overlapping. The probability that a client i is satisfied by the first row can be upper bounded by the following equation.

$$\Pr\{\text{client } i \text{ is satisfied by the first row}\} = \binom{1/p}{1} p(1-p)^{1/p-1} \geq 1/e. \tag{2.15}$$

Note that since 1s in any two rows of the coding matrix do not overlap, we can calculate the probability that a client i is satisfied by the coding matrix $\boldsymbol{A}$ as:

$$\begin{aligned} \Pr\{\text{client } i \text{ is satisfied by the coding matrix } \boldsymbol{A}\} &\geq 1 - (1 - 1/e)^{\frac{3}{\log(e/(e-1))} \log(n)} \\ &\geq 1 - 1/n^3. \end{aligned} \tag{2.16}$$

[3] We simply treat $1/p$ as integers, which does not change the problem essentially.

Hence, the probability that all clients are satisfied can be bounded as:

$$\begin{aligned}\Pr\{\text{all clients are satisfied by the coding matrix } \boldsymbol{A}\} &\geq (1-\tfrac{1}{n^3})^n \\ &\geq 1-\tfrac{1}{n^2}.\end{aligned} \tag{2.17}$$

Therefore, we have the following result.

Theorem 6. *For pliable index coding over random graph $B(m,n,p)$ ($m = O(n^\delta)$ for some constant δ) with constant p, we can achieve the optimal linear pliable index code length $K = \Theta(\log(n))$ almost surely.*

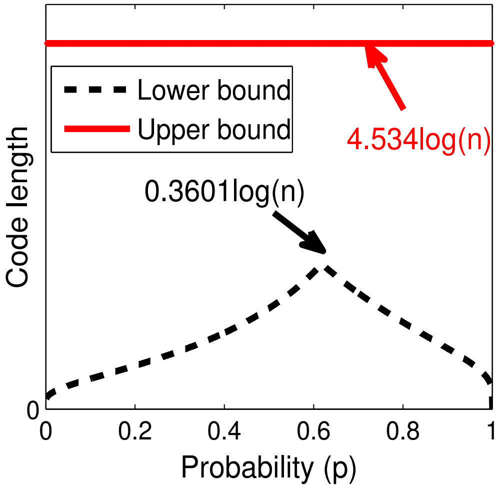


Figure 2.3: Lower and upper bounds of pliable index coding over random graphs.

To illustrate how the lower and upper bounds change with the probability p, we plot the relationship between them in Fig. 2.3.

2.8 Discussion

In this chapter, we make two observations: one on the field size for the optimal solution and the other on a connection with the minrank problem.

2.8.1 Field Size

We show through an example that a binary code is not sufficient to achieve the optimal code length. Consider the following instance with $m = 4$ and $n = 10$:

- $R_1 = \{1\}, R_2 = \{2\}, R_3 = \{3\}, R_4 = \{4\}, R_5 = \{1,2\}, R_6 = \{1,3\}, R_7 = \{1,4\}, R_8 = \{2,3\}, R_9 = \{2,4\}, R_{10} = \{3,4\}$.

This instance contains clients with request sets of all 1-message and 2-message subsets. We can easily see that the optimal code length is 2, e.g., $b_1 + b_2 + b_4$ and $b_2 + b_3 + 2b_4$, in $\mathbf{F}_3$. However, we cannot find a binary code of length 2, because we have all 1-message and 2-message request sets, requiring $\boldsymbol{a}_j \neq (0,0)^T$, for $j = 1, 2, 3, 4$ and $\boldsymbol{a}_j \neq \boldsymbol{a}_{j'}$, for $j \neq j'$. But, we have only 3 non-zero vectors $(1,0)^T, (0,1)^T, (1,1)^T$. It is not possible to assign these 3 non-zero vectors to 4 columns so as to satisfy all clients.

This example extends to that at least a field size $m-1$ of coding coefficients is needed to achieve the optimal code length for all instances with m messages. We consider an instance with m messages and $n = m + \binom{m}{2}$ clients, where the clients have all 1-message and 2-message request sets. Namely, the clients' request sets are $\{j\}$ and $\{j_1, j_2\}$, for any $j \in [m]$ and $j_1, j_2 \in [m]$.

Assume we use finite field $\mathbf{F}_q$ to realize coding. According to our decoding criterion, we need every coding vector to be nonzero and any pair of the coding vectors to be linearly independent.

- If the coding vector contains 0, then there will be 2 of them: $(1,0)^T$ and $(0,1)^T$ since any other vector in the form of $(x,0)^T$ ($x \in \mathbf{F}_q$) is linearly dependent with $(1,0)^T$ and similarly, $(0,x)^T$ ($x \in \mathbf{F}_q$) is linearly dependent with $(0,1)^T$.

- If the coding vector is in the form $(x,y)^T$, $x, y \in \mathbf{F}_q, x, y \neq 0$, then there are in total $(q-1)^2$ such vectors. However, $(x,y)^T$ is linearly dependent with $z(x,y)^T$, for $z \in \mathbf{F}_q$. There are in total $(q-1)$ distinct $z(x,y)^T$ vectors, so the total number of pair-wise independent vectors is $(q-1)^2/(q-1) = (q-1)$.

Therefore, we need $2+(q-1) \geq m$ in order to satisfy these clients, resulting in $q \geq m-1$.

2.8.2 Minrank

In index coding, the optimal linear code length is shown to equal to the minrank, which is the minimum rank of a mixed matrix (some of whose elements are to be determined) associated with the side-information graph [BBJ11]. In a similar way, we can characterize the pliable index coding problem using the minimum rank of a mixed matrix associated with the bipartite graph.

We say that a matrix $\boldsymbol{G} \in \mathbf{F}_q^{n\times m}$ fits the pliable index coding problem instance $(m, n, \{R_i\}_{i\in[n]})$, if in the i-th row ($\forall i \in [n]$):

- among all $j \in R_i$, there exists one and only one $j^* \in R_i$, such that $g_{ij^*} = 1$, and other $g_{ij} = 0$ for any $j \in R_i \backslash \{j^*\}$;
- for $j \in S_i$, g_{ij} can be any element in $\mathbf{F}_q$.

Let us denote by $\mathcal{G}$ the set of all matrices fitting the pliable index coding problem $(m, n, \{R_i\}_{i\in[n]})$, and by minrank$(\mathcal{G})$ the minimum rank among all the matrices $\boldsymbol{G} \in \mathcal{G}$. In other words, minrank$(\mathcal{G}) = \min_{\boldsymbol{G}\in\mathcal{G}} \text{rank}(\boldsymbol{G})$, where rank$(\boldsymbol{G})$ denotes the rank of matrix $\boldsymbol{G}$. The following theorem characterizes the optimal coding length:

Theorem 7. *The optimal linear code length of the pliable index coding instance* $(m, n, \{R_i\}_{i\in[n]})$ *equals to minrank*$(\mathcal{G})$.

Proof. First, let us prove that a linear code with length $K = \text{minrank}(\mathcal{G})$ exists. Assume that a matrix $\boldsymbol{G} \in \mathcal{G}$ achieves rank K. Without loss of generality, let us also assume that the first K rows of $\boldsymbol{G}$ are linearly independent. For the encoding process, we define the coding matrix $\boldsymbol{A}$ to be the first K rows of $\boldsymbol{G}$. For matrix $\boldsymbol{G}$, there is one and only one $j^* \in R_i$, such that $g_{ij^*} = 1$, and other $g_{ij} = 0$ for $j \in R_i\backslash\{j^*\}$; so that column $\boldsymbol{g}_{j^*}$ cannot be expressed as a linear combination of $\{\boldsymbol{g}_j\}_{j\in R_i\backslash\{j^*\}}$. Since all the rows of $\boldsymbol{G}$ are linear combinations of the first K rows, column $\boldsymbol{a}_{j^*}$ cannot be expressed as a linear combination of $\{\boldsymbol{a}_j\}_{j\in R_i\backslash\{j^*\}}$ either. As a result, the decoding criterion holds for client i and message j^* can be decoded by client i.

Next, let us prove that for any linear code with a $K \times m$ coding matrix $\boldsymbol{A}$ in filed $\mathbf{F}_q$ has a code length $K \geq \text{minrank}(\mathcal{G})$. We show that using the coding matrix $\boldsymbol{A}$, we can build a matrix $\boldsymbol{G} \in \mathbf{F}_q^{n\times m}$ with rank at most K that fits the index coding problem. To show this, we use the following claim.

Claim 1. *If for client i, the message j^* can be decoded, then the row vector $e_{j^*}^T$ is in the span of $\{\boldsymbol{\alpha}_l^T : l \in [K]\} \cup \{e_j^T : j \in S_i\}$, where e_j^T is a row vector with all 0s, except a 1 in the j-th position and $\boldsymbol{\alpha}_l^T$ represents the l-th row of matrix $\boldsymbol{A}$.*

This claim shows that $e_{j^*}^T$ is in the span of the union of row space of $\boldsymbol{A}$ and the side-information space. The proof of this claim can be found in [BBJ11].

For each client i, the claim states that $e_{j^*}^T = \sum_{l=1}^K \lambda_l \boldsymbol{\alpha}_l^T + \sum_{j\in S_i} \mu_j e_j^T$ for some λ_l, μ_j in field $\mathbf{F}_q$. To construct $\boldsymbol{G}$, we define the i-th row of $\boldsymbol{G}$, γ_i^T, to be the linear combination $\sum_{l=1}^K \lambda_l \boldsymbol{\alpha}_l^T$. Or equivalently, we have $\gamma_i^T = \sum_{l=1}^K \lambda_l \boldsymbol{\alpha}_l^T = e_{j^*}^T - \sum_{j\in S_i} \mu_j e_j^T$. This shows that γ_i^T has value 1 at position j^*, $-\mu_j$ at position $j \in S_i$, and 0 at positions indexed by $R_i \backslash \{j^*\}$.

Therefore, we have shown that $K \geq \text{rank}(\boldsymbol{G}) \geq \text{minrank}(\mathcal{G})$. □

2.9 Numerical Results

In this chapter, we conduct numerical experiments on our proposed algorithms. We first evaluate the performance of our BinGreedy algorithm by comparing with the algorithm in [BF12], and then evaluate the optimality gap with respect to the minrank solution in Chapter 2.8. We finally evaluate the BinGreedyT algorithm performance for t-requests case.

2.9.1 Performance Comparison

We compare the performance of our proposed algorithm BinGreedy with *RANDCOV*, which is a randomized algorithm proposed in [BF12]. RANDCOV is the current state-of-the art alternative and was theoretically shown to achieve an *average performance* upper bounded by $O(\log^2(n))$ with respect to the random code realization.

In our simulations, we set the number of messages m to be $n^{0.75}$ and numerically investigate how the code length changes with the number of clients n. We randomly generate 100 pliable index coding bipartite graph instances for each n, by connecting each client and each message with probability 0.3 in the homogeneous case; and connecting equal number of clients with each message with probabilities $0.05, 0.15, \ldots, 0.95$ in the heterogeneous case.

Fig. 2.4 shows the code length varying with n (note that the horizontal axis is in logarithmic scale). We can see that on average (averaged over 100 instances for the same n) and in the worst case, the proposed BinGreedy algorithm outperforms RANDCOV by 20%-35% in terms of the code length for homogeneous instances. In heterogeneous instances, the proposed BinGreedy algorithm outperforms the existing randomized algorithm by 20%-50%. We also observe that for heterogeneous instances, we need more transmissions than the homogeneous instances of the same size. As seen in the figure, for homogeneous instances, the code length increases almost linearly with $\log(n)$; while for heterogeneous instances, the code length increases super linearly with $\log(n)$.

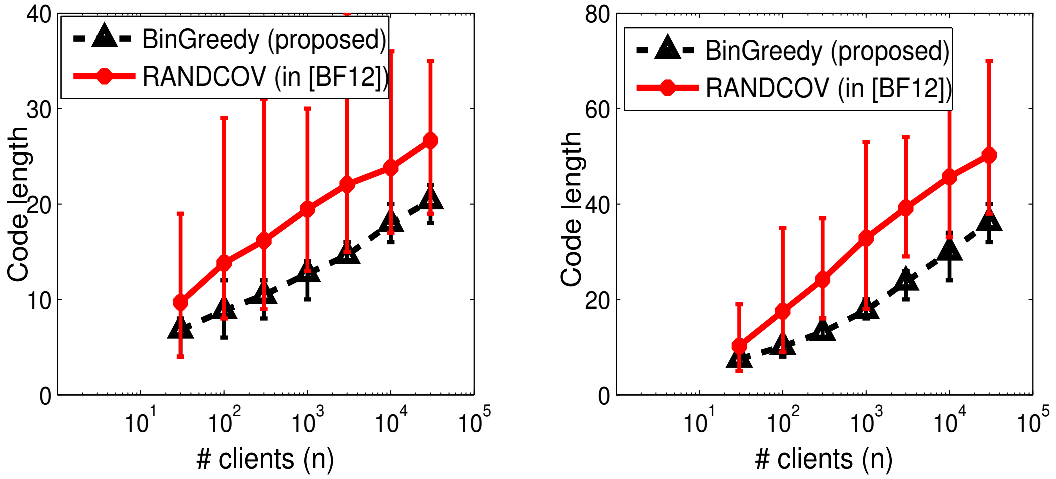
In contrast to the randomness of RANDCOV, our proposed BinGreedy algorithm runs deterministically and we expect more robustness. Indeed, we can see from Fig. 2.4 that the difference between best case and worst case instances is much larger for RANDCOV than that for the proposed BinGreedy algorithm.

2.9.2 Optimality Gap

We compare our BinGreedy performance with the optimal binary code length calculated through the minrank method in Chapter 2.8. By setting $n = 12$ and 18, we evaluate the performance of the two algorithms as m varies[4]. For each pair of m and n, we randomly generate 10 bipartite graph instances by connecting each client and each message with probability 0.3.

The *gap* for an instance I is defined as the difference of code length achieved by our

[4] Because of the exponential complexity of finding the optimal performance we can compare only for small instances.



(a) Homogeneous case

(b) Heterogeneous case

Figure 2.4: Comparison of BinGreedy and randomized algorithms (code length vs. the number of clients). The curves in the figures show the average performance over random instances and the bars at each point show the region between the best and worst case performances.

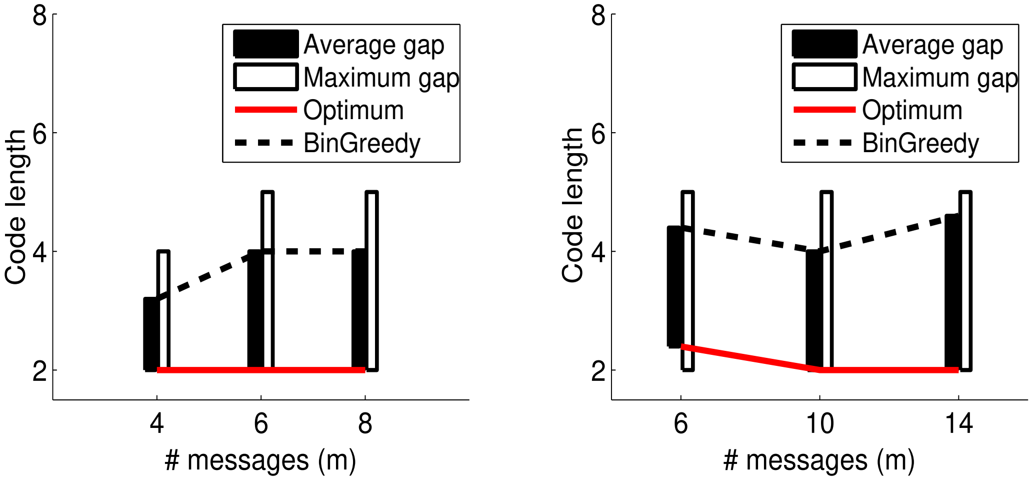
BinGreedy algorithm and by the optimal binary algorithm, i.e., $gap = BinGreedy(I) - OPT_2(I)$. We plot the *average gap* and the *maximum gap* among instances generated with the same parameters m and n. Fig. 2.5 shows that the average gap (the black bar) is around 2 for both n=12 and n=18; the maximum gap (the white bar) is 3 for both n=12 and n=18; the same as the average code lengths achieved by the BinGreedy and optimal algorithms. We also note that the approximation ratio for $n = 18$ (2.01) is slightly greater than that for n=12 (1.87). In fact, the approximate ratio is known to be no less than $\Omega(\log\log(n))$ from [SF16b], so it grows as n increases.

2.9.3 t-requests Case

In this chapter, we conduct experiments on the t-requests case using our BinGreedyT algorithm.

In our simulations, we set the number of messages m to be $n^{0.75}$. We randomly generate 100 pliable index coding bipartite graph instances for each n, by connecting each client and each message with probability 0.3.

In Fig. 2.6 (a), we investigate how the code length changes with the number of clients



(a) Number of clients $n = 12$

(b) Number of clients $n = 18$

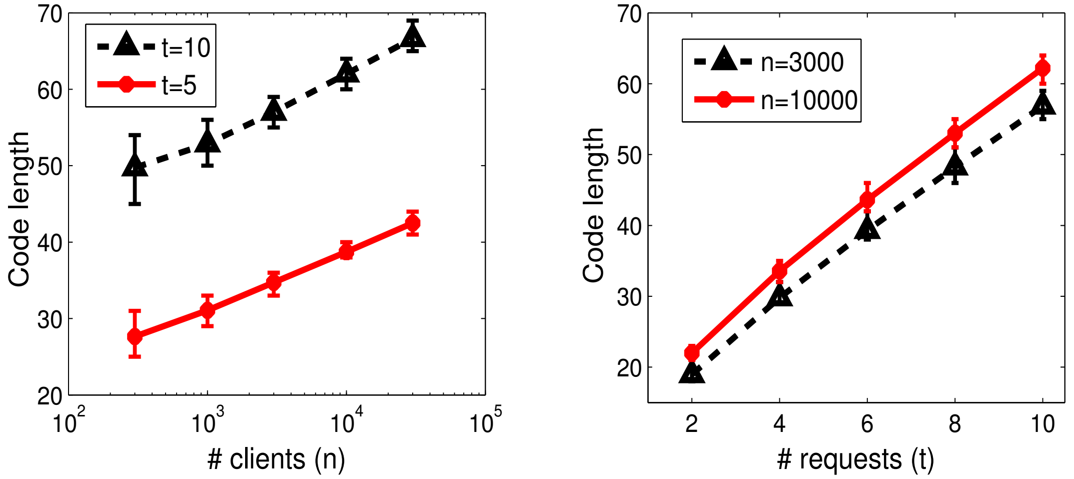
Figure 2.5: Optimality gap of BinGreedy algorithm.

n for 5-requests and 10-requests cases. We can see that for both curves, the required code length increases slightly greater than logarithmically with n (notice that the horizontal axis is in logarithmic scale), from 28 to 42 for $t = 5$ and from 50 to 69 for $t = 10$. Indeed, we show that our algorithm performs in the worst case as $O(t\log(n) + \log^2(n))$. Given a fixed t, we also observe that as n increases, the difference between code lengths in the best case and in the worst case decreases, i.e., the bar in the figure becomes shorter. This implies robustness for larger n.

In Fig. 2.6 (b), we evaluate how the number of requests t affect the the code length for $n = 3000$ and $n = 10000$. We can see that given a fixed number of clients n, the code length increases almost linearly with the number of requests t, from around 20 to 60.

2.10 Open Questions and Future Work

We note that for pliable index coding problem, we have a $\log(n)$ ratio between the lower and upper bounds of optimal code length, namely, $\Omega(\log(n))$ vs. $O(\log^2(n))$. This gap closes only for equal size side information case [BF15] or random graph instances (in Chapter 2.7) However, in general, we still do not know whether there exists an algorithm that can achieve a worst case performance of $O(\log(n))$ or is it the case that any algorithm cannot do better



(a) Code length vs. the number of clients (n). (b) Code length vs. the number of requests (t).

Figure 2.6: Performance of BinGreedyT algorithm. The curves in the figures show the average performance over random instances and the bars at each point show the region between the best and worst case performances.

than $\Omega(\log^2(n))$.

For an extensive number of Big Data applications, the content-type messages are stored at different locations or servers. Hence, a future direction is to study the pliable index coding algorithm in such a *distributed* setting. In this setting, the servers only have limited choices of encoding messages. We are interested in the fundamental bounds, as well as effective algorithms, when this constraint is imposed.

2.11 Summary

In this chapter, we study the fundamental bounds for optimal code length for pliable index coding and design polynomial-time approximation algorithms for pliable index coding. We show that our proposed algorithm achieves code length at most $O(\log^2(n))$. We modify this algorithm for the t-requests case and provide a worst case performance $O(t\log(n)+\log^2(n))$ guarantee. We construct problem instances that achieve a lower bound of $\Omega(\log(n))$ for pliable index coding and $\Omega(t+\log(n))$ for the t-requests case. We perform a probabilistic analysis over random graphs to show that the optimal code length is almost surely $\Theta(\log(n))$. We also present experimental results that show up to 50% performance benefits of our proposed algorithms and higher robustness over existing algorithms.

CHAPTER 3

Application to Bandwidth Aware Recommender Systems

3.1 Introduction

Aiming at maximizing an aggregate benefit, recommender systems make decisions about which messages to offer to users [RV97, LCL14]. These existing recommender systems are currently oblivious to the cost of distributing the content from the server to the users, which, however, may result in failure: unsatisfactory delivery is identified as a core threat to the user experience and has already caused loss of billions of revenue dollars [Con15]. Wireless consumption in particular, which is increasingly gaining popularity, is inherently subject to bandwidth constraints.

In this chapter, we ask: how much could we gain in terms of bandwidth and user satisfaction, if recommendation systems took into account not only the user preferences, but also the fact that they need to serve these users under bandwidth constraints? In other words, what if the recommender systems became bandwidth aware?

We formulate this as a new problem in the context of pliable index coding, where now each client has preferences associated with messages: a client can be satisfied by receiving any message she does not already have; however, the benefit we get is proportional to how high her preference is, for the message she gets. For instance, consider wireless stations serving sale coupons inside a shopping mall: a client walking outside a shop would be happy to receive a coupon she does not already have, but would be happier to receive (and more likely to use) a coupon closer to her interests. We note that the side-information setup fits well with the recommender systems framework [LCL14]: collecting side information about the clients and keeping track of previous content served is an integral part of recommender systems; it is a natural step to leverage this side information, not only to inform recommendations, but to also increase the communication efficiency so as to extract more benefits under communication constraints. But for the amount of interesting work in index coding (eg., [BBJ11, BK98, BKL10, ESG10, EEL15]), this is the first work, as far as we know, that explores trade-offs between user satisfaction and bandwidth.

When setting as our goal to evaluate potential benefits, a challenge we faced is that these depend on the preference model we use. There exist numerous models for expressing preferences and for taking decisions based on them; clearly we cannot exhaustively investigate all possible ranking models. We opted to sample a few models that we thought were representative, with the hope of finding consistent trends across them. One model we investigated uses the Borda count, that has each client sort m messages according to her preferences, and assigns to a message ranked l by a client a score of $m + 1 - l$ [Bor81]. We also considered a bimodal preferences model, where a fraction of the messages are much more preferable than the remaining ones. More generally, we considered an arbitrary score model, where each message j gets an arbitrary score w_{ij} by a client i.

To calculate the aggregate benefit, we count only the highest-preference message we have served to each client. This is motivated by the fact that, if a client at a certain time can see only one video (or read one article or click one ad), although her device may have downloaded multiple items, she will only see her most preferred one. We will collect the corresponding benefit. This benefit model aligns well with the index-coding rationale, where only the one message the client wants counts.

In this chapter, we examine the trade-off between benefit and bandwidth across three scenaria, both theoretically and numerically using designed algorithms. We first provide results for the case where there is no side information for each client and each client has a full ranking of the messages. We show that the problem is NP-hard, however a simple greedy polynomial time algorithm can achieve an approximation ratio of 1.58. Moreover, we provide upper and lower bounds of optimal performance as well as an average case analysis, both indicating diminishing returns: the benefits increase with the number of transmissions K only by a multiplicative factor $1 - 1/K$.

The second scenario investigates the case where each client has side-information of the same cardinality and a partial ranking over the desired messages. We prove lower bounds on the optimal benefits, and design a polynomial-time algorithm that achieves a $O(1)$ approximation ratio.

The third scenario considers a general case where each client has arbitrary size side information. This problem is hard to approximate within a ratio of $n^{1-\epsilon}$ for any $\epsilon > 0$. For this case we establish a connection with the maximum weighted independent set problem; we design and evaluate a heuristic coded algorithm that leverages this connection.

We evaluate our algorithms numerically over synthetic and real world data sets (Yahoo! advertiser bidding data sets [yah]). We find that even with one transmission we can in many cases already achieve half of the maximum possible benefit. In general, we can achieve 80% of the benefit with less than 10% of the transmissions we would need to achieve 100% of it. We also find that leveraging side information to make coded transmissions, can in some cases enable doubling the benefit over uncoded transmissions.

The work presented in this chapter was published in [SF16a].

3.2 Setup and Problem Formulation

3.2.1 Setup

We assume that a server has m messages $b_1, b_2, \ldots, b_m$, taking values in a finite field $\mathbf{F}_q$, and n clients $1, 2, \ldots, n$. Each client $i \in [n]$ already knows (has as side information) a subset of the m messages; we denote by $S_i \subseteq [m]$ the side information of client i (S_i could be the empty set), and by $R_i = [m] \backslash S_i$ the set of messages that the client may request (does not have).

Broadcast transmissions and coding

The server is connected to the clients through error-free broadcast transmissions; that is, all clients perfectly receive each server transmission. During the k-th transmission, the server transmits $x_k = \sum_{j \in [m]} a_{kj} b_j$, where $a_{kj} \in \mathbf{F}_q$ are constant coefficients and the addition and multiplication operations are performed in $\mathbf{F}_q$. Thus the server transmits either one of the uncoded messages b_j, or a linear combination of some of the messages. Assume the server broadcasts K transmissions $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$. We will denote by $D_i = \phi_{i,dec}(\{b_j\}_{j \in S_i}, \boldsymbol{x})$ the set of new messages that client i can decode, where $\phi_{i,dec}(\{b_j\}_{j \in S_i}, \boldsymbol{x})$ is the decoding function for client i.

Scores and benefit B

Each client i has a *rank or preference* $\pi_i(j)$ for each of the messages j in her request set R_i; accordingly, we get a *message score* $s_i(j)$ when message j is decoded by client i. Sometimes we omit the ranking and assume that the message scores are given directly. A client i has *client score* $s(i) = \max_{j \in D_i} s_i(j)$; that is, we only count the message of highest score among the $|D_i|$ messages she decodes. If D_i is empty we set $s(i) = 0$. The *benefit* B we get is the aggregate client score $B = \sum_{i \in [n]} s(i)$. We considered the following models for scores:

-The *Borda count* method assumes that the ranking is a permutation of the set R_i and calculates message scores as $s_i(j) = |R_i| + 1 - \pi_i(j)$ (a message ranked first gives score $|R_i|$, ranked second gives score $|R_i| - 1$, etc.)

-The *bimodal score* assumes that a fraction F_{bim} of the messages are much more desirable than the remaining $(1 - F_{bim})$ fraction. In particular we assume that the ranking $\pi_i(j)$ is a permutation of the $|R_i|$ messages, and we set $s_i(j) = G_{bim}(m+1-\pi_i(j))$ if $\pi_i(j) <= F_{bim}|R_i|$ and $s_i(j) = m+1-\pi_i(j)$ otherwise. The parameter G_{bim} determines how separated (bimodal) the two sets of messages are.

-The general model assigns an *arbitrary weight* to each score $s_i(j) = w_{ij}$.

Performance metrics

We are interested in the tradeoff between the number K of broadcast transmissions and the corresponding achievable benefit B.

3.2.2 Problem Formulation

We first express that the bandwidth aware recommendation problem treats index coding and pliable index coding as two special cases; and then introduce the three example scenaria we will examine through theoretical analysis in this chapter.

Relations to Index Coding and Pliable Index Coding

For index coding, a client i requests a specific message $j_i \in R_i$. If we set $s_i(j) = 1$ for $j = j_i$ and 0 otherwise. Thus $s(i)$ takes values either 0 or 1, depending on whether client i can decode b_{j_i} or not, and $0 \leq B \leq n$. Index coding asks for the minimum number of transmissions to achieve the maximum benefit $B = n$ possible, i.e., so that all clients receive the message they have requested.

For pliable index coding, each client is happy to receive any message she does not have (without any preference). We thus set $s_i(j) = 1$, for all i and $j \in R_i$. Then, $s(i)$ takes value 1 if client i decodes any one message in R_i, and $0 \leq B \leq n$. Pliable index coding asks for the minimum number of transmissions to achieve benefit $B = n$.

New Formulations

In the following, we sample three scenaria of the bandwidth aware recommendation problem depending on the side information size and preference model and then we derive theoretical results for these scenaria. In each scenario, we ask what is the benefit B given a fixed number of K transmissions.

•P1. No side information and full ranking

No side information implies that $R_i = [m]$. We consider the Borda count, where $\pi_i(j)$ defines for each client i a permutation of $[m]$, and $s_i(j) = m+1-\pi_i(j)$ is also a permutation of $[m]$. Thus $0 \leq B \leq nm$.

•P2. Equal size side information and partial ranking

We assume that $|S_i| = m - r$ for all clients, $\pi_i(j)$ defines for client i a permutation over the remaining r messages, and $s_i(j) = r + 1 - \pi_i(j)$. In this case $0 \leq B \leq nr$.

•P3. Arbitrary size side information and score

If the size of the side information set for each receiver is arbitrary, we cannot use a permutation of R_i as ranking of the messages to calculate the score, as it would give unfair weight to the different clients. We assume instead that (fair) scores $s_i(j) = w_{ij}$ are provided as input.

3.3 No Side Information (P1)

This is the simplest case we examine. This problem is close to the rank aggregation problems studied in the literature [Bor81, DG77, Kem59, DKN01], the difference being that only the highest ranked message a client receives counts towards the total benefit. Interestingly, while the rank aggregation problem is polynomial time using the Borda count optimal rule, taking into account only the highest score message makes the problem NP-hard.

We collect the preferences into an $n \times m$ ranking matrix Π, where the (i, j) entry is the rank of message j by client i, i.e., $\pi_i(j)$. That is, each row of this matrix expresses the ranking of messages by a client.

Illustrative example

Consider an instance with $n = 5$ clients, $m = 4$ messages and the 5×4 ranking matrix

$$\Pi = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 4 & 1 & 2 & 3 \\ 4 & 2 & 1 & 3 \\ 4 & 2 & 3 & 1 \\ 4 & 1 & 2 & 3 \end{bmatrix}. \tag{3.1}$$

We can see that, to serve to all clients their first preference (benefit $B = 20$), we need $K = 4$ transmissions, yet with only $K = 1$ transmission (second column) we can already serve to all clients either their first or their second preference (benefit $B = 17$). Moreover, in a situation where the server recommends to send to each client their first preference but only one of these messages is delivered (in time) because of bandwidth constraints, we would in the worst case achieve benefit $B = 8$ (e.g., only the first message is delivered); thus taking into account the bandwidth constraints can more than double the benefit. This difference can be magnified proportionally to a parameter G_{bim}, if instead of Borda count we used bimodal score model with gain factor G_{bim}.

3.3.1 Problem P1 is NP-hard

We next describe our NP-hard result in the following theorem.

Theorem 8. *The bandwidth aware recommendation problem with Borda score model and no side information (P1) is NP-hard.*

Proof. The proof uses a reduction from the set cover problem [Kar72]. We reiterate the set cover problem in the following. Consider a set cover problem instance, with a universe set $U = \{1, 2, \ldots, n\}$ and a family of subsets of U, $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$. The union of elements of $\mathcal{S}$ is U, i.e., $S_1 \cup S_2 \cup \ldots \cup S_r = U$ and $S_j \subseteq U$ for all $j \in [r]$. The goal of the set cover problem is to find a subset of $\mathcal{S}$, $\mathcal{S}' \subseteq \mathcal{S}$, with minimum cardinality such that $\cup_{S \in \mathcal{S}'} S = U$.

We show that a set cover problem instance can be reduced to a P1 problem instance with m messages and n clients in polynomial time. The clients will correspond to elements in the universe U, and we will have $m = (n + 2)(nr - n + 1)$ messages, that correspond to the r subsets in the family $\mathcal{S}$ plus some additional (dummy) messages for construction purposes. We will construct a corresponding $n \times m$ ranking matrix Π, and sequentially test whether a selection of $K = 1, 2, \ldots, r$ columns can achieve a benefit greater than or equal to $n(m + 1 - r)$. Recall that in the ranking matrix the rows correspond to clients (elements of U), and the columns to messages (each one of the first r columns will represent a subset in the family $\mathcal{S}$ and the remaining dummy messages).

Let us denote by $\mathcal{S}[i]$ the family of subsets that contains element $i \in U$, i.e., $\mathcal{S}[i] = \{S \in \mathcal{S} : i \in S\}$. We denote the cardinality of $\mathcal{S}[i]$ by $d_i \leq r$ and the columns corresponding to subsets in $\mathcal{S}[i]$ by $j_{i1}, j_{i2}, \ldots, j_{id_i}$. Let us define $g = nr - n + 1$ and $d = d_1 + d_2 + \ldots + d_n$.

Then $m = (n+2)g$. The ranking matrix we would like to construct has the following form:

$$\Pi = [\Pi_0, \Pi_1, \Pi_2, \ldots, \Pi_n, R], \tag{3.2}$$

where Π_0 is of size $n \times r$; Π_i $(1 \leq i \leq n)$ is of size $n \times (g - d_i)$; and R is of size $n \times (2g - r + d)$. The construction process is as follows.

• Step 1: construct submatrix Π_0. We assign rankings row by row to the matrix. For the i-th row, assign an arbitrary permutation of ranks $1, 2, \ldots, d_i$ to d_i positions corresponding to $\mathcal{S}[i]$, and assign an arbitrary permutation of ranks $g+1, g+2, \ldots, g+r-d_i$ to $(r - d_i)$ positions corresponding to $\mathcal{S}^C[i] = \mathcal{S} \backslash \mathcal{S}[i]$.

• Step 2: construct submatrix Π_i, $1 \leq i \leq n$, of size $n \times (g - d_i)$. First, assign an arbitrary permutation of the ranks $d_i + 1, d_i + 2, \ldots, g$ as the row vector of row i. For the other rows $l \neq i$, assign an arbitrary permutation of the ranks $(i+1)g+1, (i+1)g+2, \ldots, (i+1)g+(g-d_i)$ as the row vector of row l.

• Step 3: construct submatrix R. For each row i of submatrix R, assign an arbitrary permutation of the remaining $(2g - r + d)$ ranks $g + r - d_i + 1, g + r - d_i + 2, \ldots, 2g, 2g + g - d_1 + 1, \ldots, 3g, 3g + g - d_2 + 1, \ldots, 4g, \ldots, (n+1)g + g - d_n + 1, \ldots, (n+2)g, (i+1)g + 1, (i+1)g + 2, \ldots, (i+1)g + (g - d_i)$ as a row vector of row i.

Next, we prove that finding a cover of size K can be achieved by finding K columns that achieve a certain benefit; hence we can sequentially test whether K is the minimum set cover size, from which the theorem follows. We map a selection of K sets for the cover problem to the selection of the corresponding messages (columns of Π_0) to transmit; and reversely, when selecting messages to transmit, we consider as part of the cover the sets corresponding to messages/columns of Π_0 (we ignore dummy messages). Thus we say that selection of K columns to transmit "covers" a client, if the corresponding set in the set cover problem includes this client. We finish the proof of this theorem by following Lemma 6. □

Lemma 6. *The selection of K subsets in $\mathcal{S}$ can cover the universe U, if and only if we can find K columns in matrix Π that achieve benefit at least $n(m + 1 - r)$.*

Proof. • Necessity. If a selection of K subsets in $\mathcal{S}$ can cover the universe U, this selection can achieve at least a minimum rank of r for any client, resulting in a benefit at least $n(m+1-r)$ according to our ranking assignment in construction step 1.

• Sufficiency. Suppose there exists an optimal selection $\mathcal{K}$ of size K that does not cover some client $i \in [n]$ and achieves a benefit $B_{\mathcal{K}} \geq n(m+1-r)$.

We observe that the selection $\mathcal{K}$ does not contain any column in Π_i.

Indeed, if the selection $\mathcal{K}$ does not cover client i, but contains a column j in Π_i, then we can simply construct another selection by replacing column j with the column that contains the rank 1 choice of client i. Formally, we can construct $\mathcal{K}' = \mathcal{K} \cup \{\pi_i^{-1}(1)\} \backslash \{j\}$, where $\pi_i^{-1}(1)$ is the column that contains the rank 1 choice of client i. From construction, we can see that the ranks of message $\pi_i^{-1}(1)$ (column $\pi_i^{-1}(1)$) is better than j for all clients and at least client i can improve her score. So $\mathcal{K}'$ is a better selection than $\mathcal{K}$, resulting in a contradiction of $\mathcal{K}$ being an optimal selection.

Thus it suffices to consider the case that $\mathcal{K}$ does not cover client i and does not contain any column in Π_i. From construction, we note that columns corresponding to $\mathcal{S}[i]$ and Π_i contain choices of ranks $1, 2, \ldots, g$ of client i. As a result, the minimum rank that client i can achieve is $g+1$, and the maximum score client i can achieve is $m-g$. Therefore, the benefit can be bounded by

$$B_{\mathcal{K}} \leq (n-1)m + (m-g) = n(m+1-r) - 1 < n(m+1-r), \tag{3.3}$$

which results in a contradiction with $B_{\mathcal{K}} \geq n(m+1-r)$.

We finally argue that a selection $\mathcal{K}$ of K columns of the ranking matrix Π that can cover all $i \in U$, does not contain columns outside Π_0. This follows from construction, since a column j_1 in Π_i ($i = 1, 2, \ldots, n$) is dominated by any column in Π_0 that can cover client i, such that the removal of j_1 will not affect the benefit. This means that $\mathcal{K} \backslash \{j_1\}$ can achieve the same benefit as $\mathcal{K}$. In this case, the testing would stop at most at step $K-1$. Similarly, a column j_2 in R is dominated by any column in Π_0 such that the removal of j_2 will not affect the benefit. □

We illustrate the construction of matrix Π and the reduction using a simple example.

Example: Let us consider a set cover problem represented by the following adjacency matrix:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix},$$

where each element of the universe $U = \{1,2,3\}$ is represented by a row and each subset in the subset family $\mathcal{S} = \{S_1 = \{1,3\}, S_2 = \{1,2\}, S_3 = \{2,3\}, S_4 = \{1\}\}$ is represented by a column. In this case, we have $n = 3$, $r = 4$, $g = nr - n + 1 = 10$, $m = (n+2)g = 50$. Then we can construct our ranking matrix in the following form:

$$\Pi = \left[\begin{array}{cccc|cccc|cccc|cccc|cc}
1 & 2 & 11 & 3 & 4 & 5 & \dots & 10 & 31 & 32 & \dots & 38 & 41 & 42 & \dots & 48 & 12 & \dots \\
11 & 1 & 2 & 12 & 21 & 22 & \dots & 27 & 3 & 4 & \dots & 10 & 41 & 42 & \dots & 48 & 13 & \dots \\
1 & 11 & 2 & 12 & 21 & 22 & \dots & 27 & 31 & 32 & \dots & 38 & 3 & 4 & \dots & 10 & 13 & \dots \\
\multicolumn{4}{c}{\Pi_0} & \multicolumn{4}{c}{\Pi_1} & \multicolumn{4}{c}{\Pi_2} & \multicolumn{4}{c}{\Pi_3} & \multicolumn{2}{c}{R}
\end{array}\right].$$

If we choose K columns, we can see that only when K columns covering at least one column in $\{1,2,4\}$ (i.e., covering element 1), at least one column in $\{2,3\}$ (i.e., covering element 2), and at least one column in $\{1,3\}$ (i.e., covering element 3) we can make the benefit no less than $n(m+1-r) = 141$).

3.3.2 Greedy Selection Approximation Algorithm

For $K = 1$, we simply need to find the column of the matrix Π, whose elements have the highest sum (this would be the benefit B). For $K > 1$, we need to select K columns in a set $\mathcal{K}$ such that $B_{\mathcal{K}}$ is as large as possible (we denote by $B_{\mathcal{K}}$ the benefit from a choice of a set of columns $\mathcal{K}$). Alg. 3 describes a straightforward greedy selection algorithm (that we denote by AlgP1), that is sufficient to achieve a constant approximation ratio. Let $\mathcal{K}^*$ be the optimal selection of K columns, and $B^* \triangleq B_{\mathcal{K}^*}$ the optimal benefit achieved by this selection over this problem instance. We have the following theorem.

Theorem 9. *For any P1 problem instance with no side information and full ranking, given*

Algorithm 3 Greedy selection algorithm for P1.

```
**Input**: ranking matrix $\Pi$ and number of columns to select $K$.
**Output**: a set of columns $\mathcal{K}$.
**Initialization**: set $\mathcal{K} = \emptyset$, $B_{\mathcal{K}} = 0$.
**for** $k = 1 : K$ **do**
    $j = \arg\max_{j' \in [m] \setminus \mathcal{K}} B_{\mathcal{K} \cup \{j'\}}$
    find a column $j$ to maximize the benefit given current $k-1$ selected columns $\mathcal{K}$.
    $\mathcal{K} = \mathcal{K} \cup j$.
**end for**
```

that we can make K broadcast transmissions, AlgP1 can achieve an approximation ratio at least $1/(1-(1-\frac{1}{K})^K)$*, namely,*

$$\frac{B_{\mathcal{K}}}{B^*} \geq 1 - (1 - \frac{1}{K})^K. \tag{3.4}$$

Proof. We use $B(k)$ and $\delta(k)$, $k = 1, 2, \ldots, K$, to denote the benefit collected after k steps and the increase of benefit in the k-th step, respectively, using AlgP1. That is:

$$\begin{aligned} &B(1) = \delta(1); \\ &B(2) = B(1) + \delta(2); \\ &\vdots \\ &B_{\mathcal{K}} = B(K) = B(K-1) + \delta(K). \end{aligned} \tag{3.5}$$

We set $B(0) = 0$ and denote by B^* the optimal benefit and by $\mathcal{K}^*$ an optimal selection of K messages (columns of the ranking matrix) to transmit that achieve B^*. For all k, we can bound $\delta(k)$ using the following lemma.

Lemma 7. *In AlgP1, we can bound the increase in benefit in each step by:*

$$\delta(k) \geq \frac{B^* - B(k-1)}{K}, \quad k = 1, 2, \ldots, K. \tag{3.6}$$

Proof. To prove this lemma, we first observe the following. Consider two selections of messages (columns of the ranking matrix) $\mathcal{K}_1$ and $\mathcal{K}_2$, with $\mathcal{K}_1 \subseteq \mathcal{K}_2$. If we add a new column j

to these selections, we get that:

$$B_{\mathcal{K}_1 \cup \{j\}} - B_{\mathcal{K}_1} \geq B_{\mathcal{K}_2 \cup \{j\}} - B_{\mathcal{K}_2}. \tag{3.7}$$

Indeed, when we add message j (column j) to $\mathcal{K}_1$, assume C is the set of clients whose score can be improved. The left hand side of eq. (3.7), denoted by δ_1, is the increase in score due to this set of clients C and to a level determined by their ranking of message j. Because we have $\mathcal{K}_1 \subseteq \mathcal{K}_2$, when we add message j (column j) to $\mathcal{K}_2$, clearly clients not in C cannot achieve a higher score because of j and clients in C cannot achieve a benefit improvement as large as δ_1 (since some of the clients in C may have already higher scores thanks to messages in $\mathcal{K}_2 \backslash \mathcal{K}_1$).

Going back to the proof of (3.6), observe that at the beginning of the k-th step, the benefit difference from the optimal benefit is $B^* - B(k-1)$. From the pigeonhole principle, at least one of the K messages in the optimal set $\mathcal{K}^*$, let us say j, can improve the benefit at least by $\frac{B^*-B(k-1)}{K}$, since a future selection of j can only offer improvements less than or equal to what it can offer in the current stage from (3.7). □

Return back to the proof of Theorem 9. Using Lemma 7 we can prove the bound of our theorem.

$$
\begin{aligned}
B(K) \quad &= B(K-1) + \delta(K) \\
&\geq B(K-1) + \tfrac{B^* - B(K-1)}{K} \\
&= (1 - \tfrac{1}{K})B(K-1) + \tfrac{1}{K}B^* \\
&\geq (1 - \tfrac{1}{K})[(1 - \tfrac{1}{K})B(K-1) + \tfrac{1}{K}B^*] + \tfrac{1}{K}B^* \\
&= (1 - \tfrac{1}{K})^2 B(K-2) + \tfrac{1}{K}[1 + (1 - \tfrac{1}{K})]B^* \\
&\geq \ldots \\
&\geq (1 - \tfrac{1}{K})^{K-1}\tfrac{B^*}{K} + \tfrac{1}{K}[1 + (1 - \tfrac{1}{K}) + (1 - \tfrac{1}{K})^2 + \ldots + (1 - \tfrac{1}{K})^{K-2}]B^* \\
&= (1 - \tfrac{1}{K})^{K-1}\tfrac{B^*}{K} + [1 - (1 - \tfrac{1}{K})^{K-1}]B^* \\
&= [1 - (1 - \tfrac{1}{K})^K]B^* \\
&\geq [1 - \tfrac{1}{e}]B^*,
\end{aligned}
\tag{3.8}
$$

where the first four inequalities hold by repeatedly using Lemma 7; the second to the last equality holds according to the finite geometric series calculation; the last inequality holds according to the inequality $(1 - \frac{1}{K})^K \leq 1/e$. □

From Theorem 9, we can see that for any K, the approximation ratio is bounded by a constant factor $1/(1 - \frac{1}{e}) = 1.58$.

3.3.3 Bounds on the Optimal Benefit B^*

We show upper and lower bounds of optimal benefit B^*.

Theorem 10. *For any P1 instance with m messages, n clients and K transmissions, the optimal benefit B^* is lower bounded by*

$$B^* \geq \frac{Kn(m+1)}{K+1}. \tag{3.9}$$

Moreover, if $n \geq 6K\log(m)$, there exist P1 instances such that the optimal benefit B^* can *be upper bounded by*

$$B^* \leq (1+\delta)\frac{Kn(m+1)}{K+1}, \tag{3.10}$$

where $\delta = \sqrt{\frac{6K\log(m)}{n}}$.

Proof.

Lower bound

We first prove the lower bound by showing that a simple random choice of K columns achieves expected benefit $\mu \triangleq \frac{Kn(m+1)}{K+1}$ and thus, there exists a selection of K columns that achieves at least such benefit, $B \geq \mu$.

Consider an $n \times m$ ranking matrix Π. Assume we select uniformly at random K columns from all m columns, i.e., with equal probability select 1 from all $\binom{m}{K}$ possible selections. We will calculate the expected benefit from this random selection.

For a given selection, if we denote by $X_1, X_2, \ldots, X_n$ the score received by clients $1, 2, \ldots, n$, then the benefit is $B = X_1 + X_2 + \ldots + X_n$. Note that since we randomly select the K columns and the rows can have arbitrary assignments of ranking for each client, the expected scores are equal, namely, $\mathbb{E}X_1 = \mathbb{E}X_2 = \ldots = \mathbb{E}X_n$. Thus it is sufficient to calculate $\mathbb{E}X_1$.

Denote by $\mathcal{K}$ a realization of the K-selection of columns. We next just consider all $(m-K+1)$ possibilities for the score X_1 that client 1 has achieved. We will use the notation $\pi_i^{-1}(j)$ to refer to the message ranked j by client i.

- Case 1: the message ranked 1 is in $\mathcal{K}$, i.e., $\pi_1^{-1}(1) \in \mathcal{K}$, and $X_1 = m$. The probability of $\{\pi_1^{-1}(1) \in \mathcal{K}\}$ is $p_1 = \frac{K}{m}$.
- Case 2: the message ranked 1 is in $[m]\backslash\mathcal{K}$ and the message ranked 2 is in $\mathcal{K}$, i.e., $\{\pi_1^{-1}(2) \in \mathcal{K} \wedge \pi_1^{-1}(1) \notin \mathcal{K}\}$. In this case, $X_1 = m-1$ and occurs with probability $p_2 = \frac{K(m-K)}{m(m-1)}$, where $\frac{K}{m}$ is the probability of $\{\pi_1^{-1}(2) \in \mathcal{K}\}$ and $\frac{m-K}{m-1}$ is the probability of $\{\pi_1^{-1}(1) \notin \mathcal{K}\}$.

Continuing along these lines we get:

- Case j: the messages ranked $1, 2, \ldots, j-1$ are in $[m]\backslash\mathcal{K}$ and the message ranked j is in $\mathcal{K}$, i.e., $\{\pi_1^{-1}(j) \in \mathcal{K} \wedge \pi_1^{-1}(1) \notin \mathcal{K} \wedge \ldots \wedge \pi_1^{-1}(j-1) \notin \mathcal{K}\}$. In this case, $X_1 = m+1-j$, and occurs with probability $p_j = \frac{K(m-K)(m-K-1)\ldots(m-K-j+2)}{m(m-1)(m-2)\ldots(m-j+1)}$, where $\frac{K}{m}$ is the probability of $\{\pi_1^{-1}(j) \in \mathcal{K}\}$, and $\frac{(m-K)(m-K-1)\ldots(m-K-j+2)}{(m-1)(m-2)\ldots(m-j+1)}$ is the probability of $\{\pi_1^{-1}(1) \notin \mathcal{K} \wedge \ldots \wedge \pi_1^{-1}(j-1) \notin \mathcal{K}\}$. Hence, we have

$$\begin{aligned} \mathbb{E}X_1 &= p_1 m + p_2(m-1) + \ldots + p_j(m+1-j) + \ldots + p_{m-K+1}K \\ &= \tfrac{K}{m!} \sum_{j=1}^{m-K+1} [P(m-K, j-1)(m-j+1)!] = \tfrac{K(m+1)}{K+1}, \end{aligned} \tag{3.11}$$

where the third equality holds from Lemma 8 that we provide later after this proof.

Upper bound

We use a probabilistic method to construct a $n \times m$ preference ranking matrix instance Π as follows. Draw a permutation of $[m]$ from all $m!$ possible permutations iid uniformly at random, and assign it to the i-th row of the ranking matrix Π, for all $i = 1, 2, \ldots, n$.

Consider a fixed K-selection of columns, e.g., $\mathcal{K} = [K] = \{1, 2, \ldots, K\}$. Assume X_1, X_2, …, X_n are the scores received by clients $1, 2, \ldots, n$, then the benefit is $B = X_1 + X_2 + \ldots + X_n$. Due to the iid uniform selection of each row of Π, we have that $\mathbb{E}X_1 = \mathbb{E}X_2 = \ldots = \mathbb{E}X_n$. We calculate $\mathbb{E}X_1$, by listing all the possibilities for client 1's (first row of the ranking matrix) rankings and scores. Recall that we use $\pi_1^{-1}(j)$ to denote the column (message) that is ranked j by client 1, and that the score of receiving a ranking j message is $m+1-j$. We have the following $(m-K+1)$ possibilities for X_1.

• Case 1: the message ranked 1 is in $[K]$, i.e., $\{\pi_1^{-1}(1) \in [K]\}$. In this case $X_1 = m$, and the probability of this event equals $p_1 = \frac{K(m-1)!}{m!}$, where K is the number of ways we can have the message ranked 1 selected in $[K]$; $(m-1)!$ is the number of ways we can rank the remaining $(m-1)$ messages; and the total number of possible assignments is $m!$. We underline that each assignment occurs with equal probability.

• Case 2: the message ranked 1 is in $[m]\backslash[K]$ and the message ranked 2 is in $[K]$, i.e., $\{\pi_1^{-1}(2) \in [K] \wedge \pi_1^{-1}(1) \notin [K]\}$. In this case, $X_1 = m-1$, and the associated probability is $p_2 = \frac{KP(m-K,1)(m-2)!}{m!}$, where K is the number of ways we can have the message ranked 2 in $[K]$; $P(m-K, 1)$, the 1-permutation of $m-K$, is the number of ways to assign the message ranked 1 in the remaining $m-K$ positions $K+1, K+2, \ldots, m$; $(m-2)!$ is the number of ways we can assign the remaining $(m-2)$ messages; and the total number of possible assignments is $m!$.

Continuing along these lines we get:

• Case j: the messages ranked $1, 2, \ldots, j-1$ are in $[m]\backslash[K]$ and the message ranked j is in $[K]$, i.e., $\{\pi_1^{-1}(j) \in [K] \wedge \pi_1^{-1}(1) \notin [K] \wedge \ldots \wedge \pi_1^{-1}(j-1) \notin [K]\}$. In this case, $X_1 = m+1-j$, and this event occurs with probability $p_j = \frac{KP(m-K,j-1)(m-j)!}{m!}$, where K is the number of ways we can have the message ranked j in $[K]$; $P(m-K, j-1)$, the j-permutations of $m-K$, is the number of ways we can assign the messages ranked $1, 2, \ldots, j-1$ in the remaining $m-K$ positions $K+1, K+2, \ldots, m$; and $(m-j)!$ is the number of choices for the remaining $(m-j)$ messages. Therefore:

$$\begin{aligned} \mathbb{E}X_1 &= p_1 m + p_2(m-1) + \ldots + p_j(m+1-j) + \ldots + p_{m-K+1}K \\ &= \tfrac{K}{m!}\sum_{j=1}^{m-K+1}[P(m-K, j-1)(m-j+1)!] = \tfrac{K(m+1)}{K+1}, \end{aligned} \tag{3.12}$$

where the third equality holds from Lemma 8.

From the Chernoff bound, we can bound the probability that the benefit is above $B_{UPPER} = (1+\delta)\mu$ (where $\delta = \sqrt{\frac{6K\log(m)}{n}}$):

$$\Pr\{B \geq (1+\delta)\mu\} = \Pr\{\tfrac{B}{m} \geq \tfrac{1}{m}(1+\delta)\mu\} \leq e^{-\frac{\mu\delta^2}{3m}} = e^{-\frac{2K^2(m+1)\log(m)}{m(K+1)}} \triangleq \epsilon, \tag{3.13}$$

where the first equality just normalizes the random variable B, i.e., $\frac{B}{m} = \frac{X_1}{m} + \frac{X_2}{m} + \ldots + \frac{X_n}{m}$, such that $\frac{X_i}{m}$ is between 0 and 1. Note that $\mathbb{E}\frac{B}{m} = \frac{\mu}{m}$.

The inequality $\Pr\{B \geq (1+\delta)\mu\} \leq \epsilon$ implies that for the fixed K-selection of columns, $[K]$, there are at most an ϵ fraction of instances when selecting the matrix Π that can achieve a benefit no less than $(1+\delta)\mu$. Due to the uniform at random selection of Π, given any fixed K-selection, it is also the case that at most ϵ fraction of instances can achieve this benefit $(1+\delta)\mu$. There are in total $\binom{m}{K}$ possible selections of columns. The fraction of instances of matrices Π that can achieve a benefit no less than $(1+\delta)\mu$ given any of the $\binom{m}{K}$ K-selections is at most

$$\tbinom{m}{K}\epsilon < m^K e^{-\frac{2K^2(m+1)\log(m)}{m(K+1)}} = e^{(K\log(m)-\frac{2K^2(m+1)\log(m)}{m(K+1)})} = e^{(-\frac{K\log(m)[(K-1)m+2K]}{m(K+1)})} < 1, \qquad (3.14)$$

which indicates that there must exist instances, such that, for any K-selection, the average score cannot be more than B_{UPPER}. This concludes the proof of the theorem. □

Lemma 8.

$$\frac{1}{m!}\sum_{j=1}^{m-K+1}[P(m-K,j-1)(m-j+1)!] = \frac{m+1}{K+1}, \mathit{for\ any}\ K \leq m. \qquad (3.15)$$

Proof. Change the variable of this equation by setting $h = m - K$. Denote by $H(h)$ the expression on the left hand side of eq. (3.15), i.e.,

$$H(h) = \frac{1}{m!}\sum_{j=1}^{h+1}[P(h,j-1)(m-j+1)!]. \qquad (3.16)$$

To show that $H(h) = \frac{m+1}{m-h+1}$ for any $m \geq h$, we use a mathematical induction method for h and consider m as a parameter.

For $h = 0$, $\frac{1}{m!}P(0,0)m! = 1$ and for $h = 1$, $\frac{1}{m!}(P(1,0)m! + P(1,1)(m-1)!) = 1 + \frac{1}{m}$, eq. (3.15) holds. Assume eq. (3.15) holds for $h \leq m-1$. Now, for $h+1$, we have

$$\begin{aligned} H(h+1) &= \tfrac{1}{m!}\sum_{j=1}^{h+2}[P(h+1,j-1)(m-j+1)!] \\ &= \tfrac{1}{m!}[\tfrac{(h+1)!}{(h+1)!}m! + \sum_{j=2}^{h+2}\tfrac{(h+1)!}{(h+1-j+1)!}(m-j+1)!] \\ &= 1 + \tfrac{h+1}{m(m-1)!}\sum_{l=1}^{h+1}\tfrac{h!}{(h+1-l)!}(m-1-l+1)! \\ &= 1 + \tfrac{(h+1)}{m}\cdot\tfrac{m-1+1}{m-1-h+1} \\ &= \tfrac{m+1}{m-h}, \end{aligned} \qquad (3.17)$$

where the third equality holds due to a change of variable $l = j - 1$ and the fourth equality holds due to the induction hypothesis on h with parameter $m - 1$. Therefore, the equation (3.15) holds. □

We underline that the upper bound does not apply for all instances (it is trivial to create instances where we get $B^* = nm$) but only for the worst case instances. Note that if δ is small the lower and upper bound have the same order of magnitude, which implies that the bounds become tight. Moreover, if the number of clients increases to $n > O(K^3 \log(m))$, then the (worst case instances) upper bound can be simplified to $O(mn(1 - \frac{1}{K}))$, which is close to the optimal nm. In particular, if we consider all possible $n = m!$ clients that have distinct rankings (permutations of $[m]$), then the benefit achieved by *any* K-selection is $\frac{Kn(m+1)}{K+1}$.

Average case analysis

We here assume uniform distribution over the client rankings, and calculate the expected benefits. In particular, the ranking matrix Π is generated as follows: for each row (client ranking), select uniformly and independently a permutation from all $m!$ permutations of $[m]$ (with repetition).

Theorem 11. *The average benefit (averaged over the ranking matrices* Π*) in P1 using the Borda score model satisfies:*

$$\mathbb{E}_{\Pi} B \geq \mu + n\Delta(m, n, K),$$

where $\mu = \frac{Kn(m+1)}{K+1}$, $\Delta(2, n, 1) = \frac{1}{\sqrt{2\pi n}}$ *for even* n, $\Delta(2, n, 1) = \frac{1}{\sqrt{2\pi(n-1)}}$ *for odd* $n \geq 3$, $\Delta(m, n, K) \approx \frac{1}{\sqrt{2\pi n}} \sigma(m, K)$ *for large* n, *and* $\sigma(m, K) = \sqrt{\frac{(m+1)(m-K)K}{(K+1)^2(K+2)}}$ *is the standard deviation of a client's score distribution when we randomly select* K *columns.*

Note that we have already shown in Theorem 10 that the worst case benefit is μ; the above Theorem 11 shows that the average benefit is higher by at least $n\Delta(m, n, K)$.

Proof. We consider a family of instances $\mathcal{I}$, each with m messages and n clients. The ranking matrix Π is generated as follows: for each row, uniformly and independently draw a permutation from all $m!$ permutations of $[m]$ and assign it to the row.

We first show that $\Delta(2, n, 1) = \frac{1}{\sqrt{2\pi n}}$ for even n and $\Delta(2, n, 1) = \frac{1}{\sqrt{2\pi(n-1)}}$ for odd $n \geq 3$.

Consider a $n \times 2$ ranking matrix instance Π in the family $\mathcal{I}$. Define n_1 and n_2 to be the numbers of 1s and 2s in the first column. Obviously, we have $n_1 + n_2 = n$ and n_1 and n_2 are the numbers of 2s and 1s in the second column. Our strategy is to select column 1 if $n_1 \geq n_2$ and to select column 2, otherwise. For even n, the expected benefit with respect to Π is

$$\mathbb{E}B = \sum_{n_1=0}^{n/2} \frac{1}{2^n} \binom{n}{n_1} [2(n-n_1)+n_1] + \sum_{n_1=n/2+1}^{n} \frac{1}{2^n} \binom{n}{n-n_1} [(n-n_1)+2n_1], \tag{3.18}$$

where the first term corresponds to the selection of column 2 and the second term corresponds to the selection of column 1. Here, in the first term, $\binom{n}{n_1}$ is the number of choices for n_1 2s in the second column, resulting in a probability of $\frac{1}{2^n}\binom{n}{n_1}$ that the benefit is $2(n-n_1)+n_1$. The interpretation is similar for the second term. Similarly, for odd n, the expectation of benefit with respect to Π is

$$\mathbb{E}B = \sum_{n_1=0}^{\frac{n-1}{2}} \frac{1}{2^n} \binom{n}{n_1} [2(n-n_1)+n_1] + \sum_{n_1=\frac{n+1}{2}}^{n} \frac{1}{2^n} \binom{n}{n-n_1} [(n-n_1)+2n_1]. \tag{3.19}$$

By simplifying this expression (see Lemma 9), we get that:

$$\mathbb{E}B = \begin{cases} \frac{3n}{2} + \frac{n}{\sqrt{2\pi n}}, & n \text{ even}, \\ \frac{3n}{2} + \frac{n}{\sqrt{2\pi(n-1)}}, & n \geq 3, \text{ odd}. \end{cases} \tag{3.20}$$

We next consider the general term $\Delta(m, n, K)$. The strategy we use here is as follows. We randomly select K columns. If this selection can achieve a benefit no less than $\mu = \frac{Kn(m+1)}{K+1}$, we keep these columns as our selection. If not, we discard these columns, and select columns with a benefit at least μ. This is always possible according to Theorem 10.

Next, we look at the case where the benefit we actually achieve is greater than $\mu + \frac{n}{\sqrt{2\pi n}}\sigma(m, K)$.

For a fixed selection of K columns, if X_i is the score of client i, then the benefit X can be represented as $X = \sum_{i=1}^{n} X_i$. According to the central limit theorem, the distribution of $Y = \frac{X-\mu}{\sqrt{n}\sigma(m,K)}$ is approximately the standard normal distribution $\mathcal{N}(0,1)$. Given our selection algorithm, the benefit is lower bounded by:

$$B \geq \begin{cases} \mu, & X \leq \mu, \\ X, & X > \mu. \end{cases} \tag{3.21}$$

Hence, the expected benefit can be lower bounded by

$$\mathbb{E}B \geq \mu + \Pr\{X > \mu\}\mathbb{E}[X - \mu | X > \mu]. \tag{3.22}$$

The second term $\Pr\{X > \mu\}\mathbb{E}[X - \mu | X > \mu]$ can be approximately calculated as:

$$\Pr\{X > \mu\}\mathbb{E}[X - \mu | X > \mu] \approx \int_0^\infty \sqrt{n}\sigma(m, K) y \phi(y) dy = \frac{n}{\sqrt{2\pi n}}\sigma(m, K), \tag{3.23}$$

where $\phi(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$ is the probability density function of the standard normal distribution. The calculation of $\sigma(m, K)$ is in Lemma 10. From this the theorem follows. □

In addition, we can strictly lower bound the average performance by involving third moment of X_1 and using the Berry-Esseen theorem.

Corollary 1. *The expectation of benefit can be strictly lower bounded by* $\mu + \frac{n}{\sqrt{2\pi n}}\sigma(m, K)$ $(1 - e^{-\frac{m^2}{2}}) - \frac{m^3}{2\sigma^2(m,K)}$.

Proof. The Berry-Esseen theorem states that the difference of distribution of $Y = \frac{(X-\mu)}{\sqrt{n}\sigma(m,K)}$ and the normal distribution can be bounded by

$$|F_Y(y) - \Phi(y)| \leq \frac{\rho}{2\sigma^3(m, K)\sqrt{n}}, \tag{3.24}$$

for all n and y, where $F_Y(y)$ and $\Phi(y)$ are the cumulative distribution functions of Y and the normal distribution, and $\rho = \mathbb{E}|X/n - \mathbb{E}X/n|^3 < m^3$. We also know that $|X - \mu| \leq mn$.

Hence, the term $\Pr\{X > \mu\}\mathbb{E}[X-\mu|X>\mu]$ can be bounded by

$$\begin{aligned} \Pr\{X > \mu\}\mathbb{E}[X-\mu|X>\mu] &\geq \textstyle\int_0^m \sqrt{n}\sigma(m,K)y\phi(y)dy - \sqrt{n}\sigma(m,K)\frac{\rho}{2\sigma^3(m,K)\sqrt{n}} \\ &= \tfrac{n}{\sqrt{2\pi n}}\sigma(m,K)(1-e^{-\frac{m^2}{2}}) - \tfrac{m^3}{2\sigma^2(m,K)}. \end{aligned} \tag{3.25}$$

This proves the corollary. □

Lemma 9. *For $m = 2$ and $K = 1$, we have*

$$\mathbb{E}B = \begin{cases} \frac{3n}{2} + \frac{n}{\sqrt{2\pi n}}, & n \text{ even}, \\ \frac{3n}{2} + \frac{n}{\sqrt{2\pi(n-1)}}, & n \geq 3, \text{ odd}. \end{cases} \tag{3.26}$$

Proof. When n is even, we have

$$\begin{aligned} \mathbb{E}[B] &= \tfrac{1}{2^n}[\sum_{n_1=0}^{n/2}\tbinom{n}{n_1}(2(n-n_1)+n_1) + \sum_{n_2=0}^{n/2}\tbinom{n}{n_2}(2(n-n_2)+n_2) - \tbinom{n}{\frac{n}{2}}(\tfrac{3n}{2})] \\ &= \tfrac{2}{2^n}[\sum_{n_1=0}^{n/2}\tbinom{n}{n_1}(2n-n_1) - \tbinom{n}{n/2}(\tfrac{3n}{4})] \\ &= \tfrac{2}{2^n}[n(2^n + \tbinom{n}{n/2})) - n2^{(n-2)} - \tbinom{n}{n/2}(\tfrac{3n}{4})] \\ &\approx \tfrac{3n}{2} + \tfrac{n}{\sqrt{2\pi n}}, \end{aligned} \tag{3.27}$$

where the last approximation is due to the Stirling's approximation and the third equality holds because of the following two equations:

$$2^n = \sum_{n_1=0}^{n/2}\tbinom{n}{n_1} + \sum_{n_1=n/2+1}^{n}\tbinom{n}{n_1} = 2\sum_{n_1=0}^{n/2}\tbinom{n}{n_1} - \tbinom{n}{n/2}, \tag{3.28}$$

and

$$\sum_{n_1=0}^{n/2}\tbinom{n}{n_1}n_1 = n\sum_{n_1=1}^{n/2}\tfrac{(n-1)!}{(n_1-1)!(n-n_1)!} = n\sum_{n_1'=0}^{n/2-1}\tfrac{(n-1)!}{n_1'!(n-1-n_1')!} = n2^{(n-2)}. \tag{3.29}$$

When $n \geq 3$ is odd, we have

$$\begin{aligned} \mathbb{E}[B] &= \tfrac{1}{2^{(n-1)}}[\sum_{n_1=0}^{(n-1)/2} \tbinom{n}{n_1}(2n-n_1)] \\ &= \tfrac{1}{2^{(n-1)}}[n2^n - (n2^{(n-2)} - \tfrac{n}{2}\tbinom{n-1}{(n-1)/2})] \\ &\approx \tfrac{3n}{2} + \tfrac{n}{\sqrt{2\pi(n-1)}}, \end{aligned} \tag{3.30}$$

where the last approximation is due to the Stirling's approximation and the second equality holds because of the following two equations:

$$2^n = \sum_{n_1=0}^{(n-1)/2} \tbinom{n}{n_1} + \sum_{n_1=(n+1)/2}^{n} \tbinom{n}{n_1} = 2\sum_{n_1=0}^{(n-1)/2} \tbinom{n}{n_1}, \tag{3.31}$$

and

$$\sum_{n_1=0}^{(n-1)/2} \tbinom{n}{n_1} n_1 = n \sum_{n_1=1}^{(n-1)/2} \tfrac{(n-1)!}{(n_1-1)!(n-n_1)!} = n \sum_{n_1'=0}^{(n-3)/2} \tbinom{n-1}{n_1'} = n\tfrac{2^{(n-1)} - \binom{n-1}{(n-1)/2}}{2}. \tag{3.32}$$

□

Lemma 10.

$$\sigma(m, K) = \sqrt{\frac{(m+1)(m-K)K}{(K+1)^2(K+2)}} \tag{3.33}$$

Proof. We already know that the expected value of the score X_1 is $\frac{K(m+1)}{K+1}$, and the distribution of X_1 is as follows:

$$\Pr\{X_1 = m+1-j\} = \frac{KP(m-K, j-1)(m-j)!}{m!}, \; j = 1, 2, \ldots, m+1-K. \tag{3.34}$$

Next, we prove the following result using induction:

$$\mathbb{E}[X_1^2] \;\; = \sum_{j=1}^{m+1-K} \Pr\{X = m+1-j\}(m+1-j)^2 = \tfrac{K^2(m+1)}{K+1} + \tfrac{K(m-K)(m+1)}{K+2}, \tag{3.35}$$

or

$$\tfrac{1}{m!}\sum_{j=1}^{h+1} [P(h, j-1)(m-j+1)!(m-j+1)] = \tfrac{K(m+1)}{K+1} + \tfrac{(m-K)(m+1)}{K+2}. \tag{3.36}$$

We change the variable of this equation by setting $h = m - K$. We denote by $L(h)$ the following expression:

$$L(h) = \frac{1}{m!} \sum_{j=1}^{h+1} [P(h, j-1)(m-j+1)!(m-j+1)]. \tag{3.37}$$

When $h = 0$, the initial condition holds, i.e., $L(0) = m = \frac{m(m+1)}{m+1} + \frac{0(m+1)}{m-0+2}$. Assume that $L(h) = \frac{(m-h)(m+1)}{m-h+1} + \frac{h(m+1)}{K+2}$ holds for all $m > h$. Then, for $h+1$, we have

$$\begin{aligned} L(h+1) &= \tfrac{1}{m!} \sum_{j=1}^{h+2} [P(h+1, j-1)(m-j+1)!(m-j+1)] \\ &= \tfrac{1}{m!} m!m + \tfrac{1}{m!} \sum_{j=2}^{h+2} [\tfrac{(h+1)!}{(h+1-j+1)!}(m-j+1)!(m-j+1)] \\ &= m + \tfrac{h+1}{m} [\tfrac{1}{(m-1)!} \sum_{l=1}^{h+1} [\tfrac{h!}{(h+1-l)!}(m-1-l+1)!(m-1-l+1)]] \\ &= m + \tfrac{h+1}{m} [\tfrac{(m-1-h)m}{m-h} + \tfrac{hm}{m-h+1}] \\ &= \tfrac{(m-h-1)(m+1)}{m-h} + \tfrac{(h+1)(m+1)}{m-h+1}, \end{aligned} \tag{3.38}$$

where the third equality holds due to a change of variable $l = j - 1$ and the fourth equality holds due to the induction hypothesis on h with parameter $m - 1$. Therefore eq. (3.35) is proved.

Furthermore, we can calculate $\sigma^2(m, K) = \mathbb{E}[X_1^2] - \mu^2 = \frac{(m+1)(m-K)K}{(K+1)^2(K+2)}$. □

3.4 Equal Size Side Information (P2)

We now look at the case where all clients have side information of the same size $|S_i| = m - r$ and thus $|R_i| = r$, $\forall\ i$. We again assume Borda count scores.

3.4.1 Bounds on the Optimal Benefit B^*

We are interested in the optimal benefit B^* we can achieve with K transmissions (recall that $0 \leq B \leq nr$). We next prove a lower bound on the performance of the optimal algorithm through dynamic programming.

Theorem 12. *The optimal benefit B^* satisfies*

$$B^* \geq \begin{cases} \frac{nr}{4e} & \textit{for} \quad K = 1, \\ \frac{nr}{e}(\frac{1}{2} - \frac{1}{8e} + \frac{1}{16e^2}) & \textit{for} \quad K = 2, \\ \frac{nr}{e}(\frac{3}{4} - \frac{1}{4e} + \frac{1}{16e^2}) & \textit{for} \quad K = 3, \\ \frac{nr}{e}(1 - \frac{5}{8e} + \frac{13}{64e^2}) & \textit{for} \quad K = 4, \\ nr(1 - \frac{4e}{K} + \frac{12e}{K^2}) & \textit{for} \quad 5 \leq K < r \\ nr & \textit{for} \quad K = r. \end{cases} \tag{3.39}$$

The proof of this theorem is constructive: we provide a randomized algorithm and show that it achieves on average the performance prescribed in the theorem, which implies that the optimal performance can only be better. The approximation ratio of this scheme is $O(1)$, as the best achievable benefit is nr. We also note that if $K = r$, we can easily achieve $B = nr$: the server can use an MDS erasure correcting code to create r linear combinations to transmit, so that each client using her side information can solve for the r messages she misses. We next show the proof of the theorem.

Proof. We first show the strategy for one transmission and a claim to characterize the benefit. For $K = 1$, assume that the server makes the transmission $x_1 = a_1b_1 + a_2b_2 + \dots a_mb_m$ where a_j, $j \in [m]$, are the constant coding coefficients (we will call the vector $\boldsymbol{a} = (a_1, a_2, \dots, a_m)$ the coding vector). Assume we select iid random values for the coding coefficients, setting $a_j = 1$ with probability $1/r$, and $a_j = 0$ otherwise.

Claim 2. *There exists a binary coding vector $\boldsymbol{a}^\xi$ that enables at least $\frac{n\xi}{re}$ clients to decode a message in their request set they have ranked less than or equal to ξ, for $\xi = 1, 2, \dots, r$, and thus to achieve a benefit of at least $\frac{(r+1-\xi)n\xi}{re}$.*

Proof. Without loss of generality, assume that client i has the request set $R_i = \{b_{l_1}, b_{l_2}, \dots, b_{l_r}\}$ with ranking $\pi(l_1) = 1,\ \pi(l_2) = 2, \dots, \pi(l_r) = r$. Client i can decode a message with rank at least ξ, i.e., can decode some b_{l_j} with $j \leq \xi$, if and only if $a_{l_j} = 1$ and $a_{l_1} = a_{l_2} =$

$\ldots = a_{l_{j-1}} = a_{l_{j+1}} = \ldots = a_{l_r} = 0$. Indeed, we can then express the server transmission as $x_1 = b_{l_j} + \sum_{l \in S_i} a_l b_l$; client i can remove from x_1 the part $\sum_{l \in S_i} a_l b_l$ using her side information, and decode b_{l_j}. The probability that such an event happens is:

$$\binom{\xi}{1} \frac{1}{r} (1 - \frac{1}{r})^{r-1} \leq \frac{\xi}{re} \triangleq p_\xi. \tag{3.40}$$

Hence, randomly selecting a coding vector would enable on average $np_\xi = \frac{n\xi}{re}$ clients to decode messages of rank no more than ξ, and thus, from the averaging principle, there exists at least one coding vector $\boldsymbol{a}^\xi$ that also enables this. □

Return back to the proof of the theorem for $K > 1$. We consider the following dynamic programming problem with K stages, each corresponding to one transmission. At stage k, $1 \leq k \leq K$, the server can select one of r actions, which of the r possible $\boldsymbol{a}^\xi$ vectors to use. In particular, we proceed as follows:

– At the beginning of stage 1, no transmission has yet been made and there are $n_1 = n$ clients in the system. The server chooses an action $\xi_1 \in [r]$, i.e., uses the coding vector $\boldsymbol{a}^{\xi_1}$ to make a transmission. From Claim 2, this transmission enables $\frac{n_1 \xi_1}{re}$ clients to decode a message that they have ranked less than or equal to ξ_1, and thus, we can achieve a benefit $B_1 \geq \frac{(r+1-\xi_1)n\xi_1}{re}$. We remove these $\frac{n_1\xi_1}{re}$ clients from the system and denote the remaining number of clients by $n_2 = n_1(1 - \frac{\xi_1}{re})$.

– At the beginning of stage 2, we only consider the n_2 clients; similarly to before, the server chooses an action $\xi_2 \in [r]$ to enable $\frac{n_2\xi_2}{re}$ clients decode a message ranked less than or equal to ξ_2. At this point we have achieved benefit $B_2 \geq B_1 + \frac{(r+1-\xi_2)n_2\xi_2}{re}$. We remove these $\frac{n_2\xi_2}{re}$ clients from the system and denote the remaining number of clients by $n_3 = n_2(1 - \frac{\xi_2}{re})$.

– Continuing along the same lines, at the beginning of stage $k = 3, 4, \ldots, K$, we have n_k clients to consider; the server chooses an action ξ_k that enables to achieve befit $B_k \geq B_{k-1} + \frac{(r+1-\xi_k)n_k\xi_k}{re}$; and we set $n_{k+1} = n_k(1 - \frac{\xi_k}{re})$.

Let $J_k(n_k)$ be the benefit the n_k clients can receive for the *remaining* $K + 1 - k$ stages.

We have the following Bellman equation:

$$J_k(n_k) = \max_{\xi_k \in [r]} \{(r+1-\xi_k)\frac{\xi_k n_k}{re} + J_{k+1}(n_k(1-\frac{\xi_k}{re}))\}, \tag{3.41}$$

with $J_{K+1}(n_{K+1}) = 0$ (as we will only make K transmissions).

From the above equation, we can see that the benefit achieved using this scheme is $B = J_1(n)$.

- If $K = 1$, we set $\xi_1 = \frac{r}{2}$ (r even) and $\xi_1 = \frac{r+1}{2}$ (r odd), and get $J_1(n) \geq \frac{rn}{4e}$.

- If $K = 2$, we set $\xi_1 = \lceil \frac{r}{2}(1-\frac{1}{4e}) \rceil$ and $\xi_2 = \lceil \frac{r}{2} \rceil$, and get $J_1(n) \geq \frac{rn}{e}(\frac{1}{2} - \frac{1}{8e} + \frac{1}{16e^2})$.

- If $K = 3$, we set $\xi_1 = \lceil \frac{r}{2}(1-\frac{1}{2e}) \rceil$, $\xi_2 = \lceil \frac{r}{2}(1-\frac{1}{4e}) \rceil$ and $\xi_3 = \lceil \frac{r}{2} \rceil$, and get $J_1(n) \geq \frac{rn}{e}(\frac{3}{4} - \frac{1}{4e} + \frac{1}{16e^2})$.

- If $K = 4$, we set $\xi_1 = \lceil \frac{r}{2}(1-\frac{3}{4e}) \rceil$, $\xi_2 = \lceil \frac{r}{2}(1-\frac{1}{2e}) \rceil$, $\xi_3 = \lceil \frac{r}{2}(1-\frac{1}{4e}) \rceil$ and $\xi_4 = \lceil \frac{r}{2} \rceil$, and get $J_1(n) \geq \frac{rn}{e}(1 - \frac{5}{8e} + \frac{13}{64e^2})$.

- For $K > 4$, the theorem follows from Claim 3 that by setting $k = 1$, we get $J_1(n) \geq rn(1 - \frac{4e}{K} + \frac{12e}{K^2})$. □

Claim 3. *$J_\tau(n) \geq nr(1 - \frac{4e}{K+1-k} + \frac{12e}{(K+1-k)^2})$ for $K > 4$.*

Proof. We use the backward induction method for the last 5 stages. By setting $\xi_{K-4} = \lceil \frac{r}{2}(1 - \frac{1}{e} + \frac{5}{8e^2}) \rceil$, $\xi_{K-3} = \lceil \frac{r}{2}(1-\frac{3}{4e}) \rceil$, $\xi_{K-2} = \lceil \frac{r}{2}(1-\frac{1}{2e}) \rceil$, $\xi_{K-1} = \lceil \frac{r}{2}(1-\frac{1}{4e}) \rceil$ and $\xi_K = \lceil \frac{r}{2} \rceil$, we can get $J_{K-4}(n_{K-4}) \geq \frac{rn_{K-4}}{e}(\frac{5}{4} - \frac{9}{8e} + \frac{39}{64e^2}) \doteq 0.33rn_{K-4}$. Therefore, we have the initial condition:

$$\begin{aligned} J_{K-4}(n_{K-4}) &\doteq 0.33rn_{K-4} \\ &\geq rn_{K-4}(1 - \tfrac{4e}{K+1-(K-4)} + \tfrac{12e}{(K+1-(K-4))^2}) \doteq 0.13rn_{K-4}. \end{aligned}$$

Assume that $J_{k+1}(n_{k+1}) \geq rn_{k+1}(1 - \frac{4e}{K+1-(k+1)} + \frac{12e}{(K+1-(k+1))^2})$ holds for $k+1$, then consider

$J_k(n_k)$ $(k < K-4)$:

$$\begin{aligned} J_k(n_k) \quad &= \max_{\xi_k \in [r]}\{(r+1-\xi_k)\tfrac{\xi_k n_k}{re} + J_{k+1}(n_k(1-\tfrac{\xi_k}{re}))\} \\ &\geq \max_{\xi_k \in [r]}\{(r+1-\xi_k)\tfrac{\xi_k n_k}{re} + rn_k(1-\tfrac{\xi_k}{re})(1-\tfrac{4e}{K+1-(k+1)}+\tfrac{12e}{(K+1-(k-1))^2})\} \\ &\geq rn_k(1-\tfrac{4e}{K+1-k}+\tfrac{12e}{(K+1-k)^2}), \end{aligned}$$

where the first inequality holds due to the hypothesis and the property of the Bellman equation; the second inequality holds by setting ξ_k to be an integer between $\xi' = \frac{2re}{K-k} - \frac{6re}{(K-k)^2} + 1/2$ and $\xi'' = \frac{2re}{K-k} - \frac{6re}{(K-k)^2} - 1/2$. If we define $f(\xi) = (r+1-\xi)\frac{\xi n_k}{re} + rn_k(1-\frac{\xi}{re})(1-\frac{4e}{K+1-(k+1)}+\frac{12e}{(K+1-(k+1))^2})$, then we have $f(\xi_k) \geq \min\{f(\xi'), f(\xi'')\} \geq rn_k(1-\frac{4e}{K+1-k}+\frac{12e}{(K+1-k)^2})$. Therefore, Claim 3 holds. □

3.4.2 Algorithms for Problem P2

We base our proposed algorithm (that we term AlgP2) in this case, as shown in Alg. 4, on the randomized algorithm described in the proof of Theorem 12 that operates in rounds, and in each round selects what coding vector to transmit so as to satisfy a certain fraction of clients. The only random step in this algorithm is the selection of a binary coding vector in Claim 2; however, we can easily derandomize it using a deterministic algorithm in polynomial time: we sequentially visit the entries of the coding vector and decide whether to assign value 0 or 1 depending on how the benefit would increase, as described in detail as follows.

Derandomization Function for Claim 2

We here describe a polynomial-time deterministic algorithm to select a coding vector $\boldsymbol{a}^{\xi}$. We refer to the clients that can decode a message they have ranked less than or equal to ξ as the *qualified clients*. For a given coding vector $\boldsymbol{a}$, we denote the number of qualified clients by $Y[\boldsymbol{a}]$.

We sequentially assign a coding coefficient 0 or 1 to the m coding coefficients in the vector $\boldsymbol{a} = [a_1\ a_2\ \ldots a_m]$ in m steps. At the beginning of the j-th step, the first $j-1$ coefficients have been assigned some values $a_1 = \bar{a}_1,\ a_2 = \bar{a}_2, \ldots, a_{j-1} = \bar{a}_{j-1}$.

We define $Y_{\bar{\boldsymbol{a}}_{[j-1]},0} = \mathbb{E}_{\boldsymbol{a}} Y[\boldsymbol{a} | a_1 = \bar{a}_1, a_2 = \bar{a}_2, \ldots, a_{j-1} = \bar{a}_{j-1}, a_j = 0]$ to be the expected

Algorithm 4 Dynamic programming algorithm for solving P2.

```
**Input**: number of messages $m$, number of clients $n$, request sets $R_i, \forall i \in [n]$, ranking $\pi_i(j), \forall i \in [n], j \in R_i$, size of request set $r$, and number of selections $K$.
**Output**: coding matrix $\boldsymbol{A} \in \{0,1\}^{K \times m}$.
**Initialization**: set the client set $\mathcal{N} = [n]$;
for $k = 1 : K$ do
    if $k = K$ then
        Set ranking threshold $\xi_k = \lceil \frac{r}{2} \rceil$.
    else if $k = K - 1$ then
        Set ranking threshold $\xi_k = \lceil \frac{r}{2}(1 - \frac{1}{4e}) \rceil$.
    else if $k = K - 2$ then
        Set ranking threshold $\xi_k = \lceil \frac{r}{2}(1 - \frac{1}{2e}) \rceil$.
    else if $k = K - 3$ then
        Set ranking threshold $\xi_k = \lceil \frac{r}{2}(1 - \frac{3}{4e}) \rceil$.
    else if $k = K - 4$ then
        Set ranking threshold $\xi_k = \lceil \frac{r}{2}(1 - \frac{1}{e} + \frac{5}{8e^2}) \rceil$.
    else
        Set ranking threshold $\xi_k = \lceil \frac{2re}{K-k} - \frac{6re}{(K-k)^2} - \frac{1}{2} \rceil$.
    end if
    Find a row coding vector $\boldsymbol{a}_k$ as the $k$-th row of $\boldsymbol{A}$ with respect to the ranking threshold $\xi_k$ and clients $\mathcal{N}$ using derandomization function Alg. 5.
    Remove all $i$ from $\mathcal{N}$, if $i \in \mathcal{N}$ is a qualified client, given coding vector $\boldsymbol{a}_k$.
end for
```

number of qualified clients, averaged over all coding vectors with the assigned values for the first $j-1$ coding coefficients $\bar{\boldsymbol{a}}_{[j-1]}$ and a 0 for the j-th coding coefficient; $Y_{\bar{\boldsymbol{a}}_{[j-1]},1} = \mathbb{E}_{\boldsymbol{a}} Y[\boldsymbol{a} | a_1 = \bar{a}_1, a_2 = \bar{a}_2, \ldots, a_{j-1} = \bar{a}_{j-1}, a_j = 1]$ to be the expected number of qualified clients, averaged over all coding vectors with the assigned values for the first $j-1$ coding coefficients $\bar{\boldsymbol{a}}_{[j-1]}$ and a 1 for the j-th coding coefficient; and $Y_{\bar{\boldsymbol{a}}_{[j]}} = \mathbb{E}_{\boldsymbol{a}} Y[\boldsymbol{a} | a_1 = \bar{a}_1, a_2 = \bar{a}_2, \ldots, a_j = \bar{a}_j]$ to be the expected number of qualified clients, averaged over all coding vectors with the assigned values $\bar{\boldsymbol{a}}_{[j]}$ for the first j coefficients. We also use $\boldsymbol{a}_{[0]}$ to refer the empty set.

The algorithm proceeds as follows: *For step $j = 1, 2, \ldots, m$, assign the j-th coding coefficient $\bar{a}_j$ to be* 1 *if $Y_{\bar{\boldsymbol{a}}_{[j-1]},1} \geq Y_{\bar{\boldsymbol{a}}_{[j-1]},0}$ and* 0 *otherwise.*

In the derandomization function flow diagram, the steps 8-12 essentially implement the calculation of the expected values we use for the decision making. We track the probability that a client i will be a qualified client, p_i^{j-1}, for each step $j-1$. Then we choose the coding coefficient a_j by comparing the expected numbers of qualified clients if we choose a 0 and a 1 for a_j. We set a state parameter z_i^j to represent the number of coefficient assignment patterns for the remaining messages in R_i such that client i can remain qualified. For example, $z_i^j = 3$ if $|\{j' \in R_i | j' > j, \pi_i(j) \leq \xi, a_{j''} = 0, \forall j'' \in R_i \text{ and } j'' \leq j\}| = 3$.

We here argue that the coding vector $\bar{\boldsymbol{a}}$ we identify enables at least $\frac{n\xi}{re}$ clients to be qualified, i.e., decode a message in their request set they have ranked less than or equal to ξ. Let $Y_{\bar{\boldsymbol{a}}}$ be the qualified clients after the derandomized function, and let Y be the qualified clients after the randomized selection in Claim 2, where we iid at random assigned value 0 to each coding coefficient with probability p. Tracking the qualified clients, originally we have $Y = (1-p)Y_{\emptyset,0} + pY_{\emptyset,1}$; then $Y_{\emptyset,0} \geq Y$ or $Y_{\emptyset,1} \geq Y$ holds; and hence we have $Y_{\bar{\boldsymbol{a}}_{[1]}} \geq Y$. For step j, we can see that $Y_{\bar{\boldsymbol{a}}_{[j]}} = (1-p)Y_{\bar{\boldsymbol{a}}_{[j-1]},0} + pY_{\bar{\boldsymbol{a}}_{[j-1]},1}$. Hence, at least one of the two following inequalities, $Y_{\bar{\boldsymbol{a}}_{[j-1]},0} \geq Y_{\bar{\boldsymbol{a}}_{[j]}}$ and $Y_{\bar{\boldsymbol{a}}_{[j-1]},0} \geq Y_{\bar{\boldsymbol{a}}_{[j]}}$, holds. Therefore, using the derandomization function, we have $Y_{\bar{\boldsymbol{a}}_{[j]}} \geq Y_{\bar{\boldsymbol{a}}_{[j-1]}}$. Hence, $Y_{\bar{\boldsymbol{a}}} \geq Y \geq \frac{n\xi}{re}$ holds.

3.4.3 Benefits of Coding

As is the case in index coding, leveraging side information enables to use coding and convey through the same transmission different messages to clients. We next compare, over two sets of instances, the ratio between the benefit we get when we leverage side information and the benefit we get when we do not.

- *Ratio of* $\frac{n}{K}$. Assume that for each pair of clients i_1 and i_2, the request sets R_{i_1} and R_{i_2} do not overlap, i.e., $R_{i_1} \cap R_{i_2} = \emptyset$ for all $i_1 \neq i_2$. Assume that each client receives a maximum score of r if she can decode her most preferred message. In this case, the best uncoded K selections are to choose the K messages such that K clients receive the maximum score, achieving benefit Kr. With K encoded transmissions each client can decode her most preferred message. Therefore, the ratio is $nr/Kr = n/K$.
- *Ratio of* $\frac{2n}{K(r+1)}$. Consider an instance with m messages and $n = m$ clients. All clients have a request set of the same size, i.e., $|R_i| = r < K$, $\forall i$. The clients are partitioned in groups: for any two clients i_1 and i_2 in different groups, their request sets R_{i_1} and R_{i_2} do not overlap, i.e., $R_{i_1} \cap R_{i_2} = \emptyset$; for any two clients i_1 and i_2 in the same group, their request sets

Algorithm 5 Derandomization Function.

```
Input: number of messages $m$, number of clients $n$, request sets $R_i, \forall i \in [n]$, ranking $\pi_i(j), \forall i \in [n], j \in R_i$, size of request set $r$, and ranking threshold $\xi$.
Output: coding vector $\boldsymbol{a} \in \{0,1\}^m$.
Initialization: set the client set $\mathcal{N} = [n]$; set the qualification probability $p_i^0 = \frac{\xi}{r}(1 - \frac{1}{r})^{r-1}$, for all client $i \in [n]$; set the state $z_i^0 = \xi$ for each client $i \in [n]$.
for $j = 1 : m$ do
    for all $i \in \mathcal{N}$ do
        $p^j_{i,0} = p^j_{i,1} = p^{j-1}_i$; $z^j_{i,0} = z^j_{i,1} = z^{j-1}_i$, for all $j \notin R_i$. // Not affect for $j \notin R_i$.
        if $j \in R_i$ and $\pi_i(j) \le \xi$ then
            // This is the case that client $i$ ranks $j$ no more than $\xi$.
            // Update the probability that client $i$ can be qualified if $a_j$ is 0 or 1:
            $$p^j_{i,0} = \begin{cases} 0, & \text{if } z_i^{j-1} = 1 \text{ and } a_{j'} = 0 \text{ for all } j' < j \text{ and } j' \in R_i, \\ \frac{p_i^{j-1}}{1-1/r}, & \text{if } z_i^{j-1} = 1 \text{ and } a_{j'} = 1 \text{ for some } j' < j \text{ and } j' \in R_i, \\ \frac{(1-1/z_i^{j-1})p_i^{j-1}}{1-1/r}, & \text{otherwise;} \end{cases}$$
            $$p^j_{i,1} = \begin{cases} 0, & \text{if } a_{j'} = 1 \text{ for some } j' < j \text{ and } j' \in R_i, \\ \frac{r p_i^{j-1}}{z_i^{j-1}}, & \text{otherwise.} \end{cases}$$
            // Update the state of client $i$ if $a_j$ is 0 or 1:
            $$z^j_{i,0} = \begin{cases} 0, & \text{if } z_i^{j-1} = 1 \text{ and } a_{j'} = 0 \text{ for all } j' < j \text{ and } j' \in R_i, \\ 1, & \text{if } z_i^{j-1} = 1 \text{ and } a_{j'} = 1 \text{ for some } j' < j \text{ and } j' \in R_i, \\ z_i^{j-1} - 1, & \text{otherwise;} \end{cases}$$
            $$z^j_{i,1} = \begin{cases} 0, & \text{if } a_{j'} = 1 \text{ for some } j' < j \text{ and } j' \in R_i, \\ 1, & \text{otherwise.} \end{cases}$$
        end if
        if $j \in R_i$ and $\pi_i(j) > \xi$ then
            // This is the case that client $i$ ranks $j$ more than $\xi$.
            // Update the probability that client $i$ can be qualified if $a_j$ is 0 or 1:
            $$p^j_{i,0} = \frac{p_i^{j-1}}{1 - 1/r}; \quad p^j_{i,1} = 0.$$
            // Update the state of client $i$ if $a_j$ is 0 or 1:
            $$z^j_{i,0} = z^j_i; \quad z^j_{i,1} = 0.$$
        end if
    end for
    if $\sum_{i \in \mathcal{N}: j \in R_i} p^j_{i,1} \ge \sum_{i \in \mathcal{N}: j \in R_i} p^j_{i,0}$ then
        Set $a_j = 1$, $p_i^j = p^j_{i,1}$, and $z_i^j = z^j_{i,1}$.
    else
        Set $a_j = 0$, $p_i^j = p^j_{i,0}$, and $z_i^j = z^j_{i,0}$.
    end if
    Remove $i$ from $\mathcal{N}$, if $z_i^j = 0$ for all $i \in \mathcal{N}$.
end for
```

R_{i_1} and R_{i_2} are the same, i.e., $R_{i_1} = R_{i_2}$. In each group, the number of clients equals to the cardinality of the request set r, and thus we have r clients requiring r messages. We assign the associated ranking submatrix of each group to be a Latin square, i.e., each required message is ranked differently by these r clients, from 1 to r. Hence, the ranking submatrix has no same elements in the same row or in the same column. Therefore, for the uncoded K selections, this instance will give a total score of $\frac{r(r+1)K}{2}$. For the coded K selections, this instance will give a total score of nr, when we use MDS coding scheme to send all the missing messages. Hence, the ratio is $\frac{2n}{K(r+1)}$.

3.5 Arbitrary Size Side Information (P3)

We are here given as input the score $s_i(j) = w_{ij}$ that client i has for message j, and make no assumptions on the size of the side information set. This is the most general case that admits P1 and P2 as special cases. We solve this problem using a mapping to the Maximum Weighted Independent Set (MWIS) problem[1].

3.5.1 Mapping to the MWIS Problem

Assume that the server uses a binary coding vector $\boldsymbol{a} = (a_1, \ldots, a_m)$ to make a transmission $x = a_1 b_1 + \ldots + a_m b_m = b_{j_1} + b_{j_2} + \ldots + b_{j_l}$, where the indices $j_1, j_2, \ldots, j_l$ correspond to the nonzero coding coefficients.

A client i can decode the message b_{j_1} from x if and only if this is the only message appearing in x that she does not have; that is, b_{j_1} belongs in her request set ($j_1 \in R_i$) and she has already as side information the rest of the messages appearing in x ($j_2, \ldots, j_l \in S_i$). Consider now the $|R_i|$ positions in the coding vector $\boldsymbol{a}$ that correspond to the $|R_i|$ messages client i does not have. There are $|R_i|$ possible choices of coding coefficients for these positions, so that client i can decode one of these message: making exactly one of these coefficients

[1] The MWIS problem is the weighted version of the maximum independent set problem. It aims to find a set of vertices that have the maximum weighted sum in a given graph. For details of the problem, see, for example, [STY03].

one, and the remaining $|R_i| - 1$ zero. If we were to depict these coefficients sequentially, the choices are $(1, 0, 0, \ldots, 0)_i, (0, 1, 0, \ldots, 0)_i, \ldots, (0, \ldots, 0, 1)_i$, where we used the subscript i to express that these correspond to the messages in R_i. Client i can decode the first message in R_i under the first assignment, the second message under the second assignment, etc. We call these $|R_i|$ assignments the assignments for client i.

We map each of the $|R_i|$ assignments for client i, for all $i \in [n]$, to a vertex in the MWIS instance; thus in total we create $\sum_{i\in[n]} |R_i|$ vertices. We assign weight w_{ij} to the vertex corresponding to the assignment that enables client i to decode message j. We connect two vertices with an edge if the corresponding assignments cause conflict with each other: that is, there exists at least one common message to which one vertex assigns coefficient 0 and the other coefficient 1. Vertices corresponding to assignments of the same client i are pairwise connected, forming a clique, since these assignments are mutually exclusive. Vertices corresponding to assignments of different clients may be connected or not. For example, if $R_1 = \{1, 2\}$ and $R_2 = \{2, 3, 4\}$, there are 5 assignments corresponding to clients 1 and 2, denoted as $(1, 0)_1, (0, 1)_1, (1, 0, 0)_2, (0, 1, 0)_2, (0, 0, 1)_2$. The vertex $(1, 0)_1$ is connected to $(0, 1)_1$ and $(1, 0, 0)_2$, where the latter is because $(1, 0)_1$ assigns a coding coefficient 0 to message 2 and $(1, 0, 0)_2$ assigns a coding coefficient 1 to message 2, resulting in a conflict.

Given that each vertex of this graph specifies part of a coding vector, an independent set specifies (perhaps in part) a feasible coding vector that enables all clients with a vertex in this independent set to decode a message. Thus, finding a MWIS enables to construct a coding vector that leads to the maximum benefit.

3.5.2 Algorithms for Problem P3

Algorithm for $K = 1$

Given the MWIS connection, we can now translate any of the MWIS solvers to an algorithm for our problem when $K = 1$. As an example, the following theorem presents the score achievable by the MWIS polynomial time approximation algorithm in [STY03].

Theorem 13. *For problem P3, with $K = 1$ transmission, we can achieve a benefit of at least*

$\frac{W}{2(d_1-1)d_2+1}$ in polynomial time, where $W = \sum_{(i,j):j\in R_i} w_{ij}$ is the total weight of the instance, $d_1 = \max_{i\in[n]}\{R_i\} \leq m$, and $d_2 \leq n$ is the maximum number of request sets a message can belong to.

Proof. The proof follows by observing that maximum degree of each vertex in the graph is at most $2(d_1 - 1)d_2$, and directly applying Theorem 3.4 in [STY03].

Indeed, consider a vertex v that enables client i to decode message $j \in R_i$. This vertex is connected to the remaining $|R_i| - 1$ vertices of the same client, which contributes to v degree at most $d_1 - 1$. Now consider another client $i' \neq i$. If $j \in R_{i'}$, then only one of the assignments for client i' does not have a conflict with v, the one that enables client i' to decode j. Thus counting each $i' \neq i$ we may have additional degree of at most $(d_2-1)(d_1-1)$. Finally, consider $j' \in R_i \cap R_{i'}$ for some $j' \neq j$. In this case, the vertex v' that enables client i' to decode message j' will be connected to v, since v' needs the coefficient of message j' to be 1 and v requires the coefficient of message j' to be 0. This last case contributes additional degree of at most $(d_1 - 1)d_2$. □

Algorithm for general K

This algorithm applies for general K and operates in K iterations, in each iteration simply solving one instance of a MWIS problem. We presented the algorithm (that we term AlgP3) in Alg. 6. In the first iteration, we solve the MWIS described earlier to select the transmission the server makes. Next, we update the problem instance: (i) we add decoded messages into side-information sets, and (ii) if client i has decoded message j, we set $w_{ij'} = \max\{0, w_{ij'} - w_{ij}\}$ for all $j' \in R(i)$, to reflect the additional benefit that receiving message j' would bring to client i given that she has already received j. We proceed with the next iteration by solving the MWIS problem on the new instance. Observe that this scenario admits the index coding problem as a special case, where each client requires one message with score 1 and others with score 0. The hardness of approximating index coding capacity is shown through a reduction from the maximum independent set (MIS) problem [LS11]. Here, we use a similar reduction to show the hardness of approximating B^* as follows.

Proposition 1. *The P3 problem is hard (unless $NP = ZPP$) to approximate within a ratio of $n^{1-\epsilon}$ for any $\epsilon > 0$.*

Proof. For this we simply use the result that the MIS is hard (unless $NP = ZPP$) to approximate within a ratio of $n^{1-\epsilon}$ for any $\epsilon > 0$ [Has96]. We map an MIS instance on a graph $G = (V, E)$, into a P3 problem as follows.

- We create a P3 instace with $m = n = |V|$: we map each of the $|V|$ vertices in G to a client in P3; and we also create $|V|$ messages.
- A client i has message i in her request set with score $w_{ii} = 1$.
- A client i has a message $j \neq i$ in her request set R_i with score $w_{ij} = 0$ if and only if there is an edge between vertex i and vertex j in G. All the remaining messages are in the side information set of client i.

We next argue that the size of the MIS in G equals the maximum benefit that we can achieve over the constructed P3 instance if we are restricted to $K = 1$ transmission. Indeed, given an independent set S in G, we can construct a coding vector for P3 that enables each client i in S to decode message i (we simply use coding coefficients 1 for all such $i \in S$ and 0 for the remaining coding coefficients). Recall that with one transmission, a client can decode a message i in her request set if and only if the coding coefficient for this message is nonzero and the coding coefficients for all other messages in her request set are zero. This would achieve benefit $|S|$ in P3.

We argue that this is the maximum benefit we could achieve in P3: indeed, if a larger benefit was possible in P3, more than $|S|$ clients i would have been able to decode their corresponding message i. Note also that, a coding vector that achieves a maximum benefit B^*, enables B^* clients i to decode their corresponding message i, and thus directly determines an independent set of size $|S| = B^*$ in G. □

Algorithm 6 Greedy Coding Algorithm to Solve P3.

```
Input: number of messages $m$, number of clients $n$, request sets $R_i, \forall i \in [n]$, weights $w_{i,j}, \forall i \in [n], j \in R_i$, number of encoded messages $K$.
Output: $K$ encoded messages $\{x_1, x_2, \ldots, x_K\}$.
Initialization: problem instance $\mathcal{I} = (m, n, \{R_i\}_{i\in[n]}, \{w_{ij}\}_{i\in[n],j\in R_i})$, received score $s(i) = 0, \forall i \in [n]$.
for $k = 1 : K$ do
    Map the instance $\mathcal{I}$ into an MWIS instance $\mathcal{J}$.
    Solve the MWIS problem $\mathcal{J}$ and get output $\{j_1, j_2, \ldots, j_l\}$.
    Set the $k$-th encoded message to be $x_k = b_{j_1} + b_{j_2} + \ldots + b_{j_l}$.
    Update the instance $\mathcal{I}$:
    if Client $i$ ($\forall i \in [n]$) can decode message $j \in R_i$ then
        Move $j$ from $R_i$ to $S_i$.
        if $w_{ij} > 0$ then
            Update score: $s(i) = w_{ij}$.
            Set $w_{ij'} = 0$ for all $j' \in R_i$ and $w_{ij'} \leq w_{ij}$.
            Set $w_{ij'} = w_{ij'} - w_{ij}$ for all $j' \in R_i$ and $w_{ij'} > w_{ij}$.
        end if
    end if
end for
```

3.6 Numerical Evaluation

3.6.1 Over Random Instances

For Figs. 3.1-3.8, we uniformly at random generate instances with the parameters described in the captions, and present values averaged over all instances. In the following experiments, we normalize the benefit B to get B_0 (divided by the maximum benefit possible to have maximum value 1).

Trade-off between B and K Figs. 3.1 and 3.2 show for P1 (AlgP1) and P2 (AlgP2) the trade-off between the number of transmissions K and the normalized benefit B_0. We consistently observe that we can achieve a large percentage of the benefit with a small fraction of the transmissions we need to achieve the maximum benefit. For example, in Fig. 3.1, a 2% decrease in benefit can achieve a 71% bandwidth savings, and in Fig. 3.2, a 20% decrease in benefit can achieve a 91% bandwidth savings.

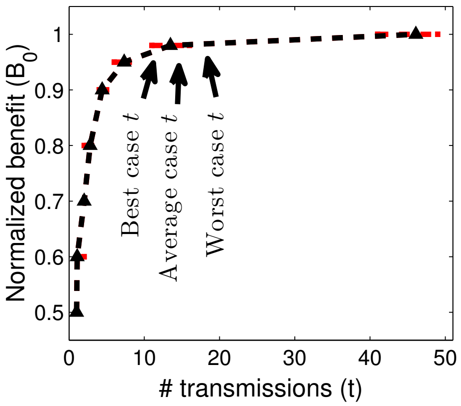


Figure 3.1: Trade-off between bandwidth K and normalized benefit B_0 for AlgP1 and P1 with m = 300, n = 50 and Borda score model, averaged over 100 random instances.

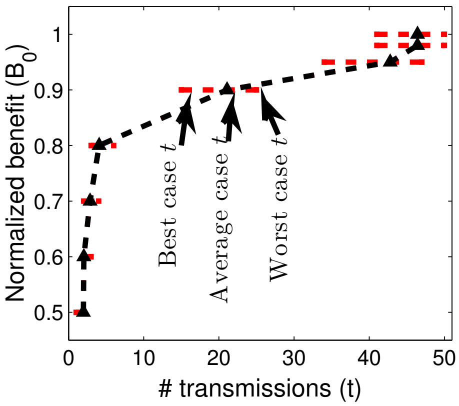


Figure 3.2: Trade-off between bandwidth K and normalized benefit B_0 for AlgP2 and P2 with m = 300, n = 50, r = 60 and Borda score model, averaged over 100 random instances.

Benefit for small K Figs. 3.3 and 3.4 highlight the normalized benefit B_0 we can achieve if the server is restricted to very few transmissions ($K = 1$ or $K = 4$) over some scenaria. Observe that with $K = 4$ transmissions we can consistently achieve more than 85% of the benefit and with $K = 1$ more than 38% of the benefit.

Table 3.1: Description of scenaria

Scenaria	Side information	Score model	Parameters	Algorithms
Scenario 1	No	Borda score	$m = 1000$, $n = 20$	AlgP1
Scenario 2	No	Bimodal score with $G_{bim} = 10$ and $F_{bim} = 0.1$	$m = 1000$, $n = 20$	AlgP1
Scenario 3	Yes, $r = 100$	Borda score	$m = 1000$, $n = 20$	AlgP2
Scenario 4	Yes, $r = 100$	Bimodal score with $G_{bim} = 10$ and $F_{bim} = 0.1$	$m = 1000$, $n = 20$	AlgP3

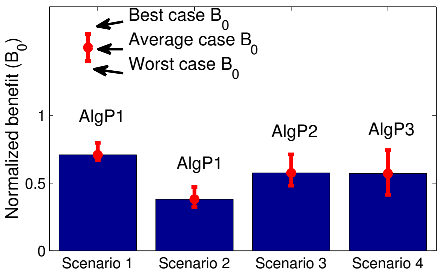


Figure 3.3: Normalized benefit for $K = 1$. The scenaria are described in Table 3.1.

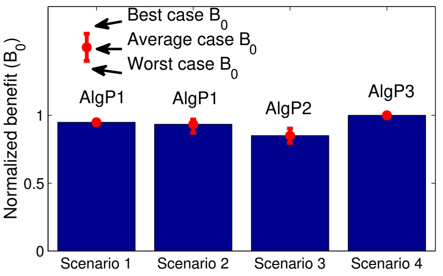


Figure 3.4: Normalized benefit for $K = 4$. The scenaria are described in Table 3.1.

Benefit from coding Figs. 3.5 and 3.6 compare, over two sets of parameters for P2, the performance of AlgP1 (we run AlgP1 by ignoring the side information and making uncoded transmissions) and AlgP2. We find that leveraging the side information and coding enables AlgP2 to double in some cases the benefit.

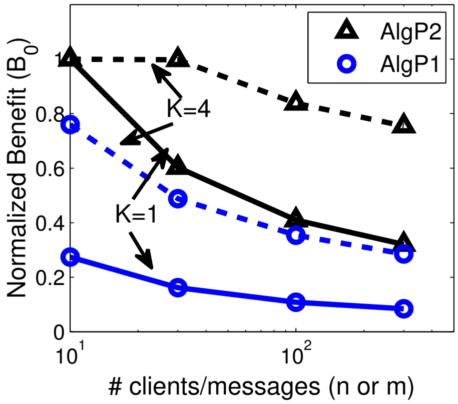


Figure 3.5: Coding gain for instances of P2 with $n = m$, $r = 0.1m$, and the Borda score model.

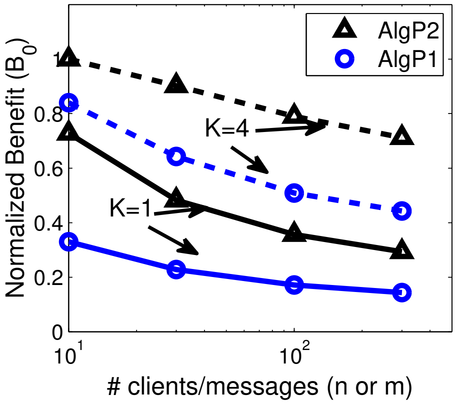


Figure 3.6: Coding gain for instances of P2 with $n = m$, $r = 0.2m$, and the Borda score model.

Bebefit over random selection Figs. 3.7 and 3.8 compare the performance of AlgP1 with that of random selection over bimodal instances of P1. Random selection assumes that the server first identifies all messages that form first preference for at least one client, and then randomly selects to transmit K of them. We find that AlgP1 achieves $52\% - 80\%$ more benefit than random selection for $K = 4$, and $60\% - 88\%$ more for $K = 1$ as m changes from 10 to 1000; and achieves $30\% - 71\%$ more benefit than random selection for $K = 4$, and $31\% - 106\%$ more for $K = 1$ as G_{bim} changes from 2 to 50.

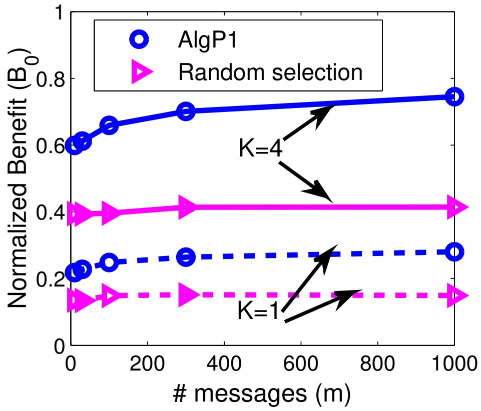


Figure 3.7: AlgP1 vs. random selection for P1 with $n = 50$, and the bimodal score model with $G_{bim} = 10$ and $F_{bim} = 0.1$, as m varies.

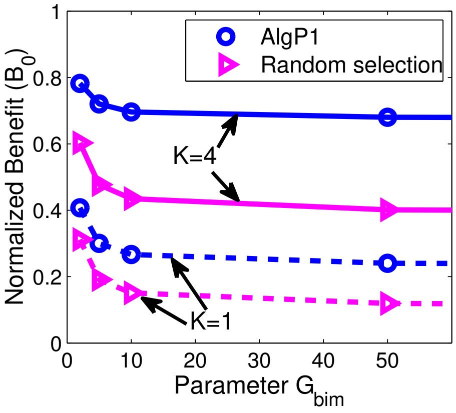


Figure 3.8: AlgP1 vs. random selection for P1 with $m = 300, n = 50$, and the bimodal score model with $F_{bim} = 0.1$, as G_{bim} varies.

3.6.2 Over Real Dataset

We extract instances from the Yahoo! Search Marketing advertiser bidding dataset, which was collected every 15 minutes over a year's period; data instances include the time stamp, the key phrase ID, the advertiser ID, and the bidding price [yah]. We generate a problem instance as follows: During each hour, n users with n search query key phrases enter the system (these are the clients). The advertiser bids to place an ad (the ads are the messages) to some of the key phrases using certain prices (we assume price zero for the rest of the key phrases). We interpret the price of a bid for a key phrase as the score of this ad (message) with respect to the key phrase (client). We assume that the messages that the advertiser does not bid for, form side information.

We compare AlgP1 and AlgP3 with the conventional Borda count method [Bor81], the Spearfoot rule-based method [DG77, DKN01], and the Kemeny's method [Kem59, DKN01] (we interpret the recommendations these algorithms make as the uncoded messages to transmit). The horizontal axis represents time (instances collected at sequential time slots). In Figs. 3.9 and 3.11 the vertical axis represents the actual benefit achieved at each time (current instance); in Figs. 3.10 and 3.12, the vertical axis represents the accumulated benefit (all previous instances). We find that all uncoded algorithms (including AlgP1) perform similarly; this is because in the data set a few of the messages concentrated the highest rankings from all clients, and thus the score model used by the algorithm did not make a difference in the message choice. However, by leveraging the side information, AlgP3 could accrue multiple times the benefit over time.

3.7 Open Question and Future Work

In this chapter, we discussed the trade-off between the benefit and the number of available transmissions. We note that the benefit depends on the preference model. For the Borda score model, since the scores of messages are linear in their preferences, we can regard the Borda score model as a *linear preference model* and it can be solved using polynomial-time approximation algorithm within a constant approximation ratio. For the general score

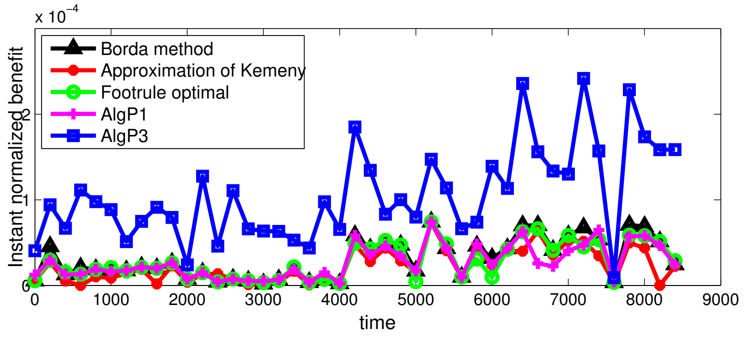


Figure 3.9: Instant benefit for $K = 2$ as a function of time (current instance).

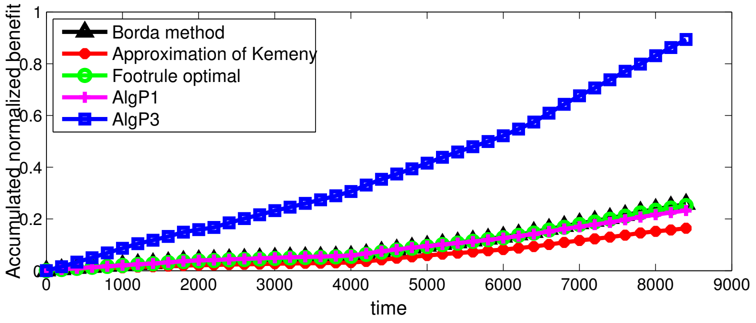


Figure 3.10: Accumulated benefit for $K = 2$ as a function of time (all previous instances).

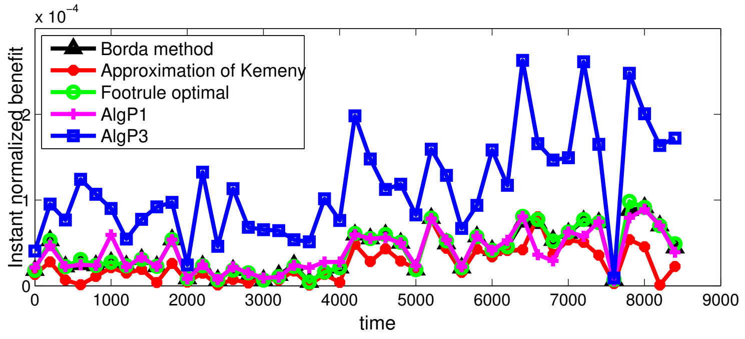


Figure 3.11: Instant benefit for $K = 4$ as a function of time (current instance).

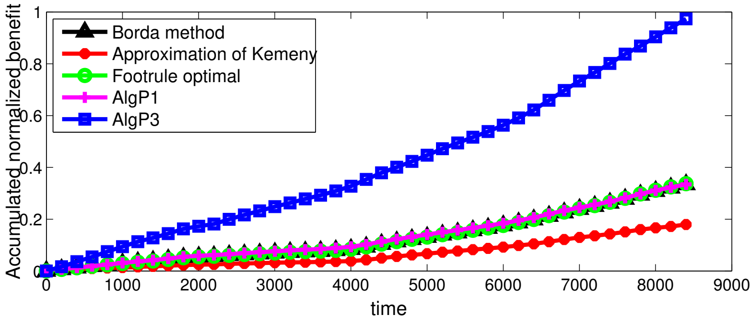


Figure 3.12: Accumulated benefit for $K = 4$ as a function of time (all previous instances).

model, as shown in Chapter 3.5, we can see that it is hard to approximate within $n^{1-\epsilon}$. Hence, one open question is how to evaluate the approximation ratio with respect to the preference model. In particular, to achieve constant approximation ratio, what characteristic should the preference model have? And what other preference models will result in a hard-to-approximate problem, such as hard to approximate within $n^{1-\epsilon}$?

3.8 Summary

In this chapter, we examine recommender systems under bandwidth constraint in a pliable index coding framework. We present three problem examples to show that although the problems are in general NP-hard, designing polynomial time approximation algorithms can still make a significant bandwidth savings by leveraging coding. We also conduct experiments over real data set to validate our arguments.

CHAPTER 4

Application to Data Shuffling in Distributed Computing

4.1 Introduction

In distributed computing systems, the communication between computing nodes forms a major bottleneck for the runtime performance [CZM11], especially those for processing Big Data applications. The coding techniques can be used to improve the communication efficiency in distributed computing systems [LLP15, AT16, LMA15].

In this chapter we consider a "master-workers" distributed computing system model, where a master node has m messages and is connected through a broadcast channel to n worker nodes. Each worker i is equipped with a cache that can store s_i messages. Shuffling occurs in iterations, where in each iteration we need to refresh the data the workers have, with a random selection of s_i out of m messages from the master node.

We can model this data shuffling problem in the index coding framework: the messages that worker nodes have from previous iterations form side information; the master node randomly interleaves the m messages and allocates some specific s_i messages to each worker, corresponding to some specific s_i requests in index coding.

From the exponential savings of the broadcast transmissions of pliable index coding over index coding shown in Chapter 2 and [BBJ11, BF15], we posit that the framework of pliable index coding could be a better fit for such applications. Our observation is that, when shuffling, we do not need to pre-specify the new messages a worker gets; we only need the worker to get a certain new messages that are randomly selected from the original messages.

For example, to train a classification model in a distributed system, a large volume of data instances can be randomly distributed into n worker nodes in tens of millions of ways, not necessarily in a specific way. This degree of freedom enables us to design more efficient coding and transmission schemes to realize semi-random shuffling.

In this chapter, we first analyze how pliable index coding would perform under the constraints of data shuffling. In particular, when data shuffling, we want each message to go to at most a specific number of workers, say c, to achieve an unbiased data distribution that looks "random-like". We capture this by imposing the constraint that each message can be used to satisfy at most c clients. That is, each client is happy to receive any message she does not have, but at most c clients can receive the same message. We show that even if $c = 1$, i.e., each message can satisfy at most one client, we can still achieve $O(n)$ benefits over index coding in some cases; this is because, we still have the freedom to select any of the $O(n!)$ interleaved versions of requests that lead to the smallest number of transmissions. We prove that the constrained pliable index coding problem is NP-hard. We show that for random instances, the optimal code length is almost surely upper bounded by $O(\min\{\frac{n}{c\log(n)}, \frac{n}{\log(m)}\})$ for $c = o(\frac{n^{1/7}}{\log^2(n)})$ and $O(\min\{\frac{n}{c} + \log(c), \frac{n}{\log(m)}\})$ for $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$.

We then design a hierarchical transmission scheme for data shuffling that utilizes constrained pliable index coding as a component. We introduce a Hamming distance measure to quantify the shuffling performance, and show that our scheme can achieve benefits $O(ns/m)$, in terms of transmissions over index coding, with linear encoding complexity at the master node, where s is the cache size and ns/m is the average number of workers that cache each message.

The work presented in this chapter was published in [SF17].

4.2 Constrained Pliable Index Coding

4.2.1 Problem Formulation

We consider a server with m messages $b_1, b_2, \ldots, b_m$ in a finite field $\mathbf{F}_q$ and n clients. Each client has as side information some subset of the messages, indexed by $S_i \subseteq [m]$. Client $i \in [n]$ has requested any one of the remaining messages, indexed by $R_i = [m]\backslash S_i$. We term this set R_i the request set.

c-constraint We require that a message j is stored by at most c clients. We call such a problem c-constrained pliable index coding and denote it by problem instance $(m, n, \{R_i\}_{i\in[n]}, c)$.

Bipartite Graph Representation In the bipartite graph, on one side the vertices correspond to messages and on the other side to clients; we connect clients to the messages they *do not* have, i.e., client i connects to the messages in R_i [BF12].

Linear Encoding The server makes K broadcast transmissions $x_1, x_2, \ldots, x_K$ over a noiseless channel. Each x_k is a linear combination of $b_1, \ldots, b_m$, namely, $x_k = a_{k1}b_1 + a_{k2}b_2 + \ldots + a_{km}b_m$, where $a_{kj} \in \mathbf{F}_q$ is the encoding coefficient. We refer to the number of transmissions, K, as the *code length* and to the $K \times m$ coefficient matrix $\boldsymbol{A}$ with entries a_{kj} as the *coding matrix*. In matrix form, we can write

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{b}, \tag{4.1}$$

where $\boldsymbol{b}$ and $\boldsymbol{x}$ are vectors that collect the original messages and encoded transmissions, respectively.

Linear Decoding Given $\boldsymbol{A}$, $\boldsymbol{x}$, and $\{b_j | j \in S_i\}$, each client i needs to solve the linear equation (4.1) to get a unique solution of b_{j_i}, for some $j_i \in R_i$. We say that client i is satisfied if she stores the decoded message b_{j_i} and b_{j_i} is decoded and stored by at most c clients. Clearly, client i can remove from the transmissions her side information messages, i.e., to recover $x_k^{(i)} = x_k - \sum_{j \in S_i} a_{kj} b_j$ from the k-th transmission. As a result, client i only needs to solve

$$\boldsymbol{A}_{R_i} \boldsymbol{b}_{R_i} = \boldsymbol{x}^{(i)}, \tag{4.2}$$

to retrieve a message b_{j_i} she does not have, where $\boldsymbol{A}_{R_i}$ is the sub-matrix of $\boldsymbol{A}$ with columns indexed by R_i; $\boldsymbol{b}_{R_i}$ is the message vector with elements indexed by R_i; and $\boldsymbol{x}^{(i)}$ is a K-dimensional column vector with elements $x_k^{(i)}$.

Building on results in Lemma 1, we have the following decoding criterion. We use $\boldsymbol{a}_j$ to denote the j-th column of matrix $\boldsymbol{A}$ and use $\text{span}\{\boldsymbol{a}_{j'} | j' \in R_i \backslash \{j\}\} = \{\sum_{j' \in R_i \backslash \{j\}} \lambda_{j'} \boldsymbol{a}_{j'} | \lambda_{j'} \in \mathbf{F}_q\}$ to denote the linear space spanned by columns of $\boldsymbol{A}$ indexed by R_i other than j.

Lemma 11. *In a constrained pliable index coding problem $(m, n, \{R_i\}_{i \in [n]}, c)$, a coding matrix $\boldsymbol{A}$ can satisfy all clients if and only if there exist messages $j_1, j_2, \ldots, j_n \in [m]$, one for each client, where no single message is repeated more than c times, i.e., $j_{i_1} = j_{i_2} = \ldots = j_{i_{c+1}}$ does not hold for any combination of $c+1$ clients $i_1, i_2, \ldots, i_{c+1} \in [n]$, such that the matrix $\boldsymbol{A}$ satisfies*

$$\boldsymbol{a}_{j_i} \notin \mathit{span}\{\boldsymbol{a}_{j'} | j' \in R_i \backslash \{j_i\}\}, \forall i \in [n]. \tag{4.3}$$

Our goal is to construct the coding matrix $\boldsymbol{A}$ with the minimum code length K. Note that the c-constraint significantly changes the pliable index coding problem. For example, assume we have m messages and n clients with no side information; then pliable index coding requires 1 transmission, while constrained pliable index coding needs n/c transmissions to satisfy all clients.

4.2.2 Main Results

4.2.2.1 Benefits Over Index Coding

Clearly, the larger the value of c, the more benefits we expect constrained pliable index coding to have over index coding (for $c = n$ we have exponential benefits [BF15, SF16c]). We here provide an example to show that it is possible to have benefits of $O(n)$ even when $c = 1$, i.e., each message can satisfy at most one client, as is the case in index coding. This equivalently shows that, if we are allowed to “interleave the demands” in index coding, we can gain $O(n)$ in terms of the number of transmissions.

We construct the following 1-constrained pliable coding instance with n messages and n clients. Client $i \in [n/2]$ requests any of the messages 1 to $n/2$ and $n/2+i$, i.e., $R_i = \{1, 2, \ldots, n/2, n/2+i\}$, for $i \in [n/2]$. Client $i \in [n]\backslash[n/2]$ requests any of the messages $n/2+1$ to n and $i-n/2$, i.e., $R_i = \{i-n/2, n/2+1, n/2+2, \ldots, n\}$, for $i \in [n]\backslash[n/2]$. All messages not in the request set form side information.

For index coding, if client i requests message i and has the same side information as above, then we need at least $n/2$ transmissions, since the first $n/2$ clients do not have the first $n/2$ messages as side information. In contrast, 1-constrained pliable index coding only requires 2 transmissions. Indeed, we can enable client $i \in [n/2]$ to decode the message $n/2+i$, by making the transmission $b_{n/2+1} + b_{n/2+2} + \ldots + b_n$, since each client $i \in [n/2]$ has all messages indexed by $[n]\backslash([n/2] \cup \{n/2+i\})$ as her side information. Similarly, we can enable client $i \in [n]\backslash[n/2]$ to decode the message $i-n/2$ by making the transmission $b_1 + b_2 + \ldots + b_{n/2}$.

4.2.2.2 Constrained Pliable Index Coding is NP-hard

It suffices to show that 1-constrained pliable index coding is NP-hard.

Theorem 14. *For a 1-constrained pliable index coding problem, deciding if the optimal code length*

- *$K = 1$ is in P.*
- *$K = 2$ is NP-complete.*

Proof.

Deciding if Optimal $K = 1$ is in P

We first show that deciding if the optimal code length equals 1 is in P. To see this, we notice that if one transmission can make each client to receive a distinct message, then the server needs to encode exact n messages for the transmission, one for each client. For a client i, if it can decode a message b_j, $j \in R_i$, then all other $n-1$ messages must be in its side information set following from the decoding criterion. Similarly, any one of the n messages for encoding is in the side information set of $n-1$ clients and requested by the remaining one client. Hence, in the bipartite graph representation, if and only if we can find n message vertices, such that each one has degree 1 and is connected to a different client vertex, then the optimal code length is 1. This can be tested by going over all message vertices, which runs in polynomial time.

Deciding if Optimal $K = 2$ is NP-complete

We next show that deciding if optimal code length equals 2 is NP-complete. To prove this, we first introduce another NP-complete problem.

Definition 1 (Distinct Labeling Problem)**.** *We are given a universal set $U = \{1, 2, \ldots, u\}$ with $|U| = u$ elements, a fixed set of Π labels $\{1, 2, \ldots, \Pi\}$, and a collection of size 3 subsets of U, i.e., $\mathcal{S} \subseteq 2^U$ and $|S| = 3$ for any $S \in \mathcal{S}$, where 2^U is the power set of U. The distinct labeling problem (DL) asks if we can label the elements using Π labels such that every subset in $\mathcal{S}$ contains elements of 3 different labels. For short, we call it Π-DL problem for such a distinct labeling problem with Π labels.*

We next show that Π-DL problem is NP-complete for $\Pi \geq 3$.

It is easy to see that the Π-DL problem is in NP. We next show that we can use a polynomial time reduction from the graph coloring problem (a.k.a., chromatic number) to the Π-DL problem.

We reiterate the well-known decision version of graph coloring problem as follows [Kar72]: it is NP-complete to decide whether the vertices of a given graph $G(V, E)$ can be colored using a fixed $\Pi \geq 3$ colors, such that no two neighboring vertices share the same color.

We perform the following mapping. We map each vertex in V and each edge in E as the universal set with $|U| = |V| + |E|$ elements. We map an edge $e \in E$ together with the two endpoints x_1, x_2 as a subset, where $e = \{x_1, x_2\}$. So there are in total $|\mathcal{S}| = |E|$ subsets.

We show that if G is Π-colorable, then we can find a solution for the Π-DL problem. We can assign a set $\{1, 2, \ldots, \Pi\}$ of colors to the vertices in V, such that no two neighboring vertices share a same color. When we map to the Π-DL problem, we notice that each edge appears in exact 1 subset, the one corresponding to this edge. Hence, we can use the following labeling scheme: label the elements corresponding to the vertices as the color used in $\{1, 2, \ldots, \Pi\}$; and label the edge using any label that is different from its two endpoints. This is a solution for the Π-DL problem.

On the other hand, if we have a solution for the Π-DL problem, we can find a solution for the graph coloring problem. We can label the elements corresponding to vertices using the Π labels. Note that if two vertices x_1 and x_2 are adjacent to each other, i.e., $\{x_1, x_2\} \in E$, then according to the definition of Π-DL problem, these two elements x_1 and x_2 must have different labels. Hence, keep the labels of each vertex element, then we get a Π-coloring of the graph G.

We then prove that deciding if the optimal code length $K = 2$ is NP-complete for constrained pliable index coding problem over a finite filed $\mathbf{F}_q$.

We observe that we can decide if a given $2 \times m$ coding matrix $\boldsymbol{A}$ can satisfy a constrained pliable index coding instance from our decoding criterion. Indeed, given a coding matrix $\boldsymbol{A}$, one can list the messages a client can decode using the decoding criterion. Then we have a bipartite subgraph representation that has n clients, some messages, and edges that connect each client with the message she can decode. We only need to check if the maximum matching in such a subgraph equals the number of clients n using polynomial time. If and only if so, this coding matrix can satisfy the problem instance.

Next, we use a reduction from the $(q+1)$-DL problem defined above to show that the constrained pliable index coding problem is NP-hard. We are given a $(q+1)$-DL problem instance with the universal set $U = \{1, 2, \ldots, u\}$ and a collection of size 3 subsets $\mathcal{S} \subseteq 2^U$. We perform the following two mappings.

• For each subset, e.g., $S = \{x, y, z\} \in \mathcal{S}$ and $x, y, z \in U$, we map into a structure as show in Fig. 4.1. We map each element in the subset S as a message vertex and add 3 client vertices c_1, c_2, c_3 in the constraint pliable index coding problem instance. We connect c_1 to

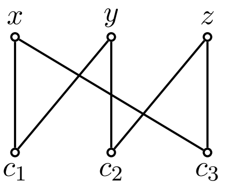


Figure 4.1: Mapping a subset into a structure.

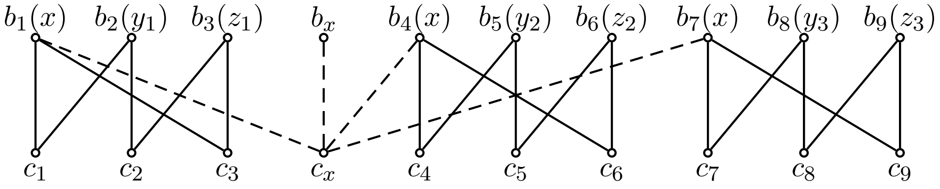


Figure 4.2: Connecting the same elements in different subsets.

x and y, connect c_2 to y and z, and connect c_3 to z and x.

- For different subsets, if they contain the same element, we connect them using the following structure as shown in Fig. 4.2. For example, if the subsets $S_1 = \{x, y_1, z_1\}$, $S_2 = \{x, y_2, z_2\}$, and $S_3 = \{x, y_3, z_3\}$ all contain the element x, we connect a client vertex c_x to all messages corresponding to x and another additional message vertex b_x.

After this mapping, we can see that we construct a constrained pliable index coding instance with $n = 3|\mathcal{S}| + |U|$ clients and $m = n$ messages. We want to show that if and only if the $(q+1)$-DL problem outputs a "Yes" answer, a code length 2 coding matrix can satisfy such a problem.

If for a "Yes" instance of $(q+1)$-DL problem, we can find a labeling scheme using $q+1$ labels to the elements. In finite field $\mathbf{F}_q$, we notice that the maximum number of vectors that are pair-wise independent is $q+1$, e.g.,

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \dots, \begin{bmatrix} 1 \\ q-1 \end{bmatrix}. \tag{4.4}$$

We consider each label as one of these $q+1$ vectors. Then for the coding matrix $\boldsymbol{A}$, we can assign the columns the same vector that correspond to the same element in subsets, e.g., b_1, b_4, and b_7 in Fig. 4.2. For the columns corresponding to an element not in subsets, e.g., b_x, we assign a different vector other than the one for the element in subsets. This is a valid coding matrix. Indeed, for 3 messages corresponding to a subset, e.g., b_1, b_2, and b_3, they are labeled using different labels from the $q+1$-DL problem solution. Then, the 3 clients corresponding to this subset, i.e., c_1, c_2, and c_3, can decode b_1, b_2, and b_3, respectively, according to the decoding criterion. The client corresponding to an element not in subsets, e.g., c_x, can decode the corresponding message not in subsets, i.e., b_x, as coding vectors corresponding to messages b_1, b_4, and b_7 are the same and different from the coding vector corresponding to message b_x.

If for the constrained pliable index coding instance, a length 2 coding matrix $\boldsymbol{A}$ can make a successful shuffling. Then we notice that client c_x should be satisfied by message b_x, since b_x only connects to c_x and $m = n$, which implies that each message needs to satisfy a client. In this case, the non-zero coding vector corresponding to b_x is not in the space spanned by other coding vectors corresponding to x in subsets, i.e., b_1, b_4, and b_7. As a result, the space spanned by coding vectors corresponding to the same element in subsets is a one dimensional space, e.g., the space spanned by coding vectors corresponding to b_1, b_4, and b_7. For clients and messages inside a subset, e.g., b_1, b_2, b_3, c_1, c_2, and c_3, there are two ways to satisfy these clients: one is b_1 to c_1, b_2 to c_2, b_3 to c_3; and the other one is b_2 to c_1, b_3 to c_2, b_1 to c_3. For both of these two ways, we notice that coding vectors corresponding to b_1, b_2, and b_3 should be pair-wise independent; otherwise, one of the clients cannot decode a new message, e.g., if coding vectors corresponding to b_1 and b_2 are dependent to each other, then c_1 cannot decode any new message. In addition, we observe that there are in total $q+1$ 1-dimensional subspaces spanned by 2-dimensional vectors over finite filed $\mathbf{F}_q$, i.e., the spaces spanned by vectors in (4.4). Therefore, if we assign each space a label, then messages in the same subsets are using different labels and messages in different subsets corresponding to a same element are using the same label, resulting in a solution of the $q+1$-DL problem.

□

4.2.2.3 Bounds for Constrained Pliable Index Coding

Similar to index coding, we show that there is also a "sandwich property" for the constrained pliable index coding problem that has the optimal code length bounded from above and below.

A Lower Bound The conventional index coding problem has a lower bound that is the maximum independent set of the undirected graph with n vertices and an edge connecting two vertices i and j if and only if clients i and j do not have messages j and i as their side-information. Hence, the interpretation of this lower bound is to find the largest set of clients (and their corresponding required messages), such that no one of them has any of these corresponding messages as side-information. Then this degenerates to a common broadcasting scenario where k broadcast transmissions are needed: a server broadcasts k messages to k clients and none of them has side information.

However, in our constraint pliable index coding scenario, such a lower bound does not hold, e.g., the instance in Chapter 4.2.2.1. We show a lower bound using different technique based on our decoding criterion as shown in the following theorem.

Theorem 15. *In the c-constrained pliable index coding problem, if we can find k clients $i_1, i_2, \dots, i_k$, such that the request sets satisfy $R_{i_1} \subseteq R_{i_2} \subseteq \dots \subseteq R_{i_k}$, then the code length should be at least k/c for such an instance.*

Proof. From the decoding criterion, we assume that there exist messages $j_1 \in R_{i_1}, j_2 \in R_{i_2}, \dots, j_k \in R_{i_k}$, such that for the coding matrix $\boldsymbol{A}$, $\boldsymbol{a}_{j_l} \notin \text{span}\{\boldsymbol{A}_{R_{i_l}\setminus\{j_l\}}\}$ for $l = 1, 2, \dots, k$. From $R_{i_1} \subseteq R_{i_2} \subseteq \dots \subseteq R_{i_k}$, we have the following: $\boldsymbol{a}_{j_k} \notin \text{span}\{\boldsymbol{a}_{j_1}, \boldsymbol{a}_{j_2}, \dots, \boldsymbol{a}_{j_{k-1}}\}$, $\boldsymbol{a}_{j_{k-1}} \notin \text{span}\{\boldsymbol{a}_{j_1}, \boldsymbol{a}_{j_2}, \dots, \boldsymbol{a}_{j_{k-2}}\}$, ..., $\boldsymbol{a}_{j_2} \notin \text{span}\{\boldsymbol{a}_{j_1}\}$. This implies that after removing redundancy, the set of vectors $\{\boldsymbol{a}_{j_1}, \boldsymbol{a}_{j_2}, \dots, \boldsymbol{a}_{j_k}\}$ are linearly independent. Hence, the coding matrix $\boldsymbol{A}$ needs to be have a rank at least k/c and the result follows. □

An Upper Bound To show an upper bound, we use the partition number of the bipartite graph that represents a problem instance. For ease of exposition, we use k colors to color the vertices of the graph. We can upper bound the number of transmissions k, if the coloring scheme satisfies the following two properties:

- For each client vertex $i \in [n]$, it has exactly 1 neighbor that has the same color;
- For each message vertex $j \in [m]$, it has at most c neighbors that have the same color.

To see this, we use k transmissions, where each transmission consists of a linear combination (with coefficient 1) of messages that have the same color, i.e., $\sum b_{j'}$ for all $b_{j'}$ with the same color.

It is not hard to see that this transmission scheme can result in a successful coding. Indeed, for client i, if the color is 'red', then the transmission corresponding to the color 'red' can help client i recover the only neighbor of i that has the same color 'red'; and also any message vertex with a color 'red' will be recovered and stored by at most c neighbors with the same color 'red'.

We can see that this minimum number of colors is just the *partition number* $\mathcal{P}_{star}(G)$ of the graph G, where $\mathcal{P}_{star}(G)$ is the minimum number of *induced star forests*[1] into which the graph can be partitioned such that any induced star is centered on a message vertex in $[m]$ with degree no more than c. An example is shown in Fig. 4.3. Note that as a special case when $c = 1$, this partition number is the minimum number of *induced matchings* into which the graph can be partitioned.

4.2.2.4 Performance Over Random Instances

We consider a random bipartite graph instance, denoted by $B(m, n, c, p)$, or B for short, where there are m messages, n clients, each message can be recovered and stored by at most c clients, and each client is connected with a message with probability p (clients have as side information all the messages they are not connected to). We assume that p is a fixed constant and define $\bar{p} = \min\{p, 1-p\}$, while $c = c(n)$ and $m = m(n) \geq n$ could be changing with n.

[1] A star is a complete bipartite graph $K_{1,l}$ with degree l. An induced star forest is an induced subgraph consists of disjoint stars.

Theorem 16 summarizes our main result.

Theorem 16. *The number of broadcast transmissions for random graph instance $B(m,n,c,p)$ is almost surely upper bounded by*

- $O(\min\{\frac{n}{c\log(n)}, \frac{n}{\log(m)}\})$*, for* $c = o(\frac{n^{1/7}}{\log^2(n)})$*; and*
- $O(\min\{\frac{n}{c} + \log(c), \frac{n}{\log(m)}\})$*, for* $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$*.*

Our proof outline is as follows. We design a transmission scheme, and show that it achieves this performance. To do so, we first define an *l-pattern* (we will give details later) that enables with a single broadcast transmission to satisfy lc clients. We then find values $l = \mathbb{L}$, m' and n' for which almost surely an $\mathbb{L}$-pattern exists in every induced subgraph B' of B with m' message vertices and n' client vertices, i.e.,

$$\Pr\{\exists B', s.t., B' \text{ contains no } \mathbb{L}\text{-patterns}\} = o(1). \tag{4.5}$$

Then the transmission scheme proceeds as follows. If there are more than n' clients in the original graph B, we pick a $\mathbb{L}$-pattern and make one transmission. We remove the satisfied clients and the used messages. If there are less than n' clients, we use at most n' transmissions to satisfy the remaining clients. Hence, we almost surely need $\frac{n}{\mathbb{L}c} + n'$ transmissions.

To minimize $\frac{n}{\mathbb{L}c} + n'$, we want n' to be small and $\mathbb{L}$ to be large. However, by decreasing n' we also decrease the values of $\mathbb{L}$ that satisfy (4.5). Hence, we need to balance the sizes of n' and $\mathbb{L}$; we use different values depending on how m, n, and c are related.

Next, we provide the proof in detail. First, we define a subgraph called *l-pattern*.

Definition 2. *An l-pattern is an induced subgraph that consists of l message vertices, lc client vertices, if the following property is satisfied:*

Each of the l messages is connected with c clients and each of the lc client is connected with only one message.

An illustration of the l-pattern is shown in Fig. 4.3.

For a given random bipartite graph B, let us denote by Y_l the number of l-patterns in

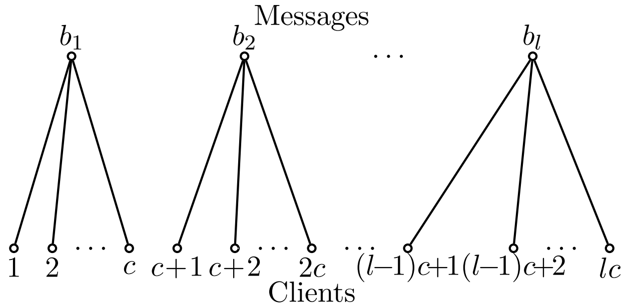


Figure 4.3: In a l-pattern, one transmission satisfies lc clients.

B. For an induced subgraph B' of B, let us denote by $Y_l^{B'}$ the number of l-patterns on the subgraph B'.

We have the following calculation for the average number of l-patterns.

$$\mathbb{E}[Y_l] = \binom{m}{l}\binom{n}{lc}\binom{lc}{c, c, \dots, c} p^{lc}(1-p)^{l(l-1)c}. \tag{4.6}$$

It is easy to see that $\mathbb{E}[Y_l]$ is decreasing with l, given other parameters fixed. Hence, we define l_0 to be the maximum integer such that $\mathbb{E}[Y_{l_0}] \geq 1$, i.e., $l_0 = \max\{l | \mathbb{E}[Y_l] \geq 1\}$. We next show that l_0 is in the order of $\log(n) + \frac{\log(m)}{c} - \log(c)$. More accurately, we show in the following lemma.

Lemma 12. *l_0 satisfies: $x_1 \leq l_0 \leq x_1 + O(1)$ for $x_1 = 1 + \frac{1}{\log(1/(1-p))}[\log(n) + \frac{\log(m)}{c} - \log(c) - \frac{\log(\log(n) + \frac{\log(m)}{c} - \log(c))}{c} - \frac{\log\log(\frac{1}{1-p})}{c} + \log(p)] + o(1)$.*

Proof. Using the binomial inequality

$$(\frac{x}{y})^y \leq \binom{x}{y} \leq (\frac{ex}{y})^y, \tag{4.7}$$

we can bound the value of $\mathbb{E}[Y_k]$ by

$$\begin{aligned} \mathbb{E}[Y_l] &\leq (\tfrac{em}{l})^l(\tfrac{en}{c})^c(\tfrac{e(n-c)}{c})^c(\tfrac{e(n-2c)}{c})^c \dots (\tfrac{e(n-lc)}{c})^c p^{lc}(1-p)^{l(l-1)c} \\ &< (\tfrac{em}{l})^l(\tfrac{en}{c})^{lc} p^{lc}(1-p)^{l(l-1)c}, \end{aligned} \tag{4.8}$$

and

$$\begin{aligned}\mathbb{E}[Y_l] \ & \geq (\tfrac{m}{l})^l(\tfrac{n}{c})^c(\tfrac{n-c}{c})^c(\tfrac{n-2c}{c})^c \dots (\tfrac{n-lc}{c})^c p^{lc}(1-p)^{l(l-1)c} \\ & > (\tfrac{m}{l})^l(\tfrac{n-lc}{c})^{lc} p^{lc}(1-p)^{l(l-1)c}.\end{aligned} \tag{4.9}$$

By taking log(·) on both sides, we get the following relationships:

$$\begin{aligned}\log(\mathbb{E}[Y_l]) < \ & l[1+\log(m)-\log(l)] + lc[1+\log(n)-\log(c)] + lc\log(p) \\ & +l(l-1)c\log(1-p), \\ \log(\mathbb{E}[Y_l]) > \ & l[\log(m)-\log(l)] + lc[\log(n-lc)-\log(c)] + lc\log(p) \\ & +l(l-1)c\log(1-p).\end{aligned} \tag{4.10}$$

Let us define two continuous functions $f_1(x) = x[\log(m) - \log(x)] + xc[\log(n - xc) - \log(c)] + xc\log(p) + x(x-1)c\log(1-p)$ and $f_2(x) = x[1+\log(m)-\log(x)] + xc[1+\log(n) - \log(c)] + xc\log(p) + x(x-1)c\log(1-p)$. Hence, we can rewrite the above inequalities as $f_1(l) < \log(\mathbb{E}[Y_l]) < f_2(l)$. Note that the two functions $f_1(x)$ and $f_2(x)$ are monotonously decreasing around $\log(n) + \frac{\log(m)}{c} - \log(c)$.

Solving the equations $f_1(x) = 0$ and $f_2(x) = 0$, we get

$$\begin{aligned} f_1(x_1) = 0, \ \text{for } x_1 = \ & 1 + \tfrac{1}{\log(1/(1-p))}[\log(n) + \tfrac{\log(m)}{c} - \log(c) \\ & -\tfrac{\log(\log(n)+\frac{\log(m)}{c}-\log(c))}{c} - \tfrac{\log\log(\frac{1}{1-p})}{t} + \log(p)] + o(1), \\ f_2(x_2) = 0, \ \text{for } x_2 = \ & 1 + \tfrac{1}{\log(1/(1-p))}[\log(n) + \tfrac{\log(m)}{c} - \log(c) + 1 + \tfrac{1}{c} \\ & -\tfrac{\log(\log(n)+\frac{\log(m)}{c}-\log(c))}{c} - \tfrac{\log\log(\frac{1}{1-p})}{c} + \log(p)] + o(1).\end{aligned} \tag{4.11}$$

We can see that both x_1 and x_2 are in the order of $\log(n) + \frac{\log(m)}{c} - \log(c)$ and $x_2 - x_1 = \frac{1}{\log(1/(1-p))}[1 + \frac{1}{c}] + o(1) \leq \frac{2}{\log(1/(1-p))} + o(1)$, which is bounded by $O(1)$.

We also have $\log(\mathbb{E}[Y_{\lceil x_2 \rceil}]) < f_2(\lceil x_2 \rceil) \leq f_2(x_2) = 0$ and $\log(\mathbb{E}[Y_{\lfloor x_1 \rfloor}]) > f_1(\lfloor x_1 \rfloor) \geq f_1(x_1) = 0$. This implies that $x_1 - 1 < \lfloor x_1 \rfloor \leq l_0 \leq \lceil x_2 \rceil - 1 < x_2$, from which the result follows. □

What we would like to show next is that the average number of l-patterns $\mathbb{E}[Y_l]$ has the property that it changes fast around the value l_0. We have the following lemma.

Lemma 13. $\mathbb{E}[Y_{l_1}]$ *satisfies:* $\mathbb{E}[Y_{l_1}] \geq (\frac{n}{ec})^{3c(1+o(1))} m^{3(1+o(1))}$, *for* $l_1 = l_0 - 3$.

Proof. We first have the following equation

$$\begin{aligned}
\frac{\mathbb{E}[Y_{l_0-3}]}{\mathbb{E}[Y_{l_0}]} &= \frac{\binom{m}{l_0-3}\binom{n}{(l_0-3)c}\binom{(l_0-3)c}{c}\binom{(l_0-4)c}{c}\ldots\binom{c}{c}p^{(l_0-3)c}(1-p)^{(l_0-3)(l_0-4)c}}{\binom{m}{l_0}\binom{n}{l_0c}\binom{l_0c}{c}\binom{(l_0-1)c}{c}\ldots\binom{c}{c}p^{l_0c}(1-p)^{l_0(l_0-1)c}} \\
&= \frac{l_0(l_0-1)(l_0-2)(c!)^3}{(m-l_0+3)(m-l_0+2)(m-l_0+1)(n-l_0c+1)(n-l_0c+2)\ldots(n-l_0c+3c)p^{3c}(1-p)^{6l_0c-12c}} \\
&\geq \frac{(c!)^3}{m^3n^{3c}(1-p)^{6c(l_0-2)}} \\
&\geq (\tfrac{n}{ec})^{3c(1+o(1))} m^{3(1+o(1))},
\end{aligned} \tag{4.12}$$

where the last inequality follows from $c! \geq e(c/e)^c$ and $(1-p)^{6c(l_0-2)} = (\frac{nm^{1/c}}{c})^{6c(1+o(1))}$, since $l_0 - 2 = \frac{1}{\log(1/(1-p))}[\log(n) + \frac{\log(m)}{c} - \log(c) + o(\log(n) + \frac{\log(m)}{c} - \log(c))]$.

Also note that $\mathbb{E}[Y_{l_0}] \geq 1$ and the result follows from (4.12). □

Similarly, we can define $l_0^{B'}$ as the maximum integer such that $\mathbb{E}[Y^{B'}_{l_0^{B'}}] \geq 1$ and define $l_1^{B'} = l_0^{B'} - 3$.

Let us denote by $\mathcal{B}(B, m', n')$ the family of all induced subgraphs of B by m' message vertices and n' client vertices. Next, we will discuss in different scenarios (in the following scenarios 1, 2, and 3) that we can find another integer $\mathbb{L} = \mathbb{L}(m, n) = l_2^{B'} \leq l_1^{B'}$, such that every induced subgraph $B' \in \mathcal{B}(B, m', n')$ almost surely contains a $\mathbb{L}$-pattern. For the fourth scenario, we will discuss separately. The 4 scenarios with parameters $\mathbb{L}$, m', and n' are formally defined as follows.

Definition 3. *We define the following* 4 *scenarios and how the corresponding parameters are related.*

- *Scenario* 1*:* $m < \exp(n^{1/15})$*. In this scenario, we set* $c = 1$*,* $\mathbb{L} = \lfloor \frac{1}{\log(1/\bar{p})}[\log(m) - 3\log\log(m) + 2\log\log(\frac{1}{\bar{p}})] \rfloor = \Theta(\log(m))$*,* $m' = \frac{m}{\log(m)}$*, and* $n' = \frac{n}{\log(m)}$*. If* $c > 1$*, we simply set* $c = 1$ *and this is a stronger constraint.*

- *Scenario* 2*:* $m \geq \exp(n^{1/15})$*. In this scenario, we set* $c = 1$*,* $\mathbb{L} = \lfloor \frac{1}{\log(1/\bar{p})}[\log(m) - 3\log\log(m) + 2\log\log(\frac{1}{\bar{p}})] \rfloor = \Theta(\log(m))$*,* $m' = m - \mathbb{L} = m(1 - o(1))$*, and* $n' = \mathbb{L}$*.*

If $c > 1$, we simply set $c = 1$ and this is a stronger constraint.

- *Scenario 3: $c = o(\frac{n^{1/7}}{\log^2(n)})$. In this scenario, we set $\mathbb{L} = \lfloor \frac{1}{\log(1/\bar{p})}[\log(n) - 3\log\log(n) - 3\log(c) + 2\log\log(\frac{1}{\bar{p}})]\rfloor = \Theta(\log(n))$, $m' = \frac{m}{\log(n)}$, and $n' = \frac{n}{c\log(n)}$.*
- *Scenario 4: $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$. In this scenario, we set $\mathbb{L} = 1$, $m' = 1$, and $n' = \frac{2c}{p}$.*

Note that the scenarios 1 and 2 are defined based on the relationship between m and n; the scenarios 3 and 4 are defined based on the relationship between c and n. There maybe overlaps between scenarios $1, 2$ and scenarios $3, 4$. We want to show the following lemma for the first 3 scenarios.

Lemma 14. *For scenarios 1, 2, and 3 with parameters defined in Definition 3, every induced subgraph $B' \in \mathcal{B}(B, m', n')$ almost surely contains a $\mathbb{L}$-pattern:*

$$\Pr\{\exists B' \in \mathcal{B}(B, m', n'), s.t., B' \textit{ contains no } \mathbb{L}\textit{-pattern}\} = o(1). \tag{4.13}$$

Proof. To prove this, we first use an "edge exposure" process to form a martingale based on the random subgraph B' [Bol01, MR10]. Specifically, we define X as a maximum number of $\mathbb{L}$-patterns in B' such that no two of them share a same message-client pair (i.e., any two $\mathbb{L}$-patterns either have no common message vertices or client vertices or both). We label the possible edges as $1, 2, \ldots, m'n'$ and denote by Z_t the random variable to indicate whether the edge t is exposed in the random graph, i.e., $Z_t = 1$ if the t-th possible edge is present in the graph and $Z_t = 0$ otherwise. Therefore, $X = f(Z_1, Z_2, \ldots, Z_{m'n'})$ is a function of the variables Z_t. Define $X_t = \mathbb{E}[X|Z_1, Z_2, \ldots, Z_t]$ as a sequence of random variables for $t = 1, 2, \ldots, m'n'$, then $\{X_t\}$ is a Doob martingale and $X_{m'n'} = X$. Obviously, the function $X = f(Z_1, Z_2, \ldots, Z_{m'n'})$ is 1-Lipschitz, namely, flipping only one indicator function, some Z_t, the value of X differs by at most 1: $|f(Z_1, \ldots, Z_t, \ldots, Z_{m'n'}) - f(Z_1, \ldots, Z_{t-1}, 1 - Z_t, Z_{t+1}, \ldots, Z_{m'n'})| \leq 1$.

Note that the subgraph B' contains no $\mathbb{L}$-pattern is equivalent to $X = 0$. We then use the Azuma's inequality

$$\Pr\{\mathbb{E}[X] - X \geq a\} \leq \exp(-\frac{a^2}{2m'n'}), \text{ for } a > 0. \tag{4.14}$$

to get

$$\Pr\{X = 0\} = \Pr\{\mathbb{E}[X] - X \geq \mathbb{E}[X]\} \leq \exp(-\frac{\mathbb{E}^2[X]}{2m'n'}). \tag{4.15}$$

Hence, to bound $\Pr\{X = 0\}$, we only need to find a lower bound of $\mathbb{E}[X]$. We use the following probabilistic argument. For subgraph B', we define $\mathcal{L}$ as the family of all $\mathbb{L}$-patterns and $\mathcal{P}$ as the family of all $\mathbb{L}$-pattern pairs that share at least a same message and a same client. Let us denote by $B_1, B_2 \in \mathcal{B}(B', \mathbb{L}, \mathbb{L}c)$ induced subgraphs of B' by $\mathbb{L}$ message vertices and $\mathbb{L}c$ client vertices. Let us also denote by X_{B_1} and X_{B_2} the variables to indicate whether the subgraphs B_1 and B_2 are $\mathbb{L}$-patterns. Let us use the notation $B_1 \sim B_2$ if two different subgraphs B_1 and B_2 share at least a same message vertex and a same client vertex. We then lower bound $\mathbb{E}[X]$ using the following scheme for scenarios $1, 2, 3$ (we will talk about how we bound $\mathbb{E}[X]$ for scenario 4 later): randomly select a subset of $\mathbb{L}$-patterns from the set $\mathcal{L}$ by picking up each $\mathbb{L}$-pattern with probability $p^{\dagger}$ (the value of which we will determine later); if two selected $\mathbb{L}$-patterns $B_1^{\dagger}$ and $B_2^{\dagger}$ form a pair in the set $\mathcal{P}$, then remove one of them. Then,

$$\mathbb{E}[X] \geq p^{\dagger}\mathbb{E}[|\mathcal{L}|] - p^{\dagger 2}\mathbb{E}[|\mathcal{P}|], \tag{4.16}$$

where the first term in the expression is the average number of selected $\mathbb{L}$-patterns in $\mathcal{L}$ and the second term is the average number of $\mathbb{L}$-patterns that are removed because a pair in $\mathcal{P}$ is selected with probability $p^{\dagger 2}$.

We observe that $\mathbb{E}[|\mathcal{L}|] = \mathbb{E}[Y_{\mathbb{L}}^{B'}]$ and next we calculate $\mathbb{E}[|\mathcal{P}|]$ and determine $p^{\dagger}$.

$$\begin{aligned}
\mathbb{E}[|\mathcal{P}|] &= \tfrac{1}{2}\textstyle\sum_{B_1 \in \mathcal{B}(B', \mathbb{L}, \mathbb{L}c)} \sum_{B_2 \in \mathcal{B}(B', \mathbb{L}, \mathbb{L}c): B_2 \sim B_1} \mathbb{E}[X_{B_1} X_{B_2}] \\
&= \tfrac{1}{2}\textstyle\sum_{B_1} \sum_{B_2 : B_2 \sim B_1} \Pr\{X_{B_1} = 1\} \Pr\{X_{B_2} = 1 | X_{B_1} = 1\} \\
&= \tfrac{1}{2}\tbinom{m'}{\mathbb{L}}\tbinom{n'}{\mathbb{L}c}\tbinom{\mathbb{L}c}{c, c, \ldots, c} \Pr\{X_{B_0} = 1\} \textstyle\sum_{B_2 : B_2 \sim B_0} \Pr\{X_{B_2} = 1 | X_{B_0} = 1\} \\
&= \tfrac{1}{2}\mathbb{E}[Y_{\mathbb{L}}^{B'}] \textstyle\sum_{B_2 : B_2 \sim B_0} \Pr\{X_{B_2} = 1 | X_{B_0} = 1\},
\end{aligned} \tag{4.17}$$

where the second equality is from the conditional probability formula, the third equality is by symmetry of the selection of B_1 and we then take a fixed selection B_0 consisting of the first $\mathbb{L}$ messages and first $\mathbb{L}c$ clients.

Hence, we only need to calculate the term $\sum_{B_2:B_2\sim B_0}\Pr\{X_{B_2}=1|X_{B_0}=1\}$ for different scenarios. We upper bound this term from above by enumerating all subgraph B_2 that has at least one common client vertex one common message vertex with B_0.

1) For scenario 1, we have

$$\textstyle\sum_{B_2:B_2\sim B_0}\Pr\{X_{B_2}=1|X_{B_0}=1\}\leq\sum_{j=1}^{\mathbb{L}}\sum_{i=1}^{\mathbb{L}}\binom{\mathbb{L}}{j}\binom{m'-\mathbb{L}}{\mathbb{L}-j}\binom{\mathbb{L}}{i}\binom{n'-\mathbb{L}}{\mathbb{L}-i}\mathbb{L}!\frac{p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}}{\bar{p}^{ij}}, \tag{4.18}$$

where the inequality is because $\bar{p}^{ij}\leq p^a(1-p)^{ij-a}$ for any non-negative integer $a\leq ij$. For simplicity, let us define the term inside the summation as $\Delta_{ij}\triangleq\binom{\mathbb{L}}{j}\binom{m'-\mathbb{L}}{\mathbb{L}-j}\binom{\mathbb{L}}{i}\binom{n'-\mathbb{L}}{\mathbb{L}-i}\mathbb{L}!\frac{p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}}{\bar{p}^{ij}}$.

We can see that for $i=1,2,\ldots,\mathbb{L}$ and $j=1,2,\ldots,\mathbb{L}-1$, we have

$$\begin{aligned}\frac{\Delta_{i,j+1}}{\Delta_{i,j}}&=\frac{(\mathbb{L}-j)^2}{(j+1)(m'-2\mathbb{L}+j+1)}\bar{p}^{-i}\\&\leq\frac{\mathbb{L}^2}{2(m'-2\mathbb{L}+2)}\bar{p}^{-\mathbb{L}}\\&\leq\frac{\mathbb{L}^2}{m'}\bar{p}^{-\mathbb{L}}\\&\leq\frac{\frac{1}{\log^2(1/\bar{p})}\log^2(m)}{m/\log(m)}\frac{m\log^2(1/\bar{p})}{\log^3(m)}\\&\leq 1.\end{aligned}\tag{4.19}$$

This implies that for all $i=1,2,\ldots,\mathbb{L}$ and $j=1,2,\ldots,\mathbb{L}$, $\Delta_{i,j}\leq\Delta_{i,1}$. Also note that for $i=1,2,\ldots,\mathbb{L}-1$,

$$\begin{aligned}\frac{\Delta_{i+1,1}}{\Delta_{i,1}}&=\frac{(\mathbb{L}-i)^2}{(i+1)(n'-2\mathbb{L}+i+1)}\bar{p}^{-1}\\&\leq\frac{\mathbb{L}^2}{n'\bar{p}}\\&\leq\frac{\log(m)^3}{n\bar{p}\log^2(1/\bar{p})}\\&=o(1),\end{aligned}\tag{4.20}$$

where the last equality holds for $m<\exp(n^{1/10})$. Hence, $\Delta_{i,j}\leq\Delta_{1,1}$ for all $i=1,2,\ldots,\mathbb{L}$ and $j=1,2,\ldots,\mathbb{L}$.

For $\Delta_{1,1}$, we have the following

$$\begin{aligned}\frac{\Delta_{1,1}}{\mathbb{E}[Y_{\mathbb{L}}^{B'}]} &= \frac{\mathbb{L}\binom{m'-\mathbb{L}}{\mathbb{L}-1}\mathbb{L}\binom{n'-\mathbb{L}}{\mathbb{L}-1}\mathbb{L}!\frac{p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}}{\bar{p}}}{\binom{m'}{\mathbb{L}}\binom{n'}{\mathbb{L}}\mathbb{L}!p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}} \\ &= \frac{\mathbb{L}^4(m'-\mathbb{L})!(m'-\mathbb{L})!(n'-\mathbb{L})!(n'-\mathbb{L})!}{\bar{p}m'!(m'-2\mathbb{L}+1)!n'!(n'-2\mathbb{L}+1)!} \\ &\leq \frac{\mathbb{L}^4\log^2(m)}{\bar{p}mn} \leq \frac{\log^6(m)}{\bar{p}mn\log^4(1/\bar{p})}.\end{aligned} \tag{4.21}$$

Plugging into (4.17), we have

$$\mathbb{E}[|\mathcal{P}|] \quad \leq \tfrac{1}{2}\mathbb{E}^2[Y_{\mathbb{L}}^{B'}]\mathbb{L}^2\tfrac{\log^6(m)}{\bar{p}\log^4(1/\bar{p})mn} \leq \mathbb{E}^2[Y_{\mathbb{L}}^{B'}]\tfrac{\log^8(m)}{2\bar{p}\log^6(1/\bar{p})mn} \tag{4.22}$$

From Lemmas 12 and 13, we have $l_0^{B'} = \frac{1}{\log(1/(1-p))}[\log(n)+\log(m)-2\log\log(m)-\log(\log(n)+\log(m)-2\log\log(m))-\log\log(\frac{1}{1-p})+\log(p)]+O(1)$ and $\mathbb{E}[Y_{l_0^{B'}-3}^{B'}] \geq (\frac{mn}{e\log^2(m)})^{3(1+o(1))}$. Obviously, we have $\mathbb{L} < l_1^{B'} = l_0^{B'} - 3$ and $\mathbb{E}[Y_{\mathbb{L}}^{B'}] \geq (\frac{mn}{e\log^2(m)})^{3(1+o(1))}$. By setting the probability $p^{\dagger} = \frac{\bar{p}\log^6(\frac{1}{\bar{p}})mn}{\mathbb{E}[Y_{\mathbb{L}}^{B'}]\log^8(m)} < 1$, we can bound the average number of X, $\mathbb{E}[X]$, in e.q. (4.16), as

$$\mathbb{E}[X] \quad \geq \frac{\bar{p}^2\log^6(\frac{1}{\bar{p}})mn}{2\log^8(m)}. \tag{4.23}$$

Plugging (4.23) into (4.15), we can bound the following probability

$$\Pr\{B' \text{ contains no } \mathbb{L}\text{-pattern}\} \leq \exp(-\frac{\bar{p}^2\log^{12}(1/\bar{p})mn}{8\log^{14}(m)}) \tag{4.24}$$

Therefore, we can bound the probability that any subgraph B' induced by m' messages and n' clients does not contain a $\mathbb{L}$-pattern:

$$\begin{aligned}&\Pr\{\exists B' \in \mathcal{B}(B, m', n'), s.t., B' \text{ contains no } \mathbb{L}\text{-pattern}\} \\ &\leq \tbinom{m}{m'}\tbinom{n}{n'}\exp(-\tfrac{\bar{p}^2\log^{12}(1/\bar{p})mn}{8\log^{14}(m)}) \\ &\leq 2^{m+n}\exp(-\tfrac{\bar{p}^2\log^{12}(1/\bar{p})mn}{8\log^{14}(m)}) = o(1).\end{aligned} \tag{4.25}$$

2) For scenario 2, we have

$$\textstyle\sum_{B_2:B_2\sim B_0}\Pr\{X_{B_2}=1|X_{B_0}=1\}\leq\sum_{j=1}^{\mathbb{L}}\binom{\mathbb{L}}{j}\binom{m'-\mathbb{L}}{\mathbb{L}-j}\mathbb{L}!\frac{p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}}{\bar{p}^{j\mathbb{L}}}. \tag{4.26}$$

Let us define the term inside the summation as $\Delta_j\triangleq\binom{\mathbb{L}}{j}\binom{m'-\mathbb{L}}{\mathbb{L}-j}\mathbb{L}!\frac{p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}}{\bar{p}^{j\mathbb{L}}}$.

Then we can see that for $j=1,2,\ldots,\mathbb{L}-1$, we have

$$\begin{aligned}\frac{\Delta_{j+1}}{\Delta_j} &= \frac{(\mathbb{L}-j)^2}{(j+1)(m'-2\mathbb{L}+j+1)}\bar{p}^{-\mathbb{L}}\\ &\leq \frac{\mathbb{L}^2}{m'}\bar{p}^{-\mathbb{L}}\\ &\leq \frac{\frac{1}{\log^2(1/\bar{p})}\log^2(m)}{m(1-o(1))}\frac{m\log^2(1/\bar{p})}{\log^3(m)}\\ &= o(1).\end{aligned} \tag{4.27}$$

This implies that for all $j=1,2,\ldots,\mathbb{L}$, $\Delta_j\leq\Delta_1$.

For Δ_1, we have the following

$$\begin{aligned}\frac{\Delta_1}{\mathbb{E}[Y_{\mathbb{L}}^{B'}]} &= \frac{\mathbb{L}\binom{m'-\mathbb{L}}{\mathbb{L}-1}\mathbb{L}!\frac{p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}}{\bar{p}}}{\binom{m'}{\mathbb{L}}\binom{n'}{\mathbb{L}}\mathbb{L}!p^{\mathbb{L}}(1-p)^{\mathbb{L}(\mathbb{L}-1)}}\\ &= \frac{\mathbb{L}(m'-\mathbb{L})!(m'-\mathbb{L})!}{\bar{p}m'!(m'-2\mathbb{L}+1)!}\\ &\leq \frac{\mathbb{L}^2}{\bar{p}m(1-o(1))}\leq\frac{\log^2(m)(1+o(1))}{\bar{p}m\log^2(1/\bar{p})}.\end{aligned} \tag{4.28}$$

Next, we have

$$\mathbb{E}[|\mathcal{P}|]\ \ \leq\tfrac{1}{2}\mathbb{E}^2[Y_{\mathbb{L}}^{B'}]\mathbb{L}\frac{\log^2(m)(1+o(1))}{\bar{p}\log^2(1/\bar{p})m}\leq\mathbb{E}^2[Y_{\mathbb{L}}^{B'}]\frac{\log^3(m)}{2\bar{p}\log^3(1/\bar{p})m} \tag{4.29}$$

From Lemmas 12 and 13, we have $l_0^{B'}=\frac{1}{\log(1/(1-p))}[\log(\mathbb{L})+\log(m-\mathbb{L})-\log(\log(\mathbb{L})+\log(m-\mathbb{L}))-\log\log(\frac{1}{1-p})+\log(p)]+O(1)=\frac{1}{\log(1/(1-p))}[\log(\mathbb{L})+\log(m-\mathbb{L})-\log(\log(\mathbb{L})+\log(m-\mathbb{L}))-\log\log(\frac{1}{1-p})+\log(p)]+O(1)=\frac{1}{\log(1/(1-p))}[\log(m)-2\log\log(\frac{1}{1-p})+\log(p)]+O(1)$ and $\mathbb{E}[Y_{l_0^{B'}-3}^{B'}]\geq(\frac{m\mathbb{L}}{e})^{3(1+o(1))}$. Obviously, we have $\mathbb{L}<l_1^{B'}=l_0^{B'}-3$ and $\mathbb{E}[Y_{\mathbb{L}}^{B'}]\geq(\frac{m\mathbb{L}}{e})^{3(1+o(1))}$. By setting the probability $p^{\dagger}=\frac{\bar{p}\log^3(\frac{1}{\bar{p}})m(1-o(1))}{\mathbb{E}[Y_{\mathbb{L}}^{B'}]\log^3(m)}<1$, we can bound the average number of X, $\mathbb{E}[X]$, by

$$\mathbb{E}[X] \;\; \geq \tfrac{\bar{p}\log^3(\frac{1}{\bar{p}})m(1-o(1))}{2\log^3(m)}. \tag{4.30}$$

We then can bound the following probability using Azuma's inequality

$$\begin{aligned} \Pr\{B' \text{ contains no } \mathbb{L}\text{-pattern}\} \;\; &\leq \exp(-\tfrac{\bar{p}^2\log^6(1/\bar{p})m^2(1-o(1))}{8\log^6(m)m'n'}) \\ &\leq \exp(-\tfrac{\bar{p}^2\log^7(1/\bar{p})m(1-o(1))}{8\log^7(m)}). \end{aligned} \tag{4.31}$$

Therefore, we can bound the probability that any subgraph B' induced by m' messages and n' clients does not contain a $\mathbb{L}$-pattern:

$$\begin{aligned} &\Pr\{\exists B' \in \mathcal{B}(B, m', n'), s.t., B' \text{ contains no } \mathbb{L}\text{-pattern}\} \\ &\leq \tbinom{m}{m'}\tbinom{n}{n'}\exp(-\tfrac{\bar{p}^2\log^7(1/\bar{p})m(1-o(1))}{8\log^7(m)}) \\ &\leq m^n 2^n \exp(-\tfrac{\bar{p}^2\log^7(1/\bar{p})m(1-o(1))}{8\log^7(m)}) = o(1), \end{aligned} \tag{4.32}$$

where the last equality follows from that $n \leq \log^{15}(m)$.

3) For scenario 3, we have

$$\begin{aligned} &\textstyle\sum_{B_2:B_2\sim B_0}\Pr\{X_{B_2} = 1 | X_{B_0} = 1\} \\ &\leq \textstyle\sum_{j=1}^{\mathbb{L}}\sum_{i=1}^{\mathbb{L}c}\tbinom{\mathbb{L}}{j}\tbinom{m'-\mathbb{L}}{\mathbb{L}-j}\tbinom{\mathbb{L}c}{i}\tbinom{n'-\mathbb{L}c}{\mathbb{L}c-i}\tbinom{\mathbb{L}c}{c,c,\ldots,c}\tfrac{p^{\mathbb{L}c}(1-p)^{\mathbb{L}c(\mathbb{L}-1)}}{\bar{p}^{ij}}, \end{aligned} \tag{4.33}$$

For simplicity, let us define the term inside the summation as $\Delta_{i,j} \triangleq \tbinom{\mathbb{L}}{j}\tbinom{m'-\mathbb{L}}{\mathbb{L}-j}\tbinom{\mathbb{L}c}{i}$ $\tbinom{n'-\mathbb{L}c}{\mathbb{L}c-i}\tbinom{\mathbb{L}c}{c,c,\ldots,c}\tfrac{p^{\mathbb{L}c}(1-p)^{\mathbb{L}c(\mathbb{L}-1)}}{\bar{p}^{ij}}$.

Then we can see that for $j = 1, 2, \ldots, \mathbb{L}$ and $i = 1, 2, \ldots, \mathbb{L}c - 1$, we have

$$\begin{aligned} \tfrac{\Delta_{i+1,j}}{\Delta_{i,j}} \;\; &= \tfrac{(\mathbb{L}c-i)^2}{(i+1)(n'-2\mathbb{L}c+i+1)}\bar{p}^{-j} \\ &\leq \tfrac{\mathbb{L}^2c^2}{n'}\bar{p}^{-\mathbb{L}} \\ &\leq \tfrac{\frac{1}{\log^2(1/\bar{p})}c^3\log(n)}{n}\tfrac{n\log^2(1/\bar{p})}{c^3\log^3(n)} \\ &\leq 1. \end{aligned} \tag{4.34}$$

This implies that for all $i = 1, 2, \ldots, \mathbb{L}c$, $\Delta_{i,j} \leq \Delta_{1,j}$.

We also note that for $j = 1, 2, \ldots, \mathbb{L} - 1$, we have

$$\begin{aligned} \tfrac{\Delta_{1,j+1}}{\Delta_{1,j}} &= \tfrac{(\mathbb{L}-j)^2}{(j+1)(m'-2\mathbb{L}+j+1)}\bar{p}^{-1} \\ &\leq \tfrac{\mathbb{L}^2}{2\bar{p}(m'-2\mathbb{L}+2)} \\ &\leq \tfrac{\mathbb{L}^2}{\bar{p}m'} \\ &\leq \tfrac{\log^3(n)}{\bar{p}\log^2(1/\bar{p})m} \\ &= o(1). \end{aligned} \tag{4.35}$$

This implies that for all $i = 1, 2, \ldots, \mathbb{L}c$ and $j = 1, 2, \ldots, \mathbb{L}$, $\Delta_{i,j} \leq \Delta_{1,j} \leq \Delta_{1,1}$.

For $\Delta_{1,1}$, we have the following

$$\begin{aligned} \tfrac{\Delta_{1,1}}{\mathbb{E}[Y_{\mathbb{L}}^{B'}]} &= \tfrac{\mathbb{L}\binom{m'-\mathbb{L}}{\mathbb{L}-1}\mathbb{L}c\binom{n'-\mathbb{L}c}{\mathbb{L}c-1}\binom{\mathbb{L}c}{c,c,\ldots,c}\frac{p^{\mathbb{L}c}(1-p)^{\mathbb{L}c(\mathbb{L}-1)}}{\bar{p}}}{\binom{m'}{\mathbb{L}}\binom{n'}{\mathbb{L}c}\binom{\mathbb{L}c}{c,c,\ldots,c}p^{\mathbb{L}c}(1-p)^{\mathbb{L}c(\mathbb{L}-1)}} \\ &\leq \tfrac{\mathbb{L}^4c^2}{\bar{p}m'n'} \\ &\leq \tfrac{c^3\log^6(n)}{\bar{p}mn\log^4(1/\bar{p})}. \end{aligned} \tag{4.36}$$

Next, we have

$$\mathbb{E}[|\mathcal{P}|] \;\leq \tfrac{1}{2}\mathbb{E}^2[Y_{\mathbb{L}}^{B'}]\mathbb{L}^2c\tfrac{c^3\log^6(n)}{\bar{p}mn\log^4(1/\bar{p})} \leq \mathbb{E}^2[Y_{\mathbb{L}}^{B'}]\tfrac{c^4\log^8(n)}{2\bar{p}\log^6(1/\bar{p})mn} \tag{4.37}$$

From Lemmas 12 and 13, we have $l_0^{B'} = \frac{1}{\log(1/(1-p))}[\log(n) + \frac{\log(m)}{c} - 2\log(c) - (1 + 1/c)\log\log(n) - \frac{\log[\log(n)+\log(m)/c-2\log(c)]}{c} - \frac{\log\log(1/(1-p))}{c} + \log(p)] + O(1)$ and $\mathbb{E}[Y_{l_0^{B'}-3}^{B'}] \geq (\frac{n}{ec^2\log(n)})^{3c(1+o(1))}(\frac{m}{\log(n)})^{3(1+o(1))}$. Obviously, we have $\mathbb{L} < l_1^{B'} = l_0^{B'} - 3$ and $\mathbb{E}[Y_{\mathbb{K}}^{B'}] \geq (\frac{n}{ec^2\log(n)})^{3c(1+o(1))}(\frac{m}{\log(n)})^{3(1+o(1))}$. By setting the probability $p^{\dagger} = \frac{\bar{p}\log^6(\frac{1}{\bar{p}})mn}{\mathbb{E}[Y_{\mathbb{L}}^{B'}]c^4\log^8(n)} < 1$, we can bound the average number of X, $\mathbb{E}[X]$, by

$$\mathbb{E}[X] \;\geq \tfrac{\bar{p}\log^6(\frac{1}{\bar{p}})mn}{2c^4\log^8(n)}. \tag{4.38}$$

We then can bound the following probability using Azuma's inequality

$$\begin{aligned}\Pr\{B' \text{ contains no } \mathbb{L}\text{-pattern}\} &\leq \exp(-\tfrac{\bar{p}^2\log^{12}(1/\bar{p})m^2n^2}{8c^8\log^{16}(n)m'n'}) \\ &\leq \exp(-\tfrac{\bar{p}^2\log^{12}(1/\bar{p})mn}{8c^7\log^{14}(n)}).\end{aligned} \tag{4.39}$$

Therefore, we can bound the probability that any subgraph B' induced by m' messages and n' clients does not contain a $\mathbb{L}$-pattern:

$$\begin{aligned}&\Pr\{\exists B' \in \mathcal{B}(B, m', n'), s.t., B' \text{ contains no } \mathbb{L}\text{-pattern}\} \\ &\leq \tbinom{m}{m'}\tbinom{n}{n'}\exp(-\tfrac{\bar{p}^2\log^{12}(1/\bar{p})mn}{8c^7\log^{14}(n)}) \\ &\leq 2^{m+n}\exp(-\tfrac{\bar{p}^2\log^{12}(1/\bar{p})mn}{8c^7\log^{14}(n)}) = o(1),\end{aligned} \tag{4.40}$$

where the last equality follows from that $c = o(\frac{n^{1/7}}{\log^2(n)})$. □

We then can reiterate Theorem 16 in a slightly different way.

Theorem 16′. *The number of broadcast transmissions for random graph instance $B(m, n, c, p)$ is almost surely upper bounded by*

- $O(\frac{n}{\log(m)})$*, for any* $c \geq 1$*;*
- $O(\frac{n}{c\log(n)})$*, for* $c = o(\frac{n^{1/7}}{\log^2(n)})$*;*
- $O(\frac{n}{c} + \log(c))$*, for* $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$*.*

Proof. For scenarios 1, 2, and 3, we can proceed using the following transmission scheme.

- Start from the original bipartite graph representation B. If there are more than m' messages and n' clients in the graph, we pick a $\mathbb{L}$-pattern to encode the messages and make one transmission. We remove the satisfied clients and the used messages. If there are less than n' clients, we can almost surely use n' transmissions to satisfy the remaining clients, which follows from that a subgraph of B that contains n' vertices on both sides almost surely have a perfect matching [ER66].

From Lemma 14, we can see that the above scheme can be done almost surely. Hence, we can almost surely use the number of transmissions $\frac{n}{\mathbb{L}c} + n'$, from which the first two parts follow.

Now, let us prove the third part of the theorem for scenario 4 with $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$. We use a slightly different but simple proof technique. By setting $n' = \frac{2c}{p}$, we use a 2-step transmission scheme.

• In the first step, we arbitrarily make n/c uncoded transmissions. After each transmission, we remove up to c satisfied clients as many as possible.

• In the second step, we divide the remaining unsatisfied clients into groups as few as possible, each with up to c clients, we use pliable index coding scheme to satisfy each of the groups.

We want to show that we can almost surely satisfy at least $n - n'$ clients by using these n/c uncoded transmissions in the first step. Hence, we can almost surely divide the remaining unsatisfied clients into at most $n'/c = 2/p$ groups and these groups almost surely take $\frac{2}{p}O(\log(c))$ broadcast transmissions from Chapter 2.4.

For a fixed uncoded transmission, e.g., message b_j, we would like to show that the probability that this transmission cannot satisfy c clients is exponentially small if the remaining unsatisfied clients is more than n'. Let us denote by D the number of connections for message vertex b_j to any n' remaining client vertices. Then obviously, $\mathbb{E}[D] = n'p = 2c$ and the probability that the uncoded transmission of b_j cannot satisfy c clients (i.e., a 1-pattern exists) can be bounded by the following Chernoff bound:

$$\begin{aligned} \Pr\{b_j \text{ cannot satisfy } c \text{ clients}\} &\leq \Pr\{D \leq c\} = \Pr\{D \leq (1-1/2)\mathbb{E}[D]\} \\ &\leq \exp(-\tfrac{2c(1/2)^2}{2}) = \exp(-\tfrac{c}{4}). \end{aligned} \tag{4.41}$$

After n/c uncoded transmissions, the probability that the number of remaining unsatisfied clients is more than n' can be bounded as follows:

$$\begin{aligned} &\Pr\{n/c \text{ uncoded transmissions cannot satisfy } n-n' \text{ clients}\} \\ &\leq \tfrac{n}{c}\exp(-\tfrac{c}{4}) \leq n^{6/7}\log^2(n)o(1)\exp(-\tfrac{n^{1/7}\Omega(1)}{4\log^2(n)}) = o(1). \end{aligned} \tag{4.42}$$

Combining the two steps, we have the number of broadcast transmissions almost surely upper bounded by $n/c + \frac{2}{p}O(\log(c)) = O(n/c + \log(c))$ for scenario 4. □

Note that for Theorem 16′, we can combine the results and have the number of broadcast transmissions almost surely upper bounded by $O(\min\{\frac{n}{\log(m)}, \frac{n}{c\log(n)}\})$ for $c = o(\frac{n^{1/7}}{\log^2(n)})$ and $O(\min\{\frac{n}{\log(m)}, \frac{n}{c} + \log(c)\})$ for $c = \Omega(\frac{n^{1/7}}{\log^2(n)})$.

4.3 Application to Distributed Computing

4.3.1 Model and Performance Metric

Consider a distributed computing system, with one master node with m messages $b_1, b_2, \ldots, b_m$ and n worker nodes. Each worker $i \in [n]$ is equipped with a cache of size s. The system solves a computational problem $x = f(b_1, b_2, \ldots, b_m)$ in iterations, where at iteration t: the master broadcasts the current estimate x^{t-1} to all workers; workers perform local computations and send to the master their new estimate x_i^t; the master node combines local estimates to get an updated estimate; he then performs data shuffling. In data shuffling, the master node makes broadcast transmissions (that may be encoded) to the workers and each worker replaces some of the old messages with the new messages that she can decode.

In machine learning and distributed learning community, although to what extent the data should be shuffled is far from understood, the idea that shuffling needs to be done such that caches store a certain fraction of different data across workers and iterations is well recognized through practice in order to achieve sufficient statistical gains.

Motivated by this practical consideration, we use a Hamming distance metric to evaluate the performance of data shuffling algorithms. Let us define the *cache state* for worker node i in time period t be $z_i^t \in \{0,1\}^m$, where the j-th bit of z_i^t takes value 1 if message b_j is in the cache of worker node i in time period t and 0 otherwise. The Hamming distance between two cache states z and z', denoted by $H(z, z')$, is the number of positions where the entries are different for z and z'. We define the Hamming distance of a shuffling scheme as the average Hamming distance across time and workers $H \triangleq \frac{1}{\binom{Tn}{2}} \sum_{1\le t,t'\le T,(i,i')\in[n]^2,(t,i)\neq(t',i')} \mathbb{E}[H(z_i^t, z_{i'}^{t'})]$, where T denotes the number of iterations.

4.3.2 Data Shuffling Scheme

4.3.2.1 Hierarchical Structure

We partition the messages into m/m_1 groups so that each group g contains m_1 disjoint messages. In our scheme, each worker i gets allocated messages from groups indexed by a set $D(i)$; each group g allocates messages to workers indexed by a set $N(g)$. We can represent this relationship using a bipartite graph: at one side there are m/m_1 groups, and at the other side there are n workers; there is a connection between worker i and group g if and only if worker i caches messages from group g, i.e., $g \in D(i)$; the degree of the worker node i is $|D(i)|$ and of the group node g is $|N(g)|$. This structure is maintained for all iterations.

In order to have a large Hamming distance H, we can impose the constraint that $|D(i) \cap D(i')| \leq 1$ for any two worker nodes i and i', namely, they have common messages in no more than one group. Moreover, to balance the messages cached in different worker nodes, we would like that $|N(g)|$ is the same for all groups. We thus select for our scheme to use $|D(i)| = \frac{s}{m_1(1-1/r)}$, and $|N(g)| = \frac{ns}{m(1-1/r)}$, where the design parameter $2 \leq r \leq m_1$ takes integer values.

Note that because of the requirement that the same message can be decoded and stored by at most c workers, we need that no more than rc workers cache messages in group g, i.e., $|N(g)| \leq cr$. For example, for $c = 1$, then at most r workers can be in $N(g)$, each one of them with $m_1(1 - 1/r)$ cached messages from this group.

4.3.2.2 Transmissions

For a message group g and associated clients $N(g)$, we consider as a constrained pliable index coding instance and design a simple transmission scheme that proceeds as follows.

- Initialization: the cache of worker i is filled with uniformly at random selected $m_1(1-1/r)$ messages from each group in $D(i)$, thus in total s messages.
- Iteration t: the master makes m/m_1 broadcast transmissions, one for every group. For each group g, the master selects uniformly at random r messages in the group and transmits

their linear combination, say $b_{j_1}+b_{j_2}+\ldots+b_{j_r}$. From the following Lemma 15, every worker in $N(g)$ can decode a new message with probability at least $1/e$. The workers who can decode a new message store it in their cache and discard an old message; they select the old message to discard uniformly at random from the messages in their cache that are also contained in the broadcast transmission, i.e., one from $\{b_{j_1}, b_{j_2}, \ldots, b_{j_r}\}$.

Lemma 15. *A worker with $m_1(1-1/r)$ cached messages from group g that receives a linear combination $b_{j_1}+b_{j_2}+\ldots+b_{j_r}$ of r messages uniformly at random selected from g, can decode a message she does not have with probability at least $1/e$.*

Proof. Without loss of generality, assume the worker has cached the messages $1, 2, \ldots, m_1(1-1/r)$ and requires a new message from the remaining m_1/r messages. The probability that there is exactly one message in the r data pieces $b_{j_1}, b_{j_2}, \ldots, b_{j_r}$ selected from the last m_1/r data pieces is lower bounded by

$$\begin{aligned}
p_1 \quad &\triangleq \Pr\{\text{The worker can decode a new message}\}\\
&\geq \frac{\binom{m_1/r}{1}\binom{m_1(1-1/r)}{r-1}}{\binom{m_1}{r}}\\
&= \frac{(m_1-\frac{m_1}{r})(m_1-\frac{m_1}{r}-1)...(m_1-\frac{m_1}{r}-r+2)}{(m_1-1)(m_1-2)...(m_1-r+1)}\\
&\geq (\frac{m_1-\frac{m_1}{r}-r-2}{m_1-r+1})^{r-1} = (1-\frac{m_1-r}{r(m_1-r+1)})^{r-1}\\
&\geq (1-\frac{1}{r})^{r-1} \geq \frac{1}{e}
\end{aligned}$$

□

4.3.3 Algorithm Performance

We here theoretically evaluate properties of the proposed algorithm.

- Communication cost. Each data shuffling phase requires m/m_1 broadcast transmissions.
- Satisfying c-constraint. As at most rc workers have cached messages from group g, from Lemma 15, we can see that on average at most rcp_1 (for some fixed $1 > p_1 \geq 1/e$) workers can update their cache with a new message during one transmission. Because we uniformly at random select which r messages to encode, each message can be decoded by cp_1 workers

on average. Hence, on average, the c-constraint is satisfied. Note that this scheme allows us to maintain the randomness property for workers in $N(g)$ (see Chapter 4.3.3.2).

• Hamming distance. Between the caches of any two workers the Hamming distance is at least $2(s - m_1 + m_1/r)$, since any two workers have common messages from at most one group.

Next, we evaluate the Hamming distance across iterations for the same worker. We first consider the Hamming distance $H|_g$ only corresponding to the messages of a specific group g. The average Hamming distance across all iterations is at least the average Hamming Distance between two consecutive iterations (see Chapter 4.3.3.1). Hence, the average Hamming distance $H|_g$ can be lower bounded by:

$$H|_g \quad \geq 0 \cdot (1 - \tfrac{1}{e}) + 2 \cdot \tfrac{1}{e} = 2/e. \tag{4.43}$$

We then consider the average Hamming distance across all the groups in $D(i)$. Since $|D(i)| = \frac{s}{m_1(1-1/r)}$, this is at least $\frac{s}{m_1(1-1/r)} 2/e = \frac{2s}{em_1(1-1/r)}$. Therefore, on average $H \geq \min\{\frac{2s}{em_1(1-1/r)}, 2(s - m_1 + m_1/r)\}$.

• Comparison to Index Coding. Index coding may require in the worst case $\Omega(n)$ broadcast transmissions and $\Theta(n/\log(n))$ for random graph instances to update one message in each cache, and thus $\Omega(ns/em_1(1-1/r))$ (in the worst case) and $\Theta(ns/em_1(1-1/r)\log(n))$ (for random graph instances) broadcast transmissions in each data shuffling iteration to guarantee a Hamming distance of $\frac{2s}{em_1(1-1/r)}$ across time. Using our proposed scheme, we need m/m_1 transmissions to achieve an average Hamming distance of $2s/em_1(1-1/r)$ across time. Note that each client stores s messages as side information, then on average each message is stored on sn/m workers. The benefits of our proposed scheme over index coding (i.e., the ratio of the numbers of transmissions for index coding scheme and for our proposed scheme) is $O(sn/m)$ (in the worst case) and $O(\frac{sn}{m\log(n)})$ (for random graph instances). Additionally, finding the optimal index coding solution is NP-hard, while our scheme has linear complexity of encoding.

4.3.3.1 Hamming Distance Analysis

We analyze the Hamming distance of our pliable index coding based shuffling. We first note that across different worker nodes, the Hamming distance is at least $2(s - m_1 + m_1/r)$, as in the outer layer of the transmission structure, two different worker nodes have common messages in no more than one group.

Next, we evaluate the Hamming distance across iterations for the same worker i. Let us define a *truncated cache state* on group g for worker i at iteration t, $z_i^t|_g \in \{0,1\}^{m_1}$, as a m_1-tuple that consists of coordinates of z_i^t corresponding to messages in group g. We first consider the Hamming distance $H|_g$ between truncated cache state on a specific group g for worker i across iterations. We claim that the average Hamming distance $H|_g$ across all iterations is at least the average Hamming Distance between two consecutive iterations, i.e., for two given iterations $t_1 < t_2$, $\Pr\{z_i^{t_1}|_g = z_i^{t_2}|_g\} \leq \Pr\{z_i^{t_1}|_g = z_i^{t_1+1}|_g\}$.

To prove this, we use a random walk model on a graph $G(V, E)$ that is constructed as follows. Each vertex $v \in V$ corresponds to one of $\binom{m_1}{m_1(1-1/r)}$ possible *truncated cache states* $z_i^t|_g$, or state for short, i.e., all binary vectors of length m_1 and weight $m_1(1 - 1/r)$. There is an edge between two states v_1 and v_2 if and only if their Hamming distance is no more than 2, i.e., each vertex v has a self-loop and there is an edge connecting two vertices of Hamming distance 2. Thus, a vertex v has $m_1^2(1/r - 1/r^2)$ connections with other vertices. Originally, worker i is in any of the $\binom{m_1}{m_1(1-1/r)}$ possible states with equal probability. Using our proposed shuffling scheme, after each iteration, worker i remains in the same state with probability $1 - p_1 \leq 1 - 1/e$ (p_1 is defined as the probability that a worker can decode a new message during each transmission) and changes to a neighboring state with probability $\frac{p_1}{em_1^2(1/r-1/r^2)}$ according to Lemma 15. Assume at iterations t_1, worker i is in some state $v_1 \in V$. At iteration t_2, worker i's state is a random variable with some distribution. Let us denote by p_v^t the probability that worker i is in state v at iteration t. Then we have the flow conservation equation:

$$\begin{aligned} p_{v_1}^{t_2} &= p_{v_1}^{t_2-1}(1-p_1) + \textstyle\sum_{v\neq v_1:\{v,v_1\}\in E} p_v^{t_2-1}\frac{p_1}{m_1^2(1/r-1/r^2)} \\ &= p_{v_1}^{t_2-1}(1-p_1) + p_{v_0}^{t_2-1}\frac{p_1}{m_1^2(1/r-1/r^2)}m_1^2(1/r-1/r^2) \\ &\leq p_{v_1}^{t_2-1}(1-p_1) + \frac{1-p_{v_1}^{t_2-1}}{m_1^2(1/r-1/r^2)}p_1 \\ &\leq 1-p_1 = p_{v_1}^{t_1+1}, \end{aligned} \tag{4.44}$$

where the second equality holds because the probabilities for worker i in v_1's neighbors, $p_v^{t_2-1}$ for $v \neq v_1 : \{v, v_1\} \in E$, are all equal by symmetry, and thus we can pick a fixed neighbor v_0 of v_1; the first inequality holds because $p_{v_0}^{t_2-1}$ is at most $\frac{1-p_{v_1}^{t_2-1}}{m_1^2(1/r-1/r^2)}$, i.e., worker i has equal probability in any of v_1's neighbors by symmetry and the probability that worker i is in one of v_1's neighbors is at most $1-p_{v_1}^{t_2-1}$; and the second inequality holds because the function $g(p_{v_1}^{t_2-1}) = p_{v_1}^{t_2-1}(1-p_1) + \frac{1-p_{v_1}^{t_2-1}}{m_1^2(1/r-1/r^2)}p_1$ is an increasing function and achieves the maximum for $p_{v_1}^{t_2-1} = 1$. Our claim is proved.

Hence, the average Hamming distance $H|_g$ can be lower bounded by:

$$H|_g \geq 0 \cdot (1-\tfrac{1}{e}) + 2 \cdot \tfrac{1}{e} = 2/e. \tag{4.45}$$

We then consider the average Hamming distance across all the groups in $D(i)$. Since $|D(i)| = \frac{s}{m_1(1-1/r)}$, this is at least $\frac{s}{m_1(1-1/r)}2/e = \frac{2s}{em_1(1-1/r)}$. Therefore, on average $H \geq \min\{\frac{2s}{em_1(1-1/r)}, 2(s-m_1+m_1/r)\}$.

4.3.3.2 Independence and Randomness Preserving Property

Originally, if the worker nodes in $N(g)$ have independently and uniformly at random cached $m_1(1-1/r)$ messages in group g, then we observe that the pliable index coding based shuffling scheme maintains this "independence and randomness" property. Without loss of generality, assume the worker nodes in $N(g)$ are $1, 2, \ldots, n_1$, where $n_1 = |N(g)|$. Again, we use the graph constructed above.

Corollary 2. *The pliable index coding based shuffling scheme maintains the "independence* *and randomness" property. Formally, if the following two properties hold for iteration t:*

$$\begin{aligned}&\Pr\{z_1^t|_g = v_1, z_2^t|_g = v_2, \ldots, z_{n_1}^t|_g = v_{n_1}\} \\ &= \Pr\{z_1^t|_g = v_1\} \Pr\{z_2^t|_g = v_2\} \ldots \Pr\{z_{n_1}^t|_g = v_{n_1}\},\end{aligned} \tag{4.46}$$

for any state tuple $(v_1, v_2, \ldots, v_{n_1}) \in V^{n_1}$, and

$$\Pr\{z_i^t|_g = v_i\} = \frac{1}{|V|}, \tag{4.47}$$

for any worker $i \in [n_1]$ and state $v_i \in V$; then these two properties also hold for iteration $t+1$:

$$\begin{aligned}&\Pr\{z_1^{t+1}|_g = v_1', z_2^{t+1}|_g = v_2', \ldots, z_{n_1}^{t+1}|_g = v_{n_1}'\} \\ &= \Pr\{z_1^{t+1}|_g = v_1'\} \Pr\{z_2^{t+1}|_g = v_2'\} \ldots \Pr\{z_{n_1}^{t+1}|_g = v_{n_1}'\},\end{aligned} \tag{4.48}$$

for any state tuple $(v_1', v_2', \ldots, v_{n_1}') \in V^{n_1}$, and

$$\Pr\{z_i^{t+1}|_g = v_i'\} = \frac{1}{|V|}, \tag{4.49}$$

for any worker $i \in [n_1]$ and state $v_i \in V$.

Proof. The second property is obvious. Indeed, by symmetry of the constructed graph, if worker i is in every state with equal probability, then after one iteration (one random walk), worker i remains in every state with equal probability.

We then show the first property. We have the following

$$\begin{aligned}&\Pr\{z_1^{t+1}|_g = v_1', \ldots, z_{n_1}^{t+1}|_g = v_{n_1}'\} \\ &= \textstyle\sum_{(v_1,\ldots,v_{n_1})} \Pr\{z_1^t|_g = v_1, \ldots, z_{n_1}^t|_g = v_{n_1}\} \cdot \\ &\qquad\qquad \Pr\Big\{z_1^{t+1}|_g = v_1', \ldots, z_{n_1}^{t+1}|_g = v_{n_1}' \Big| z_1^t|_g = v_1, \ldots, z_{n_1}^t|_g = v_{n_1}\Big\} \\ &= \tfrac{n_1}{|V|} \textstyle\sum_{(v_1,\ldots,v_{n_1})} \Pr\Big\{z_1^{t+1}|_g = v_1, \ldots, z_{n_1}^{t+1}|_g = v_{n_1} \Big| z_1^t|_g = v_1', \ldots, z_{n_1}^t|_g = v_{n_1}'\Big\} \\ &= \tfrac{n_1}{|V|},\end{aligned} \tag{4.50}$$

where the first equality holds due to the total probability theorem; the second equality holds because of the initial two properties for iteration t, i.e., e.q. (4.46) and (4.47), and the "reversibility property" of the random walk, i.e.,

$$\begin{aligned}&\Pr\{z_1^{t+1}|_g = v_1', \dots, z_{n_1}^{t+1}|_g = v_{n_1}' \Big| z_1^t|_g = v_1, \dots, z_{n_1}^t|_g = v_{n_1}\} \\ &= \Pr\{z_1^{t+1}|_g = v_1, \dots, z_{n_1}^{t+1}|_g = v_{n_1} \Big| z_1^t|_g = v_1', \dots, z_{n_1}^t|_g = v_{n_1}'\}.\end{aligned} \tag{4.51}$$

The "reversibility property" describes that the probability walking from $(v_1, \dots, v_{n_1})$ to $(v_1', \dots, v_{n_1}')$ is equal to that of walking from $(v_1', \dots, v_{n_1}')$ to $(v_1, \dots, v_{n_1})$. Indeed, if we use the same coded transmission and a reverse discarding process, then we achieve the goal. For example, if a worker has messages $\{1, 2, 3\}$ in its cache, and the transmission is $b_1+b_2+b_3+b_4$; then the worker decodes message 4 and replaces message 1 and at last has cached messages $\{2, 3, 4\}$. If we reverse the process, we start from messages $\{2, 3, 4\}$ in cache; using the same transmission $b_1 + b_2 + b_3 + b_4$, the worker decodes b_1 and replaces message 4, resulting in cached messages $\{1, 2, 3\}$. This can be done with equal probability across all workers. The corollary is proved. □

4.3.4 Experimental Results

We conduct experiments on distributed machine learning over a real data set[2] that aims to detect diseased trees in an image. We train the distributed classification model using a stochastic gradient descent method based on 1000 data instances (messages). We set the number of workers to $n = 300$ and the cache size to $s = 10$. We divide the messages into 50 groups, with 20 messages in each. We set the parameter $r = 2$, i.e., each worker has cached messages in 1 group. We carry out experiments by comparing our hierarchical pliable index coding based shuffling against: (i) no shuffling and (ii) shuffling with randomly selected messages. For case (ii), once we randomly select what message to send to each worker, we use two approaches for broadcasting: uncoded broadcast transmissions, and index coding [LLP15, CS08]. We implemented index coding using the graph coloring based heuristic approach in [CS08].

[2] https://archive.ics.uci.edu/ml/datasets/Wilt#

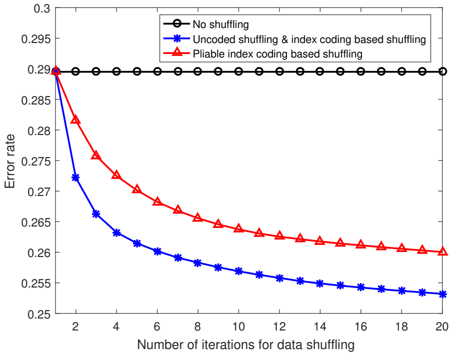


Figure 4.4: Comparison of computation performance for different data shuffling schemes.

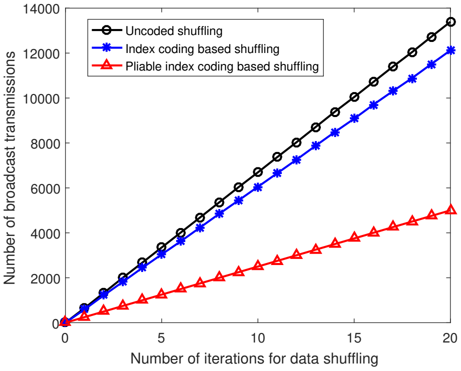


Figure 4.5: Comparison of broadcast transmissions for different data shuffling schemes.

In Fig. 4.4, we compare the computation performance of our pliable index coding based shuffling scheme with no data shuffling scheme and the data shuffling scheme with randomly selected messages (uncoded shuffling and index coding based shuffling use the same cached messages in each local computation, and only differ in the data shuffling phase). We find that data shuffling improves the performance by 11.4% as compared to no shuffling; pliable coding shuffling performs very similarly (2.6% worse) to randomized shuffling. In Fig. 4.5, we compare the number of broadcast transmissions for the data shuffling schemes: uncoded shuffling, index coding based shuffling and pliable index coding based shuffling. We find that our proposed pliable index coding based shuffling scheme saves 63% and 59% in terms of the number of transmissions as compared to the uncoded shuffling and to the index coding based shuffling, respectively.

4.4 Open Question and Future Work

In this chapter, we discuss how to use pliable index coding to reduce the communication cost in distributed computing. We consider a "master-workers" system model and show the benefits of using pliable index coding over index coding. Our future work will examine other distributed computing system models and try to investigate benefits that coding can offer for communication. We observe that the communication cost relies highly on the type of distributed algorithms. For example, to train a classifier in a distributed manner, we can average the distributed classifier parameters to get an aggregated classifier; but if we want to train a decision tree, we may also need to exchange information at each layer when constructing the tree. Hence, we would like to see which type of computation task can benefit more from our proposed framework. Moreover, our framework may enable the design of new distributed algorithms that can rely more heavily on communication, since our approach can make communication cheaper.

4.5 Summary

In this chapter, we discuss how to use pliable index coding to realize data shuffling in order to reduce communication bottleneck in distributed computing system. We first pose a constraint on pliable index coding that requires each message achieve no more than c clients and show fundamental limits on this problem: upper and lower bounds, average performance over random instances. We then propose a hierarchical structure and data shuffling scheme that admits this constrained pliable index coding as a basic building block. We find benefits up to $O(ns/m)$ over index coding, where ns/m is the average number of workers caching a message, and m, n, and s are the numbers of messages, workers, and cache size, respectively.

CHAPTER 5

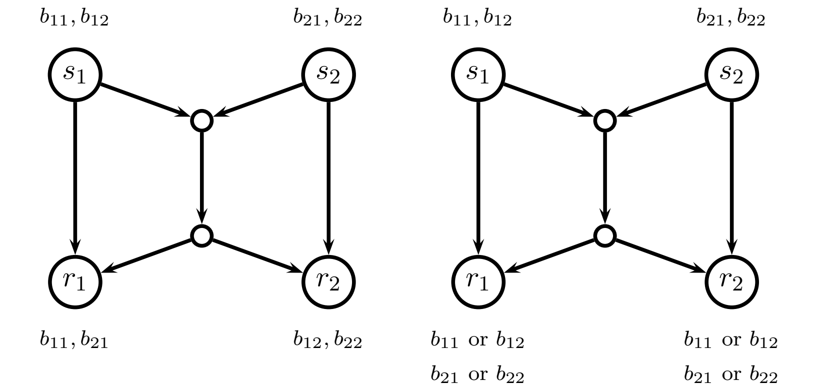
Content-Type Coding over Large Networks and Lossy Networks

5.1 Introduction

In this chapter, we sample more problems within the scope of content-type coding and show the benefits over conventional message-specific coding. We go beyond the index coding framework and consider two other content-type coding frameworks: one is over large scale networks and the other is over lossy networks.

First, we study the content-type coding over large scale networks. We are interested in two issues: the performance of content-type coding for sufficiently large networks and how the benefit over message-specific coding changes with the network size. In network coding literature, the combination network is often studied to show the performance of network coding over large scale networks, e.g., [NY04]. Here, we consider a "combination-like" network and show that if we serve types of content instead of specific messages over the network to users, we can achieve as many benefits as the size of the content type (i.e., the minimum number of messages inside a content type) allows when network size is sufficiently large.

Second, we consider a broadcast erasure channel with "ACK-NACK" feedbacks, where a source wants to send content-type messages to two receivers. We theoretically show the capacity of such a content-type broadcast erasure channel and propose an achievable transmission scheme using content-type coding. Compared with the capacity of message-specific broadcast erasure channel [GT09], we show that we can always achieve certain benefits (i.e.,



(a) Requesting specific messages. (b) Requesting one message from each type.

Figure 5.1: Example where each source has messages of a given type.

the capacity region of the content-type broadcast erasure channel always covers that of the message-specific broadcast erasure channel), and the capacity benefit can be up to 19.5% under a symmetric setting.

The work presented in this chapter was published in [SF15].

5.2 Content-Type Coding over Large Networks

5.2.1 Motivating Example

Fig. 5.1 provides a simple example of content-type coding benefits. We start from the classical butterfly network, with two sources, s_1 and s_2, two receivers, r_1 and r_2, and unit capacity directed links, where each source has 2 messages of a given type. In the example, source s_1 (say advertising hotels) has 2 unit rate messages, $\{b_{11}, b_{12}\}$, and source s_2 (say advertising car rentals) also has 2 unit rate messages $\{b_{21}, b_{22}\}$. The min-cut to each receiver is two, thus she used all the network resources by herself, we could send to her two specific messages, for instance, to r_1: b_{11} and b_{21} and to r_2: b_{12} and b_{22}. However, if we want to send to both receivers these specific requests, it is clearly not possible, as there are no coding opportunities; we can satisfy one receiver but not both. In contrast, if each receiver requires one message from each type, then we can multicast say b_{11} and b_{21} to both receivers.

5.2.2 Problem Formulation

We consider a network represented as a directed graph $G = (V, E)$, a set of m sources $\{s_1, s_2, \cdots, s_m\}$ and a set of n receivers $\{r_1, r_2, \cdots, r_n\}$. Each source has u messages of the *same* type, and different sources have *different* types of messages. We denote by $\mathcal{B}_j = \{b_{j1}, b_{j2}, \cdots, b_{ju}\}$ the set of type-j messages from source s_j.

Given a transmission scheme, we use the rate towards a receiver to measure the efficiency of the transmission scheme. In the conventional message-specific coding problem, each receiver r_i requests specific messages, one from each type, denoted by a fixed element $(b_1, b_2, \cdots, b_m) \in \mathcal{B}_1 \times \cdots \times \mathcal{B}_m$. For K transmissions, the messages requested by receiver r_i are denoted by $(b_1^K, b_2^K, \cdots, b_m^K)$, where each element is a vector $b_j^K = (b_j(1), \cdots, b_j(k), \cdots, b_j(K))$.

We denote by $\mathcal{R}_i$ the set of messages that the receiver r_i can decode after K transmissions, and we say that the transmission rate towards r_i is the number of requested messages that can be decoded by the receiver r_i per transmission, i.e.,

$$R_i = \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{m} I_{\{b_j(k) \in \mathcal{R}_i\}}. \tag{5.1}$$

In the content-type formulation, each receiver r_i requests to receive (any) one message from each type, and does not care which specific one. We denote the requested content-type messages by an arbitrary element $x_1, x_2, \cdots, x_m \in \mathcal{B}_1 \times \cdots \times \mathcal{B}_m$. For K transmissions, the messages requested by r_i are denoted by $(x_1^K, x_2^K, \cdots, x_m^K)$, where each element is a vector that is represented as $x_j^K = (x_j(1), \cdots, x_j(k), \cdots, x_j(K))$. Similarly, the rate towards r_i is defined as the number of requested messages that can be decoded by the receiver r_i per transmission, i.e.,

$$R_i^c = \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{m} I_{\{\exists x_j(k) \in \mathcal{R}_i, \text{ such that } x_j(k) \in \mathcal{B}_j\}}. \tag{5.2}$$

We would like to study the rate averaged among all receivers for message-specific coding, denoted by R, and the that for content type coding, denoted by R^c. Clearly, for message specific coding, the rate depends upon the specific message requests. We denote by R^w the worst-case rate (minimizing among all possible sets of requests), and by R^a the average rate (averaged over all possible sets of requests). We define the worst case and average case gains, respectively, as:

$$G^w = \tfrac{R^c}{R^w}, \quad G^a = \tfrac{R^c}{R^a}. \tag{5.3}$$

5.2.3 Combination-Like Network

We consider the combination-like network structure $G(m, h, u)$, where $m \leq h$. shown in Fig. 5.2. The network has four layers: the first layer is the m sources and each source connects to every node in the second layer; the second layer has h intermediate A nodes and each A node connects to a B node in the third layer, which also contains h intermediate B nodes; the fourth layer contains $n = \binom{h}{m}u^m$ receivers, where we have u^m receivers connected to each subset of size m of the B nodes.

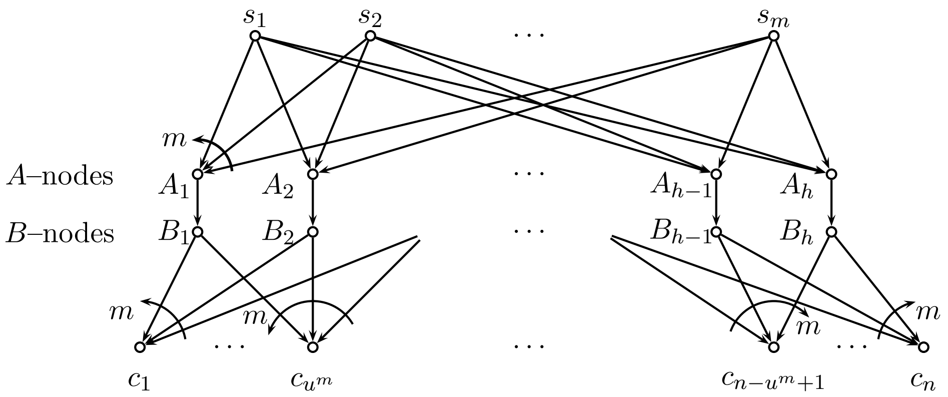


Figure 5.2: Combination-like network structure.

Theorem 17. *In a $G(m, h, u)$ network,*

$$G^w = u, \quad \lim_{h\to\infty} G^a = u. \tag{5.4}$$

Proof. We first note that in the network $G(m, h, u)$, content type coding achieves $R^c = m$, as we can use network coding to multicast one specific message from each type to all receivers.

We next show that in the network $G(m, h, u)$, the worst case message-specific coding rate is $R^w = m/u$. We construct the following receiver requirements: for every u^m receivers that are connected to the same B nodes, each request is an element of the set $\mathcal{B}_1 \times \cdots \times \mathcal{B}_m$, and no two requests are the same. Since all mu messages are requested, we need to use the same set of m A-B edges, at least u times. Using network coding we can ensure that at the end of u transmissions, each receiver gets all mu messages and thus also the m messages required by her; thus the transmission rate is m/u. Note that receiving all mu messages by each receiver is a worst case scenario in terms of rate.

Finally, we show that in a network $G(m, h, u)$, the average rate of the message-specific coding problem is bounded by

$$R^a \leq \frac{m}{u} + \frac{m}{u^{\frac{m+1}{2}}} + \frac{m^2(1+\sqrt{\ln u})}{u^{\frac{m+1}{2}}} + \frac{(m+1)^2\sqrt{\ln u}}{\sqrt{\binom{h}{m}u^{\frac{m+1}{2}}}}. \tag{5.5}$$

The idea is as follows. We consider m out of the h edges that connect A to B nodes, and the u^m receivers that can be reached only through these edges. We call these m edges and u^m receivers a basic structure. We argue that, through each such structure, we need to send almost surely all messages to have a good average - with high probability all messages are required by less than $u^{m-1}(1-\delta_1)$ receivers. Thus the rate we can get through each basic structure converges to m/u.

Then, let us discuss in detail. Let us denote by $\mathcal{E} = \{e_1, e_2, \cdots, e_h\}$ the set of h edges connecting A nodes and B nodes. We refer to m different edges in $\mathcal{E}$ and the u^m receivers that are only connected to them, as a basic structure. From construction, there are $\binom{h}{m}$ such structures. For each structure, it is straightforward to see that:

- For any basic structure, the maximum transmission rate through these m edges to the receivers in this structure is m (since the min-cut is m).

- Denote by s_{jv} the number of requirements of message $b_{jv} \in \mathcal{B}_j$ ($v \in [u]$) for receivers in a basic structure (i.e., the number of receivers requesting this message). Then the maximum transmission rate is $\sum_{l=1}^{m} \max_l s_{jv}/u^m$, where $\max_l s_{jv}$ represents the l-th maximum number among $\{s_{jv} | 1 \leq j \leq m, 1 \leq v \leq u\}$. This means that the maximum transmission rate is achieved by transmitting the m most popular messages through these m edges.

Consider any given basic structure. We first observe that $E[s_{jv}] = u^{m-1}$. We define the event $E_{jv}^{\delta_1} = \{s_{jv} > E[s_{jv}](1+\delta_1)\}$ as an abnormal event with respect to the message b_{jv}. We also define the event $E^{\delta_1} = \cup_{1 \le j \le m, 1 \le v \le u} E_{jv}^{\delta_1}$ as an abnormal event for this basic structure. From the Chernoff bound, we have that

$$\Pr\{E_{jv}^{\delta_1}\} = \Pr\{s_{jv} > E[s_{jv}](1+\delta_1)\} \le e^{-\frac{u^{m-1}\delta_1^2}{3}}, \tag{5.6}$$

and

$$p_1 = \Pr\{E^{\delta_1}\} = um \Pr\{E_{jv}^{\delta_1}\} \le ume^{-\frac{u^{m-1}\delta_1^2}{3}}. \tag{5.7}$$

Here we denote by p_1 the probability that an abnormal event happens. When an abnormal event happens for this basic structure, the rate for this structure is (at most) m. If an abnormal event does not happen for this basic structure, the rate for this structure is (at most) $mu^{m-1}(1+\delta_1)/u^m = m(1+\delta_1)/u$.

Next, we consider the $w = \binom{h}{m}$ structures. Let us denote by T_E the number of structures with an abnormal event happening. The expected value of T_E is wp_1. The probability that $\{T_E > wp_1(1+\delta_2)\}$ happens, denoted by p_2, can be bounded using the Chernoff bound:

$$p_2 = \Pr\{T_E > wp_1(1+\delta_2)\} \le e^{-\frac{wp_1\delta_2^2}{3}}, \tag{5.8}$$

Hence, if the event $\{T_E > wp_1(1+\delta_2)\}$ does not happen, the number of structures with an abnormal event is at most $wp_1(1+\delta_2)$. Therefore, the average rate can be bounded by

$$\begin{aligned} R^a &\le p_2 m + (1-p_2)[\tfrac{wp_1(1+\delta_2)m + (w - wp_1(1+\delta_2))\frac{m(1+\delta_1)}{u}}{w}] \\ &\le p_2 m + p_1(1+\delta_2)m + \tfrac{m(1+\delta_1)}{u}. \end{aligned} \tag{5.9}$$

Let us set $\delta_1 = \frac{\sqrt{\frac{3}{2}(m+3)\ln u}}{u^{\frac{m-1}{2}}}$ and $\delta_2 = \frac{\sqrt{\frac{3}{2}(m+1)\ln u}}{\sqrt{wp_1}}$. Then we have

$$\begin{aligned} p_1 &\leq mu^{-\frac{m+1}{2}}, \\ p_2 &\leq u^{-\frac{m+1}{2}}. \end{aligned} \tag{5.10}$$

Plugging e.q. (5.10) into e.q. (5.9), we can have an upper-bound for the average rate:

$$\begin{aligned} &R^a \leq p_2 m + p_1(1+\delta_2)m + \tfrac{m(1+\delta_1)}{u} \\ &\leq \tfrac{m}{u} + \tfrac{m}{u^{\frac{m+1}{2}}} + \tfrac{m^2(1+\sqrt{\ln u})}{u^{\frac{m+1}{2}}} + \tfrac{(m+1)^2\sqrt{\ln u}}{\sqrt{\binom{h}{m}u^{\frac{m+1}{2}}}}. \end{aligned} \tag{5.11}$$

Setting $h = h_1 m$, and $m = h_2 u$, where h_1 and h_2 are constants, we have,

$$R^a \to \tfrac{m}{u}, \tag{5.12}$$

as $h \to \infty$. □

From this theorem, we can see that the benefit is almost surely the size of the content type, which can be arbitrarily large.

5.3 Content-Type Coding over Erasure Networks

We here make the case that, over erasure networks with feedback, we can realize benefits by allowing the random erasures to dictate the specific messages within a content type that the receivers get - essentially we shape what we serve to the random channel realizations.

5.3.1 Problem Formulation

We consider the following content-type coding setup. A server, s, has m_1 messages of content-type 1, $\mathcal{M}_1$, and m_2 messages of content-type 2, $\mathcal{M}_2$ (eg. an ad serving broadcasting station in a mall has m_1 sale coupons for clothes and m_2 sale coupons for gadgets). Receiver c_1 wants to receive all the m_1 type-1 messages (sale coupons for clothes) and a fraction α, any αm_2 of the type-2 messages (sale coupons for gadgets); receiver c_2 wants the reverse, all the m_2 type-2 messages (coupons for gadgets) and any αm_1 type-1 messages (coupons for clothes). The server is connected to the two receivers through a broadcast erasure channel with 1-bit ACK-NACK error-free feedback; in particular, when the server broadcasts a message, each receiver c_i receives it correctly with probability $1 - \epsilon_i$, independently across time and across receivers. Each receiver causally acknowledges whether she received the message or not.

The corresponding message-specific scheme is as follows [GT09]. The server wants to send to c_1 all the messages in $\mathcal{M}_1$ and in a specific subset of $\mathcal{M}_2$, $\mathcal{M}_2^1 \subseteq \mathcal{M}_2$, of size αm_2; and to c_2, all the messages in $\mathcal{M}_2$ and in a specific subset of $\mathcal{M}_1$, $\mathcal{M}_1^2 \subseteq \mathcal{M}_1$, of size αm_1. We again have independent broadcast erasure channels with feedback.

Definition 4. *We say that rates* $(r_1,\ r_2)$*, with*

$$r_1 = \frac{m_1 + \alpha m_2}{K}, \quad r_2 = \frac{m_2 + \alpha m_1}{K}, \tag{5.13}$$

are achievable, if with K *transmissions by* s *both* c_1 *and* c_2 *receive all they require.*

5.3.2 Strategy for Message-Specific Coding

The work in [GT09] proposes the following achievability strategy and proves it is capacity achieving. Recall that we use the subscript to indicate the content type, and the superscript for the receiver.

- *Phase 1:* The source repeatedly transmits each of the messages in $\mathcal{M}_1 \backslash \mathcal{M}_1^2$ and $\mathcal{M}_2 \backslash \mathcal{M}_2^1$, until one (any one) of the two receivers acknowledges it has received it.
- *Phase 2:* The source transmits linear combinations of the messages in $\mathcal{M}_2^1$, $\mathcal{M}_1^2$, those in $\mathcal{M}_1 \backslash \mathcal{M}_1^2$ that were not received by c_1, and those in $\mathcal{M}_2 \backslash \mathcal{M}_2^1$ that were not received by c_2.

The intuition behind the algorithm is that, in Phase 1, each private message (that only one receiver requests) is either received by its legitimate receiver, or, it becomes side information for the other receiver. In Phase 2, each receiver either wants to receive each message in a linear combination or already has it as side information and can thus subtract it. The source creates linear combinations that are innovative (bring new information) to each receiver (eg., through uniform at random combining over a high enough field [GT09]). The strategy achieves the rate region:

$$\begin{cases} 0 \le r_1 \le \min\{1-\epsilon_1, \frac{1-\epsilon_2}{1-(1-\phi_2)(1-\alpha)}\} \\ 0 \le r_2 \le \min\{1-\epsilon_2, \frac{1-\epsilon_1}{1-(1-\phi_1)(1-\alpha)}\} \\ \frac{r_1}{1-\epsilon_1}\left(1-\frac{\alpha\phi_1}{1+\alpha}\right) + \frac{r_2}{1-\epsilon_{12}}\frac{1}{1+\alpha} \le 1 \\ \frac{r_1}{1-\epsilon_{12}}\frac{1}{1+\alpha} + \frac{r_2}{1-\epsilon_2}\left(1-\frac{\alpha\phi_2}{1+\alpha}\right) \le 1 \end{cases} \tag{5.14}$$

where $\epsilon_{12} = \epsilon_1\epsilon_2$, and $\phi_i = (1-\epsilon_i)/(1-\epsilon_{12})$ for $i = 1, 2$.

5.3.3 Strategy for Content-Type Coding

We propose the following strategy.

• *Phase 1:* For the messages in $\mathcal{M}_1$, denote by $\mathcal{M}_{1r}$ the messages not yet transmitted. Initially $\mathcal{M}_{1r} = \mathcal{M}_1$.

1. The server repeatedly broadcasts a message in $\mathcal{M}_{1r}$, until (at least) one of the receivers acknowledges it has successfully received it. The message is removed from $\mathcal{M}_{1r}$. If c_1 receives the messages, she puts it into a queue Q_1^1. If c_2 receives the message, she puts it into a queue Q_1^2.

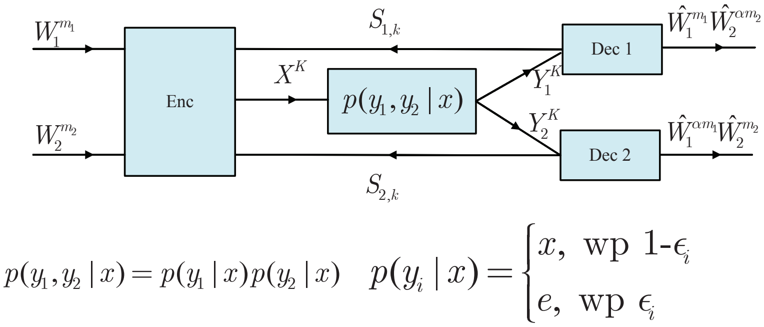
2. The server continues with transmitting a next message in $\mathcal{M}_{1r}$, until either $\mathcal{M}_{1r}$ become empty, or

$$|\mathcal{M}_{1r}| + |Q_1^2| = \alpha m_1. \tag{5.15}$$

The server follows the same procedure for message set $\mathcal{M}_2$.

• *Phase* 2*:* The source transmits linear combinations of the messages in the three sets: $\mathcal{M}_{1r} \cup \mathcal{M}_{2r}$, $Q_1^2 \backslash Q_1^1$, and $Q_2^1 \backslash Q_2^2$ until both receivers are satisfied.

The intuition behind the strategy is that, during Phase 1 we transmit messages from



$W_1^{m_1}$
$W_2^{m_2}$
Enc
X^K
$p(y_1, y_2 \mid x)$
$S_{1,k}$
$S_{2,k}$
Y_1^K
Y_2^K
Dec 1
Dec 2
$\hat{W}_1^{m_1}\hat{W}_2^{\alpha m_2}$
$\hat{W}_1^{\alpha m_1}\hat{W}_2^{m_2}$
$p(y_1, y_2 \mid x) = p(y_1 \mid x)p(y_2 \mid x)$
$p(y_i \mid x) = \begin{cases} x, \text{ wp } 1\text{-}\epsilon_i \\ e, \text{ wp } \epsilon_i \end{cases}$

Figure 5.3: Broadcast erasure channel model for content-type transmission with feedback.

$\mathcal{M}_1$ until we either run out of messages, or both receivers want to receive the remaining $\mathcal{M}_{1r}$: c_1 because she wants all the messages in $\mathcal{M}_1$ and c_2 because, on top of the Q_1^2 she has already received, she also needs the $\mathcal{M}_{1r}$ to complete the fraction αm_1. Note that originally, $|\mathcal{M}_{1r}| = m_1$ and $|Q_1^2| = 0$; at every step, the quantity in (5.15) either remains the same (if c_2 receives the message), or reduces by one (if she does not). Similarly for $\mathcal{M}_2$. In the second phase, the source again transmits linear combinations of messages that either a receiver wants, or already has and can subtract to solve for what she wants.

Using the above method, we can show the achievable rate of content-type broadcasting scheme in the following theorem.

Theorem 18. *The rate region of the* 1-2 *content-type broadcasting communication with erasures is*

$$\begin{cases} 0 \le r_1 \le \min\{1-\epsilon_1, \frac{1-\epsilon_2}{\alpha}\} \\ 0 \le r_2 \le \min\{1-\epsilon_2, \frac{1-\epsilon_1}{\alpha}\} \\ \frac{r_1}{1-\epsilon_1}[1 - \frac{\alpha(\phi_1-\alpha)^+}{1-\alpha^2}] + \frac{r_2}{1-\epsilon_1}\frac{(\phi_1-\alpha)^+}{1-\alpha^2} \le 1 \\ \frac{r_2}{1-\epsilon_2}[1 - \frac{\alpha(\phi_2-\alpha)^+}{1-\alpha^2}] + \frac{r_1}{1-\epsilon_2}\frac{(\phi_2-\alpha)^+}{1-\alpha^2} \le 1 \end{cases} \tag{5.16}$$

where $(x)^+ = \max\{x, 0\}$. *This also achieves the capacity for* 2-1 *content-type broadcasting erasure channel.*

Proof. We first show the content-type coding model in Fig. 5.3.

Achievability

To prove this theorem, we assume that m_1 and m_2 are large. Recall that we define $\epsilon_{12} = \epsilon_1\epsilon_2$, and $\phi_i = (1-\epsilon_i)/(1-\epsilon_{12})$ for $i = 1, 2$. Let us denote by m'_1 and m'_2 the number of messages transmitted from sets $\mathcal{M}_1$ and $\mathcal{M}_2$ at the end of phase 1. Therefore, the average number of transmissions needed to complete phase 1 is:

$$K_1 = \frac{m'{}_1 + m'_2}{1-\epsilon_{12}}. \tag{5.17}$$

On average, the number of messages from set $\mathcal{M}_j$ ($j = 1, 2$) received by receiver i ($i = 1, 2$) is:

$$N_j^i = m'_j \phi_i. \tag{5.18}$$

From the algorithm, we know that $m_1 - m'_1 = (\alpha m_1 - N_1^2)^+$ and $m_2 - m'_2 = (\alpha m_2 - N_2^1)^+$. Therefore, we have

$$m'_i = [1 - \frac{(\alpha - \phi_i)^+}{1-\phi_i}] m_i. \tag{5.19}$$

In phase 2, the required number of messages for receiver i ($i = 1, 2$) is then

$$m_r^i = (m'_i - N_i^i) + (m_1 - m'_1) + (m_2 - m'_2), \tag{5.20}$$

where the first term is the number of erased messages from the set $\mathcal{M}_i$ that are received by another receiver, the second and third terms are the remaining messages to be transmitted.

For phase 2, the average number of transmissions needed is

$$K_2 = \max\{\frac{m_r^1}{1-\epsilon_1}, \frac{m_r^2}{1-\epsilon_2}\}. \tag{5.21}$$

Then, the rate region can be calculated as:

$$\begin{aligned} \big\{(r_1, r_2) : r_1 \geq 0, r_2 \geq 0, r_1 = \tfrac{m_1 + \alpha m_2}{K}, \\ r_2 = \tfrac{m_2 + \alpha m_1}{K}, K_1 + K_2 \leq K\big\}, \end{aligned} \tag{5.22}$$

where K is an auxiliary variable and can be cancelled out. Plugging (5.17) and (5.21) into (5.22), we get the achievability.

Note that for $\max\{\phi_1, \phi_2\} < \alpha < \min\{\phi_1/\phi_2, \phi_2/\phi_1\}$, the conditions are simplified as $r_1 \leq 1 - \epsilon_1$ and $r_2 \leq 1 - \epsilon_2$, implying that the maximum rates can be achieved.

Converse

To prove the converse of the theorem, we use an information theory method to show that this rate region is tight. We first depict the system model in Fig. 5.3.

We aim to find the capacity region (r_1, r_2) for this broadcast channel. Without loss of generality, let us assume $|W_1| = |W_2| = |X| = |Y_1| - 1 = |Y_2| - 1 = 2$.

To prove the converse, we just need to show the first and the third equations in (5.16), and then according to the symmetry, we get the whole set of equations. For the first equation, it is equivalent to point-to-point communication, so we can directly get it. For the third equation, we consider two parts:

$$\frac{m_1}{1-\epsilon_1} + \frac{\alpha m_2}{1-\epsilon_1} \leq 1, \qquad \frac{m_1}{1-\epsilon_{12}} + \frac{m_2}{1-\epsilon_2} \leq 1,$$

First, we consider

$$\begin{aligned} K &\geq \sum_{k=1}^{K} H(X_k) \\ &\geq \sum_{k=1}^{K} H(X_k | Y_1^{k-1}, S^{k-1}) \\ &= \sum_{k=1}^{K} [H(X_k | Y_1^{k-1}, Y_2^{k-1}, S^{k-1}) \\ &\quad + I(X_k; Y_2^{k-1} | Y_1^{k-1}, S^{k-1}) \\ &= \sum_{k=1}^{K} [H(X_k | W_1^{m_1}, W_2^{m_2}, Y_1^{k-1}, Y_2^{k-1}, S^{k-1}) \\ &\quad + I(X_k; Y_2^{k-1} | Y_1^{k-1}, S^{k-1}) \\ &\quad + I(X_k; W_1^{m_1}, W_2^{m_2} | Y_1^{k-1}, Y_2^{k-1}, S^{k-1})] \end{aligned} \tag{5.23}$$

and the following two conditions:

$$\begin{aligned}
&(m_1+m_2)-K\varepsilon_K \le I(W_1^{m_1},W_2^{m_2};Y_1^K,Y_2^K,S^K)\\
&=\sum_{k=1}^{K} I(Y_{1,k},Y_{2,k},S_k;W_1^{m_1},W_2^{m_2}|Y_1^{k-1},Y_2^{k-1},S^{k-1})\\
&=\sum_{k=1}^{K} I(Y_{1,k},Y_{2,k};W_1^{m_1},W_2^{m_2}|Y_1^{k-1},Y_2^{k-1},S^{k-1},S_k)\\
&=\sum_{k=1}^{K} I(X_k;W_1^{m_1},W_2^{m_2}|Y_1^{k-1},Y_2^{k-1},S^{k-1})\Pr(S_k\neq 0)\\
&=(1-\epsilon_{12})\sum_{k=1}^{K} I(X_k;W_1^{m_1},W_2^{m_2}|Y_1^{k-1},Y_2^{k-1},S^{k-1}),
\end{aligned}\tag{5.24}$$

where the above conditions hold due to the Fano's inequality and the incidence property of S_k, and

$$\begin{aligned}
\tfrac{m_1-K\varepsilon_K}{1-\epsilon_1} \;&\le I(W_1^{m_1};X_k|Y_1^{k-1},S^{k-1})\\
\le\;& \sum_{k=1}^{K}[I(X_k;W_1^{m_1}|Y_1^{k-1},Y_2^{k-1},S^{k-1})\\
&\qquad + I(X_k;Y_2^{k-1}|Y_1^{k-1},S^{k-1})]\\
\le\;& \tfrac{m_1}{1-\epsilon_{12}}+\sum_{k=1}^{K}[I(X_k;Y_2^{k-1}|Y_1^{k-1},S^{k-1})],
\end{aligned}\tag{5.25}$$

where the last inequality follows from (using the same idea as (5.24))

$$\begin{aligned}
&m_1 \ge I(W_1^{m_1};Y_1^K,Y_2^k,S^k)\\
&\;=(1-\epsilon_{12})\sum_{k=1}^{K} I(X_k;W_1^{m_1}|Y_1^{k-1},Y_2^{k-1},S^{k-1}).
\end{aligned}\tag{5.26}$$

Plugging (5.24) and (5.25) into (5.23), we get the first part of the third equation in (5.16). Similarly, we can get the second part of the third equation in (5.16) using

$$\begin{aligned}
&(m_1+\alpha m_2)-K\varepsilon_K \le I(W_1^{m_1},W_2^{\alpha m_2};Y_1^K,S^K)\\
&=(1-\epsilon_1)\sum_{k=1}^{K} I(X_k;W_1^{m_1},W_2^{\alpha m_2}|Y_1^{k-1},S^{k-1}).\\
&\le(1-\epsilon_1)\sum_{k=1}^{K} H(X_k)\\
&\le(1-\epsilon_1)K.
\end{aligned}\tag{5.27}$$

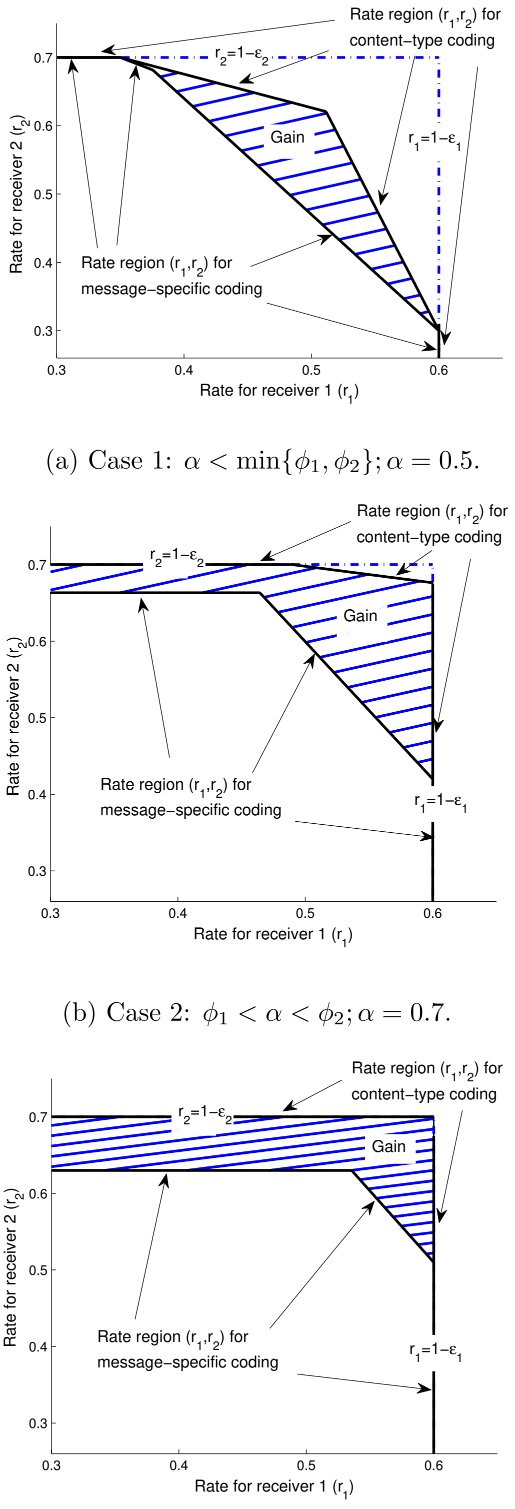
□

By comparison of (5.14) and (5.16), we can see that the capacity region of content-type broadcast erasure channel always covers that of the message-specific broadcast erasure channel. For symmetric case with $\epsilon_1 = \epsilon_2 = \epsilon$ and $r_1 = r_2 = r$, we can simplify the content-type capacity as $C_{content-type} = \min\{1-\epsilon, \frac{(1-\epsilon)(1+\epsilon)(1+\alpha)}{2+\epsilon}\}$ and the message-specific capacity as $C_{message-specific} = \frac{(1-\epsilon)(1+\epsilon)(1+\alpha)}{2+\epsilon+\epsilon\alpha}$. This shows that the gain of content-type coding is $C_{content-type}/C_{message-specific} = \frac{2+\epsilon+\epsilon\alpha}{(1+\epsilon)(1+\alpha)}$ for $\alpha \geq 1/(1+\epsilon)$ and $C_{content-type}/C_{message-specific} = \frac{2+\epsilon+\epsilon\alpha}{\max\{(1+\epsilon)(1+\alpha), 2+\epsilon\}}$. This gain achieves maximum $Gain(\epsilon) = 1 + \frac{\epsilon}{(1+\epsilon)(2+\epsilon)}$ for $\alpha = 1/(1+\epsilon)$; and $Gain(\epsilon)$ achieves maximum 1.195 for $\epsilon = 1/\sqrt{2}$ (i.e., up to 19.5% gain).

Fig. 5.4 compares the rate regions (also capacity regions) for the content-type and message coding. For content-type, we have three distinct cases, depending on the relative values of α and ϕ_i. Note that ϕ_i expresses the fraction of messages that c_i receives during Phase 1. Thus, if $\alpha < \min\{\phi_1, \phi_2\}$ (Fig. 5.4 (a)), c_1 and c_2 already receive αm_1 and αm_2 messages during Phase 1; essentially broadcasting content-type messages comes for "free", has not additional rate cost to providing c_1 with $\mathcal{M}_1$ and c_2 with $\mathcal{M}_2$. If $\min\{\phi_1, \phi_2\} < \alpha < \max\{\phi_1, \phi_2\}$, say for instance $\phi_1 < \alpha < \phi_2$ (Fig. 5.4 (b)), c_2 receives the content-type messages for free, but for c_1 we need additional transmissions in Phase 2. In $\alpha > \max\{\phi_1, \phi_2\}$ (Fig. 5.4 (c)), c_1 and c_2 require large percentages of messages from another type; interestingly, when we have $\max\{\phi_1, \phi_2\} < \alpha < \min\{\phi_1/\phi_2, \phi_2/\phi_1\}$, we can achieve the point $(1-\epsilon_1, 1-\epsilon_2)$, which implies that, all transmissions by s are useful for both receivers. Message-specific coding in general does not achieve this point.

5.4 Summary

We introduce two other content-type coding formulations: over large networks and over lossy networks. We can see that using content-type coding we can achieve arbitrary large benefit for a large "combination-like" network and capacity benefits for a 1-2 broadcast erasure channel with feedback. We believe that there are many more scenarios where we can realize



(a) Case 1: $\alpha < \min\{\phi_1, \phi_2\}; \alpha = 0.5$.

(b) Case 2: $\phi_1 < \alpha < \phi_2; \alpha = 0.7$.

(c) Case 3: $\alpha > \max\{\phi_1, \phi_2\}; \alpha = 0.85$.

Figure 5.4: Comparison of rate region, as defined in (5.13), by message-specific and content-type coding, across three cases. The shaded regions show the gains of content-type over message-specific coding. The channel parameters are $\epsilon_1 = 0.4$ and $\epsilon_2 = 0.3$, which give $\phi_1 = 0.682$ and $\phi_2 = 0.795$.

benefits, such as, downloading content-type rather than message-specific content, can help all aspects of content distribution networks, ranging from storage to coding to content delivery.

REFERENCES

[ABK13] Fatemeh Arbabjolfaei, Bernd Bandemer, Young-Han Kim, Eren Şaşoğlu, and Lele Wang. "On the capacity region for index coding." In *2013 IEEE International Symposium on Information Theory (ISIT)*, pp. 962–966, 2013.

[ACL00] Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung. "Network information flow." *IEEE Transactions on Information Theory*, **46**(4):1204–1216, 2000.

[Ari09] Erdal Arikan. "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels." *IEEE Transactions on Information Theory*, **55**(7):3051–3073, 2009.

[AT16] Mohamed Attia and Ravi Tandon. "Information theoretic limits of data shuffling for distributed learning." *arXiv preprint arXiv:1609.05181*, 2016.

[BBJ11] Ziv Bar-Yossef, Yitzhak Birk, TS Jayram, and Tomer Kol. "Index coding with side information." *IEEE Transactions on Information Theory*, **57**(3):1479–1494, 2011.

[BF12] Siddhartha Brahma and Christina Fragouli. "Pliable index coding." In *2012 IEEE International Symposium on Information Theory (ISIT)*, pp. 2251–2255, 2012.

[BF13] Siddhartha Brahma and Christina Fragouli. "Pliable index coding: the multiple requests case." In *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1142–1146, 2013.

[BF15] Siddhartha Brahma and Christina Fragouli. "Pliable index coding." *IEEE Transactions on Information Theory*, **61**(11):6192–6203, 2015.

[BK98] Y. Birk and T. Kol. "Informed-source coding-on-demand (ISCOD) over broadcast channels." In *IEEE Conference on Computer and Communications Societies (INFOCOM)*, volume 3, pp. 1257–1264, 1998.

[BKL10] Anna Blasiak, Robert Kleinberg, and Eyal Lubetzky. "Index coding via linear programming." *arXiv preprint arXiv:1004.1379*, 2010.

[BL11] Yossi Berliner and Michael Langberg. "Index coding with outerplanar side information." In *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 806–810, 2011.

[Bol01] Béla Bollobás. *Random graphs.* Cambridge Studies in Advanced Mathematics 73, 2001.

[Bol13] Béla Bollobás. *Modern graph theory.* Springer Science & Business Media, 2013.

[Bor81] Jean C de Borda. "Mémoire sur les élections au scrutin." 1781.

[CAS11] Mohammad Asad R Chaudhry, Zakia Asad, Alex Sprintson, and Michael Langberg. "On the complementary index coding problem." In *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 244–248, 2011.

[Con15] "Conviva Viewer Experience Report." *www.conviva.com*, 2013-2015.

[CS08] Mohammad Asad R Chaudhry and Alex Sprintson. "Efficient algorithms for index coding." In *IEEE International Conference on Computer Communications (INFOCOM) Workshops*, pp. 1–4, 2008.

[CZM11] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I Jordan, and Ion Stoica. "Managing data transfers in computer clusters with orchestra." In *ACM SIGCOMM Computer Communication Review*, volume 41, pp. 98–109. ACM, 2011.

[DG77] Persi Diaconis and Ronald L Graham. "Spearman's footrule as a measure of disarray." *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 262–268, 1977.

[DKN01] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. "Rank aggregation methods for the web." In *Proceedings of the 10th International Conference on World Wide Web*, pp. 613–622. ACM, 2001.

[DSC12] Son Hoang Dau, Vitaly Skachek, and Yeow Meng Chee. "On the security of index coding with side information." *IEEE Transactions on Information Theory*, **58**(6):3975–3988, 2012.

[DSC14] Son Hoang Dau, Vitaly Skachek, and Yeow Meng Chee. "Optimal index codes with near-extreme rates." *IEEE Transactions on Information Theory*, **60**(3):1515–1527, 2014.

[EEL15] Michelle Effros, Salim El Rouayheb, and Michael Langberg. "An equivalence between network coding and index coding." *IEEE Transactions on Information Theory*, **61**(5):2478–2487, 2015.

[ER66] P Erdős and Alfréd Rényi. "On the existence of a factor of degree one of a connected random graph." *Acta Mathematica Hungarica*, **17**(3-4):359–368, 1966.

[ESG08] Salim El Rouayheb, Alex Sprintson, and Costas Georghiades. "On the relation between the index coding and the network coding problems." In *2008 IEEE International Symposium on Information Theory (ISIT)*, pp. 1823–1827, 2008.

[ESG10] Salim El Rouayheb, Alex Sprintson, and Costas Georghiades. "On the index coding problem and its relation to network coding and matroid theory." *IEEE Transactions on Information Theory*, **56**(7):3187–3195, 2010.

[GRW16] Alexander Golovnev, Oded Regev, and Omri Weinstein. "The minrank of random graphs." *arXiv preprint arXiv:1607.04842*, 2016.

[GT09] L. Georgiadis and L. Tassiulas. "Broadcast erasure channel with feedback - capacity and algorithms." In *2009 IEEE Workshop on Network Coding, Theory, and Applications (NetCod)*, pp. 54–61, June 2009.

[Has96] Johan Håstad. "Clique is hard to approximate within $n^{1-\epsilon}$." In *Proceedings 37th Annual Symposium on Foundations of Computer Science (FOCS '96)*, pp. 627–636. IEEE, 1996.

[HL12] Ishay Haviv and Michael Langberg. "On linear index coding for random graphs." In *2012 IEEE International Symposium on Information Theory (ISIT)*, pp. 2231–2235, 2012.

[Kar72] Richard M Karp. "Reducibility among combinatorial problems." In *Complexity of Computer Computations*, pp. 85–103. Springer, 1972.

[Kem59] John G Kemeny. "Mathematics without numbers." *Daedalus*, **88**(4):577–591, 1959.

[KRU11] Shrinivas Kudekar, Thomas J Richardson, and Rüdiger L Urbanke. "Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC." *IEEE Transactions on Information Theory*, **57**(2):803–834, 2011.

[LCL14] Yung-Ming Li, Chia-Ling Chou, and Lien-Fa Lin. "A social recommender mechanism for location-based group commerce." *Information Sciences*, **274**:125–142, 2014.

[LLP15] Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. "Speeding up distributed machine learning using codes." *arXiv preprint arXiv:1512.02673*, 2015.

[LMA15] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. "Coded MapReduce." In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 964–971. IEEE, 2015.

[LS09] Eyal Lubetzky and Uri Stav. "Nonlinear index coding outperforming the linear optimum." *IEEE Transactions on Information Theory*, **55**(8):3544–3551, 2009.

[LS11] Michael Langberg and Alex Sprintson. "On the hardness of approximating the network coding capacity." *IEEE Transactions on Information Theory*, **57**(2):1008–1014, 2011.

[MCJ14] Hamed Maleki, Viveck R Cadambe, and Syed A Jafar. "Index coding-an interference alignment perspective." *IEEE Transactions on Information Theory*, **60**(9):5402–5432, 2014.

[MR10] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[NY04] Chi Kin Ngai and Raymond W Yeung. "Network coding gain of combination networks." In *IEEE Information Theory Workshop*, pp. 283–287. IEEE, 2004.

[Pee96] René Peeters. "Orthogonal representations over finite fields and the chromatic number of graphs." *Combinatorica*, **16**(3):417–431, 1996.

[RDR07] Matthew Richardson, Ewa Dominowska, and Robert Ragno. "Predicting clicks: estimating the click-through rate for new ads." In *Proceedings of the 16th International Conference on World Wide Web*, pp. 521–530. ACM, 2007.

[RV97] Paul Resnick and Hal R Varian. "Recommender systems." *Communications of the ACM*, **40**(3):56–58, 1997.

[SF15] Linqi Song and C Fragouli. "Content-type coding." In *2015 International Symposium on Network Coding (NetCod)*, pp. 31–35, 2015.

[SF16a] Linqi Song and Christina Fragouli. "Making recommendations bandwidth aware." *arXiv preprint arXiv:1607.03948*, 2016.

[SF16b] Linqi Song and Christina Fragouli. "A polynomial-time algorithm for pliable index coding." In *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 120–124, 2016.

[SF16c] Linqi Song and Christina Fragouli. "A polynomial-time algorithm for pliable index coding." *arXiv preprint arXiv:1610.06845*, 2016.

[SF17] Linqi Song and Christina Fragouli. "A pliable index coding approach to data shuffling." *arXiv preprint arXiv:1701.05540*, 2017.

[Sha48] Claude E Shannon. "A mathematical theory of communication." *Bell System Technical Journal*, **27**:379–423, 1948.

[SHN11] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell. "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming." In *Proceedings of the 19th ACM International Conference on Multimedia*, pp. 103–112. ACM, 2011.

[STY03] Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. "A note on greedy algorithms for the maximum weighted independent set problem." *Discrete Applied Mathematics*, **126**(2):313–322, 2003.

[yah] "Yahoo! search marketing advertiser bidding data, version 1.0." website, `https://webscope.sandbox.yahoo.com/catalog.php?datatype=a`. accessed: 2016-05-30.