Caching and Distributed Storage: Models, Limits and Designs

THÈSE N^{O} 8501 (2018)

PRÉSENTÉE LE 13 AVRIL 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE D'INFORMATION DANS LES SYSTÈMES EN RÉSEAUX
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Saeid SAHRAEI

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Prof. M. C. Gastpar, directeur de thèse
Prof. M. A. Maddah-Ali, rapporteur
Prof. S. El Rouayheb, rapporteur
Prof. M. Kapralov, rapporteur

EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2018

Abstract

A simple task of storing a database or transferring it to a different point via a communication channel turns far more complex as the size of the database grows large. Limited bandwidth available for transmission plays a central role in this predicament. In two broad contexts, Content Distribution Networks (CDN) and Distributed Storage Systems (DSS), the adverse effect of the growing size of the database on the transmission bandwidth can be mitigated by exploiting additional storage units. Characterizing the optimal tradeoff between the transmission bandwidth and the storage size is the central quest to numerous works in the recent literature, including this thesis.

In a DSS, individual servers fail routinely and must be replicated by downloading data from the remaining servers, a task referred to as the *repair process*. To render this process of repairing failed servers more straightforward and efficient, various forms of redundancy can be introduced in the system. One of the benchmarks by which the reliability of a DSS is measured is *availability*, which refers to the number of disjoint sets of servers that can help to repair any failed server. We study the interaction of this parameter with the amount of traffic generated during the repair process (the repair bandwidth) and the storage size. In particular, we propose a novel DSS architecture which can achieve much smaller repair bandwidth for the same availability, compared to the state of the art.

In the context of CDNs, the network can be highly congested during certain hours of the day and almost idle at other times. This variability of traffic can be reduced by utilizing local storage units that prefetch the data while the network is idle. This approach is referred to as caching. In this thesis we analyze a CDN that has access to independent data from various content providers. We characterize the best caching strategy in terms of the aggregate peak traffic under the constraint that coding across contents from different libraries is prohibited. Furthermore we prove that under certain set of conditions this restriction is without loss of optimality.

Keywords: Coded Caching, Distributed Storage Systems, Memory-Rate Tradeoff, Content Distribution Networks, Exact Repair, Functional Repair, Repair Bandwidth, Lattices, Shortest Vector Problem, Compute-and-Forward.

Résumé

Une tâche simple consistant à stocker une base de données ou à la transférer vers un point différent via un canal de communication devient beaucoup plus complexe à mesure que la taille de la base de données augmente. La bande passante limitée disponible pour la transmission des données joue un rôle central dans cette situation difficile. Dans deux grands contextes, les réseaux de distribution de contenu et les systèmes de stockage distribué, l'effet négatif de la taille croissante de la base de données sur la bande passante de transmission peut être atténué en exploitant des unités de stockage supplémentaires. Caractériser le compromis optimal entre la bande passante de transmission et la taille de stockage est la quête centrale de nombreux travaux dans la littérature récente, y compris cette thèse.

Dans un DSS, les serveurs individuels échouent régulièrement et doivent être répliqués en téléchargeant des données à partir des serveurs restants, une tâche appelée processus de réparation. Pour rendre ce processus de réparation des serveurs défaillants plus simple et plus efficace, diverses formes de redondance peuvent être introduites dans le système. L'un des critères de référence permettant de mesurer la fiabilité d'un DSS est *disponibilité*, qui fait référence au nombre d'ensembles de serveurs disjoints qui peuvent aider à réparer tout serveur défaillant. Nous étudions l'interaction de ce paramètre avec la quantité de trafic généré pendant le processus de réparation et la taille de stockage. En particulier, nous proposons une nouvelle architecture DSS qui permet d'obtenir une bande passante de réparation beaucoup plus petite pour la même disponibilité, par rapport à l'état de la technique.

Dans le contexte des CDN, le réseau peut être très encombré pendant certaines heures de la journée et presque inactif à d'autres moments. Cette variabilité du trafic peut être réduite en utilisant des unités de stockage locales qui prélèvent les données alors que le réseau est inactif. Cette approche est appelée mise en cache. Dans cette thèse, nous analysons un CDN qui a accès à des données indépendantes provenant de différents fournisseurs de contenu. Nous caractérisons la meilleure stratégie de mise en cache en termes de trafic de pointe agrégé sous la contrainte que le codage à travers le contenu de différentes bibliothèques est interdit. En outre, nous prouvons que dans certaines conditions, cette restriction est sans perte d'optimalité.

Mots Clés: Mise en Cache Codée, Systèmes de Stockage Distribués, Mémoire-Taux Troquer, Réseaux de Distribution de Contenu, Réparation Exacte, Réparation Fonctionnelle, Bande Passante de Réparation, Treillis, Problème de Vecteur le plus Court, Calcul-et-Transfert.

Acknowledgements

My sincere gratitude goes to my advisor Professor Michael Gastpar for his continuous support of my PhD study. His patience and impressive vast of knowledge have been the pillars on which this thesis was built. I am forever in debt for his dedication and enthusiasm throughout my time at LINX.

It was a great a honour to have Professor Emre Telatar, Professor Mohammad Ali Maddah-Ali, Professor Salim El Rouayheb and Professor Michael Kapralov on my thesis committee and to receive their insightful comments on my thesis.

My special thanks also goes to France Faille for her invaluable help with the administrative tasks, and my fellow labmates, Chen Feng, Adriano Pastore, Sung Hoon Lim, Chien-Yi Wang, Jingge Zhu, Ibrahim Issa, Giel Op't Veld, Su Li, Erixhen Sula, and Pierre Quinton for the stimulating discussions and for all our great memories at LINX.

Last but not the least, I would like to thank my parents, my brother and my sisters and my finacée, Lixia, for their unconditional love and spiritual support at all times.

Contents

Introduction 1

Of all the fundamental tradeoffs studied in the context of network information theory, the trade-off between storage and bandwidth has received significant attention in the past decade. From a business perspective, this is justified by different costs associated with the two technologies and is particularly enticing when exploiting a small amount of memory results in considerable reductions in the transmission bandwidth. The core of this thesis concerns two broad areas in which this trade-off is prominent, namely Content Distribution Networks (CDN) and Distributed Storage Systems (DSS).

Envision a network which consists of users, entertainment companies which provide multimedia content for these users, and CDNs which operate in between these two entities and are in charge of delivering the data to the users. In order for this ecosystem to successfully operate, many basic considerations must be taken into account. Let us begin with how the content providers must store their data and next move on to the operation of the CDN.

Distributed Storage Systems [1]

As the term suggests, a Distributed Storage System (DSS) is a group of servers that are designed and optimized for collectively storing a large database. By contrast, centralized storage is not suitable for such applications, due to the formidably large size of the database, and perhaps more importantly, due to the fact that individual servers are prone to failure and loss of data. Subsequently, a DSS must be reliable and efficient. Reliability translates to the requirement that a certain number of servers can fail without exposing the system to major risks such as permanent loss of the data. The failed servers must eventually be substituted with new ones which have the same functionality as their predecessors. Efficiency of a DSS is measured by the amount of traffic generated for “repairing” a failed server (henceforth referred to as the repair bandwidth) as well as by the storage size of each server. In order to see how these parameters interact with each other, let us look at two basic, yet surprisingly popular, coding schemes that a DSS can resort to.

Perhaps the simplest form of redundancy is r-replication: Let us say the size of the database is $\mathcal{M}$ bits. A DSS may first divide the database among k servers and then maintain r copies of each server. Clearly, such a configuration is resistant to failure of *any* set of $r-1$ servers. On the other end of the spectrum, a DSS can resort to a simple erasure code where the data is divided among k servers and $r-1$ additional servers are utilized to store parity checks, independent linear combinations of the data stored on the initial k servers. This construction too is resilient to failure of any $r-1$ servers. In terms of the total storage requirement this scheme clearly outperforms the r-replication: $\frac{\mathcal{M}}{k}(k+r-1)$ bits as opposed to $\frac{\mathcal{M}}{k} \cdot k \cdot r$. Nevetheless, from the perspective of the repair bandwidth, the story is entirely different. A replication scheme can repair any failed server by generating a traffic of $\frac{\mathcal{M}}{k}$ bits whereas the repair bandwidth of the simple erasure code discussed above can be as large as $\mathcal{M}$ bits. This simple example should illustrate that our two measures of efficiently cannot be simultaneously optimized, and a compromise must be made between them.

Caching in Content Distribution Networks [2]

The cost of the network bandwidth in a server-user communication system varies over time. Resorting to a first order approximation of this volatility, we shall assume that within certain time intervals, when the network congestion is low, this cost is negligible whereas a fixed cost must be paid during the congested hours. From the perspective of a CDN, it is generally desirable to transmit some part of the data to the users in the first phase, and later complete the transmission in the second. It is needless to say that for this communication scheme to succeed, the users must store the data that they receive in the first phase in their local memories. The process of substituting communication bandwidth with local memories is referred to as caching.

Unlike a DSS, the non-compatibility of minimizing storage and bandwidth in a CDN is a triviality: less needs to be transmitted to a user who already has partial access to the data. This advantage is commonly referred to as “local caching gain”. The central question in the caching literature is whether we can go beyond this linear gain and offer something more substantial. For instance, in the context of a single-server multi-user network, a “global caching gain” can be created by diversifying the content that is stored at the users in the first phase. This diversification creates broadcasting opportunities in the second phase, where a message can be simultaneously helpful for several users. But the broadcast nature of the second transmission is not crucial in achieving non-trivial caching gains. Even in a single-user single-server network, non-trivial gains can be realized, for instance by exploiting similarities among the contents that the user might be interested in.

Clearly, there are far more nuances to the design of a CDN or a DSS than the short description above. But hopefully, these two paragraphs have signified that the trade off between storage and bandwidth plays a central role in both domains. This thesis aims at contributing to a better understanding of these two fields and this tradeoff in a number of ways, that is summarized below. Other contributions which are mostly of independent nature have been listed too.

Main Contributions

- **Increasing Availability in Distributed Storage Systems via Clustering**

  We introduce Fixed Cluster Repair System (FCRS) as a novel architecture for distributed storage systems which focuses on increasing "availability" while maintaining a low repair bandwidth. Availability, refers to the number of distinct subsets of servers than can serve to repair any failed server, and is one of the benchmarks by which the reliability of a DSS is measured. Our proposed architecture consists of partitioning the servers into s clusters of equal size. Once a server fails it can choose any of the clusters other than its own for the purpose of repair. Naturally, this guarantees an availability of $s-1$. We show that for this architecture random linear codes achieve a multiplicative improvement of $2/3$ over the minimum repair bandwidth compared to the existing architectures which guarantee the same availability. Furthermore, we introduce cubic codes for the exact repair problem for FCRS. We show that cubic codes achieve an improvement of 0.79 over the repair bandwidth compared to the existing exact-repair codes that achieve the same availability. We provide an information theoretic proof of optimality of cubic codes for FCRS when the number of clusters is small.

- **Multi-Library Coded Caching**

  Envision a CDN which has access to independent databases (libraries) from several different content providers, and is serving users which may order files from any of these libraries. We refer to this as a multi-library caching network. In this context, two questions are studied. Firstly, what is the optimal caching strategy for the network if coding across different libraries is prohibited? Put simply, how much cache should each user dedicate to each library in order for the network to have the "best" overall experience in terms of the aggregate delivery rate? Avoiding a cross-library coding scheme has clear advantages in terms of the simplicity of the code and resilience to failure or corruption of individual libraries. However, a natural follow-up question is, does this restriction come at a significant price in terms of the optimal storage-bandwidth trade-off?

We provide a complete answer to the first question and a partial answer to the second. In particular, we prove that when the number of files in different libraries are equal, there is no loss at all due to this “memory-sharing” restriction.

- **GDSP, a Generic Model for Distributed Storage**

  The repair process plays a central role in the analysis of the performance of a DSS. Naturally, in order to avoid overcomplicating the theoretical analysis of a DSS, researchers impose many symmetric and idealized assumptions on the architecture of the network, some of which may prove unrealistic in practice. We take a fresh look at distributed storage systems by relaxing many of these constraints, and introducing a model which is highly flexible both in terms of the nature of the data and the architecture of the network. We call this the Generic Distributed Storage Problem or GDSP. In the analysis of GDSP we only concern ourselves with the storage requirements, as we assume the failures occur so rarely that the traffic generated due to them becomes negligible. Whether or not this assumption becomes a reality with advancing hardware technology, it is not hard to appreciate the benefits of studying such a model, in particular in highly non-symmetric configurations. We propose achievability schemes and converse bounds for specific subclasses of GDSP, and prove an optimality result in a practically motivated case.

Other Contributions

- **Finding the Best Equation in Compute-and-Forward**

  In the context of Compute-and-Forward [4], an emerging relaying strategy, a relay node is required to decode an integer linear combination of the codewords emitted by several transmitters, but is left free on how to choose the coefficients of this linear combination. A natural choice for this integer vector is one that maximizes the sum achievable rate at the relay. This maximization problem can be formulated as an instance of the shortest vector problem (SVP), a problem that is known to be NP hard in its general form, under randomized reduction [5]. Many algorithms are proposed to tackle the particular instance of SVP that appears in Compute-and-Forward, but they are either heuristic in nature or run in exponential complexity. We prove that this specific instance of SVP admits an exact solution in low polynomial complexity. Furthermore, we extend this result to the Integer Forcing paradigm [6], a generalization of Compute-and-Forward to multi-antenna relays. For this case too, we prove that the relay can find the (single) integer vector which maximizes the sum achievable in complexity polynomial in number of transmitters.

- **Two Conjectures**

In the last two chapters of this thesis we study two conjectures. The first is a well-known open problem known as the most informative bit (or most informative quantization function) conjecture. We offer an improvement over the state of the art upper-bounds for the mutual information between the inputs and arbitrary quantization functions of the outputs of binary symmetric channels. The second is a conjecture by us about a non-Shannon type information theory inequality with direct application to coded caching.

Notation and Terminology

The set of integers between a and b (inclusive) is represented as either $[a : b]$ or $\{a, a+1, \ldots, b\}$. An interval consisting of real numbers between a and b (inclusive) is denoted by $[a, b]$. For a set of integers π of size n define $a_\pi = [a_{\pi(1)}, \ldots, a_{\pi(n)}]$ where $\pi(i)$ is the i'th smallest element of π.
For two sets A and B we define

$$A \times B = \{(a, b) | a \in A \text{ and } b \in B\}.$$

For a set A and a positive integer $N \geq 2$ we define $A^N = A \times A^{N-1}$.

The set of real numbers and integers are denoted by $\mathbb{R}$ and $\mathbb{Z}$ respectively. Random variables are denoted by capital letters and the alphabets by calligraphic letters. Probability and expectation are denoted by $\mathbb{P}$ and $\mathbb{E}$ respectively. For a random variable X with support $\mathcal{X}$ we define $H(X)$ as the entropy of X. For a probability vector $\mathbf{p}$ we define

$$H(\mathbf{p}) = -\sum_i p_i \log_2 p_i.$$

For a scalar $0 \leq p \leq 1$ we define binary entropy as

$$h_2(p) = -p \log_2(p) - (1-p) \log_2(1-p).$$

The operators ceiling and floor are denoted by $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$. The round operator is denoted by either $\lceil \cdot \rfloor$ or $\lfloor \cdot \rceil$ depending on whether half integers are rounded up or down, respectively.

Preliminaries 2

Distributed Storage

The distributed storage model introduced in [1] consists of n servers that collectively store a file A of size $\mathcal{M}$. The DSS is analyzed in a sequence of snapshots or time-slots starting at $t = 0$. Let $X_{i,t}$ represent the content of the i'th server at time-slot t where $t \in \mathbb{Z}^+ \cup \{0\}$. At time-slot $t = 0$ the variables $X_{i,0}$ are arbitrarily initialized such that $H(X_{i,0}) \leq \alpha$ and $H(A|X_{\tau,0}) = 0, \ \ \forall \tau \subseteq [1 : n]$ s.t. $|\tau| = k$. This means that any subset of k servers at $t = 0$ can collectively recover the file. The servers in the network are subject to failure. We assume that at the end of each time-slot exactly one server fails. Suppose at the end of time-slot t, server X_ℓ fails. At the beginning of the next time-slot this server is replaced by a newcomer. An arbitrary set of servers $\tau \subset [1 : n]\backslash\{i\}$ s.t. $|\tau| = d$ is chosen. We refer to these servers as the repair group. The j'th server in the repair group transmits $Y^{(\tau)}_{\ell,j,t}$, a function of $X_{j,t}$ to the newcomer. We limit the size of this message to satisfy $H(Y^{(\tau)}_{\ell,j,t}) \leq \beta$. We define repair bandwidth as $\gamma = d\beta$ which is the amount of traffic generated for repairing one failed server. Upon receiving these d messages the newcomer stores $X_{\ell,t+1}$ a function of $Y^{(\tau)}_{\ell,j,t}$. We refer to this process as one round of failure and repair. For any other server $i \neq \ell$ we have $X_{i,t+1} = X_{i,t}$. In other words, apart from the failed server, the remaining servers remain intact at time-slot t. Two main repair models have been studied in the literature:

- *Exact repair*: Under the exact repair model the content of the newcomer must be identical to the failed server. Therefore, we must have

  $$X_{\ell,t+1} = X_{\ell,t} \ \forall \ t, \ell.$$

  while studying this model we may omit the subscript t for simplicity and write $X_{\ell,t} = X_\ell$.

- *Functional repair*: Under the functional repair model the newcomer may not be identical to the failed server but it must satisfy the data recovery requirement. That is, for any $\tau \subseteq [1:n]$ s.t. $|\tau| = k$ we must have

$$H(\mathcal{M}|X_{\tau,t}) = 0,\ \forall\, t \in \mathbb{Z} \cup \{0\}.$$

It is clear that the exact recovery criterion is more stringent in nature. However, in practice it is a more attractive option compared to functional repair due to its simpler implementation and maintenance [7].

The functional repair model has been thoroughly studied in [1] and a trade-off between the parameters α, $d\beta$ has been characterized.

Theorem 2.1 (Theorem 1 from [1]). *For a DSS with parameters $\mathcal{M}, n, k, d$ the achievable rate region (α, γ) under functional recovery is characterized by*

$$\alpha = \begin{cases} \frac{\mathcal{M}}{k}, & \gamma \in [f(0), +\infty) \\ \frac{\mathcal{M}-g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)) \end{cases}$$

where

$$\begin{aligned} f(i) &= \frac{2\mathcal{M}d}{(2k-i-1)i + 2k(d-k+1)} \\ g(i) &= \frac{(2d-2k+i+1)i}{2d}. \end{aligned}$$

Two points on this rate region are of particular importance. The Minimum Storage Regenerating (MSR) point where α is minimized and is given by

$$(\alpha_{\text{MSR}}, \gamma_{\text{MSR}}) = (\frac{\mathcal{M}}{k}, \frac{\mathcal{M}d}{k(d-k+1)})$$

and the Minimum Bandwidth Regenerating (MBR) where the repair bandwidth is minimized and corresponds to

$$(\alpha_{\text{MBR}}, \gamma_{\text{MBR}}) = (\frac{2\mathcal{M}d}{2kd-k^2+k}, \frac{2\mathcal{M}d}{2kd-k^2+k}).$$

As for the exact repair model, explicit codes [8, 9, 10, 11] and converse bounds [12, 13] have been studied in depth. It is known [14] that a non-vanishing gap exists between the overall achievable (α, γ) region for the two repair models.

Locality in a DSS refers to the size of the repair group for each failed server [15]. While studying the parameter locality the requirement that the repair group can be chosen arbitrarily is lifted. Therefore, for any failed server, X_ℓ, there are only a limited number of subsets $\tau \subseteq [1:n]\backslash\{\ell\}$ which can serve as its repair group. In this context the parameter repair bandwidth is typicality ignored. Therefore, “serving as repair group” simply means $H(X_\ell|X_\tau) = 0$. Furthermore, this parameter is generally studied in the realm of the exact repair model. For any server $\ell \in [1:n]$ define

$$\mathcal{S}_{\ell,r} \overset{\Delta}{=} \{\tau|\tau \subset [1:n]\backslash\{\ell\},\ |\tau| \leq r \text{ s.t. } H(X_\ell|X_\tau) = 0\}$$

We say that a DSS has locality r if $|\mathcal{S}_{\ell,r}| \geq 1$ for all $\ell \in [1:n]$.

Let $\mathcal{T}_{\ell,r}$ be a maximal subset of $\mathcal{S}_{\ell,r}$ such that $\forall \tau_1, \tau_2 \in \mathcal{T}_{\ell,r}$ we have $\tau_1 \cap \tau_2 = \emptyset$. We say that a DSS with locality r has availability $s-1$ if $|\mathcal{T}_{\ell,r}| \geq s-1,\ \forall \ell \in [1:n]$ [16, 17]. A DSS has availability $s-1$ if there exists an $r \in [1:n-1]$ such that $|\mathcal{T}_{\ell,r}| \geq s-1,\ \forall \ell \in [1:n]$. In words, availability refers to the number of disjoint repair groups for any failed server. While locality is an indicator of how efficient a DSS is in performing the repair process, availability is one of the benchmarks by which the reliability of a DSS is measured. It is possible to bring back the parameter repair bandwidth into the picture and explore its connection with availability. This will be explored in Chapter 3.

Caching

In its canonical form, a caching network consists of a server which is in possession of N independent files $\{W^1, \ldots, W^N\}$ such that $H(W^i) = F,\ \ \forall i \in [1:N]$. There are K users in the network each equipped with a cache of size M. There are two phases of communication between the server and the users. In the placement phase, the i’th user received and stores Z_i, an arbitrary function of all the files, such that, $H(Z_i) \leq MF$. In the delivery phase each user requests one file $d_i \in [1:N],\ \forall i \in [1:K]$. Next, the server broadcasts a delivery message $X_{d_{[1:K]}}$ of size $H(X_{d_{[1:K]}}) \leq RF$ in order to simultaneously satisfy all the users. The i’th user then decodes an estimate of his desired file,

$$\hat{W}^{d_i} = \mu_i(X_{d_{[1:K]}}, Z_i).$$

We say that a memory-rate pair (M, R) is achievable if there exists placement and delivery strategies such that each user can successfully recover his requested message, i.e., if for any $\epsilon > 0$ there exists a sufficiently large F such that

$$\max_{d_{[1:K]} \in [1:N]^K} \max_{k \in [1:K]} \mathbb{P}(\hat{W}^{d_i} \neq W^{d_i}) \leq \epsilon.$$

The quest is to characterize the achievable memory-rate region (M, R), i.e. to find the fundamental trade-off between the two parameters. Algorithms 1 and 2 are the placement and delivery strategies which were first proposed in [2] and later refined in [18]. The placement strategy has the property of being “uncoded”, in that it does not require any computation, apart from breaking each file into subfiles. The achievable memory-rate region of these placement and delivery algorithms is provided in Theorem 2.2. The joint placement and delivery algorithm is proved to be optimal under uncoded placement [18] and within a factor of 2 of an information-theoretic converse bound for arbitrary range of parameters [19].

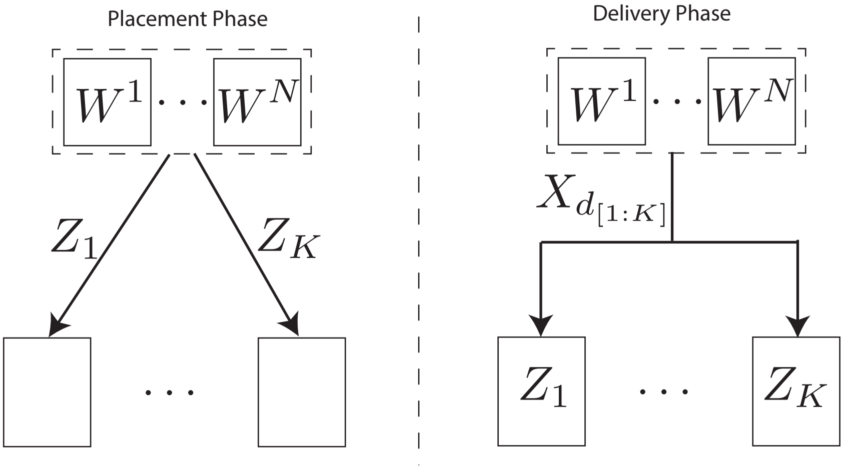


Figure 2.1: The canonical caching model consists of two phases of communication. In the placement phase, each user receives a private message of size M. This message is transmitted without full knowledge of the requests of the users. In the delivery phase, after the users announce their requests to the server, a broadcast message of size R is transmitted to satisfy all the users simultaneously.

Algorithm 1 Placement Strategy [2]

```
    Input: Parameters $N, K, M$
    Output: The cache contents $Z_{[1:K]}$
1: Let $t = \frac{KM}{N}$. Break each file into $\binom{K}{t}$ subfiles and index each by $W_\tau^{(i)}$ where $\tau \subseteq [1:K]$ and $|\tau| = t$.
2: For all $k \in [1:K]$ store $Z_k = \{W_\tau^{(i)} | i \in [1:N], \tau \subseteq [1:K], |\tau| = t, k \in \tau\}$.
```

Algorithm 2 Delivery Strategy [2, 18]

```
    Input: Parameters $N, K, M, d_{[1:K]}$
    Output: The delivery message $X_{d_{[1:K]}}$
1: Let $L \subseteq [1:K]$ be a maximal subset of users who have requested distinct files. Let $t = \frac{KM}{N}$ and $X_{d_{[1:K]}} = \emptyset$.
2: for all $T \subseteq [1:K]$ s.t. $|T| = t+1$ and $T \cap L \neq \emptyset$ do
3:     Let $X_{d_{[1:K]}} = X_{d_{[1:K]}} \cup \sum_{i \in T} W_{T \setminus \{i\}}^{(d_i)}$.
4: end for
```

Theorem 2.2 (Corollary 1 from [18])**.** *For a caching network with K users and N servers, the above caching and delivery strategy results in the following rate region.*

$$R = \frac{\binom{K}{t+1} - \binom{K-\min\{K,N\}}{t+1}}{\binom{K}{t}}$$

where $t = \frac{KM}{N}$ and $t \in [0 : K]$. If the ratio $t = \frac{KM}{N}$ is not an integer, the achievable delivery R corresponds to the lower convex envelop of the points above.

3 Increasing Availability in Distributed Storage Systems via Clustering

As discussed in the preliminaries and Theorem 2.1 the trade-off between the storage size α and the repair bandwidth γ for a DSS has been completely characterized in [1] under functional repair via a network information flow analysis. This analysis hinges on a strong assumption: a failed server must be able to choose *any* set of d servers as its repair group. Furthermore, it has been observed [1] that as the size of the repair group, the parameter d, grows large the required repair bandwidth γ can be made smaller for a fixed storage size α. Setting $d = n - 1$, we achieve the best tradeoff between α and γ. However there are important downfalls to setting d very large. A coding scheme that is designed based on say, $d = n - 1$ is not optimal for repairing multiple parallel failures. Furthermore, the servers involved in the repair process of one failed server may not be available to perform other tasks. More specifically, an architecture with large d is not suitable for applications that involve reading hot data [16, 20] where multiple parallel reads of the same block might become necessary. Mainly in the light of this latter issue, the parameter *availability* is defined in the literature. As mentioned in the preliminaries, a server in a DSS is said to have (all-symbol) availability $s - 1$ if there are $s - 1$ disjoint sets of servers that can serve as its repair group. A DSS has availability $s - 1$ if all the servers in the DSS have availability $s - 1$.

This parameter has been largely investigated in the context of locally repairable codes [15, 21], i.e. codes for which the size of the repair group can be made much smaller than k. The tradeoff between locality (size of the repair group) and availability has been extensively studied [16, 20, 17, 22, 23]. Nevertheless, in the context of locally repairable codes the parameter repair bandwidth is typically ignored and sacrificed. For instance, the achievability results in [16, 20, 17, 15] all have a repair bandwidth of *at least* $\gamma = rM/k$ where r is the locality of the code (the size of the repair group). This is easily outperformed by the codes that achieve the MBR point in [1, 9].

In this work we introduce the Fixed Cluster Repair System (FCRS) model as a novel architecture which aims at achieving a high availability while maintaining a low repair bandwidth. The main idea is to partition the servers into s clusters of equal size, and a final cluster of size $s_0 = \mod(n, s)$. As a server in a cluster fails, we allow it to choose any of the remaining clusters as its repair group (the last cluster is an exception: as it does not contain as many servers as the remaining clusters, we exempt it from serving as a repair group.) This way, we achieve an availability of $s-1$. It is noteworthy that this clustering is not relevant for the data recovery process, meaning that any set of k servers must be able to recover the file, regardless of which cluster they belong to.

The term Fixed Cluster Repair System has been specifically chosen to contrast with Adjustable Cluster Repair System (ACRS), a general model where the repair groups of two different servers do not necessarily coincide with each other. Studying an ACRS should lead us to answering a general question. Suppose we are given a DSS consisting of n servers that follow the data recovery and repair requirements discussed above, while guaranteeing an availability $s-1$. What is the trade-off between storage α and repair bandwidth γ under these constraints? To the best of our knowledge there has not been any literature so far that specifically addresses this question. However, the random linear codes as well as the explicit codes for the seminal work in [1] serve as achievability results for ACRS. In fact, since a server can choose *any* subset of d servers as its repair group in [1] where $d \in [k : n]$, it is possible to achieve an availability of $s-1$ for any $s-1 \in [1 : \lfloor \frac{n-1}{k} \rfloor]$. The random linear codes and the cubic codes designed for FCRS (Sections 3.2 and 3.3) can be viewed as achievability schemes for ACRS too for any availability $s-1 \in [1 : \lfloor \frac{n}{k} \rfloor - 1]$. While we emphasize that a comparison with the work in [1] is not in its entirety fair as the parameter availability has not been a driving motive there, we do find it instructive to demonstrate, through a comparative study, how clustering can help with achieving a low repair bandwidth and a high availability.

Our main objective in this chapter is to thoroughly analyze the FCRS under both functional and exact repair models. We will follow a network information flow analysis to completely characterize the α vs γ trade-off for the FCRS with arbitrary parameters. An interesting observation is made here. We show that the only adverse affect of increasing the number of clusters in the FCRS is the inevitable decrease in the size of the repair groups. In other words, two FCRSs with $n = ds$ and $n' = ds'$ with respectively s and s' clusters have exactly the same performance in terms of the achievable (α, γ) region. By characterizing the entire (α, γ) region, we show that for small values of γ FCRS performs better than [1]. Whereas, on the other end of the spectrum, when α is small, [1] is superior. The improvements offered by the FCRS are most visible at the MBR point itself (the point where the repair bandwidth is minimized), at which we prove an asymptotic multiplicative improvement of $\frac{2}{3}$ over the repair bandwidth compared to [1], as s, k and n grow large.

Our second contribution is to propose cubic codes for the FCRS which are designed to minimize the repair bandwidth under exact repair. Cubic codes are examples of Fractional Repetition codes [24], codes that do not require any computation to perform the repair process. More specifically, they are subclasses of Affine Resolvable Designs and generalizations of Grid Codes both discussed in [25]. When the number of clusters is small (two or three complete clusters with no residual servers), we prove that cubic codes do minimize the repair bandwidth for the FCRS. While we do not generalize this proof of optimality of cubic codes to an arbitrary number of clusters, we prove that they achieve an asymptotic (again, as s, k and n grow large) multiplicative improvement of 0.79 over the repair bandwidth compared to the MBR codes for [1].

Before we move on, it is worth noting that clustering is not a new term or technique in the analysis of Distributed Storage Systems. "Clustered Storage Systems" have been studied in a series of works [26, 27] where different repair bandwidths are associated to inter-cluster and intra-cluster repair. These models have close connections with the "rack model" [28, 29, 30] and are generally motivated by the physical architecture of the network and the fact that the cables/channels which connect the servers within one cluster or rack have higher capacities than the inter-cluster counterparts, which creates the motivation to mostly confine the repair process to within the same cluster as the failed server. They also have slightly different data recovery requirements in [26] than [1] and the model studied here. These physical considerations do not play any role in our analysis. We simply assume a completely symmetric structure where the communication bandwidth between any pair of servers is identical.

The rest of this chapter is organized as follows. In Section 3.1 we provide a precise description of FCRS. In Section 3.2 we analyze the FCRS with arbitrary parameters under the functional repair model and make a numerical as well as analytical comparison with the results in [1]. In section 3.3 we introduce cubic codes as explicit constructions targeted to minimize the repair bandwidth for the FCRS under exact repair. Comparisons with MBR codes for [1] will follow. Finally in Section 3.4, we provide a converse bound for the exact repair problem with $s \leq 3$ complete clusters (no residual servers) which indicates that cubic codes indeed minimize the repair bandwidth when the number of clusters is small.

3.1 Model Description

The FCRS is defined by three parameters n, k and s. Suppose the network consists of $n = ds + s_0$ servers where $d = \lfloor \frac{n}{s} \rfloor$ and $s_0 = \mod (n, s)$ and $2 \leq s \leq \lfloor \frac{n}{k} \rfloor$. We partition these servers into $s + 1$ clusters, s of which are of size d and the last of size s_0. We have a file $\mathcal{M}$ of size $H(\mathcal{M}) = M$. Each server $i \in [1:n]$ is equipped with a memory. We model each memory with a random variable where the server can store a function of $\mathcal{M}$. Specifically the random variable $X_{j,t}^{(i)}$ represents the content of the j'th server in the i'th cluster at time slot t where $(i,j) \in \{[1:s] \times [1:d]\} \cup \{\{s+1\} \times [1:s_0]\}$ and $t \in \mathbb{Z}^+ \cup \{0\}$. We restrict the size of each memory to be bounded by α, that is, $H(X_{j,t}^{(i)}) \leq \alpha \;\forall\, i,j,t$. For a set $E \subseteq [1:d]$ we define $X_{E,t}^{(i)} = \{X_{e,t}^{(i)} \text{ s.t. } e \in E\}$. The initial contents of the servers at $t=0$ must be chosen in such a way that any set of k servers can collectively decode the file $\mathcal{M}$, irrespective of their clusters. In other words, for any $(E_1, \ldots, E_s, E_{s+1})$ that satisfy $E_i \subseteq [1:d]$ for $i \in [1:s]$ and $E_{s+1} \in [1:s_0]$, and $\sum_{i=1}^{s+1} |E_i| = k$, we must have

$$H(\mathcal{M}|X_{E_1,0}^{(1)}, \ldots, X_{E_{s+1},0}^{(s+1)}) = 0.$$

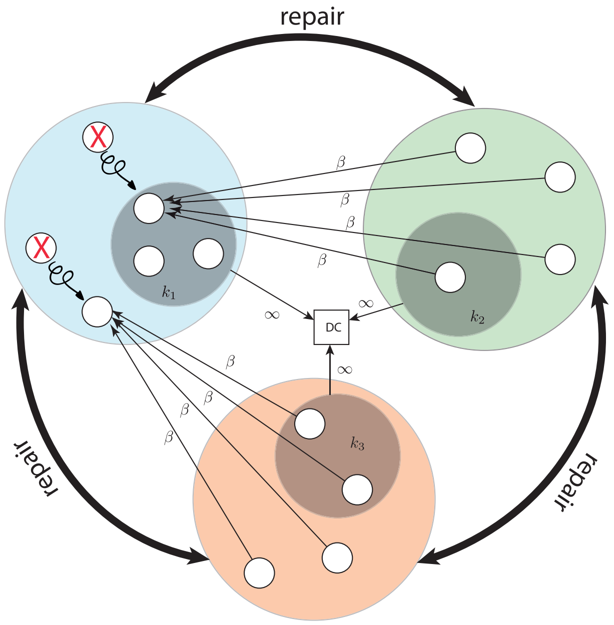


Figure 3.1: The FCRS with three complete clusters. Two nodes in the blue cluster fail which are repaired by the red and green clusters respectively. A data collector (DC) is connected to $k = k_1 + k_2 + k_3$ servers, one of which is newcomers.

The servers in the network are subject to failure. We assume that at the end of each time slot exactly one server fails. Suppose at the end of time slot t, the ℓ'th server in the r'th cluster fails. At the beginning of the next time slot this server is replaced by a newcomer. A second cluster i will be chosen *arbitrarily* such that $i \neq r$. We refer to this as the repair group. The j'th server in the repair group transmits $Y_{\ell,j,t}^{(r,i)}$, a function of $X_{j,t}^{(i)}$ to the newcomer. We limit the size of this message to satisfy $H(Y_{\ell,j,t}^{(r,i)}) \leq \beta$. Upon receiving these d messages the newcomer stores $X_{\ell,t+1}^{(r)}$ a function of $Y_{\ell,[1:d],t}^{(r,i)}$. Therefore, $H(X_{\ell,t+1}^{(r)}|Y_{\ell,[1:d],t}^{(r,i)}) = 0$. We refer to this process as one round of failure and repair. Due to this requirement, we can assume without loss of generality that $\alpha \leq d\beta$. Note that if $(i,j) \neq (\ell, r)$ then $X_{j,t+1}^{(i)} = X_{j,t}^{(i)}$. In other words, apart from the failed server, the remaining servers remain unchanged at time slot t. We will study both the functional and exact repair models.

The model described above is what we refer to as Fixed Cluster Repair System (FCRS). See Figure 3.1 for an illustration of an FCRS with three complete clusters and no residual servers ($s_0 = 0$). By contrast, a general DSS (what we referred to as ACRS) lacks many of these constraints. A DSS with parameters (n,k) consists of n servers $\{X_1, \ldots, X_n\}$ such that any k servers can recover the file $\mathcal{M}$. A DSS is said to have availability $s-1$ if for each server X_i there are $s-1$ disjoint sets of servers of respective sizes $d_1^{(i)}, \ldots, d_{s-1}^{(i)}$ that can serve as its repair group while generating repair bandwidths $d_1^{(i)}\beta_1^{(i)}, \ldots, d_{s-1}^{(i)}\beta_{s-1}^{(i)}$, respectively. The repair process can be defined either as functional or exact repair. The repair bandwidth is defined as

$$\gamma = \max_{i\in[1:n], j\in[1:s-1]} d_j^{(i)}\beta_j^{(i)}. \tag{3.1}$$

3.2 The Functional Repair Model

In this section, we present a network information flow analysis for the FCRS. Each server $X_{j,t}^{(i)}$ is modelled by a pair of nodes $X_{j,t,in}^{(i)}$ and $X_{j,t,out}^{(i)}$ that are connected with an edge. The sources is directly connected to the nodes $X_{j,0,in}^{(i)}$ with edges of infinite capacity. The nodes $X_{j,0,in}^{(i)}$ are in turn connected to $X_{j,0,out}^{(i)}$ with edges of capacity α. Suppose at end of time slot $t-1$ (where $t \geq 1$) the j'th server from the i'th cluster fails. Assume the newcomer replacing this server is repaired by connecting to the r'th cluster. We represent this by d edges which connect $X_{[1:d],t-1,out}^{(r)}$ to $X_{j,t,in}^{(i)}$. Each of these edges has a capacity of β. Furthermore, there will be an edge of capacity α from $X_{j,t,in}^{(i)}$ to $X_{j,t,out}^{(i)}$. Since we have only one failure per time slot, for all other $(i',j') \neq (i,j)$ there will be edges of infinite capacity from $X_{j',t-1,out}^{(i')}$ to $X_{j',t,in}^{(i')}$ and from $X_{j',t,in}^{(i')}$ to $X_{j',t,out}^{(i')}$. At any given time a data collector can be connected to the out nodes of any set of servers of size k with edges of infinite capacity. An illustration has been provided in Figure 3.2 which involves three clusters.

Our goal in this section is to find the minimum cut that separates any data collector from the source in this graph under all possible failure and repair patterns. As we shall see this minimum cut helps us to characterize the smallest possible value of α for any choice of $\gamma = d\beta$, such that any data collector can recover the file. Furthermore, the tradeoff (α, γ) characterized by this min-cut is achievable, for instance if we resort to random linear codes [31].

Consider a sequence of failures and repairs as depicted in Figure 3.3. Note that only two clusters participate in this sequence. First, $k_1 \geq \lceil \frac{k}{2} \rceil$ servers from the first cluster fail. All of these servers are repaired by connecting the

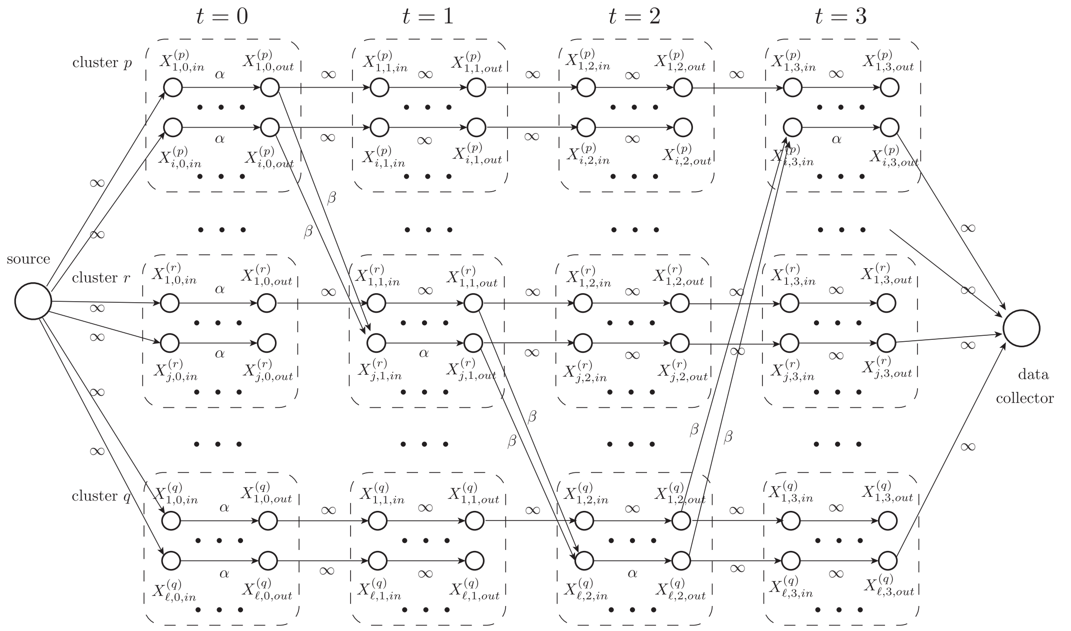


Figure 3.2: The network information flow model for the FCRS. At the end of $t = 0, 1, 2$ the servers $X^{(r)}_{j,0}, X^{(q)}_{\ell,1}$ and $X^{(p)}_{i,2}$ fail respectively. These servers are repaired by connecting to clusters p, r and q, in that order. Finally a data collector connects to k servers at time slot $t = 3$ for recovering the file $\mathcal{M}$. Note that some of these k servers are newcomers.

second cluster. Next, $k_2 = k - k_1$ servers from the second cluster fail. These servers are repaired by the first cluster. Assume a data collector connects to these k newcomers in order to recover the file $\mathcal{M}$. As we shall see soon, a simple cut-set argument shows that we must have

$$M \quad \leq \quad k_1\alpha + (d-k_1)(k-k_1)\beta.$$

Our first objective is to prove that for any choice of the parameters α and $d\beta$, there exists a $k_1 \in [\lceil \frac{k}{2} \rceil : k]$ such that this is the smallest cut which separates any data collector from the source. Let us assume that at some arbitrary point in time, t_0, a data collector is connected to k servers which we call $Z_{1,t_0}, \ldots, Z_{k,t_0}$. For any $i \in [1 : k]$ let $t_i \leq t_0$ be the smallest integer such that an edge of infinite capacity exists from $Z_{i,t',out}$ to $Z_{i,t'+1,in}$ for all $t' \in [t_i : t_0]$. If no such t_i exists, set $t_i = t_0$. We say that there is a path from Z_{i,t_i} to Z_{j,t_j} if there exists a $t' \in [t_i : t_j - 1]$ such that there is an edge of capacity β connecting $Z_{i,t',out}$ to $Z_{j,t'+1,in}$. We can order these k servers such that $i < j$ implies $t_i \leq t_j$. As a result, $i < j$ implies there is no path from Z_{j,t_j} to Z_{i,t_i}. Let us assume that such an ordering is in place. Define $e_i \in [1 : s+1]$ as the cluster to which Z_{i,t_0} belongs and let

$$c(i,j) = |\{\ell \ \ s.t. \ \ \ell \leq j \text{ and } e_\ell = i\}| \text{ for } j \in [1:k] \tag{3.2}$$

be the number of servers in $Z_{[1:j],t_0}$ which belong to the i'th cluster. Let $F(Z_{[1:k],t_0})$ be the value of the minimum cut that separates a data collector connecting to $Z_{[1:k],t_0,out}$ from the source. In order to find this cut, we must decide for any $j \in [1:k]$ whether to include both $Z_{j,t_j,in}$ and $Z_{j,t_j,out}$ on the sink (data collector) side, or to include $Z_{j,t_j,in}$ on the source side and $Z_{j,t_j,out}$ on the sink side (if we include both $Z_{j,t_j,in}$ and $Z_{j,t_j,out}$ on the source side, the value of the cut will be infinite). In the latter case the value of the cut is increased by α, whereas in the former scenario, the value of the cut is increased by *at least* $(d - \max_{i\in[1:s+1]\setminus\{e_j\}} c(i,j))\beta$. This is because any newcomer must be repaired by a cluster differently from his own. As a result, the value of this cut must satisfy

$$F(Z_{[1:k],t_0}) \geq \sum_{j=1}^{k} \min\{(d - \max_{i\in[1:s+1]\setminus\{e_j\}} c(i,j))\beta, \alpha\}.$$

Let us represent this lower bound by

$$F^*(e_{[1:k]}) \triangleq \sum_{j=1}^{k} \min\{(d - \max_{i\in[1:s+1]\setminus\{e_j\}} c(i,j))\beta, \alpha\}. \tag{3.3}$$

Note that for fixed parameters α, d and β, the expression in (3.3) is uniquely determined by the sequence $e_{[1:k]}$, hence the change of the argument in $F^*(\cdot)$ from $Z_{[1:k],t_0}$ to merely $e_{[1:k]}$. The first lemma tells us that among all different sequences $e_{[1:k]}$ the value of $F^*(e_{[1:k]})$ is minimized when this sequence has a very specific structure.

Lemma 3.1. *For any sequence* $e_{[1:k]} \in [1:s+1]^k$*, there exists a sequence* $e'_{[1:k]} \in [1:2]^k$ *such that* $F^*(e'_{[1:k]}) \leq F^*(e_{[1:k]})$.

Proof. Let $c(i,j)$ be as defined in (3.2) and

$$e'_j \triangleq \begin{cases} 1 & \text{if } c(e_j, j-1) = \max_{i\in[1:s+1]} c(i,j-1) \\ 2 & \text{Otherwise.} \end{cases}$$

for $j \in [2:k]$ and $e'_1 \triangleq 1$. Define two variables as follows.

$$v_2(j) \triangleq \begin{cases} 0 & \text{if } j = 0 \\ v_2(j-1) + e'_j - 1 & \text{if } 0 < j \leq k \end{cases} \tag{3.4}$$

and

$$v_1(j) \triangleq j - v_2(j). \tag{3.5}$$

For any finite discrete set D, let $\max^{(2)}_{i\in D} f(i)$ return the second largest value of $f(\cdot)$ over D (if the maximizer of $f(\cdot)$ over D is not unique, then $\max^{(2)}_{i\in D} f(i) = \max_{i\in D} f(i)$). We proceed by proving the following two claims: $v_1(j) =$

$\max_{i\in[1:s+1]} c(i,j)$ and $v_2(j) \geq \max^{(2)}_{i\in[1:s+1]} c(i,j)$. The first claim can be proven by induction. Trivially, $v_1(1) = 1 = \max_i c(i,1)$. Assume the hypothesis is true for $j-1$. Then

$$\begin{aligned} v_1(j) &= \mathbb{1}\{e'_j = 1\}(v_1(j-1)+1) + \mathbb{1}\{e'_j = 2\}v_1(j-1) \\ &= \mathbb{1}\{c(e_j, j-1) = \max_i c(i,j-1)\}(\max_i c(i,j-1)+1) \\ &+ \mathbb{1}\{c(e_j, j-1) < \max_i c(i,j-1)\}\max_i c(i,j-1) \\ &= \max_i c(i,j). \end{aligned}$$

The second claim follows because $v_2(j) = j - v_1(j) = j - \max_i c(i,j) = \sum_{i\neq i^*} c(i,j) \geq \max^{(2)} c(i,j)$ where $i^* = \arg\max_i c(i,j)$.

As a result, we have $\max_{i\in[1:s+1]\setminus\{e_j\}} c(i,j) \leq (e'_j - 1)v_1(j) + (2-e'_j)v_2(j)$ for any $j \in [1:k]$. Therefore,

$$F^*(e_{[1:k]}) \geq \sum_{j=1}^{k} \min\{(d - (e'_j - 1)v_1(j) - (2-e'_j)v_2(j))\beta, \alpha\}.$$

Now consider a sequence of failures and repairs occurring at $t = j \in [1:k]$ such that if $e'_j = 1$ a server from the first cluster fails and is repaired by the second cluster, and if $e'_j = 2$ then a server from the second cluster fails and is repair by the first cluster. At each time-slot the parameters $v_1(j)$ and $v_2(j)$ represent the number of failed servers from the first and the second cluster respectively. Therefore we can write

$$F^*(e'_{[1:k]}) = \sum_{j=1}^{k} \min\{(d - (e'_j - 1)v_1(j) - (2-e'_j)v_2(j))\beta, \alpha\}, \tag{3.6}$$

thus, $F^*(e'_{[1:k]}) \leq F^*(e_{[1:k]})$.

□

Suppose now that we are given an arbitrary sequence $e'_{[1:k]}$ such that $e'_i \in \{1,2\}$. The next question is then how to sort the elements of $e'_{[1:k]}$ such that the expression in Equation (3.6) is minimized. It turns out there is a simple and global answer to this equation. Assume without loss of generality that $v_1(k) \geq v_2(k)$. The next lemma tells us that the the failures from each cluster must occur consecutively, without being interrupted. Specifically, $v_1(k)$ servers must fail from the first cluster, and only then, $v_2(k)$ servers fail from the second. Such a pattern always minimizes (3.6) regardless of the value of α and $d\beta$.

Lemma 3.2. *Let $e'_{[1:k]} \in [1:2]^k$ be an arbitrary binary sequence of length k and let $v_1(j)$ and $v_2(j)$ be as defined in equations (3.5) and (3.4). Assume without loss of generality that $v_1(k) \geq v_2(k)$. We have*

$$F^*(e'_{[1:k]}) \geq v_1(k)\alpha + v_2(k)\min\{(d - v_1(k))\beta, \alpha\}. \tag{3.7}$$

This is achieved with equality if $e'_{[1:k]}$ is sorted, i.e. if $e'_{[1:v_1(k)]}$ are all ones.

Proof. Let us for simplicity define $f(\alpha) = \sum_{j=1}^{k} \min\{(d-(e'_j-1)v_1(j)-(2-e'_j)v_2(j))\beta, \alpha\}$ and $g(\alpha) = v_1(k)\alpha + v_2(k)\min\{(d-v_1(k))\beta, \alpha\}$. The proof follows from three simple observations.

- As long as $\alpha \le (d-v_1(k))\beta$ we have $f(\alpha) = g(\alpha) = k\alpha$.
- The curve $f(\alpha)$ is concave within $(d-v_1(k))\beta \le \alpha \le d\beta$ whereas the curve $g(\alpha)$ is linear within the same interval.
- $f(d\beta) = g(d\beta)$.

To see why the last claim holds, note that

$$\begin{aligned} f(d\beta) &= \sum_{j=1}^{k}(d-(e'_j-1)v_1(j)-(2-e'_j)v_2(j))\beta \\ &= kd\beta - \beta(\sum_{j=1}^{k}(e'_j-1)v_1(j)+(2-e'_j)v_2(j)). \end{aligned}$$

Since $g(d\beta) = kd\beta - v_1(k)v_2(k)\beta$, it is left to show that $\sum_{j=1}^{k}(e'_j-1)v_1(j) + (2-e'_j)v_2(j) = v_1(k)v_2(k)$. This can be proved by induction over k. For $k=1$, the result trivially holds. Let us assume it is true for $k-1$. Then:

$$\begin{aligned} &\sum_{j=1}^{k}(e'_j-1)v_1(j)+(2-e'_j)v_2(j) \\ &\quad= v_1(k-1)v_2(k-1)+(e'_k-1)v_1(k)+(2-e'_k)v_2(k) \\ &\quad= (v_1(k)+e'_k-2)(v_2(k)-e'_k+1)+(e'_k-1)v_1(k)+(2-e'_k)v_2(k) \\ &\quad= v_1(k)v_2(k). \end{aligned}$$

□

Remember that $F^*(e'_{[1:k]})$ is merely a lower bound on value of the min-cut separating any data collector from the source. But it is easy to find a cut the value of which is given by Equation (3.7). The sequence of failures and repairs leading to this cut is what is depicted in Figure 3.3: at the end of each time slot $t \in [0 : k_1 - 1]$ the server $X^{(1)}_{t+1,t}$ fails and is repaired by the second cluster. Next, at the end of each time slot $t \in [k_1 : k-1]$ the server $X^{(2)}_{t-k_1+1,t}$ fails and is repaired by the first cluster. Then a data collector connects to the k servers $X^{(1)}_{[1:k_1],k}$ and $X^{(2)}_{[1:k-k_1],k}$. For every $t \in [1 : k_1]$ we include $X^{(1)}_{t,t,in}$ on the source side of the cut and $X^{(1)}_{t,t,out}$ on the sink side. If $(d-k_1)\beta > \alpha$, we do exactly the same thing for the servers in $\{X^{(2)}_{t-k_1,t} | t \in [k_1+1 : k]\}$. Otherwise, if $(d-k_1)\beta \le \alpha$ then for all $t \in [k_1+1 : k]$ we include both $X^{(2)}_{t-k_1,t,in}$ and $X^{(2)}_{t-k_1,t,out}$ on the sink side. Since $v_1(k) = k_1$ and $v_2(k) = k - k_1$, the value of this cut is precisely what is given by Equation (3.7).

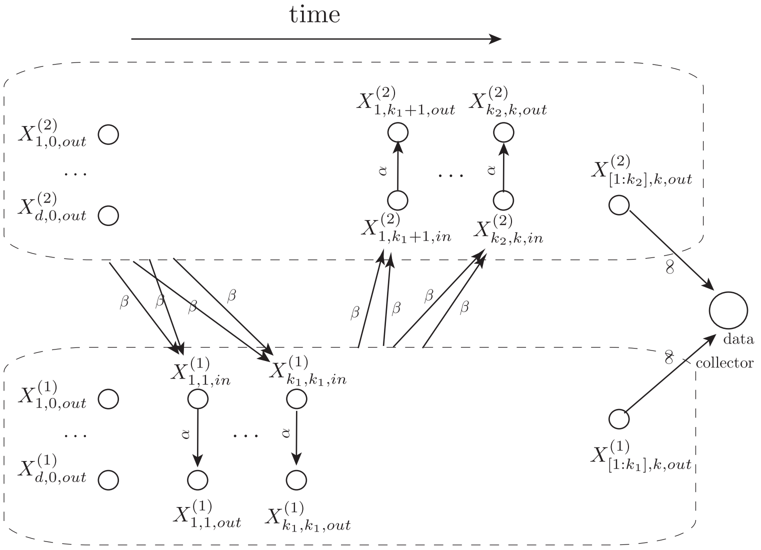


Figure 3.3: Network information flow graph corresponding to a sequence of failures and repairs where $k_1 \geq \lceil \frac{k}{2} \rceil$ servers from the first cluster fail, followed by $k_2 = k - k_1$ failures from the second cluster. Each failure in cluster 1 is repaired by cluster 2 and vice versa. A data collector is connected to the k newcomers.

We have therefore proved the claim which we made at the beginning of this section. The last question to answer is what is the optimal choice of k_1 for a specific value of α and $d\beta$. With a slight abuse of notation, let us denote by $F^*(\alpha, d\beta)$ the value of the min-cut for the FCRS with a storage of size α and a repair bandwidth of $\gamma = d\beta$.

Lemma 3.3. *Suppose we have an FCRS with parameters n, k, s and $d = \lfloor \frac{n}{s} \rfloor$. The value of the min-cut separating any data collector from the source is given by*

$$F^*(\alpha, d\beta) = \begin{cases} kd\beta - \lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil \beta & \text{if } d \leq \frac{\alpha}{\beta}, \\ k_1\alpha + (d-k_1)(k-k_1)\beta & \text{if } d+k-2k_1-1 \leq \frac{\alpha}{\beta} < \\ & \min\{d+k-2k_1+1, d\} \text{ for } k_1 \in [\lceil \frac{k}{2} \rceil : k], \\ k\alpha & \text{if } \frac{\alpha}{\beta} < d-k-1. \end{cases} \tag{3.8}$$

Proof. Let us define $g(k_1) = k_1\alpha + (k-k_1)\min\{(d-k_1)\beta, \alpha\}$. We want to minimize $g(k_1)$ over $k_1 \in [\lceil \frac{k}{2} \rceil : k]$ for a specific choice of α and $d\beta$. Without loss of generality we can assume $(d-k-1)\beta \leq \alpha \leq d\beta$. If $\alpha > d\beta$ then $g(k_1)$ is clearly minimized for $k_1 = \lceil \frac{k}{2} \rceil$ which matches with the first line of Equation (3.8). On the other hand, if $\alpha < (d-k-1)\beta$ then $g(k_1)$ is minimized at $k_1 = k$ which yields the last line in (3.8). Let us now minimize a simpler function $h(k_1) = k_1\alpha + (d-k_1)(k-k_1)\beta$. This is a second degree polynomial in k_1 and evidently its minimizer over $k_1 \in [\lceil \frac{k}{2} \rceil : k]$ is $k_1^* = \min\{\lfloor \frac{1}{2}(d+k-\alpha/\beta) \rceil, k\}$ where $\lfloor \cdot \rceil$ returns the closest integer to its argument. Note that $g(k_1) = \min\{k_1\alpha + (k-k_1)(d-k_1)\beta, k_1\alpha + (k-k_1)\alpha\} = \min\{h(k_1), k\alpha\}$, so the same k_1^* minimizes $g(\cdot)$ too. Finally, $k_1^* = \min\{\lfloor \frac{1}{2}(d+k-\alpha/\beta)\rceil, k\}$ implies $d+k-2k_1^*-1 \le \frac{\alpha}{\beta} \le d+k-2k_1^*+1$ which is the same as the second line in Equation (3.8). □

For any FCRS with parameters n, k, s and file size M we must have $M \le F^*(\alpha, d\beta)$ given by Equation (3.8), otherwise there exists a sequence of failures and repairs after which a data collector (connecting to the newcomers) is incapable of recovering the file. Furthermore, satisfying $M \le F^*(\alpha, d\beta)$ is sufficient for successfully repairing any sequence of failures, and for any data collector to recover the file, if we resort to random linear codes [31]. The function $F^*(\alpha, d\beta)$ can be inverted in order to find the minimum value of α for a specific choice of $d\beta$ and M, in much the same way as illustrated in [1]. We will sketch this - mostly replicated - proof for the sake of completeness. Let us summarize the result in the next theorem.

Theorem 3.1. *The tradeoff between α and $\gamma = d\beta$ in an FCRS with parameters n, k, s and $d = \lfloor \frac{n}{s} \rfloor$ can be characterized as*

$$\alpha^* = \begin{cases} \frac{M}{k} & \text{if } \gamma \in [f(0), \infty) \\ \frac{M - \frac{i}{d}(d-k+i)\gamma}{k-i} & \text{if } \gamma \in [f(i), f(i-1)) \end{cases} \tag{3.9}$$

where

$$f(i) \triangleq \begin{cases} \frac{Md}{(2k-i-1)i+k(d-k+1)} & \text{if } 0 \le i < \lfloor \frac{k}{2} \rfloor \\ \frac{Md}{kd - \lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil} & \text{if } i = \lfloor \frac{k}{2} \rfloor. \end{cases}$$

Proof. The function $M = F^*(\alpha, d\beta)$ can be inverted in terms of α.

$$\alpha^* = F^{-1}(M, d\beta) = \begin{cases} \frac{M}{k} & \text{if } 0 \le M < k(d-k-1)\beta \\ \frac{M-(d-k_1)(k-k_1)\beta}{k_1} & \text{if } (dk - k_1^2 - k_1)\beta \le M < (dk - k_1^2 + k_1)\beta \\ & \text{for } k_1 \in [\lceil \frac{k}{2} \rceil + 1 : k] \\ \frac{M-(d-\lceil \frac{k}{2} \rceil)\lfloor \frac{k}{2} \rfloor \beta}{\lceil \frac{k}{2} \rceil} & \text{if } (dk - \lceil \frac{k}{2} \rceil^2 - \lceil \frac{k}{2} \rceil)\beta \le M \le (dk - \lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil)\beta \end{cases}$$

If we write the conditions in terms of $d\beta$ we find

$$\alpha^* = \begin{cases} \frac{M}{k} & \text{if } d\beta \ge \frac{dM}{k(d-k-1)} \\ \frac{M-(d-k_1)(k-k_1)\beta}{k_1} & \text{if } d\beta \in [\frac{dM}{dk-k_1^2+k_1}, \frac{dM}{dk-k_1^2-k_1}] \text{ for } k_1 \in [\lceil \frac{k}{2} \rceil + 1 : k) \\ \frac{M-(d-\lceil \frac{k}{2} \rceil)\lfloor \frac{k}{2} \rfloor \beta}{\lceil \frac{k}{2} \rceil} & \text{if } d\beta \in [\frac{dM}{dk-\lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil}, \frac{dM}{dk-\lceil \frac{k}{2} \rceil^2 - \lceil \frac{k}{2} \rceil}) \end{cases}$$

which is essentially the same as Equation (3.9). We intentionally substitute $i = k - k_1$ to find an expression similar to Equation (1) in [1]. □

Let us denote by $(\alpha_{MBR,c}, \gamma_{MBR,c})$ the operating point at which the repair bandwidth of the FCRS is minimized. At this point we have

$$\gamma_{MBR,c} = f(\lfloor \frac{k}{2} \rfloor) = \frac{Md}{kd - \lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil}. \tag{3.10}$$

By plugging in this value in Equation (3.9) we find

$$\alpha_{MBR,c} = \frac{Md}{kd - \lfloor\frac{k}{2}\rfloor\lceil\frac{k}{2}\rceil} = \gamma_{MBR,c}.$$

Therefore,

$$(\alpha_{MBR,c}, \gamma_{MBR,c}) = (\frac{dM}{kd - \lfloor\frac{k}{2}\rfloor\lceil\frac{k}{2}\rceil}, \frac{dM}{kd - \lfloor\frac{k}{2}\rfloor\lceil\frac{k}{2}\rceil}).$$

Comparison with [1]

As discussed earlier, random linear codes for FCRS can be viewed as an achievability scheme for a more general problem, ACRS, where we are given a DSS and we are required to characterize the region (α, γ) for any fixed availability, where α is the storage size and γ is the repair bandwidth as defined in Equation (3.1). Interestingly, although the parameter availability has not been a motivation behind the work in [1], their scheme serves as an achievability result for this general problem too, for any availability $s-1 \in [1 : \lfloor\frac{n-1}{k}\rfloor]$. In particular, the random linear codes proposed in [1] can achieve an availability of $s-1$ if we set $d = d_o = \lfloor\frac{n-1}{s-1}\rfloor$ whereas the random linear codes for FCRS with s complete and one incomplete clusters ($n = ds + s_0$) achieve an availability of $s-1$ with $d = d_c = \lfloor\frac{n}{s}\rfloor$. In this section, we want to illustrate how FCRS can improve the tradeoff (α, γ) compared to [1] for the same availability and for certain range of parameters. To begin with, we find this comparison most interesting if neither system has any "residual servers", namely if $s-1|n-1$ and $s|n$. For instance let us select $n = k^2$ and $s = k$. For FCRS we will have k clusters each of size k and therefore $d_c = k$. For [1] we have $d_o = \frac{n-1}{s-1} = k+1$. By plugging in these values of d in Equations (3.9) and Equation (1) from [1] respectively, we can find the smallest value of α for any repair bandwidth γ. This is precisely what we have plotted in Figure 3.4 for a choice of $n = 100$ and $k = 10$ (both repair bandwidth and storage size are normalized by M). The figure suggests that at small values of repair bandwidth FCRS has a superior performance, and that there is a threshold value of γ beyond which it is outperformed by [1]. The improvements offered by FCRS are most visible at the MBR point for which we are going to provide an analytical comparison. Let us define $\gamma_{MBR,o} \overset{\triangle}{=} \frac{2d_oM}{2kd_o - k^2 + k}$ which is the value of the repair bandwidth at MBR point in [1]. The two conditions $s|n$ and $s-1|n-1$ imply that $n = (\ell s - \ell + 1)s$ for some positive integer ℓ. Under this constraint we have $d_o = \ell s + 1$ and $d_c = \ell s - \ell + 1$ and we can write

$$\frac{\gamma_{MBR,c}}{\gamma_{MBR,o}} = \frac{\frac{d_cM}{kd_c - \lfloor\frac{k}{2}\rfloor\lceil\frac{k}{2}\rceil}}{\frac{2d_oM}{2kd_o - k^2 + k}} \leq \frac{\frac{d_cM}{kd_c - \frac{k^2}{4}}}{\frac{2d_oM}{2kd_o - k^2 + k}} = \frac{\ell s + 1 - (k-1)/2}{\ell s - \ell + 1 - k/4} \cdot \frac{\ell s - \ell + 1}{\ell s + 1}.$$

This ratio is upper-bounded by 1 for almost the entire range of parameters (except when $s = 2$ or $k = 1$ or when $(s, k) \in \{(3, 2), (3, 3), (4, 2)\}$) as can be

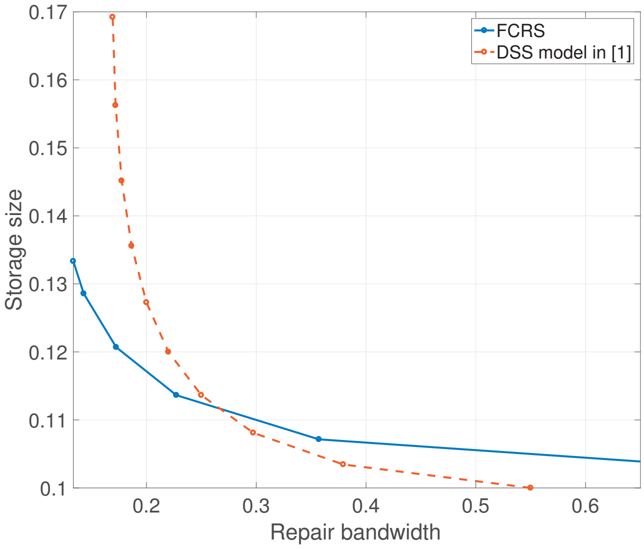


Figure 3.4: Comparison of the achievable region (α, γ) for FCRS and [1]. We have $n = 100$, $k = 10$ and both schemes are required to achieve an availability of $s - 1 = 9$.

easily verified. The ratio is smallest when s is maximal, that is, $s = \lfloor \frac{n}{k} \rfloor$. This implies $k = \ell s - \ell + 1$, which results in

$$\frac{\gamma_{MBR,c}}{\gamma_{MBR,o}} \leq \frac{2}{3} \cdot \frac{\ell s + \ell + 2}{\ell s + 1}.$$

This can be upper-bound by

$$\frac{\gamma_{MBR,c}}{\gamma_{MBR,o}} \leq \frac{2}{3} \cdot \frac{s+3}{s+1}$$

which is achieved when $\ell = 1$. This indicates an asymptotic multiplicative improvement of $2/3$ over the repair bandwidth at MBR point in comparison to [1].

It is worth noting that the assumption of "no residual server" becomes irrelevant as the parameters k and s grow large and as long as we choose $s = \lfloor \frac{n}{k} \rfloor$.

Proposition 3.1. *Let s , k and s_0 be three positive integers such that $0 \leq s_0 < \min\{k, s\}$. Let $n = sk + s_0$, $d_c = \lfloor \frac{n}{s} \rfloor = k$ and $d_o = \lfloor \frac{n-1}{s-1} \rfloor$, and define*

$$f(s, k, s_0) \triangleq \frac{\gamma_{MBR,c}}{\gamma_{MBR,o}} = \frac{\frac{d_c}{kd_c - \lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil}}{\frac{2d_o}{2kd_o - k^2 + k}}.$$

We have

$$f(s, k, s_0) \leq \frac{2}{3} \cdot \frac{(s+3)k + s - 3}{(s+1)k - 1}.$$

Proof.

$$\begin{aligned} f(s,k,s_0) &\leq \frac{\frac{d_c}{kd_c-\frac{k^2}{4}}}{\frac{2d_o}{2kd_o-k^2+k}} = \frac{d_c}{d_c-\frac{k}{4}} \cdot \frac{2d_o-k+1}{2d_o} \leq \frac{2}{3} \cdot \frac{2d_o-k+1}{d_o} \\ &\leq \frac{2}{3}(2-\frac{(k-1)(s-1)}{sk+s_0-1}) \leq \frac{2}{3}(2-\frac{(k-1)(s-1)}{(s+1)k-1}) \\ &= \frac{2}{3} \cdot \frac{(s+3)k+s-3}{(s+1)k-1}. \end{aligned}$$

□

As a result of this proposition we see that $\lim_{k\to\infty} f(s,k,s_0) \leq (\frac{2}{3}+\epsilon)\cdot\frac{s+3}{s+1}$. If we further let $s \to \infty$, the ratio of $\frac{2}{3}$ will be established.

3.3 The Exact Repair Model: Cubic Codes for the MBR Point

In this section we introduce cubic code as a coding scheme designed to minimize the repair bandwidth for the FCRS with $s+1$ clusters where $2 \leq s \leq \lfloor \frac{n}{k} \rfloor$. Cubic Codes are examples of Affine Resolvable Designs [25] which themselves are subclasses of Fractional Repetition codes [24]. They can also be viewed as generalizations of grid codes discussed in [25]. As we shall prove in the next section via a converse bound, cubic codes are information-theoretically optimal, in the sense that they minimize the repair bandwidth of FCRS with 2 and 3 clusters and no residual server. On the other hand, they have a strictly worse performance compared to random linear codes that can achieve the cutset bound analyzed in Section 3.2. This implies an inherent gap between the functional repair and exact repair models at the MBR point for the FCRS with 2 and 3 complete clusters. This is by contrast to the DSS model studied in [1] where the MBR point for functional and exact repair coincide. Despite this, we will show that cubic codes still achieve an asymptotic multiplicative improvement of 0.79 on the repair bandwidth compared to the MBR codes [8, 9] that guarantee the same availability.

Suppose as stated in Section 3.1 the network consists of $n = ds + s_0$ servers divided into s clusters of size d and one cluster of size $s_0 < s$. In this section we further assume that $s_0 < d$. If this is not true, we can increase s to s' such that $n = ds' + s'_0$ where $s'_0 < \min\{d, s'\}$ and $s' \leq \lfloor \frac{n}{k} \rfloor$. Also note that this condition is automatically satisfied if $k^2 \geq n$. Assuming the file is large enough, we break it into m independent chunks $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ so that $H(\mathcal{M}_i) = M/m$. The value of m will be determined shortly. We start by constructing a (d^{s+1}, m) MDS code over these m symbols and indexing the codewords by strings of $s+1$ digits. Let C_b represent a codeword of the MDS code where b is a string of $s+1$ digits, $b = b_{s+1} \ldots b_1$, and $b_i \in [1 : d]$.

Server j in Cluster i stores all the codewords of the form C_b where $b_i = j$ and the other indices vary. That is,

$$X_j^{(i)} = \{C_b | b_i = j\} \quad \text{for all } (i,j) \in \{[1:s] \times [1:d]\} \cup \{\{s+1\} \times [1:s_0]\}.$$

This is akin to arranging the codewords of an MDS code in an s+1-dimensional hyper-cube and requiring the servers within the i'th cluster to store hyper-planes orthogonal to the i'th axis. See Figure 3.5 for an illustration. One can also express this code in terms of its generator matrix. Let B be the generator matrix of any (d^{s+1}, m) MDS code. For any integer ℓ let $\phi_{s+1}(\ell)\dots\phi_1(\ell)$ be the $s+1$-digit expansion of ℓ in base d, where $\phi_{s+1}(\cdot)$ represents the most significant digit. For any $(i,j) \in \{[1:s] \times [1:d]\} \cup \{\{s+1\} \times [1:s_0]\}$ let $Q^{(i,j)}$ be the d^s by d^{s+1} matrix where

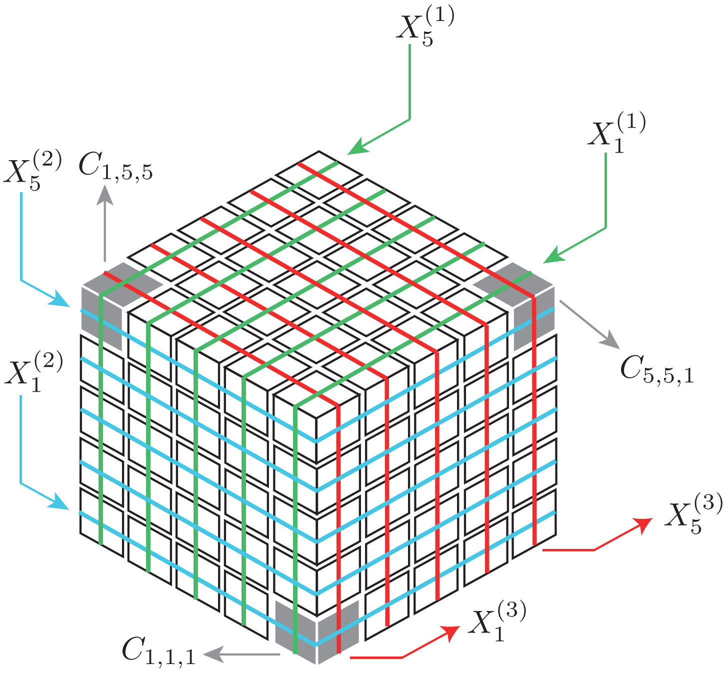


Figure 3.5: Cubic Codes for the FCRS with 3 clusters and with $n = 15$, $d = 5$. The codewords of an MDS code (the small blocks) are arranged in a three dimensional cube. The j'th node in the i'th cluster stores the codewords that form that j'th plane orthogonal to the i'th axis.

$$Q_{\ell,r}^{(i,j)} = \begin{cases} 1 & \text{if } \begin{cases} \phi_e(r) = \phi_e(\ell) & \text{for } e \in [1:i-1] \text{ and} \\ \phi_e(r) = \phi_{e-1}(\ell) & \text{for } e \in [i+1:s+1] \text{ and} \\ \phi_i(r) = j-1 & \end{cases} \\ 0 & \text{otherwise.} \end{cases}$$

Then we can write

$$X_j^{(i)} = Q^{(i,j)} B [\mathcal{M}_1, \dots, \mathcal{M}_m]^T.$$

If a server $X_\ell^{(r)}$ fails and chooses cluster i for repair, then the j'th server in cluster i transmits $Y_{\ell,j}^{(r,i)} = \{C_b | b_i = j, b_r = \ell\}$ to the newcomer. Upon receiving all such codewords $Y_{\ell,[1:d]}^{(r,i)}$, the newcomer is capable of reconstructing the failed server. Furthermore, the newcomer receives a total of d^s codewords which shows that for cubic codes $d\beta = \alpha$.

Let us now analyze the performance of this code. Based on the data recovery requirement, we know that every k servers in the network, regardless of their cluster must be able to recover the file. Consider a set of k servers chosen in such a way that k_i servers belong to cluster i where $\sum_{i=1}^{s+1} k_i = k$. These servers together provide a total of R codewords of the MDS code where

$$\begin{aligned} R &= \sum_{i=1}^{s+1} d^s k_i - \sum_{i=1}^{s+1} \sum_{j=i+1}^{s+1} d^{s-1} k_i k_j \\ &+ \sum_{i=1}^{s+1} \sum_{j=i+1}^{s+1} \sum_{\ell=j+1}^{s+1} d^{s-2} k_i k_j k_\ell - \dots \\ &= d^{s+1} - \prod_{i=1}^{s+1} (d - k_i). \end{aligned}$$

Thus, in order for the file $\mathcal{M}$ to be recoverable from these k servers, we must have

$$d^{s+1} - \prod_{i=1}^{s+1} (d - k_i) \geq m. \tag{3.11}$$

Note that this inequality must be true for any choice of the parameters k_i. Let us therefore minimize the left hand side of this inequality over the constraints $\sum_{i=1}^{s+1} k_i = k$, $k_i \geq 0$ and $k_{s+1} \leq s_0$.

$$k^*_{[1:s+1]} = \underset{k_{[1:s+1]}: \sum_{i=1}^{s+1} k_i = k \text{ and } k_{s+1} \leq s_0}{\arg\min} \quad d^{s+1} - \prod_{i=1}^{s+1} (d - k_i). \tag{3.12}$$

To solve this optimization problem, it is necessary to distinguish between two regimes.

- **Regime 1:** $s_0 \geq \lfloor \frac{k}{s+1} \rfloor$.

  Define $s_1 = \mod (k, s+1)$. The solution to (3.12) can be expressed as

  $$k_i^* = \begin{cases} \lceil \frac{k}{s+1} \rceil & \text{if } i \leq s_1 \\ \lfloor \frac{k}{s+1} \rfloor & \text{if } s_1 < i \leq s+1. \end{cases}$$

  Returning to Inequality (3.11) we can write $m = d^{s+1} - (d - \lceil \frac{k}{s+1} \rceil)^{s_1} (d - \lfloor \frac{k}{s+1} \rfloor)^{(s+1-s_1)}$. Since each server stores d^s codewords, the storage size is

$$\alpha = \frac{Md^s}{m} = \frac{Md^s}{d^{s+1} - (d - \lceil \frac{k}{s+1} \rceil)^{s_1} (d - \lfloor \frac{k}{s+1} \rfloor)^{(s+1-s_1)}}.$$

We saw that for cubic codes $\gamma = \alpha$. Therefore,

$$\gamma = \frac{Md^s}{d^{s+1} - (d - \lceil \frac{k}{s+1} \rceil)^{s_1} (d - \lfloor \frac{k}{s+1} \rfloor)^{(s+1-s_1)}}.$$

- **Regime 2:** $s_0 < \lfloor \frac{k}{s+1} \rfloor$.

  This time define $s_1 = \mod (k - s_0, s)$. The solution to (3.12) is

$$k_i^* = \begin{cases} \lceil \frac{k-s_0}{s} \rceil & \text{if } i \leq s_1 \\ \lfloor \frac{k-s_0}{s} \rfloor & \text{if } s_1 < i \leq s \\ s_0 & \text{if } i = s+1. \end{cases}$$

  Again, plugging this into (3.11) we have

$$\begin{aligned} m &= d^{s+1} - (d - \lceil \frac{k-s_0}{s} \rceil)^{s_1} (d - \lfloor \frac{k-s_0}{s} \rfloor)^{s+1-s_1} \\ &- (\lfloor \frac{k-s_0}{s} \rfloor - s_0)(d - \lceil \frac{k-s_0}{s} \rceil)^{s_1} (d - \lfloor \frac{k-s_0}{s} \rfloor)^{s-s_1} \\ &= d^{s+1} - (d - s_0)(d - \lceil \frac{k-s_0}{s} \rceil)^{s_1} (d - \lfloor \frac{k-s_0}{s} \rfloor)^{s-s_1}. \end{aligned}$$

  Consequently,

$$\gamma = \alpha = \frac{Md^s}{d^{s+1} - (d - s_0)(d - \lceil \frac{k-s_0}{s} \rceil)^{s_1} (d - \lfloor \frac{k-s_0}{s} \rfloor)^{s-s_1}}.$$

  Let us summarize this in the following theorem.

Theorem 3.2. *Suppose we have an FCRS with parameters n, k, s where $n = ds + s_0$ and $s_0 < \min\{d, s\}$. The cubic codes achieve a repair bandwidth of*

$$\gamma_{cc}(n,k,s) = \begin{cases} \frac{Md^s}{d^{s+1} - (d - \lceil \frac{k}{s+1} \rceil)^{s_1} (d - \lfloor \frac{k}{s+1} \rfloor)^{(s+1-s_1)}} & \textit{if } s_0 \geq \lfloor \frac{k}{s+1} \rfloor \\ \frac{Md^s}{d^{s+1} - (d - s_0)(d - \lceil \frac{k-s_0}{s} \rceil)^{s_2} (d - \lfloor \frac{k-s_0}{s} \rfloor)^{s-s_2}} & \textit{if } s_0 < \lfloor \frac{k}{s+1} \rfloor \end{cases} \tag{3.13}$$

where $s_1 = \mod (k, s+1)$ and $s_2 = \mod (k - s_0, s)$.

An interesting regime is when there are no residual servers, that is when $n = sd$. In this case we have

$$\gamma_{cc}(sd, k, s) = \frac{M/d}{1 - (1 - \lceil \frac{k}{s} \rceil / d)^{s_2} (1 - \lfloor \frac{k}{s} \rfloor / d)^{s - s_2}}. \tag{3.14}$$

This can be further simplified if we assume $s|k$.

$$\gamma_{cc}(sd, \ell s, s) = \frac{M/d}{1 - (1 - \frac{\ell}{d})^s}.$$

In fact, it follows from a simple argument that

$$\gamma_{cc}(sd, k, s) \leq \frac{M/d}{1 - (1 - \frac{k}{sd})^s}. \tag{3.15}$$

To see why, note that

$$\left(\frac{1 - \frac{\lceil k/s \rceil}{d}}{1 - \frac{k}{sd}} \right)^{s_2} \left(\frac{1 - \frac{\lfloor k/s \rfloor}{d}}{1 - \frac{k}{sd}} \right)^{s - s_2} \leq 1$$

which is true since the geometric mean of s numbers is upperbounded by their arithmatic mean.

It is not difficult to see that if we fix s, k and d, $\gamma_{cc}(n, k, s)$ is monotonically increasing in s_0 for $0 \leq s_0 \leq d$. This is because adding one more server to the last cluster cannot increase the expression $\min_{k_{[1:s+1]}} d^{s+1} - \prod_{i=1}^{s+1} (d - k_i)$. Based on this property and Equation (3.15) we can establish the following bound.

Corollary 3.1. *Let $\gamma_{cc}(n, k, s)$ be the repair bandwidth of cubic codes for an FCRS with parameters n, k, s where $n = ds + s_0$ and $s_0 < \min\{s, d\}$. Then*

$$\gamma_{cc}(n, k, s) \leq \frac{M/d}{1 - (1 - \frac{k}{(s+1)d})^{s+1}}.$$

This bound is sufficiently tight for our purpose and we will resort to it for our analytical comparison in the next section.

Comparison with Functional Repair and [9]

Let us start with a numerical comparison. Here we fix the number of servers n and let the availability grow gradually. For every fixed availability we compare the repair bandwidth for the three schemes: the MBR point for functional repair of FCRS in Section 3.2, that is expression (3.10), the MBR codes proposed in [9] (which corresponds to the functional repair MBR point in [1]) and finally the cubic codes, that is expression (3.13). For this numerical analysis we set $n = 400$, $k = 20$, and we let $s - 1$ grow from 1 to 19. We normalize the repair bandwidth by the size of the file. As can be seen in Figure 3.6, as s grows large cubic codes perform somewhere in between the functional repair points of FCRS and [1]. The multiplicative improvement over [1] can be measured around 0.79 at its peak, i.e. when $s - 1 = 19$. This will be theoretically justified next.

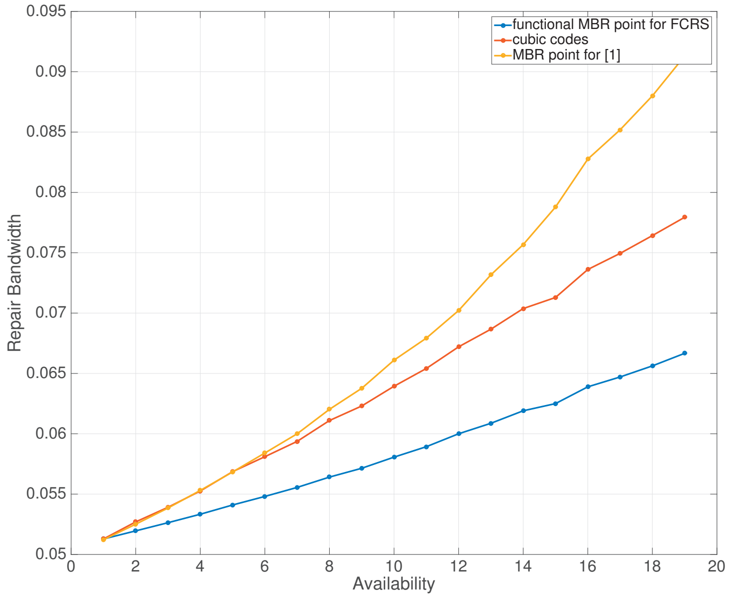


Figure 3.6: Comparison of cubic codes with the functional repair MBR point of FCRS and the MBR point of [1] for $n = 400$, $k = 20$.

Let us first bound the ratio of repair bandwidth for cubic codes and functional repair bandwidth for the FCRS. Assuming k even we have

$$\begin{aligned}\frac{\gamma_{cc}}{\gamma_{MBR,c}} &\leq \frac{\frac{M/d_c}{1-(1-\frac{k}{(s+1)d_c})^{s+1}}}{\frac{M}{k-\frac{k^2}{4d_c}}} = \frac{k(1-\frac{k}{4d_c})}{d_c - d_c(1-\frac{k}{(s+1)d_c})^{s+1}} \\ &\leq \frac{k}{d_c}(1-\frac{k}{4d_c})\frac{1}{1-e^{-k/d_c}} \leq \frac{3}{4(1-e^{-1})}.\end{aligned}$$

In conjunction with the results of Section 3.2, if we further assume that s is chosen as large as possible, i.e. $s = \lfloor \frac{n}{k} \rfloor$, we can write

$$\frac{\gamma_{cc}}{\gamma_{MBR,o}} \leq \frac{3}{4(1-e^{-1})} \cdot \frac{2}{3} \cdot \frac{(s+3)k+s-3}{(s+1)k-1}.$$

Since the expression for the ratio $\frac{\gamma_{cc}}{\gamma_{MBR,o}}$ does not depend on whether k is odd or even, the same bound holds for general k. We have therefore established the following proposition.

Proposition 3.2. *Let s , k and s_0 be three positive integers such that $s_0 < \min\{k, s\}$. Let $n = sk + s_0$, $d_c = \lfloor \frac{n}{s} \rfloor = k$ and $d_o = \lfloor \frac{n-1}{s-1} \rfloor$, and define*

$$g(s, k, s_0) \triangleq \frac{\gamma_{cc}}{\gamma_{MBR,o}} \quad \leq \quad \frac{\frac{1}{1-(1-\frac{k}{(s+1)d_c})^{s+1}}}{\frac{2d_o}{2kd_o-k^2+k}}.$$

We have

$$g(s, k, s_0) \leq \frac{1}{2(1-e^{-1})} \cdot \frac{(s+3)k + s - 3}{(s+1)k - 1}.$$

Based on this proposition, if we let $k \to \infty$, we find

$$\frac{\gamma_{cc}}{\gamma_{MBR,o}} \leq \frac{3}{4(1-e^{-1})} \cdot (\frac{2}{3} + \epsilon) \cdot \frac{s+3}{s+1} \approx 0.79 \cdot \frac{s+3}{s+1}.$$

This proves that for the same availability of $s - 1 = \lfloor \frac{n}{k} \rfloor - 1$, cubic codes achieve an asymptotic (as $k, s, n \to \infty$) multiplicative improvement of 0.79 over the minimum repair bandwidth in comparison to MBR point in [1].

3.4 Converse Bound for Exact Repair

In this section we provide an exact repair converse bound for the FCRS. The main purpose of this converse bound is to prove that the cubic codes introduced in Section 3.3 minimize the repair bandwidth for the FCRS with two or three complete clusters and no residual servers. Unfortunately a straightforward generalization of the bound to more than three clusters is loose and is omitted for this reason. As we are dealing with exact repair we can drop the time index from X and Y variables, since $X^{(i)}_{j,t} = X^{(j)}_{j,t'}$ for all t and t'. Therefore, we simply represent this by $X^{(i)}_j$. The same slight change of notation applies for the helper variables Y.

Theorem 3.3. *Cubic codes achieve the minimum repair bandwidth under exact repair for the FCRS with 2 or 3 complete clusters, i.e. when $s \in \{2, 3\}$ and $s_0 = 0$.*

Proof. We will present the proof for $s = 3$. The proof for $s = 2$ is omitted to avoid redundancy. Therefore, we have $n = 3d$. Assume k_i servers from the i'th cluster take part in the data recovery. Specifically, let $\tau_i \subseteq [1 : d]$ for $i \in [1 : 3]$ represent the set of indices of the servers from i'th cluster that are connected to a data collector, such that $|\tau_i| = k_i$ and $\sum_{i=1}^{3} k_i = k$. By taking an average over all possible such choices of τ_1, τ_2, τ_3 we can write

$$
\begin{aligned}
M &\leq \frac{1}{\binom{d}{k_1}\binom{d}{k_2}\binom{d}{k_3}} \sum_{\substack{\tau_1,\tau_2,\tau_3\\ \tau_i\subseteq[1:d],\, |\tau_i|=k_i}} H(X^{(1)}_{\tau_1}, X^{(2)}_{\tau_2}, X^{(3)}_{\tau_3}) \\
&= \frac{1}{\binom{d}{k_1}\binom{d}{k_2}\binom{d}{k_3}} \sum_{\tau_1,\tau_2,\tau_3} H(X^{(3)}_{\tau_3}|X^{(2)}_{\tau_2}, X^{(1)}_{\tau_1}) + \frac{1}{\binom{d}{k_1}\binom{d}{k_2}} \sum_{\tau_1,\tau_2} H(X^{(2)}_{\tau_2}|X^{(1)}_{\tau_1}) \\
&+ \frac{1}{\binom{d}{k_1}} \sum_{\tau_1} H(X^{(1)}_{\tau_1}). \qquad (3.16)
\end{aligned}
$$

Let us start by upper-bounding the first term.

$$
\begin{aligned}
Q_3 &\triangleq \frac{1}{\binom{d}{k_1}\binom{d}{k_2}\binom{d}{k_3}} \sum_{\tau_1,\tau_2,\tau_3} H(X^{(3)}_{\tau_3}|X^{(2)}_{\tau_2}, X^{(1)}_{\tau_1}) \\
&\overset{(*)}{\leq} \frac{1}{\binom{d}{k_1}\binom{d}{k_2}\binom{d}{k_3}} \sum_{\tau_1,\tau_2,\tau_3} H(Y^{(3,2)}_{\tau_3,[1:d]}|Y^{(3,2)}_{\tau_3,\tau_2}, X^{(1)}_{\tau_1}) \\
&= \frac{1}{\binom{d}{k_1}\binom{d}{k_2}\binom{d}{k_3}} \sum_{\tau_1,\tau_2,\tau_3} H(Y^{(3,2)}_{\tau_3,[1:d]\backslash\tau_2}|Y^{(3,2)}_{\tau_3,\tau_2}, X^{(1)}_{\tau_1}) \\
&\overset{(\#)}{\leq} \frac{1}{\binom{d}{k_1}\binom{d}{k_3}} \sum_{\tau_1,\tau_3} \frac{d-k_2}{d} H(Y^{(3,2)}_{\tau_3,[1:d]}|X^{(1)}_{\tau_1}).
\end{aligned}
$$

Inequality $(*)$ follows from the fact that $H(X^{(3)}_{\tau_3}|Y^{(3,2)}_{\tau_3,[1:d]}) = 0$ and $H(Y^{(3,2)}_{\tau_3,\tau_2}|X^{(2)}_{\tau_2}) = 0$. Inequality $(\#)$ follows from (conditional) Han's inequality. Let us continue by bounding the right hand side of this inequality.

$$
\begin{aligned}
Q_3 &\leq \frac{1}{\binom{d}{k_1}\binom{d}{k_3}} \sum_{\tau_1,\tau_3} \frac{d-k_2}{d} H(Y^{(3,2)}_{\tau_3,[1:d]}|X^{(1)}_{\tau_1}) \\
&= \frac{1}{\binom{d}{k_1}\binom{d}{k_3}} \sum_{\tau_1,\tau_3} \frac{d-k_2}{d} H(X^{(3)}_{\tau_3}|X^{(1)}_{\tau_1}) + \frac{1}{\binom{d}{k_1}\binom{d}{k_3}} \sum_{\tau_1,\tau_3} \frac{d-k_2}{d} H(Y^{(3,2)}_{\tau_3,[1:d]}|X^{(1)}_{\tau_1}, X^{(3)}_{\tau_3}) \\
&\leq \frac{1}{\binom{d}{k_1}\binom{d}{k_3}} \sum_{\tau_1,\tau_3} \frac{d-k_2}{d} H(Y^{(3,1)}_{\tau_3,[1:d]}|Y^{(3,1)}_{\tau_3,\tau_1}) + \frac{1}{\binom{d}{k_1}\binom{d}{k_3}} \sum_{\tau_1,\tau_3} \frac{d-k_2}{d} H(Y^{(3,2)}_{\tau_3,[1:d]}|X^{(3)}_{\tau_3}) \\
&\overset{(*)}{\leq} \frac{1}{\binom{d}{k_3}} \frac{(d-k_2)(d-k_1)}{d^2} \sum_{\tau_3} H(Y^{(3,1)}_{\tau_3,[1:d]}) + k_3(d-k_2)\beta - \frac{d-k_2}{d\binom{d}{k_3}} \sum_{\tau_3} H(X^{(3)}_{\tau_3}) \\
&\leq k_3 \frac{(d-k_2)(d-k_1)}{d}\beta + k_3(d-k_2)\beta - \frac{d-k_2}{d\binom{d}{k_3}} \sum_{\tau_3} H(X^{(3)}_{\tau_3}).
\end{aligned}
$$

Inequality $(*)$ follows from the fact that $H(Y|X) = H(Y) - H(X)$ if X is a function of Y and from applying Han's inequality for a second time. We go back to Equation (3.16) and bound the second term.

$$
\begin{aligned}
Q_2 &\triangleq \frac{1}{\binom{d}{k_1}\binom{d}{k_2}} \sum_{\tau_1,\tau_2} H(X^{(2)}_{\tau_2}|X^{(1)}_{\tau_1}) \leq \frac{1}{\binom{d}{k_1}\binom{d}{k_2}} \sum_{\tau_1,\tau_2} H(Y^{(2,1)}_{\tau_2,[1:d]}|Y^{(2,1)}_{\tau_2,\tau_1}) \\
&= \frac{1}{\binom{d}{k_1}\binom{d}{k_2}} \sum_{\tau_1,\tau_2} H(Y^{(2,1)}_{\tau_2,[1:d]\backslash\tau_1}|Y^{(2,1)}_{\tau_2,\tau_1}) \\
&\leq \frac{1}{\binom{d}{k_2}} \frac{d-k_1}{d} \sum_{\tau_2} H(Y^{(2,1)}_{\tau_2,[1:d]}) \leq k_2(d-k_1)\beta.
\end{aligned}
$$

Therefore, we proved

$$
\begin{aligned}
M &\leq Q_3 + Q_2 + \frac{1}{\binom{d}{k_1}} \sum_{\tau_1} H(X^{(1)}_{\tau_1}) \\
&\leq k_3 \frac{(d-k_2)(d-k_1)}{d}\beta + k_3(d-k_2)\beta - \frac{d-k_2}{d\binom{d}{k_3}} \sum_{\tau_3} H(X^{(3)}_{\tau_3}) \\
&+ k_2(d-k_1)\beta + \frac{1}{\binom{d}{k_1}} \sum_{\tau_1} H(X^{(1)}_{\tau_1}).
\end{aligned}
$$

Via an identical procedure we can more generally prove that if $\pi : \{1,2,3\} \to \{1,2,3\}$ is a bijection, then

$$
\begin{aligned}
M &\leq k_{\pi_3} \frac{(d-k_{\pi_2})(d-k_{\pi_1})}{d}\beta + k_{\pi_3}(d-k_{\pi_2})\beta \\
&- \frac{d-k_{\pi_2}}{d\binom{d}{k_{\pi_3}}} \sum_{\tau_{\pi_3}} H(X^{(\pi_3)}_{\tau_{\pi_3}}) + k_{\pi_2}(d-k_{\pi_1})\beta + \frac{1}{\binom{d}{k_{\pi_1}}} \sum_{\tau_{\pi_1}} H(X^{(\pi_1)}_{\tau_{\pi_1}}).
\end{aligned}
$$

By averaging this inequality over all possible bijections π we find

$$
\begin{aligned}
M &\leq k_3 \frac{(d-k_2)(d-k_1)}{3d}\beta + k_2\beta\frac{(d-k_3)(d-k_1)}{3d} + k_1\beta\frac{(d-k_3)(d-k_1)}{3d} \\
&+ \frac{1}{3}((d-k_1)k_2 + (d-k_1)k_3 + (d-k_2)k_1 + (d-k_2)k_3 + (d-k_3)k_1 \\
&+ (d-k_3)k_2)\beta + \frac{1}{3}(k_1k_2 + k_1k_3 + k_2k_3)\beta \\
&= \frac{\beta}{3}(2dk + k_3\frac{(d-k_2)(d-k_1)}{d} + k_2\frac{(d-k_3)(d-k_1)}{d} + k_1\frac{(d-k_2)(d-k_3)}{d} \\
&- k_1k_2 - k_1k_3 - k_2k_3) \\
&= \frac{\beta}{3d}(2d^2k + 3k_1k_2k_3 - 3d(k_1k_2 + k_1k_3 + k_2k_3) + d^2k) \\
&= \frac{\beta}{d}(d^3 - (d-k_1)(d-k_2)(d-k_3)),
\end{aligned}
$$

where we have used the fact that $\alpha \leq d\beta$ (due to the repair requirement) and $\sum k_i = k$. Note that the inequality above must hold for any choice of k_1, k_2, k_3

that satisfy $\sum k_i = k$. In particular we must be able to choose $k_i = \lceil \frac{k}{3} \rceil$ for $i \in [1 : s_2]$ and $k_i = \lfloor \frac{k}{3} \rfloor$ for $i \in [s_2 + 1 : 3]$ where $s_2 = \mod(k, 3)$. For this choice of $k_{[1:3]}$ we have

$$\gamma = d\beta \geq \frac{Md^2}{d^3 - (d - \lceil \frac{k}{3} \rceil)^{s_2}(d - \lfloor \frac{k}{3} \rfloor)^{3-s_2}} = \frac{M/d}{1 - (1 - \lceil \frac{k}{3} \rceil / d)^{s_2}(1 - \lfloor \frac{k}{3} \rfloor / d)^{3-s_2}}.$$

This is the same expression as the achievable repair bandwidth of cubic codes specified by Equation (3.14) if we set $s = 3$. □

4 Multi-Library Coded Caching

In this chapter we study the performance of a CDN which has access to the data from several different companies, as one naturally expects in practice. The model studied in [2] considers a single collection which consists of all the files required by the users. These files are of equal size and each user is interested in precisely one such file. In this sense, the model does not distinguish among heterogeneous data, and does not take into account independent requests that a user may make from different providers. Hence, for our purpose we introduce a new model; we assume the CDN has access to multiple collections of files which we refer to as *libraries.* The files on different libraries are not necessarily of equal size and each user makes independent requests from different libraries. Subsequently, our goal is to find the optimal caching strategy for such a network. That is, we are interested in minimizing the total delivery rate R assuming each user has a cache of size M.

Our main contribution is to derive inner and outer bounds for the delivery rate of the described network, and to show that when the number of files in all libraries are equal, the optimal caching strategy only requires coding across files within the same library. In other words, each user partitions his cache into several segments and dedicates one segment to each library and ignores coding opportunities across files from different libraries. Similarly, in the delivery phase, there is no need to perform coding across libraries.

The optimality of this memory-sharing strategy has interesting practical implications. Firstly, if one knows the optimal caching strategy for the single-library problem, one can simply extend it to multiple libraries. Secondly, although CDNs receive their data from multiple different companies, large streaming corporations such as Netflix are moving their traffic to their own CDNs [32] and perform independently from one another. This can be modelled as a network with several servers each having access to distinct files and having limited or no interactions with each other. The optimality of memory-sharing implies that there is no loss due to this emigration from one centralized CDN to multiple isolated ones. From a different perspective, coding across files from different servers in the placement phase introduces vulnerability to network failures; if one server goes down in the delivery phase, the users will not be able to recover the files from any other.

The basic coded caching strategy proposed in [2] has been extended to a variety of other networking scenarios, among which are decentralized [33], multi-server [34], hierarchical [35], multi-request [36] and online coded caching [37], and caching with heterogenous cache sizes [38]. Perhaps the most relevant to our work is "multi-level coded caching" [39, 40] where several popularity classes of files are served to the users via access points that are in possession of local caches. The mathematical model in [40] is similar to the model considered here, with "popularity classes" taking the role of "libraries" in our terminology. More precisely, the model in [40] is slightly more general in that it allows multiple users to have access to each cache, and is slightly less general in that it forces files on all libraries to be of the same size. The more important difference between [40] and the present work concerns the results: the caching strategies are substantially different, and while [40] establishes order-optimality (under specific constraints), the present work establishes an exact optimality result (under certain other constraints).

We continue this chapter by providing a motivating example in Section 4.1. Next, we will formally define our problem in Section 4.2 and express our main achievability and converse results in Section 4.3. In Section 4.4 we will find the optimal memory-sharing strategy and will prove that it is globally optimal when the number of files are equal in different libraries.

4.1 Motivating Example

Assume we have two libraries. Library 1 consists of two files A and B each of size F_1 and in Library 2 there are two other files C and D each of size $F_2 = 1.5F_1$. Suppose we have two users each with a cache of size $M = F_1 + F_2$. We plan to perform memory-sharing, that is to divide the cache of each user into two segments and assign each segment to one library. Let us assume we assign λM of each cache to Library 1 and the rest to the Library 2, for some $0 \leq \lambda \leq 1$. In the delivery phase, each user will request one file from each library. In other words, each user will request either of $\{A, C\}$, $\{A, D\}$, $\{B, C\}$ or $\{B, D\}$.

For each library we know the optimal coded caching strategy from [2]. Therefore, for each λ we know the minimum value of $R(\lambda) \stackrel{\triangle}{=} R_1(\lambda M) + R_2((1-\lambda)M)$. This curve is plotted in Figure 4.1 and can be described by the

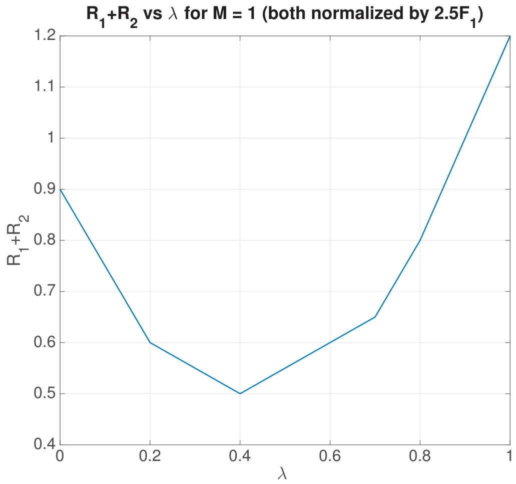


Figure 4.1: The sum of the delivery rates of Library 1 and Library 2 vs λ, the fraction of the cache dedicated to Library 1. Library 1 consists of 2 files each of size F_1 and Library 2 has two files each of size $F_2 = 1.5F_1$. The cache size of each of the two users is $M = F_1 + F_2$.

following set of equations

$$R(\lambda) = \begin{cases} \frac{9}{10} - \frac{3}{2}\lambda & \text{if } 0 \leq \lambda < \frac{1}{5}, \\ \frac{7}{10} - \frac{1}{2}\lambda & \text{if } \frac{1}{5} \leq \lambda < \frac{2}{5}, \\ \frac{3}{10} + \frac{1}{2}\lambda & \text{if } \frac{2}{5} \leq \lambda < \frac{7}{10}, \\ -\frac{2}{5} + \frac{3}{2}\lambda & \text{if } \frac{7}{10} \leq \lambda < \frac{4}{5}, \\ -\frac{4}{5} + 2\lambda & \text{if } \frac{4}{5} \leq \lambda \leq 1. \end{cases}$$

Evidently, the minimum of $R_1 + R_2$ is $\frac{1}{2}$ and is attained for $\lambda = \frac{2}{5}$. This is the same point that we obtain if we divide the cache size proportional to the size of the files on the two libraries, i.e. $\lambda = \frac{F_1}{F_1+F_2}$. As we will see in Section 4.4.1, this is always the case, regardless of the number of libraries, the number of users or the size of the cache. As long as all libraries have the same number of files, the optimal memory-sharing strategy is one that divides the cache among different libraries proportional to their size of the files. More importantly, we will see that this strategy is globally optimal. That is, coding across files from different libraries cannot help in reducing the total delivery rate.

4.2 Statement of the Problem

Our problem statement closely follows that of [2] with the difference that we classify the files into several libraries; allow the size of the files on different libraries to be different, and allow the users to request files from every library. More formally, suppose we have L libraries each consisting of N_ℓ files, $\ell \in [1 :$

$L]$. We denote by $W_n^{(\ell)}$ the n'th file on the ℓ'th library and assume all the files are independent. Furthermore, we assume the n'th file on the ℓ'th library is of size $F_n^{(\ell)} = \alpha_n^{(\ell)} F$ where $\sum_{\ell=1}^{L} \frac{1}{N_\ell} \sum_{n=1}^{N_\ell} \alpha_n^{(\ell)} = 1$. We impose this last normalization constraint in order to make our definition of rate and cache size compatible with the single-library case with equal file sizes. It is important to note that in this work we are mostly interested in the the scenario where the size of the files within each library are equal. In other words, $F_n^{(\ell)} = F^{(\ell)}$ for $n \in [1 : N_\ell]$ and $\ell \in [1 : L]$. This more general notation is introduced in order to facilitate the statement of our converse results in their full generality. We have K users each with a cache of normalized size M. The caching scheme consists of two phases, the placement phase and the delivery phase. In the placement phase each user has access to all the files and stores an arbitrary function of them of size MF in his cache Z_k. Following the notations in [2], we call these *caching functions*

$$\phi_k : \prod_{\ell=1}^{L} \prod_{n=1}^{N_\ell} [1 : 2^{F_n^{(\ell)}}] \rightarrow [1 : 2^{\lfloor FM \rfloor}], \ \forall k \in [1 : K]. \tag{4.1}$$

Note that the requests are unknown in this phase and hence ϕ_k does not depend on them. In the delivery phase, every user requests exactly one file from each library [1]. The requests made to the ℓ'th library are represented by a vector $d_{[1:K]}^{(\ell)}$ for every $\ell \in [1 : L]$ where $d_k^{(\ell)} \in [1 : N_\ell]$ for every $k \in [1 : K]$. Based on this request vector an update message $X_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}}$ of size RF is then broadcast by the server to all the users. This update message naturally depends on the requests and the files

$$X_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}} = \psi_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}} (\{W_{[1:N_\ell]}^{(\ell)}\}_{\ell=1}^{L})$$

where $\psi_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}}$ are called the *encoding functions*

$$\psi_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}} : \prod_{\ell=1}^{L} \prod_{n=1}^{N_\ell} [1 : 2^{F_n^{(\ell)}}] \rightarrow [1 : 2^{\lfloor FR \rfloor}]. \tag{4.2}$$

Each user reconstructs his desired files as a function of the content of his cache Z_k and the update message.

$$\begin{aligned} &\hat{W}_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i} {=} \mu_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i} (X_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}}, Z_k), \\ &\forall k \in [1 : K], i \in [1 : L] \end{aligned}$$

where $\mu_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i}$ are called the *decoding functions*

$$\mu_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i} : [1 : 2^{\lfloor RF \rfloor}] \times [1 : 2^{\lfloor FM \rfloor}] \rightarrow [1 : 2^{F^{(i)}_{d_k^{(i)}}}]. \tag{4.3}$$

[1] It is perhaps more realistic to assume each user orders *at most* one file from each library. However, since we are studying the worst case analysis, this will not change the results.

We say that a memory-rate pair (R, M) is achievable for a network with parameters $(L, \{\alpha^{(\ell)}_{[1:N_\ell]}\}_{\ell=1}^L, N_{[1:L]})$ if there exists a caching strategy such that for any request vector $\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^L$ each user is able to recover all his desired files. In other words, if for any $\epsilon > 0$ and F large enough, there exist caching, encoding and decoding functions for which the probability of error

$$\max_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^L \in \prod_{\ell=1}^L [1:N_\ell]^K} \max_{k\in[1:K], i\in[1:L]} \mathbb{P}(\hat{W}_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^L, k, i} \neq W^{(i)}_{d^{(i)}_k})$$

can be upper bounded by ϵ. For a network with parameters $(L, \{\alpha^{(\ell)}_{[1:N_\ell]}\}_{\ell=1}^L, N_{[1:L]})$ the memory-rate tradeoff is defined as

$$\begin{aligned} &R^*(L, M, \{\alpha^{(\ell)}_{[1:N_\ell]}\}_{\ell=1}^L, N_{[1:L]}) \triangleq \\ &\inf\{R : (M, R) \text{ is achievable}\}. \end{aligned} \tag{4.4}$$

Whenever the size of the files within each library are equal, that is $\alpha^{(\ell)}_n = \alpha^{(\ell)}$ for $n \in [1 : N_\ell]$ and $\ell \in [1 : L]$, we use the simplified notation $R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]})$ instead of $R^*(L, M, \{\alpha^{(\ell)}_{[1:N_\ell]}\}_{\ell=1}^L, N_{[1:L]})$. Our goal is to characterize the memory-rate tradeoff of a network with L libraries in terms of the memory-rate tradeoffs of networks with single libraries. To this aim we find outer and inner-bounds for the L-library network and prove that the two bounds match in special cases.

4.3 General Results: Achievability and Converse Bounds

4.3.1 Achievability

Our achievability results are based on a memory-sharing strategy. We divide the cache of each user into L segments and assign one segment to each library. We ignore coding opportunities across files from different libraries. Note that as pointed out in the previous section we are only expressing our results for the scenario where $F^{(\ell)}_n = F^{(\ell)}$, that is the files within each library are of the same size. We have the following theorem.

Theorem 4.1. *Let $R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]})$ describe the memory-rate tradeoff as defined in* (4.4) *where $\alpha^{(\ell)}_n = \alpha^{(\ell)}$ for $n \in [1 : N_\ell]$ and for $\ell \in [1 : L]$. Then, we have*

$$R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]}) \leq \sum_{\ell=1}^L \alpha^{(\ell)} R^*(1, \frac{M_\ell}{\alpha^{(\ell)}}, 1, N_\ell)$$

where M_ℓ's are arbitrary non-negative numbers satisfying $\sum_{\ell=1}^L M_\ell = M$.

Proof. Consider Network $\mathcal{A}$ with parameters $(L, \{\alpha^{(\ell)}\}_{\ell=1}^{L}, N_{[1:L]})$ and L networks $\mathcal{B}^{(\ell)}$ with parameters $(1, 1, N_\ell)$, $\ell \in [1:L]$. Suppose for every $\ell \in [1:L]$ a memory-rate pair $(\frac{FM_\ell}{F^{(\ell)}}, R_\ell) = (\frac{M_\ell}{\alpha^{(\ell)}}, R_\ell)$ is achievable for Network $\mathcal{B}^{(\ell)}$ with files $W_n^{(\ell)}$. We will prove that $(M, \sum_{\ell=1}^{L} \alpha^{(\ell)} R_\ell)$ is achievable for Network $\mathcal{A}$. Fix some $\epsilon > 0$. By definition of achievability for Network $\mathcal{B}^{(\ell)}$, $\ell \in [1:L]$ there exist caching functions

$$\phi_k^{(\ell)} : [1:2^{F^{(\ell)}}]^{N_\ell} \to [1:2^{\lfloor F^{(\ell)} \frac{M_\ell}{\alpha^{(\ell)}} \rfloor}]\ ,\ \forall k \in [1:K]$$

encoding functions

$$\psi^{(\ell)}_{d^{(\ell)}_{[1:K]}} : [1:2^{F^{(\ell)}}]^{N_\ell} \to [1:2^{\lfloor F^{(\ell)} R_\ell \rfloor}]\ ,\ \forall d^{(\ell)}_{[1:K]} \in [1:N_\ell]^K$$

and decoding functions

$$\begin{aligned} \mu^{(\ell)}_{d^{(\ell)}_{[1:K]},k} \quad : \quad & [1:2^{\lfloor F^{(\ell)} R_\ell \rfloor}] \times [1:2^{\lfloor F^{(\ell)} \frac{M_\ell}{\alpha^{(\ell)}} \rfloor}] \to [1:2^{F^{(\ell)}}], \\ & \forall k \in [1:K], d^{(\ell)}_{[1:K]} \in [1:N_\ell]^K \end{aligned}$$

such that the estimates

$$\hat{W}_{d^{(\ell)}_{[1:K]},k,\ell} \quad = \quad \mu^{(\ell)}_{d^{(\ell)}_{[1:K]},k}(X^{(\ell)}_{d^{(\ell)}_{[1:K]}}, Z_k)$$

satisfy

$$\max_{d^{(\ell)}_{[1:K]} \in [1:N_\ell]^K} \max_{k \in [1:K]} \mathbb{P}(\hat{W}_{d^{(\ell)}_{[1:K]},k,\ell} \neq W^{(\ell)}_{d^{(\ell)}_k}) < \epsilon$$

for $F = \frac{F^{(\ell)}}{\alpha^{(\ell)}}$ sufficiently large.
Now for Network $\mathcal{A}$ we define the caching functions

$$\phi_k(\{W^{(\ell)}_{[1:N_\ell]}\}_{\ell=1}^{L}) \triangleq [\phi_k^{(1)}(W^{(1)}_{[1:N_1]}), \ldots, \phi_k^{(L)}(W^{(L)}_{[1:N_L]})],\ \forall k$$

the encoding functions

$$\begin{aligned} & X_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L}} = \psi_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L}}(\{W^{(\ell)}_{[1:N_\ell]}\}_{\ell=1}^{L}) \triangleq \\ & [\psi^{(1)}_{d^{(1)}_{[1:K]}}(W^{(1)}_{[1:N_1]}), \ldots, \psi^{(L)}_{d^{(L)}_{[1:K]}}(W^{(L)}_{[1:N_L]})],\ \ \forall \{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L} \end{aligned}$$

and the decoding functions

$$\begin{aligned} & \mu_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L},k,i}(X_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L}}, Z_k) \\ & \triangleq \mu^{(i)}_{d^{(i)}_{[1:K]},k}(\psi^{(i)}_{d^{(i)}_{[1:K]}}(W^{(i)}_{[1:N_i]}), \phi_k^{(i)}(W^{(i)}_{[1:N_i]})),\ \ \forall i, k, \{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L}. \end{aligned}$$

The probability of error of this caching scheme is thus

$$\begin{aligned}
&\max_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L} \in \prod_{\ell=1}^{L}[1:N_\ell]^K} \ \max_{k\in[1:K], i\in[1:L]} \mathbb{P}(\hat{W}_{\{d^{(\ell)}_{[1:K]}\}_{\ell=1}^{L},k,i} \neq W^{(i)}_{d_k^{(i)}}) \\
&= \max_{i\in[1:L]} \ \max_{d^{(i)}_{[1:K]}\in[1:N_i]^K} \ \max_{k\in[1:K]} \mathbb{P}(\hat{W}_{d^{(i)}_{[1:K]},k,i} \neq W^{(i)}_{d_k^{(i)}}) \\
&\leq \epsilon.
\end{aligned}$$

Furthermore, this scheme has a rate equal to $\frac{1}{F}\sum_{\ell=1}^{L} F^{(\ell)} R_\ell = \sum_{\ell=1}^{L} \alpha^{(\ell)} R_\ell$ and a memory of size $\frac{1}{F}\sum_{\ell=1}^{L} \frac{F^{(\ell)} M_\ell}{\alpha^{(\ell)}} = M$. Therefore, the memory-rate pair $(M, \sum_{\ell=1}^{L} \alpha^{(\ell)} R_\ell)$ is achievable for Network $\mathcal{A}$ which proves the theorem.

□

4.3.2 Converse

Consider Network $\mathcal{A}$ with L libraries. Roughly speaking, we will prove that any caching strategy for Network $\mathcal{A}$ can also be used for Network $\mathcal{B}$ which has a single library consisting of files that are concatenation of files from different libraries of A. Intuitively, this is done by breaking each file on Network $\mathcal{B}$ into its subfiles and assuming that each subfile belongs to a separate library. This is formally stated in the next theorem.

Theorem 4.2. *Let $R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^{L}, N_{[1:L]})$ describe the memory-rate trade-off as defined in (4.4) where $\alpha_n^{(\ell)} = \alpha^{(\ell)}$ for $n \in [1 : N_\ell]$ and for $\ell \in [1 : L]$. Furthermore, assume without loss of generality that $N_1 \leq N_2 \leq \cdots \leq N_L$. Then, we have*

$$\begin{aligned}
&R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^{L}, N_{[1:L]}) \geq \\
&R^*(1, M, \beta_{[1:N_L]}, N_L).
\end{aligned}$$

The coefficients β_n for $n \in [1 : N_L]$ are defined as

$$\beta_n = \frac{\sum_{i=f(n)}^{L} \alpha^{(i)}}{\sum_{\ell=1}^{L} \alpha^{(\ell)} N_\ell} N_L$$

where $f(n)$ returns the smallest $j \in [1 : L]$ such that $n \leq N_j$.

Proof. Consider Network $\mathcal{A}$ with parameters $(L, \{\alpha^{(\ell)}\}_{\ell=1}^{L}, N_{[1:L]})$ and Network $\mathcal{B}$ with parameters $(1, \beta_{[1:N_L]}, N_L)$. Suppose a memory-rate pair (R, M) is achievable for Network $\mathcal{A}$. We will prove that the same memory-rate pair (R, M) is also achievable for Network $\mathcal{B}$. We represent the files on Network $\mathcal{B}$ by W_n which are of size $\beta_n F = \frac{\sum_{i=f(n)}^{L} \alpha^{(i)}}{\sum_{\ell=1}^{L} \alpha^{(\ell)} N_\ell} F N_L$ for $n \in [1 : N_L]$. We break each W_n into disjoint subfiles

$$W_n = [W_n^{(f(n))}, W_n^{(f(n)+1)}, \ldots, W_n^{(L)}] \tag{4.5}$$

where $W_n^{(\ell)}$ is of size $\alpha^{(\ell)}\frac{F}{\sum_{\ell=1}^{L}\alpha^{(\ell)}N_\ell}N_L$. Fix some $\epsilon > 0$. By definition of achievability for Network $\mathcal{A}$ with files $\{W_{[1:N_\ell]}^{(\ell)}\}_{\ell=1}^{L}$ there exist caching functions ϕ_k, encoding functions $\psi_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}}$ and decoding functions $\mu_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i}$ as in equations (4.1),(4.2),(4.3) such that for any request vector $\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}$ and for F large enough, each user can recover his desired files with probability of error bounded by ϵ.

Now for Network $\mathcal{B}$ we define the caching functions

$$\phi_k^{'}(W_{[1:N_L]}) \triangleq \phi_k(\{W_{[1:N_\ell]}^{(\ell)}\}_{\ell=1}^{L}),\ \forall k \in [1:K]$$

the encoding functions

$$\psi_{d'_{[1:K]}}^{'}(W_{[1:N_L]}) \triangleq \psi_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}}(\{W_{[1:N_\ell]}^{(\ell)}\}_{\ell=1}^{L}),\ \forall d'_{[1:K]} \in [1:N_L]^K$$

and the decoding functions

$$\begin{gathered}\mu_{d'_{[1:K]},k}^{'} \triangleq \\ [\mu_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,f(d'_k)}, \mu_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,f(d'_k)+1}, \ldots, \mu_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,L}] \\ \forall k \in [1:K], d'_{[1:K]} \in [1:N_L]^K\end{gathered} \tag{4.6}$$

where $d_k^{(\ell)} \triangleq \min(d'_k, N_\ell)$. Note that if $N_\ell < d'_k$, then $d_k^{(\ell)}$ is a dummy request and the reconstructed $W_{d_k^{(\ell)}}^{(\ell)}$ will be discarded as visible from equation (4.6). The probability of error of this caching scheme is less than $L\epsilon$

$$\begin{aligned}
&\max_{d'_{[1:K]}\in[1:N_L]^K}\ \max_{k\in[1:K]} \mathbb{P}(\hat{W}'_{d'_{[1:K]},k} \neq W_{d'_k}) \\
&= \max_{\substack{(d'_{[1:K]})\in[1:N_L]^K \\ d_k^{(\ell)}=\min(d'_k,N_\ell)}}\ \max_{k\in[1:K]} \mathbb{P}\left(\bigvee_{i=f(d'_k)}^{L} (\hat{W}_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i} \neq W_{d_k^{(i)}}^{(i)})\right) \\
&\leq \max_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}\in\prod_{\ell=1}^{L}[1:N_\ell]^K}\ \max_{k\in[1:K]} \mathbb{P}\left(\bigvee_{i=1}^{L}(\hat{W}_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i} \neq W_{d_k^{(i)}}^{(i)})\right) \\
&\leq L \max_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L}\in\prod_{\ell=1}^{L}[1:N_\ell]^K}\ \max_{k\in[1:K],i\in[1:L]} \mathbb{P}\left(\hat{W}_{\{d_{[1:K]}^{(\ell)}\}_{\ell=1}^{L},k,i} \neq W_{d_k^{(i)}}^{(i)}\right) \\
&\leq L\epsilon.
\end{aligned}$$

Since we are reusing the same caching, encoding and decoding functions, the memory-rate pair (R, M) is the same for both Networks $\mathcal{A}$ and $\mathcal{B}$. Since this is one achievable strategy for Network $\mathcal{B}$, we have $R^*(1, M, \beta_{[1:N_L]}, N_L) \leq R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^{L}, N_{[1:L]})$. □

4.4 Optimality Results

4.4.1 Libraries with Equal Number of Files

Suppose we have $N_\ell = N$ for $\ell \in [1:L]$. That is, all the libraries keep hold of equal number of files. We will show that if in Theorem 4.1 the M_ℓ's are chosen proportional to $F^{(\ell)}$, our inner and outer bounds will match. This implies that the simple memory-sharing strategy proposed in Theorem 4.1 is indeed optimal and cannot be outperformed by coding across files from different libraries.

Theorem 4.3. *Let $R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]})$ describe the memory-rate tradeoff as defined in* (4.4) *where $\alpha_n^{(\ell)} = \alpha^{(\ell)}$ for $n \in [1:N_\ell]$ and for $\ell \in [1:L]$. Suppose we have $N_\ell = N$ for $\ell \in [1:L]$ and $M_\ell = \alpha^{(\ell)} M$. Then*

$$R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]}) = \sum_{\ell=1}^{L} \alpha^{(\ell)} R^*(1, \frac{M_\ell}{\alpha^{(\ell)}}, 1, N_\ell). \tag{4.7}$$

Proof. From Theorem 4.2 we have

$$\begin{aligned} &R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]}) \geq \\ &R^*(1, M, \beta_{[1:N_L]}, N_L) = R^*(1, M, 1, N). \end{aligned} \tag{4.8}$$

On the other hand, from Theorem 4.1 we know that

$$\begin{aligned} &R^*(L, M, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]}) \leq \\ &\sum_{\ell=1}^{L} \alpha^{(\ell)} R^*(1, \frac{\alpha^{(\ell)} M}{\alpha^{(\ell)}}, 1, N_\ell) = \\ &\sum_{\ell=1}^{L} \alpha^{(\ell)} R^*(1, M, 1, N) = R^*(1, M, 1, N). \end{aligned} \tag{4.9}$$

The claim follows from (4.8) and (4.9). □

4.4.2 Libraries with Arbitrary Number of Files

In this section we find the optimal memory-sharing strategy when the number of files in different libraries are not necessarily equal. Whether this optimal memory-sharing strategy is globally optimal or not, is a question that we have no answer for at this point (but we conjecture that it is).

We know that the memory-rate tradeoff for a network with one library is convex. We will further assume that it is piecewise linear and has the following form

$$R^*(1, M, 1, N) = \begin{cases} \zeta_0^{(N)} - \gamma_0^{(N)} M & \text{if } 0 \leq M < \theta_1^{(N)} \\ \zeta_1^{(N)} - \gamma_1^{(N)} M & \text{if } \theta_1^{(N)} \leq M < \theta_2^{(N)} \\ \dots & \\ \zeta_{r-1}^{(N)} - \gamma_{r-1}^{(N)} M & \text{if } \theta_{r-1}^{(N)} \leq M < N \end{cases} \tag{4.10}$$

where $\gamma_0^{(N)} > \cdots > \gamma_{r-1}^{(N)} > 0$ (due to convexity) and $\zeta_{i-1}^{(N)} - \gamma_{i-1}^{(N)}\theta_i^{(N)} = \zeta_i^{(N)} - \gamma_i^{(N)}\theta_i^{(N)}$ for $i \in [1:r]$ (due to continuity) and r naturally depends on N but to simplify the notation we have used $r = r_N$. Also define $\gamma_{-1}^{(N)} = \infty$, $\gamma_r^{(N)} = 0$, $\zeta_{-1}^{(N)} = \infty$, $\zeta_r^{(N)} = 0$, $\theta_0^{(N)} = 0$, $\theta_r^{(N)} = N$ and $\theta_{r+1}^{(N)} = \infty$.

Note that if $R^*(1, M, 1, N)$ is not piecewise linear, we can readily generalize our results by approximating $R^*(1, M, 1, N)$ with a piecewise linear function of arbitrarily large number of pieces. We can now describe the optimal memory-sharing strategy for the L-library setting.

Theorem 4.4. *Suppose the memory-rate tradeoff for a network with parameters $(1, 1, N)$ has the general form of (4.10) with r_N segments. Then there exists an optimal memory-sharing strategy for a network with parameters $(L, \{\alpha^{(\ell)}\}_{\ell=1}^L, N_{[1:L]})$, i.e. a solution to*

$$M_{[1:L]}^* = \operatorname*{arg\,min}_{M_{[1:L]}, \sum_{\ell=1}^L M_\ell = M} \sum_{\ell=1}^{L} \alpha^{(\ell)} R^*(1, \frac{M_\ell}{\alpha^{(\ell)}}, 1, N_\ell)$$

that satisfies the following. There exist an $\hat{\ell} \in [1:L]$ and L integers $0 \le i_\ell \le r_{N_\ell}$, $\ell \in [1:L]$ such that

$$M_\ell^* = \begin{cases} \theta_{i_\ell}^{(N_\ell)} \alpha^{(\ell)} & \text{if } \ell \neq \hat{\ell} \\ \theta_{i_\ell}^{(N_\ell)} \alpha^{(\ell)} + M_{rem} & \text{if } \ell = \hat{\ell}. \end{cases}$$

where $0 \le M_{rem} < \alpha^{(\hat{\ell})}(\theta_{i_{\hat{\ell}}+1}^{(N_{\hat{\ell}})} - \theta_{i_{\hat{\ell}}}^{(N_{\hat{\ell}})})$ and

$$\frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}} \le \frac{\gamma_{i_{\ell'}-1}^{(N_{\ell'})}}{\alpha^{(\ell')}}, \ \forall \ell, \ell' \in [1:L]$$

and

$$\frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}} \le \frac{\gamma_{i_{\hat{\ell}}}^{(N_{\hat{\ell}})}}{\alpha^{(\hat{\ell})}}, \ \forall \ell \in [1:L]. \tag{4.11}$$

Proof. Assume there exist $\ell, \ell' \in [1:L]$ such that $\ell \neq \ell'$ and $M_\ell^* = \theta_{i_\ell}^{(N_\ell)}\alpha^{(\ell)} + M_{\text{rem}}$ and $M_{\ell'}^* = \theta_{i_{\ell'}}^{(N_{\ell'})}\alpha^{(\ell')} + M'_{\text{rem}}$ and $M_{\text{rem}} \neq 0$ and $M'_{\text{rem}} \neq 0$. Assume without loss of generality that $\frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}} \ge \frac{\gamma_{i_{\ell'}}^{(N_{\ell'})}}{\alpha^{(\ell')}}$. Now we set

$$\begin{aligned} \delta &= \min(\alpha^{(\ell)}(\theta_{i_\ell+1}^{(N_\ell)} - \theta_{i_\ell}^{(N_\ell)}) - M_{\text{rem}}, M'_{\text{rem}}), \\ M_\ell^* &\leftarrow M_\ell^* + \delta, \\ M_{\ell'}^* &\leftarrow M_{\ell'}^* - \delta. \end{aligned}$$

This moves either of M_ℓ^* or $M_{\ell'}^*$ (or both) to a corner point (that is, either of M_{rem} or M'_{rem} will be zero). Furthermore, this changes the total rate by

$$\Delta R = (\frac{\gamma_{i_{\ell'}}^{(N_{\ell'})}}{\alpha^{(\ell')}} - \frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}})\delta \le 0.$$

Therefore, there exists an optimal solution for which at most one of the libraries has $M_{\text{rem}} \neq 0$. We call this library $\hat{\ell}$. Next assume there exists a pair $\ell \neq \ell' \in [1:L] \backslash \{\hat{\ell}\}$ for which $\frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}} > \frac{\gamma_{i_{\ell'}-1}^{(N_{\ell'})}}{\alpha^{(\ell')}}$. This time we define $\delta = \min(\alpha^{(\ell)}(\theta_{i_\ell+1}^{(N_\ell)} - \theta_{i_\ell}^{(N_\ell)}), \alpha^{(\ell')}(\theta_{i_{\ell'}}^{(N_{\ell'})} - \theta_{i_{\ell'}-1}^{(N_{\ell'})}))$. Again setting $M_\ell^* \leftarrow M_\ell^* + \delta$ and $M_{\ell'}^* \leftarrow M_{\ell'}^* - \delta$ results in $\Delta R < 0$. Finally assume $\frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}} > \frac{\gamma_{i_{\hat{\ell}}}^{(N_{\hat{\ell}})}}{\alpha^{(\hat{\ell})}}$ for some ℓ. We can set $\delta = \min(\alpha^{(\ell)}(\theta_{i_\ell+1}^{(N_\ell)} - \theta_{i_\ell}^{(N_\ell)}), M_{\text{rem}})$ and $M_\ell^* \leftarrow M_\ell^* + \delta$ and $M_{\hat{\ell}}^* \leftarrow M_{\hat{\ell}}^* - \delta$ which results in $\Delta R < 0$ unless if $M_{\text{rem}} = 0$, in which case we simply choose the library with the largest $\frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}}$ to be $\hat{\ell}$. □

The solution described by Theorem 4.4 can be found in an incremental way. Assume that initially the size of the cache is zero and we gradually increase it up to $M = \sum_{\ell=1}^{L} \alpha^{(\ell)} N_\ell$. At any point we must decide how much of the cache should be allocated to each library. At the beginning it is advantageous to assign all the cache to library ℓ with the largest $\frac{\gamma_0^{(N_\ell)}}{\alpha^{(\ell)}}$, since this reduces the total delivery rate by the largest factor. This is the library which is called $\hat{\ell}$ in the theorem. This assignment continues until this library reaches the corner point $M_{\hat{\ell}} = \alpha^{(\hat{\ell})} \theta_1^{(N_{\hat{\ell}})}$. At this point $\hat{\ell}$ is re-initialized as the library with the largest right-slope and the process continues. This procedure is summarized in Algorithm 3.

As a final remark, we conjecture that this memory-sharing strategy is again globally optimal and that our converse bound is tight in this general case. In other words, we conjecture

$$\sum_{\ell=1}^{L} \alpha^{(\ell)} R^*(1, \frac{M_\ell^*}{\alpha^{(\ell)}}, 1, N_\ell) = R^*(1, M, \beta_{[1:N_L]}, N_L).$$

This would imply that even in the general case, there is no gain from coding across files from different libraries and memory-sharing suffices for minimizing the delivery rate.

Algorithm 3 Optimal Memory Allocation for L libraries

```
Set AllocM$= 0$.
Set $M_\ell = 0$ for $\ell \in [1:L]$.
Set $i_\ell = 0$, for $\ell \in [1:L]$.

while AllocM$< M$ do
    Find the library $\hat{\ell}$ that has the largest right slope
        $\hat{\ell} = \arg\max_{\ell \in [1:L]} \frac{\gamma_{i_\ell}^{(N_\ell)}}{\alpha^{(\ell)}}.$
    Set $\delta = \frac{\gamma_{i_{\hat{\ell}}}^{(N_{\hat{\ell}})}}{\alpha^{(\hat{\ell})}} \left(\theta_{i_{\hat{\ell}}+1}^{(N_{\hat{\ell}})} - \theta_{i_{\hat{\ell}}}^{(N_{\hat{\ell}})}\right)$.
    if $M \geq$AllocM $+\delta$ then
        $M_{\hat{\ell}} = M_{\hat{\ell}} + \delta$.
        AllocM =AllocM $+ \delta$.
        $i_{\hat{\ell}} = i_{\hat{\ell}} + 1$.
    else
        $M_{\hat{\ell}} = M_{\hat{\ell}} + M-$AllocM.
        AllocM$= M$.
    end if
end while
Output $M_\ell$ for $\ell \in [1:L]$.
```

5 A Generic Approach to Distributed Storage Systems

A variation of the classical distributed storage model will be introduced and analyzed in this chapter. In many ways, the new formulation can be viewed as a straightforward generalization of the classical DSS discussed in the preliminaries, whereas from other perspectives, it is more limited. This model should be of interest in studying DSSs where failures occur rarely. As a result, the traffic generated during the repair process is negligible and should not be taken as a criterion for evaluating the performance of the system. Instead, we will focus on relaxing several idealized and symmetrized properties of the classical model which, as will be discussed shortly, could be unrealistic in practice. We will first describe the new model and highlight its differences with the classical one, and next motivate it.

Let us start by looking at the classical DSS from a new perspective. Ignoring the repair requirement, this model [1] can be viewed as a complete k-uniform hyper-graph with K vertices which represent the servers and $\binom{K}{k}$ hyper-edges representing the users which are connected to the corresponding set of servers. Each vertex of the hyper-graph is equipped with a memory of size M and each hyper-edge should be able to recover a specific file A from the memories of the vertices associated with him. A minimum-storage code minimizes the total required memory MK under these constraints. As we are assuming the repair events occur rarely, this becomes the sole criterion by which we judge the performance of this DSS. Therefore, a simple MDS (maximum distance separable) code can achieve the optimal performance.

Now, let us assume instead that we have an arbitrary hyper-graph defined over K vertices and a set of N independent files $\{A_1, \dots, A_N\}$. Each hyper-edge of the graph is colored by some $c \in \{1, \dots, N\}$. We have an entire memory budget of M which we distribute among different vertices of the hyper-graph. To each vertex i we assign a memory of size M_i such that $\sum_i M_i \leq M$.

Each server stores a function h_i of the files in his memory which satisfies $H(h_i) \leq M_i$. These functions must be designed so that if $S = \{s_1, \ldots, s_\ell\}$ is a hyper-edge of the graph colored with c, then $H(A_c|h_{s_1}, \ldots, h_{s_\ell}) = 0$. The question is what is the minimum total memory budget M which allows us to accomplish this task for a particular colored hyper-graph. We refer to this as the Generic Distributed Storage Problem (GDSP). We will establish close connections between this problem and well known problems in the network information theory literature. These connections also imply that the problem cannot be easily solved in its full generality. Therefore, we focus in this work on two sub-models. Firstly, and for most of this chapter, we will study a graph (a hyper-graph where size of each hyper-edge is two) where edges are colored arbitrarily. In Section 5.1, we will propose an achievability strategy for a - practically motivated - subclass of such graphs which we refer to as “smoothly colored graphs”. We will show that under certain constraints our strategy is optimal. Secondly, we will briefly investigate a hyper-graph in the presence of only one file. For such a hyper-graph in Section 5.2 we will characterize the minimum total required memory as the solution to an LP.

To motivate the extension of the distributed storage problem considered in this work, let us start by reconsidering the classical version. There, a key requirement is that *every* sufficiently large subset of the servers must enable full recovery of the entire file. For several scenarios of potential practical interest, this requirement could be unnecessarily stringent. Consider for example a setting with different classes of servers. Some servers could be more powerful than others, or more reliable. Then, a natural consideration would be to suppose that every file recovery would always involve at least one of the more powerful servers. In other words, one would not impose a requirement that a subset consisting of only less powerful servers must enable file recovery. This naturally leads to a more general hyper-graph model, beyond the uniform complete ones studied in the classical setup. A practical framework where such a combination of more and less powerful servers might appear are caching networks for content distribution. In such networks, there are auxiliary servers that help speed up data delivery. However, it will generally not be possible to fully recover the desired content only from auxiliary servers. Rather, an additional call to one of the (more powerful, but typically overloaded) main servers will be necessary.

The second generalization of our work concerns the file itself: In the classical problem, there is a single file. In our extension, we allow for several files, and each user is requesting only one of the files. Again, such a scenario is of potential practical interest, for example, in a geographical setting: let us suppose for the sake of argument that the servers are geographically distributed in a large area, and let us envision content distribution that is location-specific, as in many of the commercial video distribution services. Here, some servers will serve only one geographical sub-area while other servers will serve multiple, leading to a hyper-graph where each hyper-edge will potentially seek to recover a different file, specific to the geographical location.

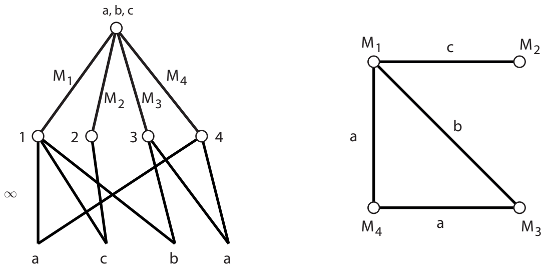


Figure 5.1: How large does $M_1 + M_2 + M_3 + M_4$ need to be, such that the network (left) admits a rate of 1? Equivalently, what is the solution to the GDSP defined over the graph on the right?

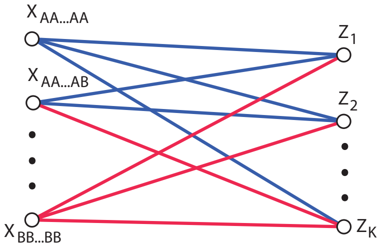


Figure 5.2: The coded caching problem [2] can be viewed as GDSP defined over a complete bipartite graph. Blue edges must be able to recover file A and the red ones, file B.

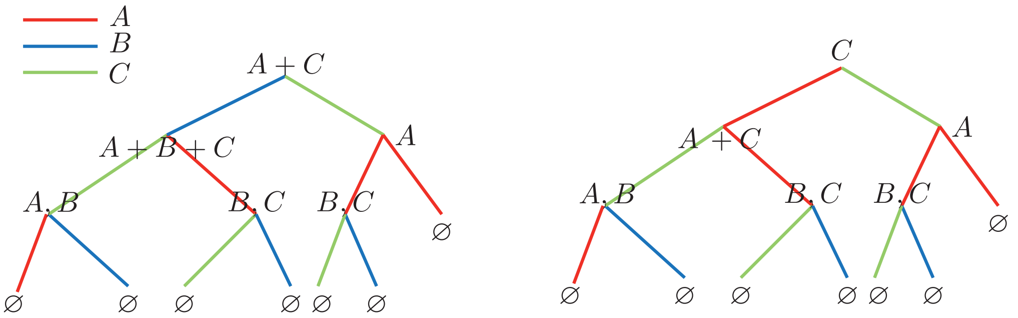


Figure 5.3: Left: GDSP over a tree. The root has to solve an instance of Index Coding with Coded Side-Information [41], assuming his children have stored the correct functions. Right: the problem cannot be solved greedily, by looking at one's own subtree (unless the root has a depth of 2 or less).

Connections with the literature

GDSP can be seen as a special instance of the single source network information flow problem. The source is connected to K intermediate nodes with links with capacities $M_1, \ldots, M_K$ respectively. These nodes represent the vertices in the GDSP. We have $|E|$ sinks corresponding to the hyper-edges of the GDSP. For any hyper-edge $S = \{s_1, \ldots, s_\ell\}$, we connect all the intermediate nodes

$s_1, \ldots, s_\ell$ to the corresponding sink with links with infinite capacities. Our problem is equivalent to finding the minimum sum rate $\sum_{i=1}^{K} M_i$ such that, given proper manipulation of the data at the intermediate nodes, each sink can recover its desired message at a rate of 1 bit per second (see Figure 5.1 for an illustration). For the network in Figure 5.1 one can use Theorem 5.2 to show that the minimum required total memory is $M^* = 3$.

The canonical coded caching model [2] can be equivalently represented by a GDSP defined over a complete bipartite graph where on one side we have K vertices standing for the user memories and on the other side we have N^K vertices representing the delivery messages (see Figure 5.2). While in general one can study the overall trade-off among $(M_1, \ldots, M_K, R_1, \ldots, R_{N^K})$, the analysis in [2] is limited to $M_i = M$ and $R_j = R$ whereas we are interested in minimizing $\sum_i M_i + \sum_j R_j$.

If we restrict GDSP to a tree, it is easy to show that there exists an optimal storage scheme where the leaves do not store anything. In such a scheme, naturally, the parents of the leaves carry the burden of recovering the files requested by the edges between them and the leaves. For the parents of the parents, the problem turns into an instance of the index coding. In fact, this simple argument shows that any instance of the index coding problem with uncoded side information can be seen as a special case of GDSP defined over a tree of diameter at most 4. GDSP defined over a tree of diameter 6 is closely connected to the problem of index coding with coded side-information [41]. See Figure 5.3 for an illustration.

There are also close connections between our work and Femto Caching [42] where caching over an arbitrary hyper-graph is studied. The vertices and edges in GDSP correspond to femto-cells and users in the Femto Caching model, respectively. There are however several differences. To mention a few, in the Femto Caching model: all the caches are assumed to be of the same size; the file requests per hyper-edge are unknown and are modeled by a popularity distribution (which does not vary across different users) and last but not least coding across different files is not permitted and the main quest is to find the best storage strategy under this restriction.

5.1 Graphs with Multiple Files

Let us first formally define the problem. We have a set of N independent files $\{A_1, \ldots, A_N\}$ where A_i consists of F independent and uniformly distributed symbols over $\mathbb{F}_q$ where q is a sufficiently large prime number. We have a colored graph $G = (\mathcal{V}, E)$ where $\mathcal{V} = \{v_1, \ldots, v_K\}$ is a set of K vertices and E is a set of tuples of the form $(\{i,j\}, c)$ where $i, j \in \{1, \ldots, K\}$, $i \neq j$ and $c \in \mathcal{N} = \{1, \ldots, N\}$. For any $\{i,j\}$ there is at most one such tuple in E, that is, if $(\{i,j\}, c) \in E$ and $(\{i',j'\}, c') \in E$ and $c \neq c'$ then $\{i,j\} \neq \{i',j'\}$. The parameter c specifies the color of the edge or the file that the edge is interested in. Each vertex i of the graph is equipped with a memory of size $M_{i,F}F$ where he stores a function the files, that is, $h_{i,F} = h_{i,F}(A_1, \ldots, A_N)$ such that both conditions below are satisfied.

$$H(h_{i,F}) \leq M_{i,F}F \tag{5.1}$$

$$H(A_c|h_{i,F}, h_{j,F}) = 0 \text{ for all } (\{i,j\}, c) \in E. \tag{5.2}$$

Note that all the entropy terms are calculated base q. For a given colored graph G, we say that a memory allocation $(M_{1,F}, \ldots, M_{K,F})$ is valid if there exists functions $h_{i,F}(\cdot)$ that satisfy (5.1) and (5.2). In this case, we call $(h_{1,F}(\cdot), \ldots, h_{1,F}(\cdot))$ a valid assignment too. We say that a normalized sum rate of M is achievable if there exists a sequence $\{(M_{1,F}, \ldots, M_{K,F})\}_{F=1}^{\infty}$ such that $(M_{1,F}, \ldots, M_{K,F})$ is a valid assignment for all F and

$$\lim_{F\to\infty} \sum_{i=1}^{K} M_{i,F} \leq M. \tag{5.3}$$

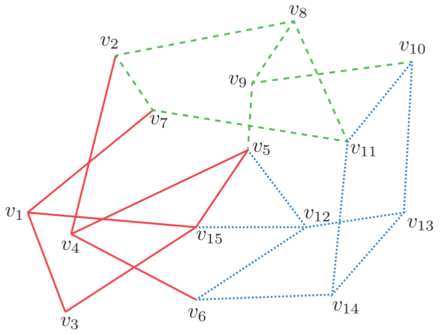
Our goal is to find the minimum normalized sum achievable rate M^* for a given colored graph G. That is $M^* = \inf\{M \Big| M \text{ is achievable}\}$. When clear from the context, we omit the subscript F from $M_{i,F}$ and $h_{i,F}$ to simplify the notation.

Motivated by the arguments presented earlier in this chapter, let us now introduce a model which we refer to as a "smoothly colored graph". Intuitively, a smoothly colored graph is one that can be "partitioned" into several clusters each of which representing a certain geographical location. The edges connecting the vertices within each cluster are colored differently from the other clusters, whereas the edges that connect vertices from two different clusters can be colored similarly to either of the two clusters. Such cross edges represent users which have access to servers from both clusters. Let us define this concept more formally.

Definition 5.1 (Smoothly Colored Graphs)**.** *We say that a graph $G = (\mathcal{V}, E)$ is smoothly colored with respect to a partitioning $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_L$ of $\mathcal{N}$ if the set $\mathcal{V}$ can be partitioned into L subsets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_L$ such that if $v_i, v_j \in \mathcal{V}_\ell$ and $(\{i,j\}, c) \in E$ then $c \in \mathcal{N}_\ell$ and if $v_i \in \mathcal{V}_\ell$ and $v_j \in \mathcal{V}_{\ell'}$ and $(\{i,j\}, c) \in E$ then $c \in \mathcal{N}_\ell \cup \mathcal{N}'_\ell$. For $i, j \in \{1, 2, \ldots, L\}$, we represent by $\mathcal{F}_{i,j}$ the subset of vertices in $\mathcal{V}_j$ which are connected to at least one vertex outside of $\mathcal{V}_j$ with an edge colored with some $c \in \mathcal{N}_i$. Formally,*

$$\begin{aligned} \mathcal{F}_{i,j} \;\; = \;\; & \Big\{ v \in \mathcal{V}_j \Big| \exists u \notin \mathcal{V}_j, \; c \in \mathcal{N}_i \;, \; s.t. \;\; (\{u,v\}, c) \in E \Big\}, \\ & \forall i,j \in \{1, \ldots, L\}. \end{aligned}$$

An example of a smoothly colored graph with three clusters and $|\mathcal{N}_1| = |\mathcal{N}_2| = |\mathcal{N}_3| = 1$ has been depicted in Figure 5.4.



$\mathcal{V}_1 = \{v_1, \ldots, v_6\}$
$\mathcal{V}_2 = \{v_7, \ldots, v_{10}\}$
$\mathcal{V}_3 = \{v_{11}, \ldots, v_{15}\}$
$\mathcal{F}_{1,1} = \{v_1, v_3\}$
$\mathcal{F}_{2,1} = \{v_2, v_5\}$
$\mathcal{F}_{3,1} = \{v_5, v_6\}$
$\mathcal{F}_{1,2} = \{v_7\}$
$\mathcal{F}_{2,2} = \{v_7, v_8, v_9\}$
$\mathcal{F}_{3,2} = \{v_{10}\}$
$\mathcal{F}_{1,3} = \{v_{15}\}$
$\mathcal{F}_{2,3} = \{v_{11}\}$
$\mathcal{F}_{3,3} = \{v_{11}, v_{12}, v_{13}, v_{14}\}$

Figure 5.4: A smoothly colored graph with three clusters.

Our approach is to reduce the GDSP over a smoothly colored graph to several smaller instances of GDSP over subgraphs representing different clusters. This can be interesting for several reasons. Firstly, if each cluster is colored with only one color, we can use proposition 5.1 from Section 5.2 in order to provide an exact solution for each cluster and consequently, find the exact solution for the overall network. Secondly, even within the realm of linear codes (for a fixed F), the complexity of an exhaustive algorithm grows exponentially with N. Therefore, any preprocessing that reduces N can significantly improve the running time of the overall algorithm.

Suppose solve(G) is an optimal algorithm that given a graph G returns any valid assignment $(M_1^*, \ldots, M_K^*)$ for which $\sum_{i=1}^K M_i^* = M^*$. Consider now Algorithm 4 which given a graph G and an arbitrary partitioning of the colors into $\mathcal{N}_1, \ldots, \mathcal{N}_L$ returns SUP$(G, \mathcal{N}_1, \ldots, \mathcal{N}_L)$, a superposition of solve$(G_1), \ldots,$ solve(G_L) where G_ℓ is a subgraph of G which only retains the edges colored by $c \in \mathcal{N}_\ell$ and eliminates all the other edges.

Algorithm 4 Superposition Algorithm

```
Input: $G = (\mathcal{V}, E)$ and a partitioning of colors into $\mathcal{N}_1, \ldots, \mathcal{N}_L$.
Output: $(M_1, \ldots, M_K) = \mathrm{SUP}(G, \mathcal{N}_1, \ldots, \mathcal{N}_L)$
Construct the subgraphs $G_\ell = (\mathcal{V}, E_\ell)$ such that $(\{i,j\}, c) \in E_\ell$ if and only if $(\{i,j\}, c) \in E$ and $c \in \mathcal{N}_\ell$.
Let $(M_1^{(\ell)}, \ldots, M_K^{(\ell)}) = solve(G_\ell)$ for all $\ell \in \{1, \ldots, L\}$.
return $(\sum_{\ell=1}^L M_1^{(\ell)}, \ldots, \sum_{\ell=1}^L M_K^{(\ell)})$.
```

The following theorem tells us that for a smoothly colored graph with $|\mathcal{N}_i| = 1$, and under the constraint $\mathcal{F}_{k,j} \cap \mathcal{F}_{\ell,j} = \varnothing$ for $k \neq \ell$, Algorithm 4 returns an exact solution.

Theorem 5.1. *Suppose a graph $G = (\mathcal{V}, E)$ is smoothly colored with respect to $\mathcal{N}_1, \ldots, \mathcal{N}_L$ where $|\mathcal{N}_\ell| = 1$ for all ℓ. Suppose further that $\mathcal{F}_{k,j} \cap \mathcal{F}_{\ell,j} = \varnothing$ for all $j, k, \ell \in \{1, \ldots, L\}$, $k \neq \ell$. Let $(\hat{M}_1, \ldots, \hat{M}_K) = SUP(G, \mathcal{N}_1, \ldots, \mathcal{N}_L)$ and $(M_1^*, \ldots, M_K^*) = solve(G)$ and, $\hat{M} = \sum_{i=1}^K \hat{M}_i$ and $M^* = \sum_{i=1}^K M_i^*$. We have $\hat{M} = M^*$.*

Proof. Suppose $(h_1^*, \ldots, h_K^*)$ is a valid assignment for G which satisfies $H(h_\ell^*) \leq M_\ell^*$. Without loss of generality let us assume $\mathcal{N}_\ell = \{\ell\}$. Consider the following memory assignment for G_ℓ. For all $v_i \in \mathcal{V}_\ell \backslash \bigcup_{k\neq\ell} \mathcal{F}_{k,\ell}$ we set $M_i^{(\ell)} = M_i^*$. For all $v_i \in \mathcal{F}_{k,\ell}$ for $k \neq \ell$, we set $M_i^{(\ell)} = M_i^* - (1 - \min_j M_j^*)^+$ where the minimum is over all j such that $v_j \in \mathcal{V}_k$ and $(\{i,j\},k) \in E$. (Note that x^+ stands for $\max\{x, 0\}$). For all $v_i \in \mathcal{F}_{\ell,k}$ we set $M_i^{(\ell)} = (1 - \min_j M_j^*)^+$ where the minimum is over all j such that $v_j \in \mathcal{V}_\ell$ and $(\{i,j\},\ell) \in E$. Finally, for all $v_i \in \mathcal{V} \backslash \left(\bigcup_{k\neq\ell} \mathcal{F}_{\ell,k} \cup \mathcal{V}_\ell\right)$ we set $M_i^{(\ell)} = 0$. We first prove that this is a valid assignment for G_ℓ. Since $|\mathcal{N}_\ell| = 1$, by proposition 5.1 we only need to prove that $M_i^{(\ell)} + M_j^{(\ell)} \geq 1$ if $(\{i,j\},\ell) \in E$. We consider several different cases.

- $i, j \in \mathcal{V}_\ell \backslash \bigcup_{k\neq\ell} \mathcal{F}_{k,\ell}$. In this case we have $M_i^{(\ell)} + M_j^{(\ell)} = M_i^* + M_j^*$. Since $(M_1^*, \ldots, M_K^*)$ is a valid assignment for G, we must have that $M_i^* + M_j^* \geq 1$, by proposition 5.1.

- $i \in \mathcal{V}_\ell \backslash \bigcup_{k\neq\ell} \mathcal{F}_{k,\ell}$ and $j \in \mathcal{F}_{\ell,k}$ for some $k \neq \ell$.

  We have $M_i^{(\ell)} + M_j^{(\ell)} = M_i^* + (1 - \min_{j'} M_{j'}^*)^+ \geq 1$ since by definition
  $$M_i^* \geq \min_{j': v_{j'} \in \mathcal{V}_\ell, (\{i,j\},\ell) \in E} M_{j'}^*.$$

- $i \in \mathcal{V}_\ell \backslash \bigcup_{k\neq\ell} \mathcal{F}_{k,\ell}$ and $j \in \mathcal{F}_{k,\ell}$. We can write $M_i^{(\ell)} + M_j^{(\ell)} = M_i^* + M_j^* - (1 - \min_{j'} M_{j'}^*)^+$. If $1 - \min_{j'} M_{j'}^* < 0$, we trivially have $M_i^{(\ell)} + M_j^{(\ell)} \geq 1$. Suppose $1 - \min_{j'} M_{j'}^* \geq 0$. For any $j' \in \mathcal{V}_k$ for which $(\{j, j'\}, k) \in E$ we have
  $$\begin{aligned} & F(M_i^* + M_j^* + M_{j'}^* - 1) \\ \geq \quad & H(h_i^*) + H(h_j^*) + H(h_{j'}^*) - F \\ \geq \quad & H(h_i^*, h_j^*, h_{j'}^*) - F \\ \overset{(\#)}{\geq} \quad & 2F + H(h_i^*, h_j^*, h_{j'}^* | A_k, A_\ell) - F \geq F \end{aligned}$$
  where (#) follows from the fact that $(h_1^*, \ldots, h_K^*)$ is a valid assignment and therefore, in graph G the triple $(h_i^*, h_j^*, h_{j'}^*)$ must be able to reproduce the files A_k and A_ℓ. Thus, $M_i^{(\ell)} + M_j^{(\ell)} = M_i^* + M_j^* + \min_{j'} M_{j'}^* - 1 \geq 1$.

- $i, j \in \mathcal{F}_{k,\ell}$ for some $k \neq \ell$. We can write $M_i^{(\ell)} + M_j^{(\ell)} = M_i^* - (1 - \min_{i'} M_{i'}^*)^+ + M_j^* - (1 - \min_{j'} M_{j'}^*)^+$. Again, if $1 - \min_{i'} M_{i'}^* < 0$ or $1 - \min_{j'} M_{j'}^* < 0$, the proof is simple. Suppose both these expressions are non-negative. For any $i', j' \in \mathcal{F}_{\ell,k}$ for which $(\{i, i'\}, k), (\{j, j'\}, k) \in E$

we have

$$\begin{aligned}
&F(M_i^* + M_{i'}^* + M_j^* + M_{j'}^* - 2)\\
\geq\ & H(h_i^*, h_{i'}^*) + H(h_j^*, h_{j'}^*) - 2F\\
\geq\ & 2F + H(h_i^*, h_{i'}^* | A_k) + H(h_j^*, h_{j'}^* | A_k) - 2F\\
\geq\ & H(h_i^*, h_j^*, h_{i'}^*, h_{j'}^* | A_k)\\
\geq\ & H(h_i^*, h_j^* | A_k) \geq F + H(h_i^*, h_j^* | A_k, A_\ell) \geq F.
\end{aligned}$$

And therefore, $M_i^{(\ell)} + M_j^{(\ell)} \geq 1$. Note that we might have $i' = j'$ but this does not affect the analysis above.

- $i \in \mathcal{F}_{k,\ell}$ and $j \in \mathcal{F}_{k',\ell}$ for $k \neq k'$. The analysis is very similar to the previous case.

Let $(M_1, \ldots, M_K) = (\sum_{\ell=1}^L M_1^{(\ell)}, \ldots, \sum_{\ell=1}^L M_L^{(\ell)})$ and let $M = \sum_{k=1}^K M_k$. Since $(M_1, \ldots, M_K)$ is a superposition of L different (not necessarily optimal) solutions for $G_1, \ldots, G_L$, clearly we have that $M \geq \hat{M}$. But we have $M = M^*$, because:

$$\begin{aligned}
M &= \sum_{k: v_k \in \mathcal{V} \setminus \bigcup_{i\neq j} \mathcal{F}_{i,j}} \sum_{\ell} M_k^{(\ell)} + \sum_{k: v_k \in \mathcal{F}_{i,j}, i\neq j} \sum_{\ell \in \{i,j\}} M_k^{(\ell)}\\
&= \sum_{k: v_k \in \mathcal{V} \setminus \bigcup_{i\neq j} \mathcal{F}_{i,j}} M_k^* + \sum_{k: v_k \in \mathcal{F}_{i,j}, i\neq j} (1 - \min_{i': v_{i'} \in \mathcal{V}_i \cdots} M_{i'}^*)^+\\
&+ \sum_{k: v_k \in \mathcal{F}_{i,j}, i\neq j} M_k^* - (1 - \min_{j': v_{j'} \in \mathcal{V}_i \cdots} M_{j'}^*)^+\\
&= \sum_{k: v_k \in \mathcal{V} \setminus \bigcup_{i\neq j} \mathcal{F}_{i,j}} M_k^* + \sum_{k: v_k \in \mathcal{F}_{i,j}, i\neq j} M_k^* = M^*.
\end{aligned}$$

Therefore, we established that $\hat{M} \leq M^*$. Trivially, we also have that $M^* \leq \hat{M}$ which proves $\hat{M} = M^*$. □

At a first glance, it is tempting to conjecture that the constraint $|\mathcal{N}_i| = 1$ imposed by Theorem 5.1 is not of fundamental importance and can be relaxed. Nevertheless, this is only partially true. As we will see shortly, even when $L = 2$, $|\mathcal{N}_1| = 1$ and $|\mathcal{N}_2| > 1$, Algorithm 4 may fail to return an optimal solution. On the bright side, we can still make the following claim.

Theorem 5.2. *Suppose a graph $G = (\mathcal{V}, E)$ is smoothly colored with respect to a partitioning $\mathcal{N}_1$ and $\mathcal{N}_2$ where $|\mathcal{N}_1| = 1$. Suppose further that $\mathcal{F}_{2,j} = \varnothing$ for $j = 1, 2$. Let $(\hat{M}_1, \ldots, \hat{M}_K) = SUP(G, \mathcal{N}_1, \mathcal{N}_2)$ and $(M_1^*, \ldots, M_K^*) = solve(G)$ and, $\hat{M} = \sum_{i=1}^K \hat{M}_i$ and $M^* = \sum_{i=1}^K M_i^*$. We have $\hat{M} = M^*$.*

Proof. Suppose $(h_1^*, \ldots, h_K^*)$ is a valid assignment for G which satisfies $H(h_\ell^*) \leq M_\ell^*$. Without loss of generality, assume $\mathcal{N}_1 = \{1\}$ and $\mathcal{N}_2 = \{2, \ldots, N\}$. Consider the following assignment for G_1. For all $v_i \in \mathcal{V}_1$ we set $M_i^{(1)} = M_i^*$ and for all $v_i \in \mathcal{F}_{1,2}$ we set $FM_i^{(1)} = I(h_i^*; A_1)$. We set $M_i^{(1)} = 0$ for all the other vertices. As for G_2, we propose the following assignment: for all $v_i \in \mathcal{V}_2$ let $FM_i^{(2)} = H(h_u^* | A_1 = \bar{a})$ where $\bar{a} \in \mathbb{F}_q^n$ is the solution to

$$\bar{a} = \arg\min_{a \in \mathbb{F}_q^n} \sum_{u \in \mathcal{V}_2} H(h_u^* | A_1 = a)$$

For all the remaining vertices we set $M_i^{(2)} = 0$. Note that without loss of generality, one can assume that $\bar{a}$ is a string of zeros. (If not, one can easily modify the functions h_u^* such that this property is held. This can be done without changing the required memory, and the recoverability of the files.)
Similar to the previous proof, we will show that these two are valid assignments. Firstly, $(M_1^{(2)}, \ldots, M_K^{(2)})$ is a valid assignment for G_2 because for all nodes in $\mathcal{V}_2$ we can store $h_u^{(2)} = h_u^*(\mathbf{0}, A_2, \ldots, A_N)$. We have $H(h_u^*(\mathbf{0}, A_2, \ldots, A_N)) = H(h_u^* | A_1 = \mathbf{0})$. For all $\{u, v\} \in \mathcal{V}_2$ where $(\{u, v\}, c) \in E$ for some $c \neq 1$, since:

$$H(A_c \Big| h_u^*(A_1, A_2, \ldots, A_N), h_v^*(A_1, A_2, \ldots, A_N)) = 0$$

we must have that

$$H(A_c \Big| h_u^*(a, A_2, \ldots, A_N), h_v^*(a, A_2, \ldots, A_N), A_1 = a) = 0$$

for all $a \in \mathbb{F}_q^n$ including $a = \mathbf{0}$. Therefore,

$$H(A_c \Big| h_u^*(\mathbf{0}, A_2, \ldots, A_N), h_v^*(\mathbf{0}, A_2, \ldots, A_N), A_1 = \mathbf{0}) = 0$$

and thus $H(A_c \Big| h_u^*(\mathbf{0}, A_2, \ldots, A_N), h_v^*(\mathbf{0}, A_2, \ldots, A_N)) = 0$ (because A_1 is independent of $(A_2, \ldots, A_N)$).

Secondly, $(M_1^{(1)}, \ldots, M_K^{(1)})$ is a valid assignment for G_1 because if $u \in \mathcal{V}_1$ and $v \in \mathcal{V}_2$ and $(\{u, v\}, 1) \in E$ then

$$\begin{aligned} F(M_u^{(1)} + M_v^{(1)}) &= FM_u^* + I(h_v^*; A_1) \\ &= H(h_u^*) + H(h_v^*) - H(h_v^* | A_1) \\ &\geq H(h_u^*, h_v^*) - H(h_v^*, h_u^* | A_1) \\ &= H(A_1) = F. \end{aligned}$$

Finally, since

$$\sum_{u \in \mathcal{V}_2} H(h_u^* | A_1 = \mathbf{0}) \leq \sum_{u \in \mathcal{V}_2} H(h_u^* | A_1)$$

it follows that

$$\begin{aligned} FM &= F(\sum_{u \in \mathcal{V}} (M_u^{(1)} + M_u^{(2)})) \\ &\leq F \sum_{u \in \mathcal{V}_1} M_u^* + \sum_{u \in \mathcal{V}_2} I(h_u^*; A_1) + H(h_u^* | A_1) \\ &= F \sum_{u \in \mathcal{V}_1} M_u^* + \sum_{u \in \mathcal{V}_2} H(h_u^*) = F \sum_{u \in \mathcal{V}} M_u^* = FM^*. \end{aligned}$$

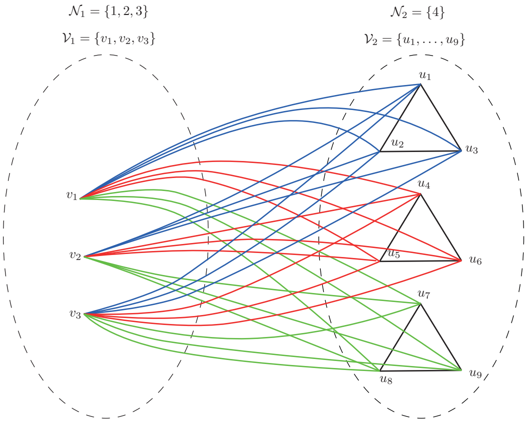

$\mathcal{N}_1 = \{1,2,3\}$
$\mathcal{V}_1 = \{v_1, v_2, v_3\}$
$\mathcal{N}_2 = \{4\}$
$\mathcal{V}_2 = \{u_1, \ldots, u_9\}$
v_1
v_2
v_3
u_1
u_2
u_3
u_4
u_5
u_6
u_7
u_8
u_9

Figure 5.5: If $|\mathcal{N}_1| > 1$ and $\mathcal{F}_{1,j} \neq \varnothing$ then Algorithm 4 is suboptimal in general.

Naturally, $\hat{M} \leq M$ and therefore, $\hat{M} = M^*$. □

The condition $\mathcal{F}_{2,j} = \varnothing, j = 1,2$ imposed by Theorem 5.2 is clearly stronger than the constraint $\mathcal{F}_{1,j} \cap \mathcal{F}_{2,j} = \varnothing, j = 1,2$ from Theorem 5.1. It tells us that the cross-edges must be all colored similarly to the monochromatic cluster.

If not, Algorithm 4 may be strictly sub-optimal. An example is depicted in Figure 5.5. This is a complete bipartite graph superimposed with the edges $(\{u_{i+3\ell}, u_{j+3\ell}\}, 4)$ for all $\{i,j\} \subset \{1,2,3\}$ and $\ell \in \{0,1,2\}$. All the black edges are interested in A_4, while the blue, red and green edges are interested in A_1, A_2 and A_3 respectively. We show that applying Algorithm 1 on this network with partitioning $\mathcal{N}_1$ and $\mathcal{N}_2$ provides a strictly suboptimal solution. Suppose G_1 and G_2 are the two subgraphs obtained from Algorithm 1. By applying Theorem 5.2 twice on G_1 one can find its optimal solution: $h_{v_i} = \{A_1, A_2, A_3\}$ for $i \in \{1,2,3\}$ (and all the other node store nothing). Then the solution to G_2 can be easily verified as $h_{u_1} = h_{u_4} = h_{u_7} = A_4^{(1)}$ and $h_{u_2} = h_{u_5} = h_{u_8} = A_4^{(2)}$ and $h_{u_3} = h_{u_6} = h_{u_9} = A_4^{(1)} + A_4^{(2)}$ where A_4 is assumed to have 2 symbols $A_4^{(1)}$ and $A_4^{(2)}$ and the summations are modulo q. Therefore, algorithm 1 provided a solution with $\hat{M} = 13.5$. On the other hand, we can do strictly better via the following strategy: $h_{v_i} = A_4$ for $i \in \{1,2,3\}$ and $h_{u_1} = A_1$, $h_{u_2} = A_1 + A_4$, $h_{u_3} = A_1 + 2A_4$, $h_{u_4} = A_2$, $h_{u_5} = A_2 + A_4$, $h_{u_6} = A_2 + 2A_4$, $h_{u_7} = A_3$, $h_{u_8} = A_3 + A_4$, $h_{u_9} = A_3 + 2A_4$. which results in $M = 12$.

5.2 Hyper-graphs with One File

In this section we briefly look at the GDSP defined over an arbitrary hypergraph but only in the presence of one file. Suppose $G = (\mathcal{V}, E)$ where E is an arbitrary subset of the power set of $\mathcal{V}$. We define the concepts of valid memory allocation and normalized sum achievable rate similarly to chapter 5.1. We only replace (5.2) by

$$H(A|h_{s_1,F},\ldots,h_{s_\ell,F}) = 0 \text{ for all } S = \{s_1,\ldots,s_\ell\} \in E.$$

We have the following simple proposition which directly follows from the analogy that we established between GDSP and network information flow in Section 5 and the min-cut max-flow theorem [43].

Proposition 5.1. *Suppose we have a hyper-graph $G = (\mathcal{V}, E)$ with K vertices. Let M^* be the minimum normalized sum achievable rate for G. Then M^* is the solution to the following LP.*

$$\begin{aligned} M^* &= \min_{M_1,\ldots,M_K} \sum_{u=1}^{K} M_u \;\; s.t. \\ M_u &\geq 0, \quad \forall u \in \{1,\ldots,K\}, \\ \sum_{u\in S} M_u &\geq 1, \quad \forall S \in E. \end{aligned}$$

6 The Shortest Vector Problem and Compute-and-Forward

A lattice is a structured collection of vectors in $\mathbb{R}^n$ which are characterized by integer linear combinations of a given set of vectors. Specifically, if we represent these vectors by the columns of a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, we can define a lattice as

$$\Lambda = \{\mathbf{Aa} | \mathbf{a} \in \mathbb{Z}^n\}.$$

The matrix $\mathbf{A}$ is known as the basis of the lattice. The basis of a lattice is not unique. Any matrix $\mathbf{A}$ whose columns are n linearly independent vectors in Λ can serve as a basis matrix for Λ.

The Shortest Vector Problem (SVP) is the problem of finding the shortest non-zero vector in a lattice. More specifically, given a basis $\mathbf{A}$ we can write

$$a^* = \underset{a\in\mathbb{Z}^n\backslash\{\mathbf{0}\}}{\arg\min} \|\mathbf{Aa}\|.$$

The shortest vector of a lattice is then $\mathbf{Aa}^*$. The optimization problem can be rephrases as

$$\mathbf{a}^* = \underset{\mathbf{a}\in\mathbb{Z}^n\backslash\{\mathbf{0}\}}{\arg\min} \mathbf{a}^T\mathbf{A}^T\mathbf{Aa} = \underset{\mathbf{a}\in\mathbb{Z}^n\backslash\{\mathbf{0}\}}{\arg\min} \mathbf{a}^T\mathbf{Ga} \tag{6.1}$$

where the matrix $\mathbf{G} = \mathbf{A}^T\mathbf{A}$ is called the Gramian matrix of $\mathbf{A}$.

A closely connected problem to SVP is the Closest Vector Problem (CVP). CVP is the problem of finding the closest vector in a lattice to a given vector in space. Specifically, given a lattice basis $\mathbf{A}$, and a vector $\mathbf{y} \in \mathbb{R}^n$ we are interested in

$$\mathbf{a}^* = \underset{\mathbf{a}\in\mathbb{Z}^n}{\arg\min} \|\mathbf{Aa} - \mathbf{y}\|^2 = \underset{\mathbf{a}\in\mathbb{Z}^n}{\arg\min} \mathbf{a}^T\mathbf{Ga} - 2\mathbf{y}^T\mathbf{Aa} + \mathbf{y}^T\mathbf{y}. \tag{6.2}$$

The CVP and the SVP are known to be NP-hard under randomized reduction [44, 45]. In fact, for a general lattice, there is not even an efficient constant-factor approximation algorithm known for these problems. Discovering such algorithms would have significant implications in terms of the hierarchy of complexity classes [46, 47, 45]. Currently, the best known polynomial complexity approximation algorithms for the SVP/CVP only achieve exponential approximation factors [48, 49].

On the bright side, efficient algorithms for special lattices have been known for a long time. For instance Gauss found an algorithm for solving the SVP in dimension two. Conway in [50] provides exact algorithms for a class of root lattices in higher dimensions. Based on [51] McKilliam [52] showed that if an obtuse superbase for a lattice is known, the SVP and the CVP can be solved in polynomial complexity.

In this work, we introduce new classes of lattices where the SVP and the CVP are of polynomial complexity. These classes are inspired by recently proposed cooperative communication strategies referred to as "Compute-and-Forward" [4] and "Integer-Forcing" [6]. Optimizing the computation rate achieved by these strategies involves solving particular instances of the SVP. We will first show that under certain conditions on the eigenvalues of the matrix $\mathbf{G}$ in Equations (6.1) and (6.2), the solution to the SVP and the CVP can be found in complexity order

$$O\left(n^{k+1}(2\lceil\psi\rceil+2)^{k+1}\right). \tag{6.3}$$

We will then show that the instances of the SVP that we are interested in satisfy these conditions. As for Integer-Forcing, the parameters k and n respectively stand for the number of antennas at the receiver and the number of transmitters. Here $\psi = \sqrt{1+P\gamma_{max}^2}$ depends on P, the transmission power and γ_{max}^2, the largest eigenvalue of $\mathbf{H}\mathbf{H}^T$ where $\mathbf{H}$ is the channel matrix. For Compute-and-Forward the complexity can be further reduced to

$$O(n\psi\log(n\psi)) \tag{6.4}$$

where $\psi = \sqrt{P\|\mathbf{h}\|^2+1}$ and $\mathbf{h}$ is the channel vector.

We will then proceed by introducing a larger class of lattices for which a constant approximation factor for the SVP and the CVP can be found in polynomial complexity.

There is a sizable literature concerning the particular instance of the SVP that appears in Compute-and-Forward and Integer-Forcing. Modifications to Sphere Decoding and Schnorr-Euchner algorithms [53, 54], Quadratic Programming Relaxation [55], Branch and Bound algorithm [56], Slowest Descent method [57], modifications to the LLL algorithm [58, 48], exhaustive search over a reduced search space [59], etc are some of the approaches taken by the community. Some of these works have excellent (sometimes linear) *average* complexity based on simulation results. Our work is distinguished in that we provide rigorous theoretical guarantees on both complexity (worst-case) and correctness of the algorithm. In addition, direct extensions of our work have been shown to demonstrate linearithmic average complexity [60, 61]. Our algorithm has further been extended to the Compute-and-Forward problem over rings of Gaussian integers and Eisenstein integers [62, 63]. Finally, another line of research concerns the recoverability of the messages at the receivers in a multi-relay scenario [64, 65, 66], where decoding the best equation by each relay node may not be the best approach.

The rest of this chapter is organized as follows. We will briefly discuss the Compute-and-Forward technique and its connection with the SVP in Section 6.2. We will demonstrate how this particular instance of the SVP can be solved efficiently. In Section 6.3 we will extend our results to Integer-Forcing. Next, in Section 6.4 we will show that the SVP and the CVP can be approximation up to a constant factor for a much larger class of lattices. Finally, in Section 6.5 we will discuss an open problem in the context of lattice reduction.

6.1 Notation

In addition to the notation in the introduction of the thesis, we introduce the following notation for this particular chapter. Vectors and matrices are denoted by boldface lowercase and capital letters respectively. All vectors are column vectors by default. All vector inequalities are elementwise. For instance $\mathbf{a} \leq \mathbf{b}$ is true if and only if $\mathbf{a}$ and $\mathbf{b}$ are of the same length n and $a_i \leq b_i$, $\forall i \in [1:n]$. For an $n \times m$ matrix $\mathbf{A}$ and for a set $\pi \subseteq [1:n]$ we define $\mathbf{A}_\pi$ as the submatrix of $\mathbf{A}$ which consists of the rows indexed in π. Similarly, for a vector $\mathbf{a}$ and a set of integers π we define $\mathbf{a}_\pi$ as a sub-vector of $\mathbf{a}$ which consists of the elements indexed in π. For a square matrix $\mathbf{A}$, the operator $diag(\mathbf{A})$ returns a column vector which consists of the diagonal elements of $\mathbf{A}$. We use $\|\cdot\|$ to represent the 2-norm of a vector. The identity matrix is denoted by $\mathbf{I}$ and $\mathbf{1}$ and $\mathbf{0}$ represent the all-one and all-zero vectors respectively.

6.2 IP^1 Matrices and Compute-and-Forward

The initial motivation behind this work is the problem of maximizing the achievable computation rate of Compute-and-Forward. In this section, we will present a short introduction to this relaying technique and establish its connection with the SVP. We will then demonstrate how this particular instance of the SVP can be solved in polynomial complexity.

6.2.1 Compute-and-Forward

Compute-and-Forward is an emerging relaying technique in wireless multiuser networks. Contrary to the conventional approaches, Compute-and-Forward does not view interference as inherently undesirable. The key idea is to recover integer linear combinations of transmitted codewords as opposed to decoding individual transmitted messages. Nested lattice codes ensure that these integer linear combinations are codewords themselves. Compute-and-Forward has the potential to increase the achievable rate compared to the traditional relaying techniques, as the analysis suggests in [4, 67, 68].

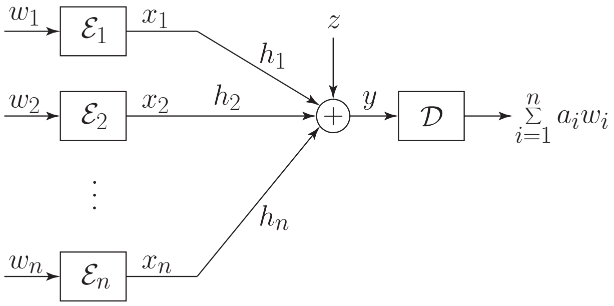


Figure 6.1: n transmitters send their messages to one relay. The relay decodes an integer linear combination of the codewords.

Figure 6.1 demonstrates a Compute-and-Forward scenario, where n transmitting nodes, each with transmission power P, share a wireless channel to send their messages to a relay node. We assume no knowledge of the channel states at the transmitters. The relay receives a noisy linear combination of the transmitted messages, namely

$$y = \sum_{i=1}^{n} h_i x_i + z$$

where x_i and h_i respectively represent the signal transmitted by node i and the effect of the channel from node i to the decoder, and z is the additive white Gaussian noise of unit variance. The relay then recovers $\sum a_i \omega_i$, an integer linear combination of the transmitted codewords. It has been proved [4] that the achievable rate of Compute-and-Forward satisfies:

$$R(\mathbf{h}, \mathbf{a}) = \frac{1}{2} \log^+ \left(\left(\|\mathbf{a}\|^2 - \frac{P |\mathbf{h}^T \mathbf{a}|^2}{1 + P \|\mathbf{h}\|^2} \right)^{-1} \right). \tag{6.5}$$

As evident from Equation (6.5), the achievable rate depends on the choice of the integer vector $\mathbf{a}$. From the perspective of a single decoder, a reasonable choice for $\mathbf{a}$ is one that maximizes R:

$$\mathbf{a}^* = \underset{\mathbf{a} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}}{\arg\max} \frac{1}{2} \log^+ \left(\left(\|\mathbf{a}\|^2 - \frac{P |\mathbf{h}^T \mathbf{a}|^2}{1 + P \|\mathbf{h}\|^2} \right)^{-1} \right) \tag{6.6}$$

which can be simplified as

$$\mathbf{a}^* = \underset{\mathbf{a} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}}{\arg\min} f(\mathbf{a}) = \mathbf{a}^T \mathbf{G} \mathbf{a} \tag{6.7}$$

where

$$\mathbf{G} = \mathbf{I} - \frac{P}{1 + P\|\mathbf{h}\|^2}\mathbf{h}\mathbf{h}^T \tag{6.8}$$

is a positive-definite matrix. Comparing this optimization problem with Equation (6.1), we see that Equation (6.7) is an instance of the SVP.

6.2.2 IP^1 Matrices and the Main Results

The positive-definite matrix $\mathbf{G}$ in Equation (6.8) falls in the following category of matrices which we refer to as IP^1. This term is chosen to emphasize a decomposition of the form $\mathbf{I} - \mathbf{P}$ where $\mathbf{P}$ is of rank 1.

Definition 6.1. *A positive-definite matrix $\mathbf{G}$ is called IP^1 if $\mathbf{G} = \mathbf{I} - \alpha\mathbf{v}\mathbf{v}^T$ where $\mathbf{v}$ is a normalized column vector in $\mathbb{R}^n$ and $0 \le \alpha < 1$ is a real number.*

The following theorem, albeit provable mostly by elementary manipulations of integer inequalities, establishes an important fact that provides the foundation of our SVP algorithm for IP^1 matrices. (The proof can be seen as a special case of Theorem 6.2. All the proofs are in Section 6.6.)

Theorem 6.1. *Suppose $\mathbf{a}^*$ is the solution to (6.1) for an IP^1 matrix $\mathbf{G}$. Then at least one of the following statements is true*

- $\mathbf{a}^*$ *satisfies*

$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{v}x < \mathbf{a}^* + \frac{1}{2}\mathbf{1} \tag{6.9}$$

  and thus

$$\mathbf{a}^* = \lceil \mathbf{v}x \rfloor \tag{6.10}$$

  for some $x \in \mathbb{R}^+$.

- $\mathbf{a}^*$ *is a standard unit vector, up to a sign.*

It follows from Theorem 6.1 that for the special lattices of interest, the shortest vector can be obtained by solving an optimization problem over only one variable. It is shown in [4] that the solution to (6.7) satisfies

$$\|\mathbf{a}^*\| \le \sqrt{P\|\mathbf{h}\|^2 + 1}. \tag{6.11}$$

Equation (6.11) transforms into

$$\|\mathbf{a}^*\| \le \frac{1}{\sqrt{1-\alpha}} \tag{6.12}$$

for a general IP^1 matrix. Consequently, the search only has to be done over a bounded region and in the vicinity of the line in the direction of $\mathbf{v}$ crossing the center. A separate examination of the standard unit vectors must also be performed. This is a significant reduction in the number of candidate vectors compared to other exhaustive search algorithms that apply to general lattices as in [69], where such a structure is naturally absent and all the lattice vectors within an n-dimensional sphere must be enumerated.

Remark 6.1. *The formula given by Theorem 6.1 has some resemblance to the results of [70] and [71]. However the span of these works are Coxeter lattices [72] and the goal is to find faster algorithms for problems (CVP) which are already known to be polynomially solvable. It is not difficult to see that the Gramian matrix for Coxeter lattices have an IP^k decomposition with small k ($k \leq 2$). Nevertheless, Coxeter lattices are highly limited in structure. For instance, there are fewer than $n+1$ different Coxeter lattices in $\mathbb{R}^{n+1}$. All these lattices are rank-deficient and the solution to their SVP can be given in closed form. For instance, for A_n^1, the shortest vector is $e_1 - \frac{1}{n+1}(\sum_{i=1}^{n+1} e_i)$ and for A_n^{n+1}, it is $e_1 - e_2$.*

6.2.3 SVP Algorithm for IP^1 Matrices

In line with Theorem 6.1 we define $\mathbf{a}(x) = \lceil \mathbf{v}x \rfloor$. Furthermore, let $\psi = \frac{1}{\sqrt{1-\alpha}}$, so that we have $\|\mathbf{a}^*\| \leq \psi$. Note that Theorem 6.1 reduces the problem to a one-dimensional optimization task. Since every $a_i(x)$, the i'th element of the vector $\mathbf{a}(x)$, is a piecewise constant function of x, so is the objective function

$$f(\mathbf{a}(x)) = \lceil \mathbf{v}x \rfloor^T \mathbf{G} \lceil \mathbf{v}x \rfloor.$$

Overall, the goal is to find a set of vectors which fully represent all the intervals in which $f(\cdot)$ is constant and choose the vector that minimizes $f(\cdot)$. Being a piecewise constant function, $f(\cdot)$ can be represented as:

$$f(\mathbf{a}(x)) = \begin{cases} r_i \ , & \text{if } \xi_i < x < \xi_{i+1} \ , \ i = 0, 1, \dots \\ s_i \ , & \text{if } x = \xi_i \ , \ i = 0, 1, \dots \end{cases} \tag{6.13}$$

ξ_i's are sorted real numbers denoting the points of discontinuity of $f(\cdot)$. Since $f(\cdot)$ is a continuous function of $\mathbf{a}$, these are in fact the discontinuity points of $\mathbf{a}(x)$ (or a subset of them) or equivalently the points where $a_i(x)$ is discontinuous, for some $i = 1 \dots n$. We can see from Equation (6.9) that any x satisfying

$$a_i^* - \frac{1}{2} < xv_i < a_i^* + \frac{1}{2} \ , \quad i = 1 \dots n \ , \ v_i \neq 0 \tag{6.14}$$

minimizes $f(\cdot)$. As a result, x belongs to the interior of an interval and not the boundary. Therefore, in the process of minimizing $f(\cdot)$, one can ignore the s_i values in (6.13), and find the minimizer of the objective function among the r_i values.

$$\min_{\mathbf{a} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}} f(\mathbf{a}) = \min_{i=0,1\dots} r_i. \tag{6.15}$$

Since $\frac{\xi_i+\xi_{i+1}}{2}$ belongs to the interior of the interval (ξ_i, ξ_{i+1}), we can rewrite r_i as $r_i = f(\mathbf{a}(\frac{\xi_i+\xi_{i+1}}{2}))$. Hence:

$$\min_{\mathbf{a}\in\mathbb{Z}^n\setminus\{\mathbf{0}\}} f(\mathbf{a}) = \min_{i=0,1\dots} f(\mathbf{a}(\frac{\xi_i+\xi_{i+1}}{2})). \tag{6.16}$$

As we discussed, ξ_i's are the points where at least one of the elements of the vector $\mathbf{a}$ faces discontinuity. Since we have $a_i(x) = \lceil v_i x \rfloor$, the discontinuity points of $a_i(x)$ are the points where $v_i x$ is a half-integer, or equivalently the points of the form $x = \frac{c}{|v_i|}$ where c is a positive half-integer and $v_i \neq 0$. From Equation (6.12) we can also see that $|a_i^*| \leq \psi$ and therefore, $0 < c \leq \lceil\psi\rceil + \frac{1}{2}$. To conclude this argument, we write:

$$\xi_i \in \bigcup_{j=1}^{n} \Phi_j \;\; , \;\; i = 0, 1, \cdots \tag{6.17}$$

where

$$\begin{aligned} \Phi_j &= \left\{ \frac{c}{|v_j|} \;\middle|\; 0 < c \leq \lceil\psi\rceil + \frac{1}{2} \;,\; c - \frac{1}{2} \in \mathbb{Z} \right\} \;,\; v_j \neq 0, \\ \Phi_j &= \varnothing \;,\; v_j = 0 \;,\; j = 1\dots n. \end{aligned}$$

Thus, the algorithm starts by calculating the sets Φ_j and their union Φ, sorting the elements of Φ and then running the optimization problem described by (6.16). The standard unit vectors will also be individually checked. The number of elements in Φ_j is upper-bounded by $\lceil\psi\rceil + 1$ and thus the number of elements in Φ is upper-bounded by $n(\lceil\psi\rceil + 1)$. The value of $f(\cdot)$ can be calculated in constant time. This is thanks to the special structure of the matrix $\mathbf{G}$. In fact, the objective function can be rewritten as:

$$f(\mathbf{a}) = \sum a_i^2 - \alpha \left(\sum a_i v_i\right)^2. \tag{6.18}$$

We keep track of every a_i and the two terms $\sum a_i^2$ and $\sum a_i v_i$. Since the discontinuity points are sorted, at each step only one of the a_i's changes and therefore the two terms can be updated in constant time. Consequently the new value of $f(\mathbf{a})$ can also be calculated in constant time. (In order to remember which a_i is being updated at each step, we assign a label to every member of Φ which indicates to which Φ_j it originally belonged).

It is easy to see that the complexity of the algorithm is determined by the sorting step. Since Φ has at most $n(\lceil\psi\rceil + 1)$ members, the complexity is

$$O(n\psi \log(n\psi)) \tag{6.19}$$

where $\psi = \frac{1}{\sqrt{1-\alpha}}$ for a general IP^1 matrix (Equation (6.12)) and $\psi = \sqrt{1 + P\|\mathbf{h}\|^2}$ for the Compute-and-Forward problem (Equation (6.11)).

The procedure is summarized in Algorithm 5. To provide further insight, it is worth noting that the efficiency of the algorithm is due to two factors. Firstly, the number of candidate lattice vectors to be enumerated is bounded by a polynomial function of n. Secondly, thanks to the special structure of these lattice vectors (imposed by Theorem 6.1) their efficient enumeration is possible. To appreciate the importance of the second factor, an analogy can be made with sphere decoding [69] where the radius of the sphere is chosen in such a way that the number of lattice vectors within the sphere is polynomially bounded. However, performing an efficient exhaustive search over all the candidate vectors within the sphere remains as the main challenge.

Algorithm 5 SVP for IP^1 matrices

```
Input: The IP$^1$ matrix $\mathbf{G} = \mathbf{I} - \alpha \mathbf{v}\mathbf{v}^T$.
Output: $\mathbf{a}^*$ the solution to SVP for $\mathbf{G}$.
Initialization :
$\mathbf{u}_i \leftarrow$ standard unit vector in the direction of the $i$-th axis
$\psi \leftarrow \frac{1}{\sqrt{1-\alpha}}$
$\Phi \leftarrow \varnothing$
$f_{\min} \leftarrow \min(diag(\mathbf{G}))$
$\mathbf{a}^* \leftarrow \mathbf{u}_{\arg\min(diag(\mathbf{G}))}$

Phase 1:
for all $i \in \{1, \ldots, n\}$, and $v_i \neq 0$ do
    for all $c$ , $0 < c \leq \lceil \psi \rceil + \frac{1}{2}$ , $c - \frac{1}{2} \in \mathbb{Z}$ do
        $x \leftarrow \frac{c}{|v_i|}$
        $\Phi \leftarrow \Phi \cup \{(x, i)\}$
    end for
end for

Phase 2:
sort $\Phi$ by the first element of the members (in an increasing order).
set $T_1 \leftarrow 0$ , $T_2 \leftarrow 0$ and $\mathbf{a} \leftarrow \mathbf{0}$.
for every $(x, j) \in \Phi$ (sweeping the set from left to right) do
    $a_j \leftarrow a_j + \text{sign}(v_j)$
    $T_1 \leftarrow T_1 + 2a_j - 1$
    $T_2 \leftarrow T_2 + |v_j|$
    $f_{\text{new}} \leftarrow T_1 - \alpha T_2^2$
    if $f_{\text{new}} < f_{\min}$ then
        $\mathbf{a}^* \leftarrow \mathbf{a}$
        $f_{\min} \leftarrow f_{\text{new}}$
    end if
end for
return $\mathbf{a}^*$
```

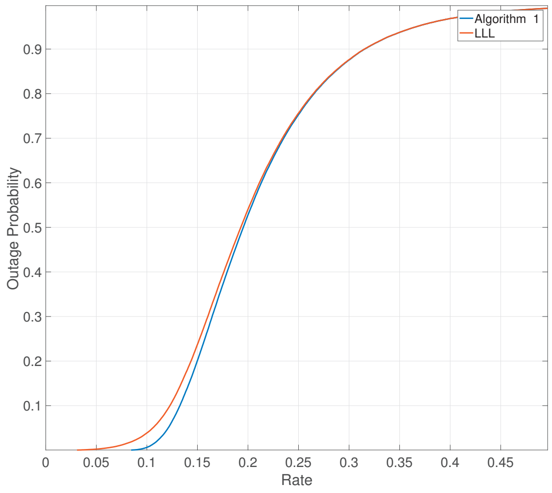


Figure 6.2: A comparison of LLL vs. Algorithm 5 in terms of outage probability. Simulations are done with the following parameters: $P = 20$, $n = 20$ and $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Prior to this work, it had been suggested [6] to use approximation algorithms such as LLL in order to solve this instance of the SVP. It is well-known that despite its notorious worst-case performance (only exponential guarantee), LLL performs well in practice. In order to provide a quantitative judgment, we performed a MATLAB simulation by generating 100,000 realizations of channels distributed as $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with 20 users and with $P = 20$. Indeed, LLL returned the optimal solution in almost 91% of the cases. Nevertheless, in the few cases where LLL fails, the difference in the achievable rate is rather significant (around 7% on average). An outage rate curve has been provided in Figure 6.2 for the sake of comparison. In particular the gap in the outage probability is non-negligible at small transmission rates, i.e. $R \approx 0.1$ where the outage probability of Algorithm 5 is only around 0.6% as opposed to LLL's 3.8%. Many other algorithms exist in the literature as discussed in the introduction. Some of these works have excellent performance based on simulation results. Nevertheless a strict theoretical guarantee is missing on either their correctness or their complexity. (We refrain from presenting empirical complexity comparisons in terms of running time or otherwise owing to the strong dependence on the precise implementation of each algorithm.)

6.2.4 Asymmetric Compute-and-Forward and DP^1 Matrices

A slightly more general model compared to Definition 6.1 is when the positive-definite matrix $\mathbf{G}$ is equal to

$$\mathbf{G} = \mathbf{D} - \alpha \mathbf{v}\mathbf{v}^T \tag{6.20}$$

where $\mathbf{D}$ is an arbitrary diagonal matrix with strictly positive diagonal elements, $\mathbf{v}$ is a normalized column vector and $0 \leq \alpha < 1$. We refer to such matrices as DP^1. Very similar to Theorem 6.1 we have

Theorem 6.2. *Suppose* $\mathbf{a}^*$ *is the solution to* (6.1) *where* $\mathbf{G}$ *is DP*^{1}*, that is* $\mathbf{G} = \mathbf{D} - \alpha \mathbf{v}\mathbf{v}^T$ *as in Equation* (6.20). *Then at least one of the following statements is true*

- $\mathbf{a}^*$ *satisfies*
$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D}^{-1}\mathbf{v}x < \mathbf{a}^* + \frac{1}{2}\mathbf{1} \tag{6.21}$$
and thus
$$\mathbf{a}^* = \lceil \mathbf{D}^{-1}\mathbf{v}x \rfloor \tag{6.22}$$
for some $x \in \mathbb{R}^+$.

- $\mathbf{a}^*$ *is a standard unit vector, up to a sign.*

Furthermore,

$$\|\mathbf{a}^*\| \leq \psi = \sqrt{\frac{G_{min}}{\lambda_{min}}}$$

where G_{min} *is the smallest diagonal element of the matrix* $\mathbf{G}$ *and* λ_{min} *is the smallest eigenvalue of* $\mathbf{G}$.

Algorithm 5 can then be readily extended to solve the SVP for DP^1 matrices. The following modifications are necessary. In step 8, v_i should be replaced by $\frac{v_i}{D_{ii}}$. Furthermore, $f(\mathbf{a})$ is no longer of the form (6.18), but instead

$$f(\mathbf{a}) = \sum D_{ii}a_i^2 - \alpha \left(\sum a_i v_i\right)^2. \tag{6.23}$$

Therefore, at step 16, T_1 should be updated as $T_1 = T_1 + D_{jj}(2a_j - 1)$. Finally, the value of ψ should be replaced with the more general expression $\psi = \sqrt{\frac{G_{min}}{\lambda_{min}}}$ to represent the new bound on the value of $\|\mathbf{a}^*\| \leq \psi$. With this change in value of ψ, the complexity again follows (6.19).

The interest in DP^1 matrices originates from their application in Asymmetric Compute-and-Forward [73]. Concisely, if in the Compute-and-Forward scheme we allow the transmitters to transmit at different rates, the achievable computation rate of the k'th transmitter is proved [73] to be equal to:

$$R_k(\mathbf{h}, \mathbf{a}, \mathbf{B}) = \left[\frac{1}{2}\log\left(\|\mathbf{B}\mathbf{a}\|^2 - \frac{P|\mathbf{h}^T\mathbf{B}\mathbf{a}|^2}{1 + P\|\mathbf{h}\|^2}\right)^{-1} + \frac{1}{2}\log B_{kk}^2\right]^+ \tag{6.24}$$

where $\mathbf{B}$ is an arbitrary diagonal matrix with positive diagonal elements. These diagonal elements are chosen by the respective transmitters based on their channel state information. Clearly, the integer vector $\mathbf{a}$ that maximizes the achievable rate is the same for all of the transmitters:

$$\mathbf{a}^* = \arg\min_{\mathbf{a}\in\mathbb{Z}^n\backslash\{\mathbf{0}\}} \mathbf{a}^T\mathbf{G}\mathbf{a}, \tag{6.25}$$

where $\mathbf{G}$ is given by:

$$\mathbf{G} = \mathbf{B}\left(\mathbf{I} - \frac{P}{1+P\|\mathbf{h}\|^2}\mathbf{h}\mathbf{h}^T\right)\mathbf{B} \tag{6.26}$$

which is a DP^1 matrix. Therefore, the extension of Algorithm 5 can be used to find the vector $\mathbf{a}$ which simultaneously maximizes the achievable rate for all transmitters.

It should be noted that for DP^1 matrices, the algorithm requires the decomposition of $\mathbf{G}$ as $\mathbf{D} - \alpha\mathbf{v}\mathbf{v}^T$. This information might be given a priori as with Asymmetric Compute-and-Forward or the decomposition could be found using the so called diagonal and low rank matrix decomposition techniques studied in [74, 75].

6.3 IP^k Matrices and Integer-Forcing

6.3.1 Integer-Forcing

In this section we provide a generalization of Theorem 6.1 and the corresponding algorithm by relaxing several constraints that we imposed on the structure of the Gramian matrix $\mathbf{G}$. The generalized theorem can be applied to maximize the achievable computation rate of Integer-Forcing studied in [6]. The scenario is very similar to the previous section, with the difference that the relay node now has multiple antennas. Our objective remains the same: decode the best integer linear combination of the received codewords. Assume there are n transmitters with transmission power P and the receiver node has k antennas. Let $\mathbf{h}_i$ be the channel vector from the transmitting nodes to the i-th antenna of the relay. Also, let $\mathbf{H}$ be the $n \times k$ matrix whose columns are the $\mathbf{h}_i$ vectors. It directly follows from the results of [6] that the achievable computation rate satisfies the following equation:

$$R(\mathbf{a}) = -\frac{1}{2}\log \mathbf{a}^T\mathbf{G}\mathbf{a} \tag{6.27}$$

where

$$\mathbf{G} = \mathbf{W}\mathbf{R}\mathbf{W}^T. \tag{6.28}$$

Here $\mathbf{W}$ is a unitary matrix in $\mathbb{R}^{n\times n}$ whose columns are the eigenvectors of $\mathbf{H}\mathbf{H}^T$, and $\mathbf{R}$ is a diagonal square matrix with the first k diagonal elements satisfying

$$r_i = \frac{1}{1+P\gamma_i^2} \;\; , \;\; i = 1\cdots k$$

and the last $n-k$ diagonal elements equal to 1. Finally, γ_i^2 is the i-th eigenvalue of $\mathbf{HH}^T$ (same order as the columns of $\mathbf{W}$).
Our goal is to find

$$\mathbf{a}^* = \arg\min_{\mathbf{a}\in\mathbb{Z}^n\setminus\{\mathbf{0}\}} \mathbf{a}^T\mathbf{Ga}$$

as in the single antenna case.

6.3.2 IP^k Matrices and the Main Results

We first mention a generalization of Theorem 6.1 and next we show that the Gramian matrix which appears in Equation (6.28) satisfies the constrains of the new theorem. To begin with, we define the following:

Definition 6.2 (IP^k matrices)**.** *A positive-definite matrix* $\mathbf{G}$ *is called IP*^{k} *if* $\mathbf{G} = \mathbf{I} - \mathbf{P}$ *where* $\mathbf{P}$ *is a positive semi-definite matrix of rank* k *and* $\mathbf{I}$ *is the identity matrix.*

We find it convenient to write $\mathbf{G}$ as

$$\mathbf{G} = \mathbf{I} - \mathbf{VV}^T \tag{6.29}$$

where $\mathbf{V}$ is an $n \times k$ matrix. Such a decomposition is not unique, but our arguments will be valid regardless of how the matrix $\mathbf{V}$ is chosen.

Theorem 6.3 is a generalization of Theorem 6.1 with a similar claim: the SVP for IP^k matrices can be reduced to a search problem over only k dimensions.

Theorem 6.3. *Suppose* $\mathbf{a}^*$ *is the solution to* (6.1) *where* $\mathbf{G}$ *is IP*^{k}*, that is* $\mathbf{G} = \mathbf{I} - \mathbf{VV}^T$ *as in Equation* (6.29)*. Then at least one of the following statements is true*

- *There exists a vector* $\mathbf{x} \in \mathbb{R}^k$ *such that* $\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{Vx} < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$ *and thus* $\mathbf{a}^* = \lceil \mathbf{Vx} \rfloor$*.*
- $\mathbf{a}^*$ *is a standard unit vector, up to a sign.*

Furthermore, $\|\mathbf{a}^*\| \leq \psi = \sqrt{\frac{G_{min}}{\lambda_{min}}}$ *where* G_{min} *is the smallest diagonal element of* $\mathbf{G}$ *and* λ_{min} *is the smallest eigenvalue of* $\mathbf{G}$*.*

Note that Theorem 6.1 is a special case of Theorem 6.3 where $k = 1$. The bound on the norm of $\mathbf{a}^*$ turns into $\frac{1}{\sqrt{1-\alpha}}$, for IP^1 matrices since we have $\lambda_{min} = 1 - \alpha$ and $G_{min} \leq 1$.

The Gramian matrix in equation (6.28) also satisfies the constraints of Theorem 6.3: Since $\mathbf{W}$ is a unitary matrix, $\mathbf{G}$ can be rewritten as $\mathbf{I}-\mathbf{W}(\mathbf{I} - \mathbf{R})\mathbf{W}^T$. The matrix $\mathbf{W}(\mathbf{I} - \mathbf{R})\mathbf{W}^T$ is of rank k (since $\mathbf{I} - \mathbf{R}$ has only k non-zero diagonal entries), and positive semi-definite. The bound given by the theorem translates into $\|\mathbf{a}^*\| \leq \sqrt{1+P\gamma_{max}^2}$ where γ_{max} is the maximum γ_i value. This is because $G_{min} \leq 1$ and the eigenvalues of $\mathbf{G}$ are equal to $\frac{1}{1+P\gamma_i^2}$ (with the same eigenvectors as $\mathbf{HH}^T$) or 1.

We will now show how to solve the SVP for an IP^k matrix using Theorem 6.3.

6.3.3 SVP Algorithm for IP^k Matrices

Similar to the case $k = 1$ we see that

$$f(\mathbf{a}(\mathbf{x})) = \lceil \mathbf{Vx} \rfloor^T \mathbf{G} \lceil \mathbf{Vx} \rfloor$$

is piecewise constant as a function of the vector $\mathbf{x}$ (this is because $\lceil \mathbf{Vx} \rfloor$ is a piecewise constant function of $\mathbf{x}$). Our objective is very similar to before: enumerate all the regions in space in which the objective function $f(\cdot)$ is constant and choose the one that minimizes $f(\cdot)$. From Theorem 6.3 we know that the vector $\mathbf{a}^*$ satisfies the $2n$ inequalities:

$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{Vx} < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$$

for some $\mathbf{x} \in \mathbb{R}^k$. In other words, $\mathbf{x}$ belongs to the interior of the cell described by these half-spaces. By analogy to the case $k = 1$ we aim at enumerating all such cells and finding the one which minimizes the objective function. To start with, we observe that each such cell is bounded by a set of hyperplanes of the form:

$$\mathbf{V}_{\{i\}}\mathbf{x} = c$$

where c is a half integer. Due to the bound given by Theorem 6.3 we could show that the hyperplanes are restricted to $|c| \leq (\lceil \psi \rceil + \frac{1}{2})$ which gives us a total of $n(2\lceil \psi \rceil + 2)$ hyperplanes. The problem of efficient enumeration of all the cells resulting from a partitioning of the space by a set of hyperplanes is the subject of a field called Hyperplane Arrangements. Very efficient algorithms have been developed over the past few decades. The general idea behind most of these algorithms is the following: we assign a normal vector with a specific direction to every hyperplane. Since a cell is bounded by hyperplanes, it must be entirely located on one side of each hyperplane. Therefore, a cell can be represented by a sign vector ν of length m where m is the number of hyperplanes. Each ν_i is either $+1$ or -1 depending on whether the cell is located on the positive side or the negative side of the corresponding hyperplane. Although there are 2^m possible configurations for the sign vector ν, at most $O(m^k)$ cells are created by the intersection of m hyperplanes. The enumeration algorithm will aim at finding those sign vectors that correspond to the actual cells. We will discuss two existing algorithms due to [76] and [77]. The first one is very simple to understand and implement but it might face numerical issues in case of degeneracies, i.e. when there are more than $k + 1$ hyperplanes intersecting at the same point. It is clear however that in practice, such an event occurs with probability zero. The second algorithm is slightly more complicated but it covers degeneracies too.

Simple Cell Enumeration Algorithm [76]

The idea is to first find all the vertices of the cells by finding the intersection of every k hyperplanes. We can represent each vertex by a sign vector of length m where each entry belongs to $\{-1, 0, 1\}$ depending on whether the vertex is on the left hand side of the corresponding hyperplane, located on it, or on its right hand side. Assuming there are no degeneracies, each such vertex belongs to exactly k hyperplanes (and not more). Consequently, every vertex has a sign vector with exactly k elements equal to zero. In this case, each vertex belongs to exactly 2^k cells whose sign vectors can be found by taking all possible assignments of $\{-1, 1\}$ to the zero elements of the sign vector of the vertex. Repeating this procedure for every vertex, we will have enumerated every cell of the arrangement, albeit in a redundant way. The running time of the algorithm is $O(m^{k+1})$. In case of degeneracies, [76] suggests that we slightly perturb the hyperplanes which however may not perform very well for highly degenerate matrices due to numerical issues.

Output-sensitive Cell Enumeration [77]

We represent every cell by a node in a graph. Two nodes are connected by an edge if and only if the corresponding cells are adjacent in space; in other words, if the sign vectors of the two cells differ in exactly one element. Intuitively such a graph is always connected. The algorithm aims at finding a spanning tree of this graph rooted at an arbitrary node. It also provides an interior point of each cell (here we are only interested in these interior points and not the sign vectors). There are two challenges. Firstly, we do not have a global knowledge of the graph and starting from each node we need to discover the neighboring nodes in an efficient way. Secondly, in order to form the spanning tree we must uniquely determine the parent of each node. The Output-sensitive Cell Enumeration algorithm in [77] uses two subroutines, namely adjlist() and parent() to address these two problems. The pseudocode is provided in Algorithm 6.

The overall complexity of the algorithm is $O(m|C|)$ where m is the number of hyperplanes and C is the number of cells which in turn is upper-bounded by $O(m^k)$. In our case, the number of hyperplanes is $n(2\lceil\psi\rceil + 2)$. Therefore the algorithm runs in $O\left(n^{k+1}(2\lceil\psi\rceil + 2)^{k+1}\right)$.

In order to ensure that all the cells are enumerated, it is necessary to make the first call to Algorithm 6 with the parameter $\mathbf{c} = \{+, \ldots, +\}$, that is the all plus sign vector. To guarantee that the all plus sign vector corresponds to an actual region, we change the direction of the hyperplanes in such a way that the origin is on the plus side of every hyperplane. Algorithm 7 will find the solution to the SVP for an IP^k matrix by first finding a list of all the hyperplanes, then calling Algorithm 6 in order to find an interior point of each cell, and finally calculating the value of the objective function over an interior point of each such cell in $O(kn)$. Since there are at most $O(m^k)$ cells, the complexity of Algorithm 7 is the same as Algorithm 6 (for constant k) that is

$$O\left(n^{k+1}(2\lceil\psi\rceil + 2)^{k+1}\right). \tag{6.30}$$

Algorithm 6 CellEnum($\mathbf{A}, \mathbf{b}, \mathbf{c}$)

```
Input: The root cell $\mathbf{c}$ represented by its sign vector. The hyperplanes
given by the matrix $\mathbf{A}$ and the vector $\mathbf{b}$
Output: An interior point of each cell in the subtree rooted at $\mathbf{c}$.

begin
Output an interior point of $\mathbf{c}$.
Find adjlist($\mathbf{c}$); the list of all neighbors of $\mathbf{c}$
for each $\mathbf{d} \in$ adjlist($\mathbf{c}$) do
    if parent($\mathbf{d}$) = $\mathbf{c}$ then
        CellEnum($\mathbf{A}, \mathbf{b}, \mathbf{d}$)
    end if
end for
```

6.3.4 Asymmetric MIMO Compute-and-Forward and DP^k Matrices

Similar to IP^1 matrices, we can slightly extend the results by replacing the identity matrix in Equation (6.29) with an arbitrary diagonal matrix with strictly positive diagonal values. More precisely, we define the DP^k matrices as positive-definite matrices of the form

$$\mathbf{G} = \mathbf{D} - \mathbf{V}\mathbf{V}^T \tag{6.31}$$

where $\mathbf{V} \in \mathbb{R}^{n\times k}$ and $\mathbf{D}$ is diagonal with strictly positive diagonal elements. The following Theorem holds:

Theorem 6.4. *Suppose* $\mathbf{a}^*$ *is the solution to* (6.1) *where* $\mathbf{G}$ *is DP*^{k}*, that is* $\mathbf{G} = \mathbf{D} - \mathbf{V}\mathbf{V}^T$ *as in Equation* (6.31)*. Then at least one of the following statements is true*

- *There exists a vector* $\mathbf{x} \in \mathbb{R}^k$ *such that* $\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D}^{-1}\mathbf{V}\mathbf{x} < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$ *and thus* $\mathbf{a}^* = \lceil\mathbf{D}^{-1}\mathbf{V}\mathbf{x}\rfloor$.
- $\mathbf{a}^*$ *is a standard unit vector, up to a sign.*

Algorithm 7 SVP for IP^k matrices

```
Input: The IP$^k$ Matrix $\mathbf{G} = \mathbf{I} - \mathbf{V}\mathbf{V}^T$.
Output: $\mathbf{a}^*$ the solution to the SVP for $\mathbf{G}$.

Initialization
$\mathbf{u}_i :=$ standard unit vector in the direction of $i$-th axis
$\psi := \sqrt{\frac{G_{\min}}{\lambda_{min}}}$
$f(\mathbf{a}) := \mathbf{a}^T\mathbf{G}\mathbf{a}$
$f_{\min} = \min(diag(\mathbf{G}))$
$\mathbf{a}^* = \mathbf{u}_{\arg\min(diag(\mathbf{G}))}$

begin
Form the matrix $\bar{\mathbf{V}} = \left[\begin{array}{c|c|c} \mathbf{V}^T & \cdots & \mathbf{V}^T \end{array}\right]^T$ by repeating $\mathbf{V}$, $(\lceil\psi\rceil + 1)$ times.
    Then $\bar{\mathbf{V}} \leftarrow \left[\begin{array}{c|c} \bar{\mathbf{V}}^T & -\bar{\mathbf{V}}^T \end{array}\right]^T$.
Form the vector $\bar{\mathbf{c}} = \left[\begin{array}{c|c|c} \mathbf{c}_1^T & \cdots & \mathbf{c}_L^T \end{array}\right]^T$ where $L = \lceil\psi\rceil + 1$ and $\mathbf{c}_i = (\frac{1}{2} - i)\mathbf{1}$
    and $\mathbf{1}$ is of length $n$. Then $\bar{\mathbf{c}} \leftarrow \left[\begin{array}{c|c} \bar{\mathbf{c}}^T & \bar{\mathbf{c}}^T \end{array}\right]^T$.
$\Phi = \text{CellEnum}(\bar{\mathbf{V}}, \bar{\mathbf{c}}, \{+, ..., +\})$
for each $\mathbf{d} \in \Phi$ do
    Find $\mathbf{a} = \lceil \mathbf{V}\mathbf{d} \rfloor$
    if $f(\mathbf{a}) < f_{\min}$ AND $\mathbf{a}$ is not the all-zero vector then
        Set $\mathbf{a}^* = \mathbf{a}$.
        Set $f_{\min} = f(\mathbf{a})$.
    end if
end for
```

Furthermore, $\|\mathbf{a}^*\| \leq \psi = \sqrt{\frac{G_{min}}{\lambda_{min}}}$ *where* G_{min} *is the smallest diagonal element of* $\mathbf{G}$ *and* λ_{min} *is the smallest eigenvalue of* $\mathbf{G}$*.*

Algorithm 7 can be reused with the following modifications in order to solve the SVP for DP^k matrices. In step 6 we now have $\bar{\mathbf{V}}$ as the vertical concatenation of the matrix $\mathbf{D}^{-1}\mathbf{V}$. Moreover, step 10 should be replaced by $\mathbf{a} = \lceil \mathbf{D}^{-1}\mathbf{V}\mathbf{d} \rfloor$. The complexity of the algorithm is again given by (6.30).

This new version of the algorithm can help us with maximizing the achievable computation rate of all transmitters in an asymmetric MIMO Compute-and-Forward scheme [73], assuming that the receiver aims at decoding a single integer linear combination of transmitted messages. In this case the achievable computation rate for the k'th transmitter is given [73] by

$$R_k(\mathbf{h}, \mathbf{a}, \mathbf{B}) = -\frac{1}{2} \log \frac{\mathbf{a}^T\mathbf{B}\mathbf{W}\mathbf{R}\mathbf{W}^T\mathbf{B}\mathbf{a}}{B_{ii}^2} \tag{6.32}$$

where $\mathbf{W}$ and $\mathbf{R}$ are as in (6.28) and $\mathbf{B}$ is an arbitrary diagonal matrix with positive diagonal elements selected at the transmitters. We can simultaneously maximize the achievable rate for all transmitters by solving the SVP for the matrix $\mathbf{G} = \mathbf{BWRW}^T\mathbf{B}$. We know from earlier discussion that $\mathbf{WRW}$ is IP^k from which it directly follows that $\mathbf{BWRW}^T\mathbf{B}$ is DP^k. Hence, our modified algorithm can be used to solve this instance of SVP.

6.4 Approximate SVP and CVP for $\widetilde{\mathrm{IP}}_\gamma^k$ Matrices

6.4.1 $\widetilde{\mathrm{IP}}_\gamma^k$ Matrices

In this chapter we introduce a larger class of lattices for which both the SVP and CVP can be approximated up to a constant factor. As evident from Definition 6.2, the eigenvalues of an IP^k matrix have a very particular structure: $n-k$ eigenvalues are equal to 1 and the remaining k eigenvalues are between 0 and 1. The main idea here is to relax this rather tight constraint and allow the eigenvalues to change within a neighborhood of these values. We will show that if these variations are small, the solution to the SVP and the CVP can be approximated within a constant factor. We start by defining the concept of IP^k-approximation of positive-definite matrices.

Definition 6.3. *Let $\mathbf{Q} \in \mathbb{R}^{n\times n}$ be a symmetric matrix with eigenvalues in $(0,1]$ and with the following eigendecomposition:*

$$\mathbf{Q} = \mathbf{V}^T\mathbf{\Lambda}\mathbf{V}.$$

For $k = 0,\ldots,n$, we define the IP^k-approximation of $\mathbf{Q}$ as:

$$\mathcal{I}_k(\mathbf{Q}) = \mathbf{V}^T\hat{\mathbf{\Lambda}}\mathbf{V}$$

where $\hat{\mathbf{\Lambda}}$ is obtained by setting the largest $n-k$ diagonal elements of $\mathbf{\Lambda}$ to one.

The assumption that the eigenvalues must be less than or equal to one is not of fundamental importance. If the eigenvalues of a Gramian matrix $\mathbf{G}$ are larger than one, we can normalize all the eigenvalues by the largest one. This translation does not have any effect on the solution of SVP (for CVP we will also have to scale the vector $\mathbf{y}$).

Definition 6.4. *A symmetric matrix $\tilde{\mathbf{G}} \in \mathbb{R}^{n\times n}$ with eigenvalues in $(0,1]$ is called $\widetilde{IP}_\gamma^k$ if $\mathcal{I}_k(\tilde{\mathbf{G}}) - \tilde{\mathbf{G}}$ has all its eigenvalues smaller or equal to γ, where γ is a constant satisfying $0 \le \gamma < 1$.*

In particular, the largest $n-k$ eigenvalues of an $\widetilde{\mathrm{IP}}_\gamma^k$ matrix cannot be arbitrarily close to zero. They must be within a constant (γ) gap of one. Figure 6.3 represents the sorted eigenvalues of an $\widetilde{\mathrm{IP}}_\gamma^k$ matrix (marked by black circles) and its IP^k-approximation (red crosses).

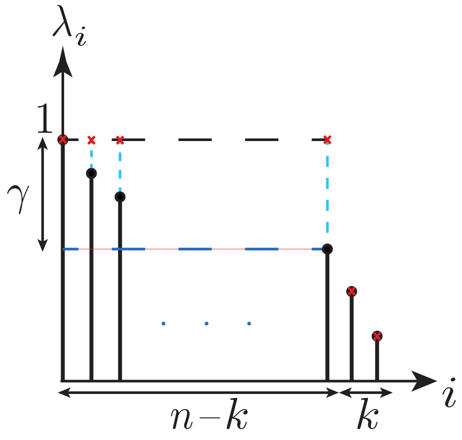


Figure 6.3: Eigenvalues of an $\widetilde{\text{IP}}_\gamma^k$ matrix (black circles) and its $\tilde{\text{IP}}^k$-approximation (red crosses).

6.4.2 Approximate SVP Algorithm for $\widetilde{\text{IP}}_\gamma^k$ Matrices

The following Theorem establishes a close connection between the solution of the SVP for an $\widetilde{\text{IP}}_\gamma^k$ matrix and for its IP^k-approximation.

Theorem 6.5. *Let*

$$f(\mathbf{a}) = \mathbf{a}^T\tilde{\mathbf{G}}\mathbf{a}$$

where $\tilde{\mathbf{G}}$ *is* $\widetilde{IP}_\gamma^k$. *Assume* $\mathbf{a}^*$ *is the solution to*

$$\mathbf{a}^* = \arg\min_{\mathbf{a}\in\mathbf{Z}^n\setminus\{\mathbf{0}\}} f(\mathbf{a})$$

and $\hat{\mathbf{a}}$ *satisfies*

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a}\in\mathbf{Z}^n\setminus\{\mathbf{0}\}} \mathbf{a}^T\mathcal{I}_k(\tilde{\mathbf{G}})\mathbf{a}$$

Then we have:

$$f(\hat{\mathbf{a}}) < \frac{1}{1-\gamma} f(\mathbf{a}^*). \tag{6.33}$$

Theorem 6.5 suggests that instead of solving the SVP for the $\widetilde{\text{IP}}_\gamma^k$ matrix $\tilde{\mathbf{G}}$, we can solve the problem for $\mathcal{I}_k(\tilde{\mathbf{G}})$, the IP^k-approximation of $\tilde{\mathbf{G}}$. The solution achieves a constant $(\frac{1}{1-\gamma})$ approximation factor on the original problem. As we saw in Section 6.3 the SVP for $\mathcal{I}_k(\tilde{\mathbf{G}})$ (which is an IP^k matrix) can be found in polynomial complexity. Therefore, Algorithm 8 is proposed to approximate the SVP for $\tilde{\mathbf{G}}$. A trivial improvement here would be to perform this minimization task over the original objective function. In other words, we can change line 3 of Algorithm 7 to $f(\mathbf{a}) := \mathbf{a}^T\tilde{\mathbf{G}}\mathbf{a}$.

Algorithm 8 $\frac{1}{1-\gamma}$-approximation algorithm for the SVP for $\widetilde{\mathrm{IP}}_\gamma^k$ matrices

```
Input: $\widetilde{\mathrm{IP}}_\gamma^k$ Matrix $\tilde{\mathbf{G}}$
Find $\hat{\mathbf{G}} = \mathcal{I}_k(\tilde{\mathbf{G}})$
return $\hat{\mathbf{a}}$ , the output of Algorithm 7 applied on $\hat{\mathbf{G}}$.
```

Finding the IP^k approximation of a matrix can be done in the same complexity order as finding its eigen-decomposition, that is $O(n^3)$, therefore, the overall complexity still follows Equation (6.30) (for $k > 1$) where ψ is equal to $\sqrt{\frac{G_{min}}{\lambda_{min}}}$, the ratio of the smallest diagonal element and the smallest eigenvalue of the matrix $\hat{\mathbf{G}}$. Although λ_{min} is equal for the two matrices $\hat{\mathbf{G}}$ and $\tilde{\mathbf{G}}$, the parameter G_{min} is in general larger for the matrix $\hat{\mathbf{G}}$ compared to $\tilde{\mathbf{G}}$. Nonetheless, it is evident that all the diagonal entries of $\hat{\mathbf{G}}$ are upper bounded by 1. Therefore we could still claim that as long as $\frac{1}{\lambda_{min}}$ is upper bounded by a polynomial function of n, the algorithm runs in polynomial complexity.

6.4.3 Extension of the Results to the CVP

Our results can be readily generalized to the CVP. For instance, the CVP for an IP^k matrix can be solved in an almost identical approach to the SVP. Furthermore, a similar constant-factor approximation for the CVP for $\widetilde{\mathrm{IP}}_\gamma^k$ matrices can be obtained. This is particularly interesting since in general, the algorithms that are used for solving the CVP are more sophisticated compared to the SVP.

The following theorem tells us that the same dimensionality reduction that appears in the SVP for IP^k, also holds for the CVP:

Theorem 6.6. *Suppose* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{G} = \mathbf{A}^T\mathbf{A}$ *is IP*^{k}*, that is*

$$\mathbf{G} = \mathbf{I} - \mathbf{V}\mathbf{V}^T$$

as in Equation (6.29)*. The solution to the Closest Vector Problem*

$$\mathbf{a}^* = \arg\min_{\mathbf{a}\in\mathbb{Z}^n} \|\mathbf{A}\mathbf{a} - \mathbf{y}\|^2$$

satisfies:

$$\mathbf{a}^* = \lceil \mathbf{V}\mathbf{x} + \mathbf{A}^T\mathbf{y} \rfloor \tag{6.34}$$

for some $\mathbf{x} \in \mathbb{R}^k$*. Furthermore,*

$$\|\mathbf{a}^* - \mathbf{A}^{-1}\mathbf{y}\| \leq \psi = \sqrt{\frac{G_{max}}{\lambda_{min}}} \tag{6.35}$$

where G_{max} *and* λ_{min} *are the largest diagonal element and the smallest eigenvalue of the matrix* $\mathbf{G}$*, respectively.*

The same Hyperplane Arrangement technique as in Section 6.3 can be applied here. The main difference is that now the hyperplanes are shifted compared to the case of SVP. Similar to Algorithm 7 we need to ensure that the all plus sign vector corresponds to an actual region. We will do this by changing the direction of the hyperplanes in such a way that the origin is on the plus side of every hyperplane. See Algorithm 9.

Algorithm 9 CVP for IP^k matrices

```
Input: The IP$^k$ Matrix $\mathbf{G} = \mathbf{I} - \mathbf{V}\mathbf{V}^T$ and the latice basis $\mathbf{A}$ that satisfies $\mathbf{G} = \mathbf{A}^T\mathbf{A}$ and the vector $\mathbf{y} \in \mathbb{R}^n$.
Output: $\mathbf{a}^*$ the solution to the CVP for $\mathbf{A}$ and $\mathbf{y}$.

Initialization
$\psi := \sqrt{\frac{G_{max}}{\lambda_{min}}}$
$f(\mathbf{a}) := \mathbf{a}^T\mathbf{G}\mathbf{a} - 2\mathbf{y}^T\mathbf{A}\mathbf{a} + \mathbf{y}^T\mathbf{y}$
$f_{\min} = \infty$

begin
Form the matrix $\bar{\mathbf{V}} = \left[\begin{array}{c|c|c} \mathbf{V}^T & \cdots & \mathbf{V}^T \end{array}\right]^T$ by repeating $\mathbf{V}$, $2(\lceil\psi\rceil+1)$ times.
Form the vector $\bar{\mathbf{c}} = \left[\begin{array}{c|c|c} \mathbf{c}_{L_1}^T & \cdots & \mathbf{c}_{L_2}^T \end{array}\right]^T$ where $L_1 = -\lceil\psi\rceil$ and $L_2 = \lceil\psi\rceil+1$
    and $\mathbf{c}_i = (i - \frac{1}{2} + \lfloor\mathbf{A}^{-1}\mathbf{y}\rfloor)\mathbf{1} - \mathbf{A}^T\mathbf{y}$ and $\mathbf{1}$ is of length $n$.
for $i = 1$ to $2n(\lceil\psi\rceil + 1)$ do
    if $\bar{c}_i > 0$ then
        $\bar{c}_i \leftarrow -\bar{c}_i$.
        $\bar{V}_{\{i\}} \leftarrow -\bar{V}_{\{i\}}$.
    end if
end for
$\Phi = \text{CellEnum}(\bar{\mathbf{V}}, \bar{\mathbf{c}}, \{+, \cdots, +\})$
for each $\mathbf{d} \in \Phi$ do
    Find $\mathbf{a} = \lceil\mathbf{V}\mathbf{d} + \mathbf{A}^T\mathbf{y}\rfloor$
    if $f(\mathbf{a}) < f_{\min}$ then
        Set $\mathbf{a}^* = \mathbf{a}$.
        Set $f_{\min} = f(\mathbf{a})$.
    end if
end for
```

These modifications do not affect the complexity order of the algorithm. Equation (6.30) still describes the complexity except for the fact that now $\psi = \sqrt{\frac{G_{max}}{\lambda_{min}}}$.

The results can be extended to DP^k matrices. In this case, the optimal coefficient vector is given by

Theorem 6.7. *The solution to the CVP for DP^k matrices satisfies:*

$$\mathbf{a}^* = \lceil \mathbf{D}^{-1}(\mathbf{V}\mathbf{x} + \mathbf{A}^T\mathbf{y}) \rfloor \tag{6.36}$$

for some $\mathbf{x} \in \mathbb{R}^k$. *Furthermore,*

$$\|\mathbf{a}^* - \mathbf{A}^{-1}\mathbf{y}\| \leq \psi = \sqrt{\frac{G_{max}}{\lambda_{min}}}. \tag{6.37}$$

We will now propose a generalization of our approximation algorithm for the CVP. First note that for the positive-definite matrix $\mathbf{G}$ with normalized and sorted eigenvalues λ_i we have $\mathbf{G} = \mathbf{A}^T\mathbf{A}$ if and only if

$$\mathbf{A} = \mathbf{U}\mathbf{B} \tag{6.38}$$

where $\mathbf{U}$ is an arbitrary unitary matrix and $\mathbf{B}$ is a symmetric matrix with the same eigenvectors as $\mathbf{G}$ and with eigenvalues β_i (sorted) that satisfy:

$$\beta_i^2 = \lambda_i.$$

Without loss of generality, we can also assume β_i's are positive (if they are negative, we can transfer the sign to the unitary matrix $\mathbf{U}$). Under this assumption, if $\mathbf{G}$ is IP^k then the matrix $\mathbf{B}$ must be IP^k too. Finally, we can generalize Theorem 6.5 for CVP. Here, besides mapping the matrix $\tilde{\mathbf{G}}$ to its IP^k approximation, we will also need to map the vector $\mathbf{y}$, whose nearest neighbor is of interest, to a different vector in space.

Theorem 6.8. *Let* $\mathbf{y}$ *be an arbitrary vector in* $\mathbb{R}^n$ *and* $\tilde{\mathbf{A}} \in \mathbb{R}^{n\times n}$ *be a full-rank matrix satisfying* $\tilde{\mathbf{A}} = \mathbf{U}\tilde{\mathbf{B}}$ *as in Equation* (6.38). *Furthermore assume* $\tilde{\mathbf{B}}^T\tilde{\mathbf{B}}$ *is* $\widetilde{IP}^k_\gamma$. *Define*

$$f(\mathbf{a}) = \|\tilde{\mathbf{A}}\mathbf{a} - \mathbf{y}\|^2.$$

Suppose $\mathbf{a}^*$ *is the solution to*

$$\mathbf{a}^* = \arg\min_{\mathbf{a}\in\mathbf{Z}^n} f(\mathbf{a})$$

and $\hat{\mathbf{a}}$ *satisfies*

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a}\in\mathbf{Z}^n} \|\hat{\mathbf{A}}\mathbf{a} - \hat{\mathbf{y}}\|^2$$

where $\hat{\mathbf{A}} = \mathcal{I}_k(\tilde{\mathbf{B}})$ *and* $\hat{\mathbf{y}} = \hat{\mathbf{A}}\tilde{\mathbf{A}}^{-1}\mathbf{y}$. *we have that:*

$$f(\hat{\mathbf{a}}) < \frac{1}{1-\gamma} f(\mathbf{a}^*).$$

To summarize, we propose Algorithm 10 for approximating the CVP for $\widetilde{\mathrm{IP}}_{\gamma}^{k}$ matrices.

Algorithm 10 $\frac{1}{1-\gamma}$-approximation algorithm for the CVP for $\widetilde{\mathrm{IP}}_{\gamma}^{k}$ matrices

```
Input: The vector $\mathbf{y}$, the full-rank square matrix $\tilde{\mathbf{A}} = \mathbf{U}\tilde{\mathbf{B}}$ as in Equation (6.38) with $\tilde{\mathbf{B}}^T\tilde{\mathbf{B}}$ being $\widetilde{\mathrm{IP}}_{\gamma}^{k}$.
1: Find $\hat{\mathbf{A}} = \mathcal{I}_k(\tilde{\mathbf{B}})$ and $\hat{\mathbf{y}} = \hat{\mathbf{A}}\tilde{\mathbf{A}}^{-1}\mathbf{y}$.
2: return $\hat{\mathbf{a}}$ , the output of Algorithm 9 applied on $\hat{\mathbf{y}}$ and $\hat{\mathbf{A}}$.
```

Again, the complexity follows Equation (6.30) where ψ is equal to $\sqrt{\frac{G_{max}}{\lambda_{min}}}$, the ratio of the largest diagonal element and the smallest eigenvalue of the matrix $\hat{\mathbf{G}}$

Remark 6.2. *A similar approximation factor of* $\frac{1}{1-\gamma}$ *can be obtained for a general positive-definite matrix of the form* $\mathbf{G} = \sqrt{\mathbf{D}}(\mathbf{I} - \mathbf{P})\sqrt{\mathbf{D}}$ *if we instead solve the SVP or CVP for the matrix* $\hat{\mathbf{G}} = \sqrt{\mathbf{D}}\mathcal{I}_k(\mathbf{I} - \mathbf{P})\sqrt{\mathbf{D}}$.

6.4.4 Application: MIMO Detection; a Trade-off Between Complexity and Accuracy

We study a potential application of Algorithm 10 in the context of MIMO detection. Consider communication over a general MIMO channel without CSIT:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}.$$

Assume the noise vector $\mathbf{z}$ is i.i.d. Gaussian. The receiver performs ML detection to estimate $\mathbf{x}$. After shifting and scaling, and assuming the alphabet size is large enough, we get the following optimization problem [78]:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{Z}^n} \|\bar{\mathbf{y}} - \bar{\mathbf{H}}\mathbf{x}\|^2.$$

This can be seen as an instance of the CVP, where the lattice matrix is $\bar{\mathbf{H}}$. Note that the ML detector outputs the correct value of $\mathbf{x}$, if the norm of the equivalent noise vector $\bar{\mathbf{z}}$ is smaller than half the minimum distance of the vectors (or equivalently, half the length of the shortest vector) of the lattice characterized by $\bar{\mathbf{H}}$. Let us call this parameter d_{min}^{H}. Thus we have:

$$P_{er} \leq P(\|\bar{\mathbf{z}}\| > d_{min}^{H}/2) \tag{6.39}$$

which can be expressed in terms of the CCDF of the Chi-squared distribution. Let us define λ_{max}^{H} as the maximum eigenvalue of $\bar{\mathbf{H}}$. For any integer $k = 0, \ldots, n$, suppose $\gamma(k)$ is the smallest positive number for which the lattice $\frac{1}{\lambda_{max}^{H}}\bar{\mathbf{H}}$ is $\widetilde{\mathrm{IP}}_{\gamma(k)}^{k}$. We will show that Algorithm 10 achieves the following error probability:

$$P_{er}^{alg10} \leq P\left(\|\bar{\mathbf{z}}\| > \frac{d_{min}^H}{1+1/\sqrt{1-\gamma(k)}}\right). \tag{6.40}$$

Note that if for some integer k we have $\gamma(k) = 0$, the matrix $\frac{1}{\lambda_{max}^H}\bar{\mathbf{H}}$ will be IP^k, and not surprisingly the algorithm returns the ML solution which achieves the same error probability as in Equation (6.39). But in general there will be a trade off between the complexity of the decoder and the achievable error probability: as we let k decrease to zero, Algorithm 10 runs faster (as evident from Equation (6.30)) but $\gamma(k)$ becomes larger which indicates a higher error probability, according to Equation (6.40).

To prove Equation (6.40), let us suppose that $\|\bar{\mathbf{z}}\| < \frac{d_{min}^H}{1+1/\sqrt{1-\gamma(k)}}$. Assume $\hat{\mathbf{x}}$ is the output of Algorithm 10 applied on $\frac{1}{\lambda_{max}^H}\bar{\mathbf{y}}$ and $\frac{1}{\lambda_{max}^H}\bar{\mathbf{H}}$. If $\hat{\mathbf{x}} \neq \mathbf{x}$, the best achievable approximation factor is:

$$\begin{aligned}
\left(\frac{\|\bar{\mathbf{H}}\hat{\mathbf{x}}-\bar{\mathbf{y}}\|}{\|\bar{\mathbf{H}}\mathbf{x}-\bar{\mathbf{y}}\|}\right)^2 &\geq \left(\frac{d_{min}^H-\|\bar{\mathbf{z}}\|}{\|\bar{\mathbf{z}}\|}\right)^2 \\
&> \left(\frac{d_{min}^H-\frac{d_{min}^H}{1+1/\sqrt{1-\gamma(k)}}}{\frac{d_{min}^H}{1+1/\sqrt{1-\gamma(k)}}}\right)^2 \\
&= \left(\frac{1}{\sqrt{1-\gamma(k)}}\right)^2 = \frac{1}{1-\gamma(k)}.
\end{aligned}$$

This contradicts with the fact that Algorithm 10 achieves an $\frac{1}{1-\gamma(k)}$ approximation factor. Thus we must have that $\hat{\mathbf{x}} = \mathbf{x}$. As a result, as long as $\|\bar{\mathbf{z}}\| < \frac{d_{min}^H}{1+1/\sqrt{1-\gamma(k)}}$, Algorithm 10 outputs the correct value of $\mathbf{x}$. This proves Equation (6.40).

6.5 Open Problem: $\widetilde{\mathrm{IP}}^k$-reduced Basis

The basis matrix of a lattice is not unique. For any lattice $\mathcal{L}(\mathbf{A})$ there are infinitely many bases. All these bases are related via linear transformation by unimodular matrices. In other words if we have $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B})$ then there exists a unimodular matrix $\mathbf{T}$ such that $\mathbf{B} = \mathbf{AT}$.

The field of lattice reduction aims at finding such unimodular transformations for arbitrary lattice bases, and producing new bases with more desirable properties. The new basis is usually called a reduced basis of the lattice. There are different notions of lattice reduction. For instance Minkowski's criteria for calling a basis $\mathbf{A}$ reduced is that the shortest vector (or column) of this basis, $\mathbf{v}_1$ must be the shortest vector of the lattice $\mathcal{L}(\mathbf{A})$; the second shortest vector of $\mathbf{A}$ must be the second shortest vector of the lattice among all the vectors that are linearly independent of $\mathbf{v}_1$ and so on. Of course, there is no polynomial time algorithm known that can find such a reduced basis (as otherwise, the SVP would have been solved and much more). Another notion is the LLL-reduced basis due to [48]. An LLL reduced basis can be found in polynomial time. However the shortest vector of an LLL-reduced basis can be exponentially longer than the shortest vector of the lattice.

Here, based on the concept of $\widetilde{\mathrm{IP}}^k_\gamma$ matrices we introduce a new notion of reduced basis. Specifically, we define

Definition 6.5. *A lattice basis* $\mathbf{A}$ *is called* $\widetilde{IP}^k$*-reduced if it holds that* $\gamma_{\mathbf{A}}(k) \leq \gamma_{\mathbf{B}}(k)$ *for any matrix* $\mathbf{B}$ *that satisfies* $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{A})$*. Here* $\gamma_{\mathbf{A}}(k)$ *is the smallest* γ *for which the matrix* $\mathbf{G} = \mathbf{A}^T\mathbf{A}$ *(after normalizing the eigenvalues) is* $\widetilde{IP}^k_\gamma$*.*

In other words, given an arbitrary lattice basis, we are interested in finding a new basis for the same lattice which minimizes the value of γ for a particular k. This is demonstrated in Figure 6.4.

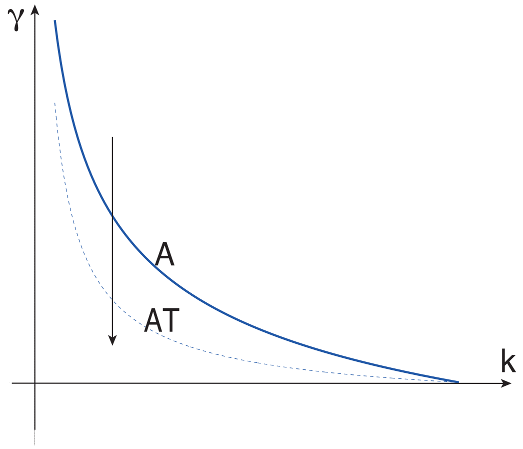


Figure 6.4: An arbitrary lattice basis and its $\widetilde{\mathrm{IP}}^k$ reduction

It should be clear why we are interested in such a basis. We want to achieve the best possible approximation factor for SVP and CVP through Algorithms 8 and 10 which run in complexity order of $O(n^{k+1}(2\lceil\psi\rceil+2)^{k+1})$. Currently we do not know any algorithm that can find an $\widetilde{\mathrm{IP}}^k_\gamma$-reduced basis for an arbitrary lattice. Finding an efficient algorithm which performs this task could have quite interesting implications in terms of approximating the SVP or CVP for a general lattice.

6.6 Appendix

Proof of Theorem 6.7. We will prove the claim for DP^k matrices, that is $\mathbf{G} = \mathbf{D} - \mathbf{P} = \mathbf{D} - \mathbf{V}\mathbf{V}^T$. Theorem 6.6 follows as a special case. To simplify the notation, we define $\mathbf{z} = \mathbf{A}^T\mathbf{y}$. We can rewrite $f(\mathbf{a}) = \mathbf{a^T G a} - 2\mathbf{z}^T\mathbf{a} + \mathbf{y}^T\mathbf{y}$ as follows:

$$f(\mathbf{a}) = \sum_{i=1}^{n}(D_{ii} - P_{ii})a_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{i-1} P_{ij}a_ia_j - 2\sum_{i=1}^{n} z_ia_i + \mathbf{y}^T\mathbf{y}.$$

First note that since $\mathbf{P}$ is positive semi-definite, we have $P_{ii} \geq 0$ for $i = 1, \ldots, n$. If for some j we have $P_{jj} = 0$, then we must have $P_{ij} = P_{ji} = 0$ and $V_{ji} = 0$ for $i = 1, \ldots, n$. The optimal value for a_j in this case is simply $a_j^* = \left\lceil \frac{z_j}{D_{jj}} \right\rfloor$ which satisfies the claim of the theorem. The problem can then be reduced to $n-1$ dimensions. Thus without loss of generality we assume $P_{ii} > 0$ for the rest of the proof.

Assume that we already know the optimal value for all a_i^* elements except for one element, a_j. Note that f is a convex parabola in a_j (this is because $D_{jj} - P_{jj} = G_{jj}$ is a diagonal element of a positive-definite matrix) thus the optimal integer value for a_j is the closest integer to its optimal real value. By taking partial derivative with respect to a_j, the optimal real value of a_j is easily seen to be equal to

$$\frac{z_j + \sum_{i=1, i\neq j}^{n} P_{ij}a_i^*}{D_{jj} - P_{jj}}.$$

Taking the closest integer to the real valued solution, we find:

$$\begin{aligned} \Rightarrow a_j^* &= \left\lceil \frac{z_j + \sum_{i=1, i\neq j}^{n} P_{ij}a_i^*}{D_{jj} - P_{jj}} \right\rfloor, \ \textit{or} \\ a_j^* &= \left\lfloor \frac{z_j + \sum_{i=1, i\neq j}^{n} P_{ij}a_i^*}{D_{jj} - P_{jj}} \right\rceil. \end{aligned} \tag{6.41}$$

Due to the symmetry of the parabola, both functions return equally correct solutions for a_j^*.

Note that this expression must be true for any j: If for $\mathbf{a}^*$ and for some j, a_j^* does not satisfy at least one of these two equations, we can achieve a strictly smaller value over f by replacing a_j^* with the value given above, and so $\mathbf{a}^*$ cannot be optimal. From Equation (6.41) we have that:

$$a_j^* + \frac{1}{2} \geq \frac{z_j + \sum_{i=1, i\neq j}^{n} P_{ij}a_i^*}{D_{jj} - P_{jj}} \quad , \ \textit{and} \tag{6.42}$$

$$a_j^* - \frac{1}{2} \leq \frac{z_j + \sum_{i=1, i\neq j}^{n} P_{ij}a_i^*}{D_{jj} - P_{jj}}. \tag{6.43}$$

Starting with Equation (6.42), we multiply both sides by the denominator, and add the term $a_j^*P_{jj}$ to obtain:

$$(a_j^* + \frac{1}{2})D_{jj} \geq z_j + \sum_{i=1}^{n} P_{ij}a_i^* + \frac{1}{2}P_{jj}.$$

Dropping the positive term $\frac{1}{2}P_{jj}$ we conclude

$$(a_j^* + \frac{1}{2})D_{jj} > z_j + \sum_{i=1}^{n} P_{ij} a_i^*.$$

As a result, we have

$$(a_j^* + \frac{1}{2}) > \frac{z_j + \sum_{i=1}^{n} P_{ij} a_i^*}{D_{jj}} \ , \ j = 1 \dots n.$$

Writing this inequality in vector format, we obtain

$$\mathbf{a}^* + \frac{1}{2}\mathbf{1} > \mathbf{D}^{-1}(\mathbf{z} + \mathbf{P}^T \mathbf{a}^*) = \mathbf{D}^{-1}(\mathbf{z} + \mathbf{V}(\mathbf{V}^T \mathbf{a}^*)) \tag{6.44}$$

In a similar fashion one can show that Equation (6.43) results in

$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D}^{-1}(\mathbf{z} + \mathbf{V}(\mathbf{V}^T \mathbf{a}^*)). \tag{6.45}$$

Defining $\mathbf{x} = \mathbf{V}^T \mathbf{a}^*$, it follows from (6.44) and (6.45) that

$$\mathbf{a}^* - \frac{1}{2}\mathbf{1} < \mathbf{D}^{-\mathbf{1}}(\mathbf{V}\mathbf{x} + \mathbf{z}) < \mathbf{a}^* + \frac{1}{2}\mathbf{1}$$

$$\Rightarrow \mathbf{a}^* = \lceil \mathbf{D}^{-\mathbf{1}}(\mathbf{V}\mathbf{x} + \mathbf{z}) \rfloor = \lceil \mathbf{D}^{-1}(\mathbf{V}\mathbf{x} + \mathbf{A}^T \mathbf{y}) \rfloor.$$

This completes the proof of Equation (6.34).
To prove Equation (6.37), note that

$$\|\mathbf{A}\mathbf{a}^* - \mathbf{y}\|^2 = \|\mathbf{A}(\mathbf{a}^* - \mathbf{A}^{-1}\mathbf{y})\|^2 \geq \lambda_{min} \|\mathbf{a}^* - \mathbf{A}^{-1}\mathbf{y}\|^2.$$

It is also evident that $\|\mathbf{A}\mathbf{a}^* - \mathbf{y}\|^2 \leq G_{max}$, that is the square distance of the closest vector of the lattice to $\mathbf{y}$ is less than the largest diagonal element of $\mathbf{G}$. To see this, note that for any vector in the Voronoi region of the origin there are N successive minima of the lattice $(\nu_1, \dots, \nu_N)$ such that $p = \sum_{i=1}^{N} \theta_i \nu_i$ where $\theta_i \geq 0$ and $\sum_{i=1}^{N} \theta_i \leq 1$. The claim follows since $\sqrt{G_{max}}$ cannot be smaller than the length of the N'th successive minima of the lattice.

$$G_{max} \geq \|\mathbf{A}\mathbf{a}^* - \mathbf{y}\|^2 \geq \lambda_{min} \|\mathbf{a}^* - \mathbf{A}^{-1}\mathbf{y}\|^2$$

from which we can conclude: $\|\mathbf{a}^* - \mathbf{A}^{-1}\mathbf{y}\| \leq \sqrt{\frac{G_{max}}{\lambda_{min}}}$. □

Proof of Theorem 6.5. Proof of Theorem 6.5 can be seen as a special case of proof of Theorem 6.8 for $\mathbf{y} = \mathbf{0}$. The extra condition of $\mathbf{a} \neq \mathbf{0}$ for the SVP does not cause any problem here. □

Proof of Theorem 6.4. The proof of Theorem 6.4 is almost identical to that of Theorem 6.7 after setting $\mathbf{y} = \mathbf{0}$. The requirement for checking the standard unit vectors follows from the fact that $\mathbf{a}$ cannot be the all zero vector. In other words, all the elements of the vector $\mathbf{a}$ must satisfy (6.41) unless $\mathbf{a}^*$ is a standard unit vector in the direction of the j'th axis in which case replacing a_j^* by (6.41) will lead to the all zero vector.

To prove the bound on norm of $\mathbf{a}^*$ note that the square norm of the shortest vector of the lattice must be smaller or equal to G_{min}. Therefore,

$$G_{min} \geq \|\mathbf{A}\mathbf{a}^*\|^2 \geq \lambda_{min}\|\mathbf{a}^*\|^2 \to \|\mathbf{a}^*\| \leq \sqrt{\frac{G_{min}}{\lambda_{min}}}.$$

□

Proof of Theorem 6.2. This is clearly a special case of Theorem 6.4. The fact that we can limit the search to $x \in \mathbb{R}^+$ instead of the whole $\mathbb{R}$ follows trivially from: $\mathbf{a}^T\mathbf{G}\mathbf{a} = (-\mathbf{a})^T\mathbf{G}(-\mathbf{a})$. □

Proof of Theorem 6.8. Define $\mathbf{c} = \tilde{\mathbf{B}}^{-1}\mathbf{U}^T\mathbf{y}$. We have

$$\begin{aligned} f(\mathbf{a}^*) &= \|\tilde{\mathbf{B}}\mathbf{a}^* - \tilde{\mathbf{B}}\tilde{\mathbf{B}}^{-1}\mathbf{U}^T\mathbf{y}\|^2 = \|\tilde{\mathbf{B}}(\mathbf{a}^* - \mathbf{c})\|^2 \\ &= \sum_{i=1}^{n} \beta_i^2((\mathbf{a}^* - \mathbf{c})^T\mathbf{v}_i)^2 \\ &\geq \sum_{i=1}^{k}(1-\gamma)\beta_i^2((\mathbf{a}^* - \mathbf{c})^T\mathbf{v}_i)^2 + \sum_{i=k+1}^{n}(1-\gamma)((\mathbf{a}^* - \mathbf{c})^T\mathbf{v}_i)^2 \\ &= (1-\gamma)\|\hat{\mathbf{A}}(\mathbf{a}^* - \mathbf{c})\|^2 \\ &= (1-\gamma)\|\hat{\mathbf{A}}\mathbf{a}^* - \hat{\mathbf{y}}\|^2 \\ &\geq (1-\gamma)\|\hat{\mathbf{A}}\hat{\mathbf{a}} - \hat{\mathbf{y}}\|^2 \\ &\geq (1-\gamma)f(\hat{\mathbf{a}}) \end{aligned}$$

where the last step is due to the fact that $\|\tilde{\mathbf{A}}\mathbf{a} - \mathbf{y}\| \leq \|\hat{\mathbf{A}}\mathbf{a} - \hat{\mathbf{y}}\|$ for any arbitrary vector $\mathbf{a}$. □

7 The Most Informative Bit Conjecture

The most informative bit conjecture first introduced in [79] can be formulated as follows.

Conjecture 7.1. *Let $\{(X_i, Y_i)\}_{i=1}^n$ be an i.i.d. sequence of length n such that $X_i \in \{0, 1\}$ is a Ber$(1/2)$ random variable and $Y_i \in \{0, 1\}$ is the output of a Binary Symmetric Channel (BSC) with input X_i, i.e. $\mathbb{P}(Y_i = 0|X_i = 0) = P(Y_i = 1|X_i = 1) = 1 - p$. Let $B : \{0, 1\}^n \to \{0, 1\}$ be an arbitrary function. We have*

$$I(B(Y_{[1:n]}); X_{[1:n]}) \leq 1 - h_2(p).$$

There is a simple interpretation to the conjecture. Suppose we have a BSC over which we transmit n independent bits. At the receiver we are allowed to compute any binary function of the outputs of these n transmissions. The conjecture states that a simple $B(Y_{[1:n]}) = Y_1$, typically referred to as the dictator function, conveys the most information about the inputs of the channel, among all possible binary functions. As of now the conjecture is still open, with the exception of extremal values of p, that is when $p \approx 0$ or $p \approx 1/2$ [80]. Several variations of the problem have been studied in the literature. For instance the following two weaker variations of the problem have been proven.

Theorem 7.1 ([79]). *Under the constraints of Conjecture 7.1 we have*

$$\sum_{i=1}^{n} I(B(Y_{[1:n]}); X_i) \leq 1 - h_2(p).$$

Theorem 7.2 ([81]). *Let $\{(X_i, Y_i)\}_{i=1}^n$ be as described in Conjecture 7.1. Let $B, C : \{0, 1\}^n \to \{0, 1\}$ be two arbitrary binary functions. Then*

$$I(B(Y_{[1:n]}); C(X_{[1:n]})) \leq 1 - h_2(p).$$

A generalization of the conjecture to non-binary functions $B : \{0,1\}^n \to \{0,1\}^k$ - referred to as "the most informative quantization function" - asks whether the following holds

$$I(B(Y_{[1:n]}); X_{[1:n]}) \leq k(1 - h_2(p)).$$

This generalized conjecture has been disproven [82], however the following result has been established.

Theorem 7.3 ([82])**.** *Let $\{(X_i, Y_i)\}_{i=1}^n$ be as described in conjecture 7.1. Assume $B : \{0,1\}^n \to \{0,1\}^k$ is an arbitrary function. We have*

$$I(B(Y_{[1:n]}); X_{[1:n]}) \leq k(1 - 2p)^2.$$

In this work we tighten and generalize this bound via our next theorem. Apart from strengthening the bound, perhaps the more important contribution of this work is to introduce an alternative - arguably more straightforward - proof technique for Theorem 7.3. The simplicity of the proof is encouraging in that a variation of the arguments presented here may result in further advance towards proving Conjecture 7.1. We first present a simple lemma and next proceed to our main theorem and its proof.

Lemma 7.1. *The function*

$$f(\mathbf{x}, \mathbf{y}) = H(\mathbf{x}) + H(\mathbf{y}) - H(p\mathbf{x} + (1 - p)\mathbf{y}) - H((1 - p)\mathbf{x} + p\mathbf{y})$$

defined over probability vectors $\mathbf{x}$ and $\mathbf{y}$ is concave in $(\mathbf{x}, \mathbf{y})$ for any $p \in [0, 1]$.

Proof. Let X, Y, U_1, U_2 be four independent discrete random variables. Let X and Y have support $[1 : n]$ such that $\mathbb{P}(X = i) = x_i$, $\mathbb{P}(Y = 0) = y_i$. Let U_1 and U_2 have support $\{0, 1\}$ such that $\mathbb{P}(U_1 = 0) = \mathbb{P}(U_2 = 0) = p$. Define $D_1 = U_1Y + (1 - U_1)X$ and $D_2 = U_2X + (1 - U_2)Y$. We have

$$f(\mathbf{x}, \mathbf{y}) = -I(D_1; U_1) - I(D_2; U_2).$$

Both $I(D_i; U_i)$ are convex in $p_{D_i|U_i} = [\mathbf{x}, \mathbf{y}]$. So, $f(\mathbf{x}, \mathbf{y})$ is concave in $(\mathbf{x}, \mathbf{y})$. □

Theorem 7.4. *Let $\{(X_i, Y_i)\}_{i=1}^n$ be as defined in Conjecture 7.1. Assume $B : \{0,1\}^n \to [0 : q - 1]$ is an arbitrary q-ary function. We have* [1]

$$I(B(Y_{[1:n]}); X_{[1:n]}) \leq H(B)(1 - 2p)^2.$$

[1] Note that Theorem 7.4 reproduces Theorem 7.3 if B has uniform marginal distribution and $q = 2^k$.

Proof of Theorem 7.4. Define two functions $C_0, C_1 : \{0,1\}^{n-1} \to [0:q-1]$ as follows.

$$\begin{aligned} C_0(Y_{[1:n-1]}) &= B(Y_{[1:n-1]}, Y_n = 0), \\ C_1(Y_{[1:n-1]}) &= B(Y_{[1:n-1]}, Y_n = 1). \end{aligned}$$

Let $P_{ij} = \mathbb{P}(C_i = j | X_{[1:n-1]})$ for $i \in \{0,1\}$ and $j \in [0:q-1]$. Let $\mathbf{P}_i = P_{i,[0:q-1]}$ and $\mathbf{S}_i = \mathbb{E}(\mathbf{P}_i)$ for $i \in \{0,1\}$ where the expectations are over $X_1, \ldots, X_{n-1}$. We prove the theorem by induction over n. The base case is trivial. Suppose our claim holds for $n-1$. Therefore we have

$$\begin{aligned} I(C_0(Y_{[1:n-1]}); X_{[1:n-1]}) &\leq H(C_0)(1-2p)^2, \\ I(C_1(Y_{[1:n-1]}); X_{[1:n-1]}) &\leq H(C_1)(1-2p)^2. \end{aligned}$$

Define

$$\mathcal{A}(\mathbf{P}_0, \mathbf{P}_1) = -H(B|X_{[1:n]}) + 0.5H(C_0|X_{[1:n-1]}) + 0.5H(C_1|X_{[1:n-1]}).$$

First note that

$$H(B|X_{[1:n]}) = - \ 0.5\mathbb{E}\left(H((1-p)\mathbf{P}_0 + p\mathbf{P}_1)\right) - 0.5\mathbb{E}\left(H((1-p)\mathbf{P}_1 + p\mathbf{P}_0)\right).$$

Therefore,

$$\begin{aligned} \mathcal{A}(\mathbf{P}_0, \mathbf{P}_1) = &- 0.5\mathbb{E}\left(H((1-p)\mathbf{P}_0 + p\mathbf{P}_1)\right) - 0.5\mathbb{E}\left(H((1-p)\mathbf{P}_1 + p\mathbf{P}_0)\right) \\ &+ 0.5\mathbb{E}(H(\mathbf{P}_0)) + 0.5\mathbb{E}(H(\mathbf{P}_1)) \\ &\leq -0.5H((1-p)\mathbf{S}_0 + p\mathbf{S}_1) - 0.5H((1-p)\mathbf{S}_1 + p\mathbf{S}_0) \\ &+ 0.5H(\mathbf{S}_0) + 0.5H(\mathbf{S}_0) \end{aligned}$$

where the inequality is due to Lemma 7.1. We are now ready to bound $I(B; X_{[1:n-1]})$.

$$\begin{aligned} I(B; X_{[1:n]}) &= H(B) - 0.5H(C_0) - 0.5H(C_1) + \mathcal{A}(\mathbf{P}_0, \mathbf{P}_1) \\ &+ 0.5I(C_0; X_{[1:n-1]}) + 0.5I(C_1; X_{[1:n-1]}) \\ &\leq H(0.5\mathbf{S}_0 + 0.5\mathbf{S}_1) \\ &- 0.5H((1-p)\mathbf{S}_0 + p\mathbf{S}_1) - 0.5H((1-p)\mathbf{S}_1 + p\mathbf{S}_0) \\ &+ 0.5(2p-1)^2 H(\mathbf{S}_0) + 0.5(2p-1)^2 H(\mathbf{S}_1) \\ &- (2p-1)^2 H(0.5\mathbf{S}_0 + 0.5\mathbf{S}_1) + (2p-1)^2 H(0.5\mathbf{S}_0 + 0.5\mathbf{S}_1). \end{aligned}$$

The last term is our desired bound. It is left to prove that the remaining terms add up to a non-positive number. Therefore, we need to show that

$$\begin{aligned} &(2p-1)^2 \left[H(0.5\mathbf{S}_0 + 0.5\mathbf{S}_1) - 0.5H(\mathbf{S}_0) - 0.5H(\mathbf{S}_1)\right] \\ &\quad \geq \ H(0.5\mathbf{S}_0 + 0.5\mathbf{S}_1) - 0.5H((1-p)\mathbf{S}_0 + p\mathbf{S}_1) - 0.5H(p\mathbf{S}_0 + (1-p)\mathbf{S}_1). \end{aligned}$$

Let us define

$$c(p) = \max_{\mathbf{S}_0,\mathbf{S}_1} f(\mathbf{S}_0, \mathbf{S}_1),$$

where

$$f(\mathbf{S}_0, \mathbf{S}_1) = \frac{H(0.5\mathbf{S}_0 + 0.5\mathbf{S}_1) - 0.5H((1-p)\mathbf{S}_0 + p\mathbf{S}_1) - 0.5H(p\mathbf{S}_0 + (1-p)\mathbf{S}_1)}{H(0.5\mathbf{S}_0 + 0.5\mathbf{S}_1) - 0.5H(\mathbf{S}_0) - 0.5H(\mathbf{S}_1)}$$

and the maximum is taken over all arbitrary probability vectors $\mathbf{S}_0$ and $\mathbf{S}_1$. If we can show that $c(p) \leq (2p-1)^2$ we are done. We can rewrite the function $f(\mathbf{S}_0, \mathbf{S}_1)$ as

$$f(\mathbf{S}_0, \mathbf{S}_1) = \frac{\sum_{i=1}^{n-1} d_i(1 - h_2(0.5 + \epsilon_i(2p-1)))}{\sum_{i=1}^{n-1} d_i(1 - h_2(0.5 + \epsilon_i))}$$

where $d_i = \frac{S_{0i}+S_{1i}}{2}$ and $\epsilon_i = \frac{S_{0i}-S_{1i}}{2(S_{0i}+S_{1i})}$.

$$\begin{aligned}
f(\mathbf{S}_0, \mathbf{S}_1) &\leq \max_{i\in[1:n-1]} \frac{1 - h_2(0.5 + \epsilon_i(2p-1))}{1 - h_2(0.5 + \epsilon_i)} \\
&\leq \max_{|\epsilon|\leq 0.5} \frac{1 - h_2(0.5 + \epsilon(2p-1))}{1 - h_2(0.5 + \epsilon)} \\
&= \lim_{\epsilon\to 0} \frac{1 - h_2(0.5 + \epsilon(2p-1))}{1 - h_2(0.5 + \epsilon)} \\
&= (2p-1)^2.
\end{aligned}$$

Therefore, $c(p) \leq (2p-1)^2$ which completes the proof.

□

8 A Conjectured Inequality with Applications to Coded Caching

The achievable rate region in [18] is known to be optimal under uncoded placement for all parameters K and N. Restricting Equation (4) in [18] to $N = 2$ and minimizing $R + M$, one can show that

$$R + M = \begin{cases} 2 - \frac{K}{4(K-1)} & \text{if } K \text{ is even} \\ 2 - \frac{K+1}{4K} & \text{if } K \text{ is odd} \end{cases}$$

is achievable for the caching problem with 2 files. Similar inner bound can be established from other achievability results some of which rely on coded placement, for instance [83, 84, 85, 86].

In a parallel line of research [87, 88, 89, 90, 91, 19] significant progress has been made in deriving information-theoretic converse bounds for the coded caching problem. These bounds are inherently different from the counting argument presented in [18], as they establish impossibility results that are not restricted to uncoded placement, but *any* caching technique. Unfortunately however, these bounds offer little when it comes to small choices of N, in particular $N = 2$. Until recently, the best known information theoretic converse bounds for $N = 2$ and arbitrary K were restricted to the following three equations

$$\begin{aligned} R + 2M &\geq 2 \\ R + M &\geq 1.5 \\ 2R + M &\geq 2 \end{aligned}$$

first derived in [2]. In [92] Tian proved that

$$K(K+1)M + 2(K-1)KR \geq 2(K-1)(K+1) \tag{8.1}$$

along with several other converse bounds that do not depend on K as long as $K \geq 4$. In particular,

$$R + M \geq 5/3 \tag{8.2}$$

when $N = 2$ and $K \geq 3$. To this date, this is the tightest known information-theoretic converse bound of the form $R + M \geq f(K)$ for the caching problem with $N = 2$. We propose the following conjecture, implying that the caching scheme in [18] minimizes $R + M$ for $N = 2$ and *arbitrary* K.

Conjecture 8.1. *Suppose (M, R) is an achievable memory-rate pair for the coded caching problem with parameters K and $N = 2$. We must have*

$$R + M \geq \begin{cases} 2 - \frac{K}{4(K-1)} & \textit{if } K \textit{ is even,} \\ 2 - \frac{K+1}{4K} & \textit{if } K \textit{ is odd.} \end{cases}$$

We will make some progress towards proving this conjecture. In particular, we will reduce this to a more intuitive conjecture which we can prove when $K = 3$. This also serves as an alternative proof for Equation (8.2).

To start with, assume we have a random variable S, and K pairs of arbitrary random variables $\{V_i, U_i\}_{i=1}^K$ such that any pair of them completely describe S. In other words, $H(S|U_i, V_i) = 0$ for all $i \in [1 : K]$. What can we say about the minimum value of $\sum_{i \neq j} H(V_i, U_j)$? The corresponding problem in set theory has an easy answer.

Proposition 8.1. *Assume we have a set S of size $|S| = n$. Let $\{V_i, U_i\}_{i=1}^K$ be $2K$ sets that satisfy $S \subseteq V_i \cup U_i$ for all $i \in [1 : K]$. We have the following (tight) inequality.*

$$\frac{1}{n} \sum_{\substack{i \neq j \\ i,j=1}}^{K} |V_i \cup U_j| \geq \begin{cases} K(K-1) - \frac{K^2}{4} & \textit{if } K \textit{ is even,} \\ K(K-1) - \frac{K^2-1}{4} & \textit{if } K \textit{ is odd.} \end{cases}$$

The proof of this proposition follows from a simple counting argument which we omit for conciseness. The question is, does the same inequality hold in the realm of random variables?

Conjecture 8.2. *Suppose we have a random variable S of entropy $H(S) = F$. Let $\{V_i, U_i\}_{i=1}^K$ be $2K$ random variables that satisfy $H(S|V_i, U_i) = 0$ for all $i \in [1 : K]$. We have the following (tight) inequality.*

$$\frac{1}{F} \sum_{\substack{i \neq j \\ i,j=1}}^{K} H(V_i, U_j) \geq \begin{cases} K(K-1) - \frac{K^2}{4} & \textit{if } K \textit{ is even,} \\ K(K-1) - \frac{K^2-1}{4} & \textit{if } K \textit{ is odd.} \end{cases}$$

We will prove that Conjecture 8.1 follows from Conjecture 8.2. Furthermore, we will show that Conjecture 8.2 holds when $K = 3$. The validity of the conjecture for $K = 4$ follows trivially from $K = 3$ (same property holds in general. If the conjecture holds for $K = 2\ell - 1$, it also holds for $K = 2\ell$). The inequality, if true, is of non-Shannon type for $K \geq 5$, as can be verified with Information Theory Inequality Provers such as the one available on xitip.epfl.ch.

Theorem 8.1. *If Conjecture 8.2 is true, so is Conjecture 8.1.*

Theorem 8.2. *Conjecture 8.2 holds when* $K = 3$.

The rest of this chapter is dedicated to proving these two theorems. But first, let us establish a simple lemma which will come handy in our proofs.

Lemma 8.1. *For three arbitrary random variables* X, Y, Z *we have*

$$I(X;Y) + I(X;Z) \leq I(Y;Z) + H(X). \tag{8.3}$$

Proof. After taking everything to the right hand side of the inequality, we have

$$\begin{aligned} & H(X) + I(Y;Z) - I(X;Z) - I(X;Y) \\ & \quad = [H(X|Z) + I(X;Z)] + I(Y;Z) - I(X;Z) - [I(X;Y|Z) + I(X;Y;Z)] \\ & \quad = [H(X|Z) - I(X;Y|Z)] + I(Y;Z) - I(X;Y;Z) \\ & \quad = H(X|Y,Z) + I(Y;Z|X) \geq 0. \end{aligned}$$

□

Proof of Theorem 8.1. Let us denote our two files by A and B. Let X_i represent the delivery message when the i'th user demands A and the remaining $K - 1$ users demand B. Let $\pi : \{1, 2, 3\} \to [1 : K]$ be an arbitrary bijection. For any such bijection we can write

$$H(X_{\pi(1)}) + H(X_{\pi(2)}) + 2H(Z_{\pi(3)}) \quad \leq \quad RF + RF + 2MF.$$

Furthermore,

$$\begin{aligned} H(X_{\pi(1)}) \quad + \quad & H(X_{\pi(2)}) + 2H(Z_{\pi(3)}) \geq H(X_{\pi(1)}, Z_{\pi(3)}) + H(X_{\pi(2)}, Z_{\pi(3)}) \\ \geq \quad & H(X_{\pi(1)}, Z_{\pi(3)}|B) + H(X_{\pi(2)}, Z_{\pi(3)}|B). \end{aligned}$$

where the last step follows from the fact that file B can be recovered from $(X_{\pi(1)}, Z_{\pi(3)})$ as well as from $(X_{\pi(2)}, Z_{\pi(3)})$. As a result of this, we have

$$2(R + M) \geq 2 + \frac{1}{F} H(X_{\pi(1)}, Z_{\pi(3)}|B) + \frac{1}{F} H(X_{\pi(2)}, Z_{\pi(3)}|B). \tag{8.4}$$

Let us average this expression over all possible bijections π. We find

$$K(K-1)(R+M) \geq K(K-1) + \frac{1}{F} \sum_{\substack{i \neq j \\ i,j=1}}^{K} H(X_i, Z_j|B). \tag{8.5}$$

It follows from Conjecture 8.2[1] that since $H(A|X_i, Z_i, B) = 0$ and $H(A|B) = F$, we must have

$$\frac{1}{F} \sum_{\substack{i,j=1 \\ i \neq j}}^{K} H(X_i, Z_j|B) \quad \geq \quad \begin{cases} K(K-1) - \frac{K^2}{4} & \text{if K is even,} \\ K(K-1) - \frac{K^2-1}{4} & \text{if K is odd.} \end{cases} \tag{8.6}$$

It follows from Equations (8.5) and (8.6) that

$$R + M \geq \begin{cases} 2 - \frac{K}{4(K-1)} & \text{if } K \text{ is even,} \\ 2 - \frac{K+1}{4K} & \text{if } K \text{ is odd.} \end{cases} \tag{8.7}$$

□

Proof of Theorem 8.2. We first apply Lemma 8.1 on three random variables U_2, V_1, V_3 as well as on V_1, V_3, U_3.

$$\begin{aligned} I(V_1; V_3) &\geq I(V_1; U_2) + I(V_3; U_2) - H(U_2) \\ I(V_3; U_3) &\geq I(V_1; U_3) + I(V_1; V_3) - H(V_1) \end{aligned}$$

From these two inequalities we conclude

$$\begin{aligned} & I(V_1; U_2) + I(V_1; U_3) + I(V_3; U_2) \\ & \quad \leq \; H(V_1) + H(U_2) + I(V_3; U_3). \end{aligned}$$

Similarly, we can write

$$\begin{aligned} & I(V_2; U_1) + I(V_3; U_1) + I(V_2; U_3) \\ & \quad \leq \; H(V_2) + H(U_1) + I(V_3; U_3). \end{aligned}$$

By summing up the last two inequalities, and following a similar approach we obtain

$$\begin{aligned} \sum_{\substack{i \neq j \\ i,j=1}}^{3} I(V_i; U_j) &\leq H(V_1) + H(U_1) + H(V_2) + H(U_2) + 2I(V_3; U_3), \\ \sum_{\substack{i \neq j \\ i,j=1}}^{3} I(V_i; U_j) &\leq H(V_1) + H(U_1) + H(V_3) + H(U_3) + 2I(V_2; U_2), \\ \sum_{\substack{i \neq j \\ i,j=1}}^{3} I(V_i; U_j) &\leq H(V_2) + H(U_2) + H(V_3) + H(U_3) + 2I(V_1; U_1). \end{aligned}$$

[1] The conditional version of Conjecture 8.2 readily follows from its non-conditional form.

Averaging the last three inequalities, we will have

$$\sum_{\substack{i\neq j\\ i,j=1}}^{3} I(V_i;U_j) \;\leq\; \frac{2}{3}\sum_{i=1}^{3}[H(V_i)+H(U_i)]+\frac{2}{3}\sum_{i=1}^{3} I(V_i;U_i).$$

Substituting $I(V_i;U_j)=H(V_i)+H(U_j)-H(V_i,U_j)$ we find

$$\begin{aligned}\sum_{\substack{i\neq j\\ i,j=1}}^{3} H(V_i,U_j) &\geq \frac{4}{3}\sum_{i=1}^{3}[H(V_i)+H(U_i)]-\frac{2}{3}\sum_{i=1}^{3} I(V_i;U_i)\\ &\geq \frac{4}{3}\sum_{i=1}^{3}[H(V_i,U_i)]\geq\frac{4}{3}\sum_{i=1}^{3}[I(V_i,U_i;S)]\\ &= 4H(S)-\frac{4}{3}\sum_{i=1}^{3}[H(S|V_i,U_i)].\end{aligned}$$

Therefore,

$$\frac{1}{F}\sum_{\substack{i\neq j\\ i,j=1}}^{3} H(V_i;U_j) \;\geq\; 4.$$

□

Bibliography

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

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

[3] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," in *Proceedings of the VLDB Endowment*, vol. 6, no. 5, 2013, pp. 325–336.

[4] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6463–6486, 2011.

[5] M. Ajtai, "The shortest vector problem in L 2 is NP-hard for randomized reductions," in *30th annual ACM symposium on Theory of computing*. ACM, 1998, pp. 10–19.

[6] J. Zhan, B. Nazer, U. Erez, and M. Gastpar, "Integer-forcing linear receivers," *IEEE Transactions on Information Theory*, vol. 60, no. 12, pp. 7661–7685, 2014.

[7] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[8] K. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *47th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2009, pp. 1243–1249.

[9] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5227–5239, 2011.

[10] N. B. Shah, K. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2134–2158, 2012.

[11] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymptotic interference alignment for optimal repair of mds codes in distributed storage," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 2974–2987, 2013.

[12] B. Sasidharan, K. Senthoor, and P. V. Kumar, "An improved outer bound on the storage-repair-bandwidth tradeoff of exact-repair regenerating codes," *arXiv preprint arXiv:1312.6079*, 2013.

[13] N. Prakash and M. N. Krishnan, "The storage-repair-bandwidth trade-off of exact repair linear regenerating codes for the case d= k= n- 1," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 859–863.

[14] C. Tian, "Rate region of the (4, 3, 3) exact-repair regenerating codes," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2013, pp. 1426–1430.

[15] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Transactions on Information Theory*, vol. 60, no. 10, pp. 5843–5855, 2014.

[16] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," *IEEE Transactions on Information Theory*, vol. 62, no. 8, pp. 4481–4493, 2016.

[17] L. Pamies-Juarez, H. D. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2013, pp. 892–896.

[18] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, 2017.

[19] ——, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *arXiv preprint arXiv:1702.04563*, 2017.

[20] I. Tamo and A. Barg, "Bounds on locally recoverable codes with multiple recovering sets," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2014, pp. 691–695.

[21] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 6661–6671, 2016.

[22] A. Wang, Z. Zhang, and M. Liu, "Achieving arbitrary locality and availability in binary codes," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 1866–1870.

[23] P. Huang, E. Yaakobi, H. Uchikawa, and P. H. Siegel, "Linear locally repairable codes with availability," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 1871–1875.

[24] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *48th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2010, pp. 1510–1517.

[25] O. Olmez and A. Ramamoorthy, "Fractional repetition codes with flexible repair from combinatorial designs," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1565–1591, 2016.

[26] N. Prakash, V. Abdrashitov, and M. Médard, "A generalization of regenerating codes for clustered storage systems," in *54th Annual Allerton Conference on Communication, Control, and Computing*, 2018.

[27] ——, "The storage vs repair-bandwidth trade-off for clustered storage systems," *arXiv preprint arXiv:1701.04909*, 2017.

[28] B. Gastón, J. Pujol, and M. Villanueva, "A realistic distributed storage system: the rack model," *arXiv preprint arXiv:1302.5657*, 2013.

[29] J. Pernas, C. Yuen, B. Gastón, and J. Pujol, "Non-homogeneous two-rack model for distributed storage systems," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*. IEEE, 2013, pp. 1237–1241.

[30] M. A. Tebbi, T. H. Chan, and C. W. Sung, "A code design framework for multi-rack distributed storage," in *IEEE Information Theory Workshop (ITW)*. IEEE, 2014, pp. 55–59.

[31] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[32] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig, "Open connect everywhere: A glimpse at the internet ecosystem through the lens of the netflix cdn," *arXiv preprint arXiv:1606.05519*, 2016.

[33] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, 2015.

[34] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *arXiv preprint arXiv:1503.00265*, 2015.

[35] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. Diggavi, "Hierarchical coded caching," in *IEEE International Symposium on Information Theory (ISIT).* IEEE, 2014, pp. 2142–2146.

[36] M. Ji, A. Tulino, J. Llorca, and G. Caire, "Caching-aided coded multicasting with multiple random requests," in *IEEE Information Theory Workshop (ITW).* IEEE, 2015, pp. 1–5.

[37] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," in *IEEE International Conference on Communications (ICC).* IEEE, 2014, pp. 1878–1883.

[38] S. Wang, W. Li, X. Tian, and H. Liu, "Fundamental limits of heterogenous cache," *arXiv preprint arXiv:1504.01123*, 2015.

[39] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," *arXiv preprint arXiv:1404.6560*, 2014.

[40] ——, "Multi-level coded caching," in *IEEE International Symposium on Information Theory (ISIT).* IEEE, 2014, pp. 56–60.

[41] N. Lee, A. G. Dimakis, and R. W. Heath, "Index coding with coded side-information," *IEEE Communications Letters*, vol. 19, no. 3, pp. 319–322, 2015.

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

[43] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[44] D. Micciancio, "The hardness of the closest vector problem with preprocessing," *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 1212–1215, 2001.

[45] ——, "The shortest vector in a lattice is hard to approximate to within some constant," *SIAM journal on Computing*, vol. 30, no. 6, pp. 2008–2035, 2001.

[46] S. Khot, "Hardness of approximating the shortest vector problem in lattices," in *45th Symposium on Foundations of Computer Science (FOCS).* IEEE, 2004, pp. 126–135.

[47] M. Alekhnovich, S. A. Khot, G. Kindler, and N. K. Vishnoi, "Hardness of approximating the closest vector problem with pre-processing," in *46th Symposium on Foundations of Computer Science (FOCS).* IEEE, 2005, pp. 216–225.

[48] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[49] N. Gama and P. Q. Nguyen, "Finding short lattice vectors within mordell's inequality," in *40th annual ACM symposium on Theory of computing.* ACM, 2008, pp. 207–216.

[50] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups.* Springer, 1999, vol. 290.

[51] ——, "Low-dimensional lattices. VI. Voronoi reduction of three-dimensional lattices," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 436, no. 1896, pp. 55–68, 1992.

[52] R. McKilliam and A. Grant, "Finding short vectors in a lattice of voronoi's first kind," in *IEEE International Symposium on Information Theory (ISIT).* IEEE, 2012, pp. 2157–2160.

[53] J. Wen, B. Zhou, W. H. Mow, and X.-W. Chang, "Compute-and-forward protocol design based on improved sphere decoding," *arXiv preprint arXiv:1410.4278*, 2014.

[54] ——, "An efficient algorithm for optimally solving a shortest vector problem in compute-and-forward design," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 6541–6555, 2016.

[55] B. Zhou and W. H. Mow, "A quadratic programming relaxation approach to compute-and-forward network coding design," in *IEEE International Symposium on Information Theory (ISIT).* IEEE, 2014, pp. 2296–2300.

[56] J. Richter, C. Scheunert, and E. Jorswieck, "An efficient branch-and-bound algorithm for compute-and-forward," in *23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC).* IEEE, 2012, pp. 77–82.

[57] L. Wei and W. Chen, "Integer-forcing linear receiver design with slowest descent method," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2788–2796, 2013.

[58] A. Sakzad, J. Harshan, and E. Viterbo, "Integer-forcing mimo linear receivers based on lattice reduction," *IEEE Transactions on Wireless Communications*, vol. 12, no. 10, pp. 4905–4915, 2013.

[59] M. Hejazi and M. Nasiri-Kenari, “Simplified compute-and-forward and its performance analysis,” *IET Communications*, vol. 7, no. 18, pp. 2054–2063, 2013.

[60] J. Wen and X.-W. Chang, “A linearithmic time algorithm for a shortest vector problem in compute-and-forward design,” in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 2344–2348.

[61] Q. Huang and A. Burr, “Low complexity coefficient selection algorithms for compute-and-forward,” in *83rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2016, pp. 1–5.

[62] Q. T. Sun, J. Yuan, T. Huang, and K. W. Shum, “Lattice network codes based on Eisenstein integers,” *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2713–2725, 2013.

[63] W. Liu and C. Ling, “Efficient integer coefficient search for compute-and-forward,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 12, pp. 8039–8050, 2016.

[64] A. Barreal, J. Pääkkönen, D. Karpuk, C. Hollanti, and O. Tirkkonen, “A low-complexity message recovery method for compute-and-forward relaying,” in *IEEE Information Theory Workshop (ITW)*. IEEE, 2015, pp. 39–43.

[65] J. Richter, J. Hejtmánek, E. A. Jorswieck, and J. Sykora, “Non-cooperative compute-and-forward strategies in gaussian multi-source multi-relay networks,” in *82nd Vehicular Technology Conference (VTC Fall)*. IEEE, 2015, pp. 1–5.

[66] B. Zhou and W. H. Mow, “Efficient compute-and-forward design with low communication overhead,” in *22nd Asia-Pacific Conference on Communications (APCC)*. IEEE, 2016, pp. 291–295.

[67] W. Nam, S.-Y. Chung, and Y. H. Lee, “Capacity bounds for two-way relay channels,” in *IEEE International Zurich Seminar on Communications*. IEEE, 2008, pp. 144–147.

[68] M. P. Wilson, K. Narayanan, H. D. Pfister, and A. Sprintson, “Joint physical layer coding and network coding for bidirectional relaying,” *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5641–5654, 2010.

[69] B. Hassibi and H. Vikalo, “On the sphere-decoding algorithm i. expected complexity,” *IEEE transactions on signal processing*, vol. 53, no. 8, pp. 2806–2818, 2005.

[70] R. G. McKilliam, I. V. L. Clarkson, W. D. Smith, and B. G. Quinn, “A linear-time nearest point algorithm for the lattice An*,” in *International*

Symposium on Information Theory and its Applications (ISITA). IEEE, 2008, pp. 1–5.

[71] R. G. McKilliam, W. D. Smith, and I. V. L. Clarkson, "Linear-time nearest point algorithms for coxeter lattices," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 1015–1022, 2010.

[72] J. Martinet, *Perfect lattices in Euclidean spaces.* Springer Science & Business Media, 2013, vol. 327.

[73] J. Zhu and M. Gastpar, "Asymmetric compute-and-forward with CSIT," *arXiv preprint arXiv:1401.3189*, 2014.

[74] J. Saunderson, V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky, "Diagonal and low-rank matrix decompositions, correlation matrices, and ellipsoid fitting," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1395–1416, 2012.

[75] A. Shapiro, "Weighted minimum trace factor analysis," *Psychometrika*, vol. 47, no. 3, pp. 243–264, 1982.

[76] T. Gerstner and M. Holtz, "Algorithms for the cell enumeration and orthant decomposition of hyperplane arrangements," *Working paper, Institute for Numerical Simulation, University of Bonn*, 2006.

[77] N. H. Sleumer, "Output-sensitive cell enumeration in hyperplane arrangements," *Nordic journal of computing*, vol. 6, no. 2, pp. 137–147, 1999.

[78] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, 2003.

[79] G. R. Kumar and T. A. Courtade, "Which boolean functions are most informative?" in *IEEE International Symposium on Information Theory (ISIT).* IEEE, 2013, pp. 226–230.

[80] O. Ordentlich, O. Shayevitz, and O. Weinstein, "Dictatorship is the most informative balanced function at the extremes," *arXiv preprint arXiv:1505.05794*, 2015.

[81] G. Pichler, P. Piantanida, and G. Matz, "Dictator functions maximize mutual information," *arXiv preprint arXiv:1604.02109*, 2016.

[82] V. Chandar and A. Tchamkerten, "Most informative quantization functions," in *IEEE Information Theory and Applications Workshop (ITA), 2014. Available online http://perso.telecom-paristech.fr/tchamker/CTAT.pdf.*

[83] C. Tian and J. Chen, "Caching and delivery via interference elimination," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 830–834.

[84] C. Tian and K. Zhang, "From uncoded prefetching to coded prefetching in coded caching," *arXiv preprint arXiv:1704.07901*, 2017.

[85] J. Gómez-Vilardebó, "Fundamental limits of caching: improved bounds with coded prefetching," *arXiv preprint arXiv:1612.09071*, 2016.

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

[87] C.-Y. Wang, S. H. Lim, and M. Gastpar, "A new converse bound for coded caching," in *IEEE Information Theory and Applications Workshop (ITA)*. IEEE, 2016, pp. 1–6.

[88] C.-Y. Wang, S. S. Bidokhti, and M. Wigger, "Improved converses and gap-results for coded caching," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2428–2432.

[89] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," *arXiv preprint arXiv:1501.06003*, 2015.

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

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

[92] ——, "Symmetry, outer bounds, and code constructions: A computer-aided investigation on the fundamental limits of caching," *arXiv preprint arXiv:1611.00024*, 2016.

Curriculum Vitae

Saeid Sahraei
School of Computer and Communication Sciences
École Polytechnique Fédéderal de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
Email: saeid.sahraei@epfl.ch

Education

2013 – 2018	Docteur ès Sciences, École Polytechnique Fédéderal de Lausanne, Lausanne, Switzerland.
2010 – 2013	M.Sc, École Polytechnique Fédéderal de Lausanne, Lausanne, Switzerland.
2006 – 2010	B.Sc, Sharif University of Technology, Tehran, Iran.

Publications

Journal Papers and Manuscripts

- Saeid Sahraei and Michael Gastpar, "Polynomially Solvable Instances of the Shortest and Closest Vector Problems with Applications to Compute-and-Forward" *IEEE transactions on Information Theory,* vol. 63, no. 12, pp. 7780-7792, 2017.
- Saeid Sahraei and Michael Gastpar, "Increasing Availability in Distributed Storage Systems via Clustering" *submitted to IEEE transactions on Information Theory, September 2017.*
Available at https://arxiv.org/pdf/1710.02653.pdf

Conference Papers

- Saeid Sahraei and Michael Gastpar. "GDSP: A Graphical Perspective on the Distributed Storage Systems." *International Symposium on In-*

formation Theory (ISIT), IEEE, 2017.

- Saeid Sahraei and Michael Gastpar. “Multi-Library Coded Caching.” *54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2016.

- Saeid Sahraei and Michael Gastpar. “K Users Caching Two Files: An Improved Achievable Rate.” *50th Annual Conference on Information Sciences and Systems (CISS)*, IEEE, 2016.

- Saeid Sahraei and Michael Gastpar. “Compute-and-forward: Finding the best equation.” *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2014.

- Saeid Sahraei and Farid Ashtiani. “Effect of power randomization on saturation throughput of IEEE 802.11 WLAN.” *International Conference on Communications (ICC)*, IEEE, 2010.