Error-Correcting Codes for Networks, Storage and Computation

Thesis by

Wael Halbawi

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

Caltech

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2017
Defended June 5, 2017

To my beloved family:
my mama and baba, Alia & Iyad,
and my brother, Anas.

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude and appreciation to my advisor Prof. Babak Hassibi. Babak's constant support has been essential for the completion of this journey. I will always remember our first meeting in early 2014; what was scheduled to be a thirty-minute meeting ended up being a two-hour discussion touching on a variety of topics – both technical and non-technical – and a soothing dose of encouragement. Thank you, Babak, for the freedom, trust and all the interesting conversations.

My sincere thanks go to my thesis committee: Prof. Shuki Bruck, Prof. Alexandros Dimakis, Dr. Dariush Divsalar and Prof. Victoria Kostina. Thank you for taking the time to provide feedback on this thesis. Special thanks are also due to Prof. P. P. Vaidyanathan who was part of my candidacy committee.

I would also like to express my gratitude to Dr. Tracey Ho, who was my advisor when I first joined Caltech; I thank Tracey for giving me the opportunity to study at this wonderful place. The motivation and encouragement of Prof. Joseph Jean Boutros were never absent throughout my undergraduate and graduate studies. Thank you, Joseph, for the many afternoons we spent in your office discussing coding theory, art, music and other things.

During the last few years, I had the pleasure of interacting and working with incredibly talented people: Hongyi Yao, Iwan Duursma, Matthew Thill, Anatoly Khina, Zihan Liu, Son Hoang Dau, Navid Azizan and Fariborz Salehi. A lot of the results reported in this thesis are a direct consequence of our collaborative efforts.

The warmth and friendliness of the Hassibi Lab is solely due to its members: Wei, Matt, Kishore, Christos, Ramya, Ehsan, Navid, Toli, Philipp, Fariborz, Ahmed and Hikmet. Thank you all for maintaining a stimulating and exciting atmosphere in Moore 155. I am lucky to have shared an office first with Christos and then with Fariborz, with whom I became good friends, enjoyed many discussions and shared lots of laughs. The efforts of Shirley and Katie ensured that the group functioned smoothly; thank you for your help and patience.

International orientation is where I met my friends Alex, Carlos, Ruby, Christos and Krishna, with whom I got to discover and enjoy beautiful Pasadena. Thank you for your friendship throughout this journey. I will always cherish the fun times we spent together and will never forget our adventures in different parts of the world.

The last six years would've been very different without George, Noor and Sami. I will always be grateful for your friendship and everything that comes with it: good memories, lots of laughter, lengthy conversations and terrific company. My dearest friends, you have been an integral part of this journey and I cannot emphasize this strongly enough.

At the very end, I owe it all to my family: my mom Alia, my dad Iyad and my brother Anas. Your love and care, during both joyous and tough times, is the reason that I was able to complete this endeavor. Your eternal support is my strength and without it none of this would've been possible. I will never be able to give back what you gave me, but the least that I can do, and I hope I can do more in the future, is to dedicate my work to you.

ABSTRACT

The advent of the information age has bestowed upon us three challenges related to the way we deal with data. Firstly, there is an unprecedented demand for transmitting data at high rates. Secondly, the massive amounts of data being collected from various sources needs to be stored across time. Thirdly, there is a need to process the data collected and perform computations on it in order to extract meaningful information out of it. The interconnected nature of modern systems designed to perform these tasks has unraveled new difficulties when it comes to ensuring their resilience against sources of performance degradation. In the context of network communication and distributed data storage, system-level noise and adversarial errors have to be combated with efficient error correction schemes. In the case of distributed computation, the heterogeneous nature of computing clusters can potentially diminish the speedups promised by parallel algorithms, calling for schemes that mitigate the effect of slow machines and communication delay.

This thesis addresses the problem of designing efficient fault tolerance schemes for the three scenarios just described. In the network communication setting, a family of multiple-source multicast networks that employ linear network coding is considered for which capacity-achieving distributed error-correcting codes, based on classical algebraic constructions, are designed. The codes require no coordination between the source nodes and are end to end: except for the source nodes and the destination node, the operation of the network remains unchanged.

In the context of data storage, *balanced* error-correcting codes are constructed so that the encoding effort required is balanced out across the storage nodes. In particular, it is shown that for a fixed row weight, any cyclic Reed–Solomon code possesses a generator matrix in which the number of nonzeros is the same across the columns. In the balanced and sparsest case, where each row of the generator matrix is a minimum distance codeword, the maximal encoding time over the storage nodes is minimized, a property that is appealing in write-intensive settings. Analogous constructions are presented for a locally recoverable code construction due to Tamo and Barg.

Lastly, the problem of mitigating stragglers in a distributed computation setup is addressed, where a function of some dataset is computed in parallel. Using Reed–Solomon coding techniques, a scheme is proposed that allows for the recovery of the function under consideration from the minimum number of machines possible. The only assumption made on the function is that it is additively separable, which renders the scheme useful in distributed gradient descent implementations. Furthermore, a theoretical model for the run time of the scheme is presented. When the return time of the machines is modeled probabilistically, the model can be used to optimally pick the scheme's parameters so that the expected computation time is minimized. The recovery is performed using an algorithm that runs in quadratic time and linear space, a notable improvement compared to state-of-the-art schemes.

The unifying theme of the three scenarios is the construction of error-correcting codes whose encoding functions adhere to certain constraints. It is shown that in many cases, these constraints can be satisfied by classical constructions. As a result, the schemes presented are deterministic, operate over small finite fields and can be decoded using efficient algorithms.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

1.1 Motivation

The rate at which we are amassing data is unprecedented. Today, sources of data are ubiquitous, and interconnected in complex networks. With that arises a demand requiring us to ensure that data is communicated efficiently, stored reliably and processed robustly. Traditionally, networked solutions allow us to utilize this data in better ways but today, this has become a necessity. For these networks to scale, one would like to employ cheap and low-power devices that are potentially less reliable, while maintaining an effective and predictable overall system. The use of fault-tolerance schemes, such as error-correcting codes, in such settings is necessary to mitigate the performance degradation resulting from these devices, or from the natural consequences that accompany large-scale networked systems.

The notion of error-correcting codes was first formally conceived in late 1940s by Richard Hamming [Ham47; Ham48; Ham50] as a means of introducing redundancy into information to protect it from errors. In his seminal work [Sha48], Shannon showed that arbitrarily reliable communication at the highest possible rates is (in principle) achievable through the use of such error-correcting codes. The setting considered by Shannon is one in which a single transmitter sends a message through a noisy channel to a single receiver. This model is well studied and has found use in modern practice, which also led to the development of efficient error correction techniques throughout the second half of the twentieth century. Such techniques have also served as a basis for network management protocols in which a network of communicating nodes is viewed as a collection of point-to-point links, where data is *routed* across this network with the potential use of an error-correcting code between nodes if necessary. Such methodology is relatively simple to implement, but far from optimal. Indeed, new paradigms revolve around the basic idea that it is beneficial for the nodes in a network to operate on – or code across – the information they receive rather than simply route it.

But as with any new invention, new challenges arise when one adopts a sophisticated scheme such as the one described. Indeed, the fact that nodes code across the data packets they receive leads to catastrophic performance in the presence of errors.

As a result, it is necessary to devise new error correction techniques that are suited to these networks. Furthermore, efficient schemes are always desirable and one would hope for those that minimally affect the underlying operation of the network. Indeed, such properties are essential for them to find place in practice.

In the context of data storage, reliable data centers, each comprising many storage nodes, are central to the operation of modern internet services such as online social networks, cloud computing services, etc. Today, it is common to utilize multiple data centers, possibly spread across geographic distances, to support a single service. Companies resort to several techniques to prevent data loss occurring from events such as disk failures and natural disasters. Replicating information across nodes is one method that is attractive due to its simplicity and efficiency. Nonetheless, replication requires a large storage overhead in that guaranteeing a certain level of robustness requires the most redundancy; one node holds data while the rest are mere copies of it. Using error-correcting codes for data storage, however, is an attractive choice that first appeared in the mid eighties with the introduction of RAID systems[1]. Since then, more and more companies are relying on coding as a solution to combat data loss. In the event of disk/node failure, the lost data is reconstructed by contacting an appropriate set of surviving nodes. Key metrics involved in the design of error-correcting codes for a storage system include:

- Capacity: The amount of information that can be stored.
- Level of resilience: The number of storage nodes allowed to fail before data loss occurs.
- Read complexity: The amount of resources required to read the data stored, in the absence of node failures.
- Write complexity: The amount of resources required to encode the data and store it on the storage nodes.
- Recovery complexity: The amount of resources required to reconstruct erroneous/erased data.

The optimization of these metrics is heavily dependent on the implementation scenario. For some applications, fast read speeds are essential, while in others it is

[1] RAID stands for Redundant Array of Inexpensive Disks. Various modes of this method exist where RAID1 employs replication while RAID2 through RAID6 employ error-correcting codes.

worth sacrificing that for an added level of protection. In another example, nodes might fail at a very low rate allowing the use of a higher capacity coding scheme capable of handling only a small number of errors. Data centers storing frequently read data, “hot data" in modern parlance, require error-correcting codes with high recovery speeds. In archival storage systems such as Amazon Glacier [Ama] and Google Coldline [Goo], one can potentially opt for codes that optimize write complexity thereby reducing energy consumption while ensuring that the code used offers high levels of resilience.

The third area of consideration is that of distributed computation. The growing number of computation problems involving massive datasets is rendering the use of distributed algorithms deployed on computer clusters necessary. In such setup, a job is divided into many tasks which are distributed across many machines to be performed in parallel. Frameworks such as Hadoop [Apaa] and Spark [Apab] have facilitated the adoption of this paradigm in various domains. In principle, the expected gain from parallelizing an algorithm should scale linearly in the number of processors being used. However, there are several factors that usually prohibit this gain. Modern clusters are shared resources that can potentially serve multiple users simultaneously. In this case, the response time for a job submitted by a user heavily depends on the jobs submitted by others. This is especially prevalent in cloud computing services such as Amazon EC2, where lower-tier machines are known to have variable performance. Network congestion and background maintenance services [DB13], amongst others, also contribute to the latency observed in these clusters.

Machines that require a longer-than-expected time to respond after being assigned a task are known as stragglers, and different solutions to alleviate their effect are available. The simplest is that of initiating multiple replicas of the same task and then waiting for the first response. While seemingly simple and attractive, this solution does lead to a waste of resources. Recent methods, inspired by error-correcting codes, preprocess the tasks in a way such that a predetermined number of them are sufficient to recover the solution of problem being solved. Such preprocessing methods have to be efficient so that their utility does not end up eliminating the gains achievable from parallel processing.

In summary, the three scenarios just considered call for the design of error-correcting codes with the following properties:

- **Efficient:** The complexity required for implementing and using the error-correcting code should be low enough not to affect the performance of the underlying sytem, whether it is a communication network, a storage system or a cluster of computers.
- **Predictable:** The fault-tolerance level guaranteed by a code should be consistent. In particular, deterministic guarantees are often desirable in practice.
- **Optimal:** Error-correcting codes that optimally trade-off their design parameters ensure the best utilization of resources.

Main Message

The three scenarios of network communication, data storage and distributed computation considered in this thesis are unified by the problem of constructing linear error-correcting codes with certain structure that are efficient, predictable and optimal. In some cases, this structure arises from constraints inherent to the problem itself. For example, we will see that in the case of network error correction, the topological structure of the network translates to constraints on the encoding functions that define the error-correcting code. In other cases, we find that imposing a certain structure can optimize particular metrics pertinent to the code's performance. This will become evident in the data storage scenario, where structure can be used to speed up the encoding process and facilitate its implementation in a distributed manner. In the case of distributed computation, this structure will provide us with schemes that offer significant speedups compared to traditional parallel computing. We provide explicit constructions of error-correcting codes, based on classical algebraic error-correcting codes, that enjoy efficient encoding and decoding, have deterministic guarantees on their error-correcting capabilities and are optimal given their parameters.

1.2 Background and Literature Review

This section formally introduces the problems considered in this thesis along with some mathematical preliminaries necessary for their treatment.

Error-Correcting Codes

An error-correcting code $\mathcal{C}$ is simply a subset of $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is the finite field of q elements. The parameter n is known as the length of the code and $|\mathcal{C}|$ is its cardinality. If we let $\mathcal{M} = \{M_1, \ldots, M_{|\mathcal{C}|}\}$ be a set of messages, one can define an error-correcting code through an encoding function f that maps a message M_i to a codeword $\mathbf{c}_i \in \mathcal{C}$. The goal then is to convey M_i to a receiver through some noisy channel by transmitting $\mathbf{c}_i$. Formally, one would like to find a decoding function g such that if the receiver sees a corrupted version of $\mathbf{c}_i$,

$$\mathbf{y} = \mathbf{c}_i + \mathbf{e},$$

it is able to reconstruct an estimate of M_i via

$$\hat{M}_i = g(\mathbf{c}_i). \tag{1.1}$$

Let $\mathsf{wt}(\mathbf{x}) = |\{i : x_i \neq 0\}|$ denote the Hamming weight of the vector $\mathbf{x}$. This function induces a metric on $\mathbb{F}_q^n$ via

$$d(\mathbf{x}, \mathbf{y}) := \mathsf{wt}(\mathbf{x} - \mathbf{y}).$$

Guarantees on the code's error correction capabilities are defined in terms of the maximum weight of any $\mathbf{e}$, such that for any M_i, the estimate in (1.1) is equal to the transmitted message. This quantity is a function of what is known as the minimum distance of the code, defined as

$$d(\mathcal{C}) = \min_{\mathbf{x} \neq \mathbf{y} \in \mathcal{C}} d(\mathbf{x}, \mathbf{y}),$$

where at least one of the arguments is nonzero. It follows that if $\mathsf{wt}(\mathbf{e}) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$, then one can recover M_i uniquely.

We consider error-correcting codes that are subspaces of $\mathbb{F}_q^n$. In this case, we can define k as the dimension of $\mathcal{C}$ when viewed as a vector space over $\mathbb{F}_q$. Here, the message set $\mathcal{M}$ can be taken as $\mathbb{F}_q^k$ and one can define the function f through the generator matrix of the code $\mathbf{G} \in \mathbb{F}_q^{k \times n}$. A message $\mathbf{m}$ is now mapped to a codeword by the relation

$$\mathbf{c} = \mathbf{m}\mathbf{G}.$$

Generator matrices in which some k columns form an identity matrix are called *systematic*. These are desirable in certain applications as the message $\mathbf{m}$ appears explicitly in its encoding.

The minimum distance of a linear code $\mathcal{C}$ reduces to

$$d(\mathcal{C}) = \min_{0 \neq \mathbf{c} \in \mathcal{C}} \mathsf{wt}(\mathbf{c}).$$

A fundamental bound due to Singleton is one that relates n, k and $d(\mathcal{C})$ via

$$d(\mathcal{C}) \leq n - k + 1.$$

Codes that achieve this bound with equality are known as maximum-distance separable (MDS) and can correct the largest numbers of errors/erasures. A construction due to Reed and Solomon is one of the most famous [RS60].

Reed–Solomon Codes

We will heavily rely on Reed–Solomon codes in various parts of this thesis. For this reason, we devote this section to describing them and stating properties that are relevant to the results presented.

A Reed–Solomon code $\mathsf{RS}[n,k]$ is defined as the image of the set of polynomials in $\mathbb{F}_q[x]$ of degree less than k evaluated at n distinct elements in $\mathbb{F}_q$, called the coordinates of the code. Formally, we have

$$\mathsf{RS}[n,k] = \{(m(\alpha_1),\dots,m(\alpha_n)) : \mathsf{deg}\,(m(x)) < k\},$$

where each $m(\alpha_i)$ is a codeword *symbol* and $\mathsf{deg}\,(m(x))$ denotes the degree of $m(x)$. With every message $m = (m_0,\dots,m_{k-1})$, we associate the polynomial $m(x) = \sum_{i=0}^{k-1} m_i x^i$, which is then evaluated at the code's coordinates. Indeed, this mapping is linear and so the code's generator matrix is given by

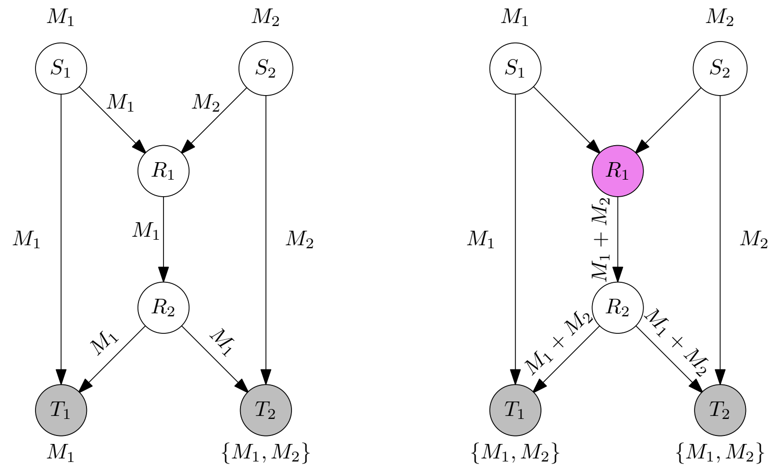
$$\mathbf{G} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{bmatrix}.$$

A Reed–Solomon code is MDS. This is readily seen from the fact that a polynomial of degree $k-1$ cannot have more than $k-1$ roots and so $d(\mathsf{RS}[n,k]) \geq n-k+1$. Combining this with the Singleton bound establishes the claim. For this reason, and many others, they are widely adopted in practice. Sample applications range from consumer technologies such as CDs, DVDs and Blu-ray Discs, to industry standards such as RAID6 and Digital Video Broadcasting (DVB).

More details on error-correcting codes can be obtained by consulting classical texts such as [MS77] and [McE02].

Network Error Correction

Network coding was introduced as a paradigm for operating networks in the seminal work [Ahl+00], where it was shown that traditional routing is insufficient to achieve the capacity of certain networks: the maximum throughput theoretically possible. Their solution, termed Network Coding, allows intermediate nodes in a network to



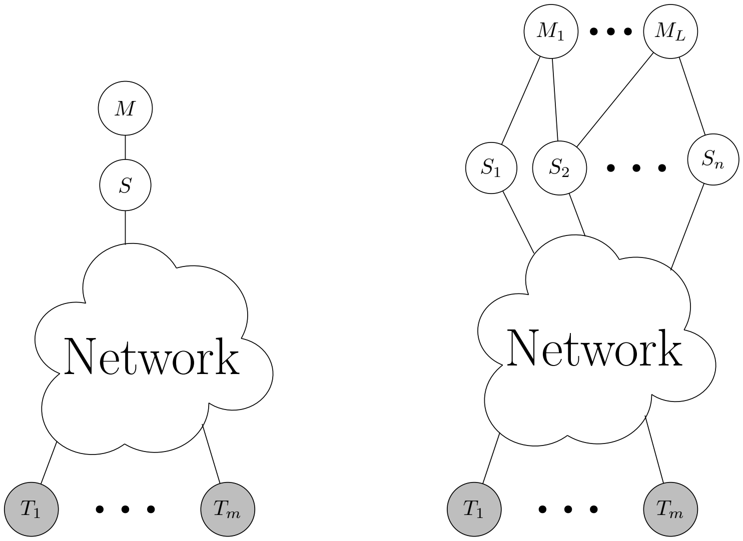
(a) Routing: Relay node R_1 has to decide whether to forward M_1 or M_2. In this case, only one of the two receivers will obtain both messages.

(b) Coding: Relay node R_1 can sum its incoming messages, which enables both receivers to obtain M_1 and M_2.

Figure 1.1: The butterfly network is the canonical example for which network coding is necessary to achieve capacity. Sources S_1 and S_2 carry messages M_1 and M_2, respectively, and each destination node T_i is interested in receiving both messages. Both messages are unit sized and the links in the network can carry one unit of information per time instant. The packet traversing a link is indicated by its label.

operate on the packets they receive before forwarding them to other nodes in the network. For an example, refer to Figure 1.1. From there, significant effort was dedicated to the study of multicast networks (Figure 1.2). In such a network, a single source node is interested in communicating a message to a collection of destinations. It was shown in [LYC03] that linear network coding suffices to achieve the capacity of these networks. Since then several frameworks have been developed to construct capacity-achieving network codes and two results that stand out are those of Ho et al. [Ho+06] and Jaggi et al. [Jag+07].

The extension of the single source multicast problem to one with multiple sources was studied in [Ho+06], where it was shown that if all nodes in the network forward random linear combinations of the packets they receive, the destination nodes can recover the original set of messages with high probability. In a network with L messages, we can model each message M_i as a collection of R_i data packets organized in a matrix $\mathbf{M}_i \in \mathbb{F}_q^{R_i \times n}$, where R_i is the *rate* of message M_i in $\mathbb{F}_q$ symbols per unit time. Due to the linearity of the network code, a receiver collects



(a) Single-source multicast network. (b) Multi-source multicast network.

Figure 1.2: In a single-source multicast network, a collection of receivers are interested in recovering the source's single message. In the multi-source multicast setting, a collection of source nodes jointly hold a set of messages which is to be recovered at each receiver.

N packets obtained through

$$\mathbf{Y} = \sum_{i=1}^{L} \mathbf{T}_i \mathbf{M}_i,$$

where $\mathbf{T}_i \in \mathbb{F}_q^{N \times R_i}$ captures the aggregate effect of the network code on the message M_i.

Naturally, one is interested in scenarios where a network operates in the presence of errors resulting from faulty links or due to an adversary, say. In this case, attainable communication rates are characterized by the capacity (capacity region in the multi-source case) of the network in the presence of z erroneous links. In the single-source multicast setting, where the message M is of size R symbols, the result is known as the network Singleton bound and was presented in [YC06] as

$$R \leq C - 2z.$$

Here, the quanity C denotes the size of the min-cut[2] between the source node and any receiver. The work of Dikaliotis et al. [Dik+10] characterizes the capacity region of multi-source multicast with L messages $\mathcal{M} = \{M_1, \ldots, M_L\}$, each of

[2] In graph theory, the min-cut between two subsets of vertices is the set of edges of smallest cardinality which, once removed, separates them.

rate R_i, as

$$\sum_{M_i \in \mathcal{M}'} R_i \leq C_{\mathcal{M}'} - 2z, \quad \forall \mathcal{M}' \subseteq \mathcal{M}. \tag{1.2}$$

where $C_{\mathcal{M}'}$ is the size of min-cut that separates $\mathcal{M}'$ from any receiver T_j.

Here, the network transfers the messages from sources to a destination node according to

$$\mathbf{Y} = \sum_{i=1}^{L} \mathbf{T}_i \mathbf{M}_i + \mathbf{Z},$$

where the matrix $\mathbf{Z}$ captures the effect of the z error packets introduced into the network. By linearity of the network code, we have that $\mathsf{rank}\left(\mathbf{Z}\right) \leq z$.

The pioneering work of Kötter and Kschischang [KK08] introduced the notion of subspace coding to the realm of single-source multicast network error correction. It was noticed that by encoding the message M in a choice of subspace $\mathcal{V}$ rather than a particular basis representing it, the receiver can decode M if the intersection between $\mathcal{V}$ and the rowspace of $\mathbf{Y}$ is large enough. Silva et al. built on top of this framework and provided constructions in [SKK08] based on Gabidulin codes [Gab85]. A Gabidulin code can be viewed as a linear subspace of matrices $\mathcal{C} \subseteq \mathbb{F}_q^{n \times m}$ where the distance between two codewords is measured using the usual rank function. In particular, the distance between two codewords $\mathbf{C_1}, \mathbf{C}_2$ is given by

$$d_{\mathsf{R}}(\mathbf{C_1}, \mathbf{C}_2) = \mathsf{rank}\left(\mathbf{C_1} - \mathbf{C}_2\right).$$

Hence, the source node maps its message to a codeword $\mathbf{C}$ in a Gabidulin code of dimension R and length n, where n is the min-cut of the network, so that the receiver observes,

$$\mathbf{Y} = \mathbf{T}\mathbf{C} + \mathbf{Z}.$$

Successful decoding occurs if $d_R(\mathbf{Y}, \mathbf{C}) \leq z$. A variety of efficient decoders for Gabidulin codes are known to exist [SK09a; WAS13; Loi06] which allows the receiver node to decode the message M efficiently. Furthermore, the construction just presented is essentially optimal in terms of the field size required. Gabidulin codes exist only when $m \geq n$ and this requirement suffices in this setting.

The picture is a little bit less rosy in the multi-source setting. The direct application of single-source codes as proposed in [JFD09] provides a practical and distributed coding scheme but can only attain a strict subregion of (1.2). On the other hand, a carefully crafted capacity-achieving construction was developed in [Dik+10]. The construction is defined as follows: Source S_i encodes its message using a z-error correction Gabidulin code of length n_i and dimension R_i, over a finite field $\mathbb{F}_{p_i}$. Here $n_i = R_i + 2z$ and $p_i = p^{\prod_{j=1}^{i} n_j}$ for prime p. It is shown that this finite field nesting technique along with a recursive decoding algorithm allows any receiver to recover all messages. Unfortunately, the exponentially growing finite field and the recursive decoding algorithm deem this construction impractical.

This gap between single-source and multi-source network error correction is addressed in the first two chapters of this thesis. In particular, we aim to construct a distributed network error-correcting code that achieves the capacity of multi-source multicast networks. By distributed, we mean that the source nodes encode the message they have independently of others. By carefully selecting the encoding functions at the sources, one would want the packets obtained by the receiver to appear as

$$\mathbf{Y} = \mathbf{T}\tilde{\mathbf{C}} + \mathbf{Z},$$

where $\tilde{\mathbf{C}}$ comes from a single-source Gabidulin code capable of correcting z errors in the rank metric. Furthermore, one would hope for a construction whose underlying field scales slowly in $C_{\mathcal{M}}$, the min-cut of the network. In this way, the receiver, through a single use of a Gabidulin code decoder, can efficiently recover all messages.

The source-message relations (as depicted by Figure 1.2b) dictate certain constraints on the encoding functions that one is allowed to use. Suppose we restrict ourselves to linear encoding functions and consider the example whereby the communication network is depicted in Figure 1.3, where the rate of each M_i is $R_i = 1$.

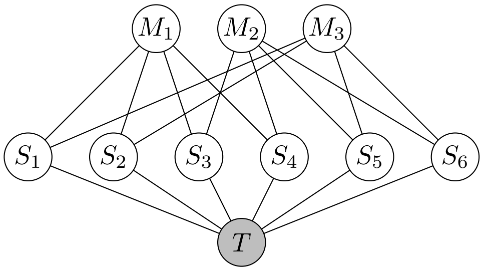


Figure 1.3: In this multi-source multicast network, each source node S_i is connected directly to the receiver T. We assume that the rate of each message M_i is $R_i = 1$.

A distributed error-correcting code possesses a generator matrix of the following form,

$$\mathbf{G} = \begin{bmatrix} g_{1,1} & g_{1,2} & g_{1,3} & g_{1,4} & 0 & 0 \\ 0 & 0 & g_{2,3} & g_{2,4} & g_{2,5} & g_{2,6} \\ g_{3,1} & g_{3,2} & 0 & 0 & g_{3,5} & g_{3,6} \end{bmatrix}.$$

Here, we associate the i^{th} row of $\mathbf{G}$ with M_i and the j^{th} column with S_j. In effect, the node S_j encodes the messages it has access to by linearly combining them using the coefficients in the corresponding column. The destination node receives a collection of six elements of $\mathbb{F}_q$ of which z are corrupted by an adversary. For the destination node to decode correctly, the variables $g_{i,j}$ are carefully selected from $\mathbb{F}_q$ so that the code generated by $\mathbf{G}$ can correct z errors.

The way we select the $g_{i,j}$'s heavily affects the decoding complexity of the code, and so a careful choice is required to accomplish this and ensure optimal error correction capabilities. In chapters 2 and 3, we formalize this framework and provide constructions for certain classes of multicast networks so that efficient decoders can be used at the destination node to recover the messages.

Distributed Data Storage

The introduction of error-correction techniques to the data storage realm began when Patterson et al. invented RAID [PGK88]. In the simplest form of the scheme, known as RAID5, one uses n disks where $n-1$ of them carry information while the last one stores their exclusive-OR (XOR). Such scheme can tolerate any single failure. RAID6 on the other hand, which is based on Reed–Solomon codes, employs two parity disks to tolerate any two failures. In addition to minimum distance, figures of merit by which a code's performance is judged include encoding complexity, decoding complexity and repair complexity. Encoding complexity measures the effort required to compute the parity symbols when the data is encoded. Furthermore, it is also of interest to measure the computation required to modify code symbols when a single data symbol is updated, known as the update complexity. Decoding complexity measures the computational effort exerted when the information stored in the storage system is to be read. Indeed, this depends on whether errors/erasures have occurred, in which case a decoder must be utilized to recover the data. Repair complexity, also commonly known today as *locality*, quantifies the smallest number of code symbols from which any other code symbol can be computed.

With an eye toward constructions with low encoding complexity, Blaum and coauthors introduced the commonly used EVENODD codes in [Bla+95] as array codes[3] that can correct any two erased columns. EVENODD codes are known to have low encoding and decoding complexity; both require $O(m^2)$ additions in $\mathbb{F}_p$ when the codewords are in $\mathbb{F}_p^{(m-1)\times m}$. A generalization of EVENODD was presented

[3] An array code is one in which every codeword can be viewed as a matrix.

in [BBV96] to handle, under some assumptions, up to eight column erasures.

Today's distributed storage systems also use Reed–Solomon codes, given their wider range of parameters. See [Mur+14] and [Clo] for practical examples. Even though these codes can tolerate an arbitrary number of erasures, practitioners have noticed that while storage nodes fail regularly, a small number of them fail at a time. This has led researchers to view error-correcting codes from the lens of *repair*. The first framework proposed to address this problem is that of *regenerating codes*. Here, one assumes that a error-correcting code is used to encode a data file $\mathbf{m} \in \mathbb{F}_q^k$, where q is a prime power, to a codeword $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$, where each c_i is placed on a storage node S_i. This leads to the natural association between the columns of the code's generator matrix and the storage nodes. Now, suppose that a single storage node fails; a question asked is the following: What is the total amount of data that needs to be retrieved from surviving nodes to enable the repair of the failed node? In principle, one can encode the file using an MDS code and then repair a failed node by retrieving data from k other surviving nodes. In the seminal work of Dimakis et al.[DPR06], it was shown that, using certain codes, one can contact more than k nodes but download less data overall, less than the size of the whole data file in particular. This work formulated what became to be known as the storage versus repair-bandwidth trade-off, which characterizes the balance that needs to be struck between the amount of data that needs to be stored per node, versus the amount that needs to be downloaded when a repair job is invoked. Several contributions followed in [Ras+09; RSK11; Sha+12a; Sha+12b]. Due to their ubiquity, recent efforts have been dedicated to the study of repairing Reed–Solomon codes [GW15].

A parallel line of work focuses on the notion *local repair*. First formulated in [OD11] and [Gop+12], the notion of local repair aims to address a similar problem as that of regenerating codes. However, it focuses on the following question: Given an error-correcting code $\mathcal{C}$ of length n and dimension k, what is the smallest number r such that for any codeword, any code symbol can be recovered by accessing r other code symbols? Codes that address this problem are known as locally repairable codes (LRCs). Formally, an LRC $\mathcal{C} \subset \mathbb{F}_q^n$ has locality r if for each $i \in \{1, \ldots, n\}$, there is a subset $\mathcal{I}_i \subset \{1, \ldots, n\} \setminus i$, where $|\mathcal{I}_i| \leq r$ such that $c_i = f_i(\{c_j : j \in \mathcal{I}_i\})$. The coordinates $c_i \cup \{c_j : j \in \mathcal{I}_i\}$ form what is known as a local repair group. A Singleton-type bound of the form

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2 \tag{1.3}$$

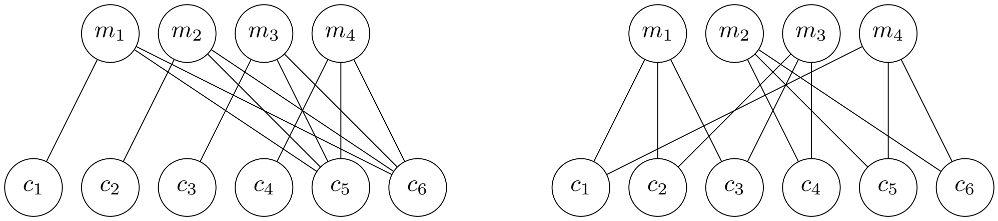
was derived for linear codes in [Gop+12] and later generalized in [PD12] to all codes[4]. The works [Raw+12; Sil+13] provided optimal constructions (with respect to (1.3)) based on Gabidulin codes, where the underlying finite field scales exponentially in n. Tamo and Papailiopoulos presented in [TPD13] a concatenated code construction whose underlying field scaled exponentially in the code dimension. Tamo and Barg [TB14; TB15] presented elegant constructions, which can be viewed as subcodes of Reed–Solomon codes, over finite fields that scale linearly in n.

Once an error-correcting code for a storage system is fixed, it is preferred to have its generator matrix in systematic form. Indeed, in the absence of erasures/errors, a systematic code allows the data collector to read off the message symbols directly without any added read complexity. It is well known, though, that the parity columns in a systematic generator matrix of an MDS code are completely dense, i.e. all entries in these columns have to be nonzero. As an example, a systematic MDS code of length 6 and dimension 4 has a generator matrix of the following mask:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & \times & \times \\ 0 & 1 & 0 & 0 & \times & \times \\ 0 & 0 & 1 & 0 & \times & \times \\ 0 & 0 & 0 & 1 & \times & \times \end{bmatrix}. \tag{1.4}$$

The structure of this mask can be captured by a graph as depicted in Figure 1.4a. While systematic codes are essential in “read" intensive applications, they do require the parity disks to exert maximal encoding effort when computing their symbols. But what if “reads" are not very common? Can we relieve this stress from the parity nodes without sacrificing the code’s error correction capabilities? This question was first considered by Dau et al. [Dau+13] in an abstract context where the authors were interested in constructing *balanced* and *sparsest* generator matrices for MDS codes. For an MDS code of length n and dimension k, this translates to having every row be of weight $n-k+1$ and every column of weight $\left\lfloor \frac{k(n-k+1)}{n} \right\rfloor$ or $\left\lceil \frac{k(n-k+1)}{n} \right\rceil$. The authors showed that a random construction over a large field is possible with high probability. A balanced and sparse generator matrix of an MDS code is indeed suitable for distributed storage systems responsible for “cold" data [Goo; Ama; Fac]. In this setting, the data being stored is of archival nature so it’s barely read. Given this observation, one might consider employing a balanced and sparsest MDS code

[4] In the case of nonlinear codes, it was assumed that the code was of cardinality q^k, where q is the order of the underlying finite field.



(a) Systematic Code: The two parity symbols are connected to all message symbols, as necessary for the code to be MDS.

(b) Balanced Code: Every code symbol is connected to the same number of message symbols, and vice versa.

Figure 1.4: A code's generator matrix can be represented by a bipartite graph on the set of message symbols and the set of code symbols, where an edge between m_i and c_j implies that c_j is (potentially) a function of m_i. The converse to this statement is also true: the absence of an edge implies that c_j cannot be a function of m_i.

in an archival storage system to balance out the encoding effort across all storage nodes. As an example, the generator matrix of an MDS code of length 6 and dimension 4 is given by

$$\mathbf{G} = \begin{bmatrix} \times & \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & \times & \times & \times & 0 & 0 \\ \times & 0 & 0 & 0 & \times & \times \end{bmatrix},$$

and is represented pictorially in Figure 1.4b. It turns out that, with a careful choice of the nonzero entries, the matrix $\mathbf{G}$ can be made to generate a Reed–Solomon code over $\mathbb{F}_7$. In Chapter 5, we describe this process in detail and prove that it is always possible to construct a balanced and sparsest generator matrix for any cyclic Reed–Solomon code. Furthermore, an analogous result for Tamo–Barg codes is provided.

Distributed Computation

Mitigating the effect of straggling machines in a distributed computation setup by utilizing error-correcting codes is a relatively new concept. In comparison to replication-based techniques [WJW14; WJW15], Lee et al. showed in [Lee+16] how substantial speed-ups can be realized by using MDS codes to perform *coded computation*. Formally, suppose that one is interested in computing a function of the form

$$f(\mathcal{X}) := f(x_1, \ldots, x_N), \tag{1.5}$$

where the $x_i \in \mathbb{F}^p$ comprise the given dataset $\mathcal{X} = \{x_1, \ldots, x_N\}$ and $\mathbb{F}$ is an arbitrary field. Certain forms of f facilitate its computation in parallel. In particular, suppose that one can find functions h and g so that

$$f(\mathcal{X}) = h(g(\mathcal{X}_1), \dots, g(\mathcal{X}_k)), \tag{1.6}$$

where the sets $\mathcal{X}_1, \dots, \mathcal{X}_k$ form a disjoint partition of $\mathcal{X}$. To evaluate $f(\mathcal{X})$ using a cluster of n machines, a taskmaster paritions $\mathcal{X}$ into n disjoint partitions and assigns the computation of $g(\mathcal{X}_i)$ to each machine S_i. The machines perform their computation in parallel and send back the results to the taskmaster, which applies the function h on $\{g(\mathcal{X}_i)\}_{i=1}^n$ and recovers $f(\mathcal{X})$. In a practical scenario, the taskmaster has to wait for all n machines to return before it can evaluate $f(\mathcal{X})$. Due to the heterogeneous nature of shared computing clusters, some of the n machines might experience severe delay before their results are sent back to the taskmaster [DB13]. In this case, one can utilize coding by asking the machines to compute the functions in a redundant fashion. In particular, we partition the dataset into $k < n$ partitions and assign to each machine S_i the computation of $\{g(\mathcal{X}_{i_j})\}_{j=1}^w$, where $2 \leq w \leq k$. Then, each S_i sends back to the taskmaster a prescribed linear combination of the form

$$c_i = \sum_{j=1}^{w} b_{i,j} g(\mathcal{X}_{i_j}).$$

The coefficients $b_{i,j}$ are chosen in a way such that once a total of f machines indexed by $l_1, \dots, l_f$ return, the task master can *decode* $\{g(\mathcal{X}_{i_j})\}_{j=1}^w$ from $c_{l_1}, \dots, c_{l_f}$ and apply the function h to compute $f(\mathcal{X})$. Suppose that the computation of interest is the multiplication of the matrix $\mathbf{X}$, corresponding to $\mathcal{X}$, by a vector $\mathbf{y}$. For this application, the functions f, g are

$$f(\mathcal{X}) = \begin{bmatrix} — & x_1^{\mathsf{t}} & — \\ & \vdots & \\ — & x_n^{\mathsf{t}} & — \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix},$$

$$g(\mathcal{X}_j) = \begin{bmatrix} — & x_{(j-1)N/k+1}^{\mathsf{t}} & — \\ & \vdots & \\ — & x_{jN/k}^{\mathsf{t}} & — \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix},$$

and h is simply the identity function. In [Lee+16], it was shown that the function f can be recovered from any k machines by choosing the $b_{i,j}$ as the entries of the generator matrix of an MDS code of length n and dimension k over $\mathbb{F}$. Furthermore, due to the linearity of f, the scheme can be implemented in way such that the taskmaster encodes the matrices $\{\mathbf{X}_j\}_{j=1}^k$ corresponding to $\{\mathcal{X}_j\}_{j=1}^k$ as

$$\mathbf{C}_i = \sum_{j=1}^{k} b_{i,j}\mathbf{X}_j$$

and provides $\mathbf{C}_i$ to S_i along with the vector $\mathbf{y}$. Once S_i returns $\mathbf{C}_i\mathbf{y}$, the taskmaster can recover $\mathbf{X}\mathbf{y}$ from any k such products. The work of [DCG16] presented a scheme in which the resulting matrices $\mathbf{C}_j$ are sparse, hence speeding up the computation at each machine. In this way, each machine performs a single matrix-vector multiplication compared to w of them. In [LMA16a], Li et al. consider coded distributed matrix multiplication in the context of a MapReduce framework, where they established a trade-off between the computational latency and the communication load of such scheme. They further proposed a coding scheme whose performance comes within a multiplicative gap of the fundamental tradeoff. In [Rei+17], coded matrix multiplication is considered in a heterogeneous setting where the machines comprising the cluster are of different capabilities. A coding scheme is proposed that achieves logarithmic (in the total number of machines) speedup when compared to other schemes.

In the context of machine learning, Tandon et al considered in [Tan+16] a distributed gradient descent scenario in the presence of stragglers. Formally, suppose one is interested in fitting a vector of parameters $\beta \in \mathbb{R}^p$ to a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, subject to a loss function L that is *separable* in data, i.e.

$$L(\mathcal{D};\beta) = \sum_{i=1}^{N} \ell(x_i, y_i, \beta),$$

where ℓ is the loss function evaluated at (x_i, y_i). The separability of the function L in the data allows us to write its gradient with respect to β as $\nabla L(\mathcal{D};\beta) = \sum_{i=1}^{N} \nabla \ell(x_i, y_i, \beta)$. The fitting problem can now be solved using gradient descent in a distributed fashion. In [Tan+16], a similar coding-theoretic framework was proposed for the recovery of the full gradient $\nabla L(\mathcal{D};\beta)$ in the presence of stragglers. The proposed scheme, which is based on random MDS codes over $\mathbb{R}$, partitions the dataset $\mathcal{D}$ into n partitions and each machine is provided with a specific subset of size w, determined by the maximum number of stragglers s to be tolerated. For a given s, their scheme achieves a theoretical lower bound on w, thereby minimizing the workload per machine. The taskmaster recovers the full gradient from the fastest $f := n - s$ machines by inverting a matrix of order f, with complexity $O(f^3)$. In such a setting, it is natural to assume a probabilistic model for the response time of each machine and then choose w so that the expected time required to compute $\nabla L(\mathcal{D}; \beta)$ is minimized. Toward this end, we adapt the framework of [Tan+16] and present in Chapter 6 a deterministic coding scheme, along with an $O(f^2)$ complexity decoder, that tolerates the largest number of stragglers for a fixed amount of data assigned to each machine. Furthermore, we introduce a delay model that serves as a guide for selecting the scheme's parameters so that the expected time required to compute $\nabla L(\mathcal{D}, \beta)$ is minimized.

1.3 Summary of Contributions

This thesis aims to provide deterministic and efficient code constructions for three scenarios that are inherently distributed, while maintaining the highest error-correction guarantees theoretically permissible. Technically speaking, we are presented with the challenge of designing error-correcting codes whose encoding functions satisfy certain constraints. In some cases, such as network error correction, these constraints are dictated by the underlying topology, while in others, such as distributed data storage and distributed computation, designing these constraints carefully can result in desirable computational properties. A detailed outline of these contributions are presented in the sequel.

Network Error Correction

In Chapter 2, we study the problem of network error correction in the context of simple multiple access networks (SMANs). In such network, a group of source nodes collectively hold a set of messages and are directly connected, via unit-capacity relay links, to a receiver node. The network operates in the presence of z erroneous relay links. An example of this network is given in Figure 1.3. When the number of messages is $L = 3$, we show that the capacity region of this network, given by (1.2), is achieved by a construction we call distributed Reed–Solomon codes. These are subcodes of classical Reed–Solomon codes whose generator matrices obey the encoding constraints imposed by the SMAN in consideration. Being subcodes of Reed–Solomon codes, they can be decoded by a variety of efficient decoders [WB86; Mas69; GS99]. Furthermore, the finite field over which the codes operate scales linearly in the number of source nodes and so in that respect they are optimal.

In Chapter 3, we provide analogous constructions, based on Gabidulin codes, for general multiple-source multicast networks, when the number of messages is $L = 3$. In particular, this provides the first construction for this setting that is both capacity-achieving and efficient. The proposed scheme requires decoding complexity that is polynomial in the network size and constant in the number of messages, compared to existing techniques where the decoding complexity was exponential in the number of messages. The code is fully distributed and is oblivious to the underlying network code; only the source nodes need to perform additional computation. The decoder at the receiver node remains unchanged and can be chosen as one of the many efficient Gabidulin code decoders [SKK08; SK09a; WAS13; Loi06].

Distributed Data Storage

In Chapter 4, we formalize the problem of coding with encoding constraints, where a given bipartite graph (see Figure 1.4) describes the feasible relations between message symbols and code symbols. A upper bound on the minimum distance of any error-correcting code that obeys the encoding constraints is presented and is shown to be achievable under certain technical conditions. We show that it is possible to achieve this bound using a carefully constructed subcode of any MDS code, and in particular Reed–Solomon codes. When the code is required to be systematic, we refine our bound and show that it is achievable, also using MDS codes.

In Chapter 5, we construct generator matrices for Reed–Solomon codes that are balanced. In this setting, we are provided with the flexibility of designing the mask of the generator matrix so that it exhibits desirable properties. In particular, for any desired row weight w where $n - k + 1 \leq w \leq n - 1$, we show how to construct a generator matrix $\mathbf{G}$ for a cyclic Reed–Solomon code of length n and dimension k so that the following two properties hold:

- Every row is of weight w.
- Every column is of weight $\left\lceil \frac{kw}{n} \right\rceil$ or $\left\lfloor \frac{kw}{n} \right\rfloor$.

Such matrix is called a *balanced* matrix. We emphasize the usefulness of such a construction by advocating a distributed storage system in which the encoding is done distributedly, i.e. every storage node in the system encodes a preallocated set of message symbols independently of the rest. Furthermore, a generator matrix of the form just described ensures that, for any fixed w, the maximal encoding time over all storage nodes is minimized. As a result, such a scheme is computationally balanced as no storage node behaves like a bottleneck. While our codes are not systematic, the update complexity is still minimal for any chosen w, i.e. updating one message symbol impacts exactly w storage nodes. From a theoretical standpoint, it is worth mentioning that our results generalize the setup of [Dau+13] and provide efficient constructions for it. We also present balanced Tamo–Barg codes for cyclic versions of the code, where w can be selected from a wide range of parameters. All of the presented constructions operate over the same finite field of the underlying codes, namely $\mathbb{F}_q$ where $q \geq n$, and so the computational efficiency remains intact.

Distributed Computation

In Chapter 6, we consider a cluster of n machines assigned the task of computing the gradient $\nabla L(\mathcal{D}; \beta)$ of a loss function L with respect to a parameter vector β, where the dataset $\mathcal{D}$ is partitioned into k disjoint sets $\{\mathcal{D}_i\}_{i=1}^{k}$. We construct a deterministic coding scheme, based on Reed–Solomon codes, that facilitates this computation in the presence of stragglers. Our scheme differs from others in that it is determined by a pair (w, k), where w is the number of data partitions assigned to each machine. Given such a pair, we carefully construct a balanced matrix that prescribes the coding coefficients for each machine. For every (w, k), our scheme achieves a lower bound, due to Tandon et al. [Tan+16], on the number of machines $f(w, k, n)$ needed for recovering $\nabla L(\mathcal{D}; \beta)$. An algorithm is developed to recover $\nabla L(\mathcal{D}, \beta)$ in $O(f^2)$ time and, contrary to previous works, avoids computing an entire matrix inverse; our decoder is also efficient in space.

Furthermore, we propose a delay model that incorporates the decoding complexity and from there, we provide an expression in terms of w, k and n for the expected time required to compute $\nabla L(\mathcal{D}; \beta)$. The expression accounts for the time spent by each machine for computation (and encoding), the delay in the response time modeled as a Pareto random variable, and the decoding time. The expression can be numerically optimized to find the minimizing w and k. Since w and k determine the fraction of data given to a machine, our setup provides a practical way to design schemes that ensure minimal response time, when the machines utilized have to adhere to a memory constraint. We supplement our theoretical findings with numerical simulations and validate the effectiveness of the proposed scheme.

Chapter 2

DISTRIBUTED REED–SOLOMON CODES

2.1 Introduction

We consider the problem of error correction in multi-source multicast networks. In general, such network is one where a set of source nodes are to communicate, with the aid of a network, a set of messages to a collection of destination nodes. We restrict ourselves to *simple multiple access networks* (SMAN), a family of networks which we formalize in this chapter and can capture various network error correction and key distribution scenarios. In a SMAN, a single destination node wishes to reconstruct a set of messages available at the source nodes with the aid of a single layer of relay nodes (see Figure 2.1). Each relay node can communicate with a subset of the source nodes, and is connected to the destination by a unit[1] capacity link. We wish to design a distributed code that can correct arbitrary errors on up to z links that connect the relay nodes to the destination. Equivalently, one wishes to be resilient to z erroneous relay nodes. This problem has been considered previously by [YHN11] in the context of decentralized distribution of keys from a pool, where it was shown to be a special case of the general multiple-access network error-correction problem, whose capacity region was established in [Dik+10]. It also applies generally to other distributed data storage/retrieval scenarios where different nodes store different subsets of the source messages.

In this chapter, we introduce a framework for constructing a computationally efficient coding scheme, based on Reed–Solomon codes, for correcting errors in SMANs. Specifically, the relay nodes encode the messages they have access to in a distributed fashion such that the destination node receives a set of code symbols which, when viewed as a single vector, form a codeword from a single Reed–Solomon code. This permits the destination node to recover the desired messages efficiently using standard decoders designed for Reed–Solomon codes [Mas69; WB86]. This scheme obviates the need for encoding over successively larger nested finite fields at each source as in the prior construction of [Dik+10]. We prove that the proposed coding scheme achieves the full capacity region for such networks with up to three sources.

[1] We are considering an algebraic setting so a unit corresponds to the size of one element from the underlying finite field.

Related Work

The multi-source multicast problem was first introduced by Dikaliotis et al. in [Dik+10]. Under the assumption of adverserial errors, the capacity region for such networks was established, and an error correction scheme based on Gabidulin codes [Gab85].

Closer to the realm of this work, the work of [Dau+13], motivated by distributed sensor networks, introduced the notion of balanced and sparsest MDS codes. There, one is interested in construction MDS codes with generator matrices that are sparse and have balanced column weights. The work of Dau et al. in [DSY14; DSY15] analyzes, amongst others, the problem considered in this chapter and conjecture that Reed–Solomon codes attain the capacity region of SMANs for any number of sources.

Also closely related is the work of Yan et al. in [YS11; YSZ14], in which the data exchange problem under weak security is considered. In the network considered, the nodes, each of which holds a subset of messages, communicate cooperatively, via error-free broadcast transmissions, in order to disseminate all messages subject to the presence of an eavesdropper. The existence of a secure transmission scheme is proven, using probabilistic arguments, by showing how to construct an MDS code subject to sparsity constraints, similar to the ones we consider, over an exponentially growing finite field.

2.2 Preliminaries

We begin this section by presenting a formal definition of an SMAN. Next, we show how the problem of constructing a linear error-correcting code for such a network is algebraically equivalent to building a structured generator matrix for a code capable of correcting z errors. The required structure is identified by the graph defining the SMAN. We will consider algebraic codes, Reed–Solomon codes in particular, defined over $\mathbb{F}_q$, the finite field of q elements.

Simple Multiple Access Networks

Definition 2.1. *A simple multiple access network is represented by a bipartite graph $G = (\mathcal{S}, \mathcal{V}, \mathcal{E})$, where the set of source nodes is $\mathcal{S} = \{S_1, \dots, S_m\}$, the set of relay nodes is $\mathcal{V} = \{v_1, \dots, v_n\}$ and the edge set is given by $\mathcal{E}$. A source node S_i is connected to a relay node v_j, via a relay link, if and only if $(i, j) \in \mathcal{E}$. Each S_i carries a message M_i of size r_i symbols from $\mathbb{F}_q$ and the destination node D is interested in retrieving all M_i's by connecting to all relay nodes.*

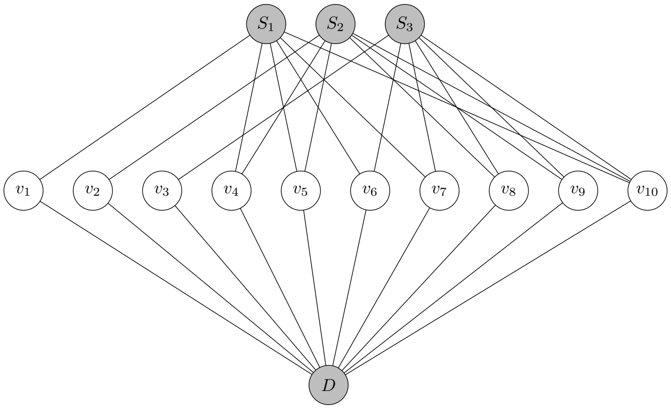


Figure 2.1: An example of an SMAN with 3 source nodes and 10 relay nodes.

An adjacency matrix $\mathbf{A}$ is associated with an SMAN, where the rows and columns represent $\mathcal{S}$ and $\mathcal{V}$, respectively, and $\mathbf{A}_{i,j} = 1$ if and only if $(i, j) \in \mathcal{E}$. An example of an SMAN with three source nodes and ten relay nodes is given in Figure 2.1. The corresponding adjacency matrix is

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

We consider an SMAN in which z of the links connecting the relays to the destination are erroneous. For each $\mathcal{S}' \subseteq \mathcal{S}$, the minimum cut capacity (min-cut) from $\mathcal{S}'$ to D is denoted by $C_{\mathcal{S}'}$. From [Dik+10], the capacity region $\mathcal{R}$ of an SMAN is given by cut set bounds for each subset of sources.

Theorem 2.1 (Dikaliotis et al. [Dik+10])**.** *The capacity region of an SMAN is the set of all rate vectors* $\mathbf{r} = (r_1, r_2, \ldots, r_m)$ *such that*

$$\sum_{S_i \in \mathcal{S}'} r_i \leq C_{\mathcal{S}'} - 2z, \forall \mathcal{S}' \subseteq \mathcal{S}. \tag{2.2}$$

Furthermore, it suffices to carry out linear coding at internal network nodes, where each v_i *transmits linear combinations of received symbols over* $\mathbb{F}_q$*.*

It is easy to see that in a SMAN, the min-cut $C_{\mathcal{S}'}$ is given by the number of relay nodes that are connected to at least one node in $\mathcal{S}'$. Formally, we have $C_{\mathcal{S}'} = |\{v_j \in \mathcal{V} : \exists S_i \in \mathcal{S}', (i, j) \in \mathcal{E}\}|$.

Having characterized the capacity region of an SMAN, we are interested in constructing codes that can achieve every rate vector in this region. The main result of this chapter ensures that Reed–Solomon codes can be used to do so when the network has three sources.

Problem Formulation

Given an SMAN defined by $G = (\mathcal{S}, \mathcal{V}, \mathcal{E})$, one is interested in constructing a distributed error-correcting code that can achieve the capacity region of the network in the presence of z erroneous relay links. Here, a distributed code is one in which each relay node encodes the information symbols it has access to independently of the others. Since each S_i carries an r_i sized message, this can be represented a vector $\mathbf{m}_i \in \mathbb{F}_q^{1\times r_i}$. The encoding symbol associated with v_j is now given by

$$c_j = \sum_{i:(i,j)\in\mathcal{E}} \mathbf{m}_i \mathbf{g}_{i,j}, \tag{2.3}$$

where $\mathbf{g}_{i,j} \in \mathbb{F}_q^{r_i}$ is the encoding vector that v_j uses to encode $\mathbf{m}_i$. Collectively, these vectors can organized in a matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \vdots \\ \mathbf{G}_m \end{bmatrix} = \left[\begin{array}{cccc} \mathbf{g}_{1,1} & \mathbf{g}_{1,2} & \cdots & \mathbf{g}_{1,n} \\ \hline \mathbf{g}_{2,1} & \mathbf{g}_{2,2} & \cdots & \mathbf{g}_{2,n} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{g}_{m,1} & \mathbf{g}_{m,2} & \cdots & \mathbf{g}_{m,n} \end{array}\right]. \tag{2.4}$$

Some of these vectors will necessarily be equal to the zero vector of appropriate size. In particular, if $[\mathbf{A}]_{i,j} = 0$, then $\mathbf{g}_{i,j} = \mathbf{0}$. As an example, the generator matrix corresponding to $\mathbf{A}$ in (2.1) is given below.

$$\mathbf{G} = \left[\begin{array}{cccccccccc} \mathbf{g}_{1,1} & \mathbf{0} & \mathbf{0} & \mathbf{g}_{1,4} & \mathbf{g}_{1,5} & \mathbf{g}_{1,6} & \mathbf{g}_{1,7} & \mathbf{0} & \mathbf{0} & \mathbf{g}_{1,10} \\ \hline \mathbf{0} & \mathbf{g}_{2,2} & \mathbf{0} & \mathbf{g}_{2,4} & \mathbf{g}_{2,5} & \mathbf{0} & \mathbf{0} & \mathbf{g}_{2,8} & \mathbf{g}_{2,9} & \mathbf{g}_{2,10} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{g}_{3,3} & \mathbf{0} & \mathbf{0} & \mathbf{g}_{3,6} & \mathbf{g}_{3,7} & \mathbf{g}_{3,8} & \mathbf{g}_{3,9} & \mathbf{g}_{3,10} \end{array}\right].$$

This formulation captures the entire coding operation performed in the network. Indeed, if we collect the message vectors as $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_m)$, then the destination node receives

$$\mathbf{y} = \mathbf{c} + \mathbf{e},$$

where $\mathbf{c} = \mathbf{m}\mathbf{G} \in \mathbb{F}_q^n$ is the concatenation of the code symbols transmitted by the relay nodes. The goal now is to recover $\mathbf{m}$ from $\mathbf{y}$ when the weight of $\mathbf{e}$ is at most z. Furthermore, we would like to do this for any rate vector $(r_1, \ldots, r_m)$ in the capacity region of G. The main result of this chapter shows that when $m = 3$, the matrix $\mathbf{G}$ can be specified in a way such that it spans a subcode of a z-error correcting Reed–Solomon code. The underlying field $\mathbb{F}_q$ of the code is size-optimal, meaning we require q to be the smallest prime power greater than n. As a result, the destination node can efficiently recover the the source messages using classical Reed–Solomon decoders.

Distributed Reed–Solomon Codes

To construct a distributed Reed-Solomon code for a given SMAN with n intermediate relay nodes and z corrupt relay links/nodes, we start with a Reed–Solomon code $\mathsf{RS}[n,k]$ of length n and minimum distance $2z+1$, so that the dimension of this code is $k = n - 2z$. As described earlier, the adjacency matrix of the network specifies constraints on the generator matrix of the distributed code. Each row of $\mathbf{G}_i \in \mathbb{F}_q^{r_i \times n}$ in (2.4) has the sparsity pattern of $\mathbf{A}_i$, the i^{th} row of $\mathbf{A}$. For each S_i, the main task then is to find r_i codewords in $\mathsf{RS}[n,k]$ that have the same support as $\mathbf{A}_i$. Once these codewords are collected in $\mathbf{G}_i$ for each i, we would like the overall matrix in (2.4) to be of rank $r := \sum_{i=1}^{m} r_i$. This will imply that these codewords span an r-dimensional subcode of $\mathsf{RS}[n,k]$ which is also z-error correcting.

Main Result

The main result of this chapter is proving the existence of distributed Reed–Solomon codes, over a small finite field, for any SMAN with three source nodes. More precisely, we prove the following theorem.

Theorem 2.2 (Main Result)**.** *Let $G = (\mathcal{S}, \mathcal{V}, \mathcal{E})$ be an SMAN with three sources, in which z relay links are erroneous. Then, a distributed Reed–Solomon code over $\mathbb{F}_q$ exists capable of correcting z errors, where $q \geq |\mathcal{V}|$ is sufficient.*

The theorem is proved via a series of results presented in the next section. In particular, we will characterize three cases under which any SMAN is categorized. Each case is treated separately and a code construction is provided. Together, three main propositions unite to form the basis of the main theorem.

2.3 Construction

By relying on the polynomial nature of Reed–Solomon codes, we can identify each codeword with a polynomial of degree less than k. Let $\alpha \in \mathbb{F}_q$ be a primitive element and associate the j^{th} column of $\mathbf{A}$ with α^j. One can form a polynomial $p_i(x)$ that vanishes on α^j whenever $\mathbf{A}_{i,j} = 0$. Once this polynomial is sampled at $\mathcal{A} = \{\alpha, \ldots, \alpha^n\}$, the resulting codeword has the same support as $\mathbf{A}_i$.

Observation 2.1. *Suppose that* $[\mathbf{A}]_{i,j} = 0$ *for* $j \in \mathcal{R}_i$ *and let* $p_i(x) = \prod_{j \in \mathcal{R}_i}(x - \alpha^j)$. *Then* $p_i(\alpha^j) = 0$ *if and only if* $[\mathbf{A}]_{i,j} = 0$. *In other words, the codeword* $\mathbf{c}$ *corresponding to* $p_i(x)$ *has the same support as* $\mathbf{A}_i$.

This observation allows us to focus on polynomials rather than codewords, and reformulate the capacity region (2.2). Therefore, for each $i = 1, \dots, m$, our goal is to find a set of r_i polynomials $\mathcal{T}_i$ each of degree less than k and each of which has $p_i(x)$ as a factor. Furthermore, the polynomials in $\cup_{i=1}^{m} \mathcal{T}_i$ are linearly independent over $\mathbb{F}_q$.

Reformulating the Capacity Region of an SMAN

Definition 2.2. *The set of polynomials of degree less than* k *in which each element vanishes on* $\mathcal{R} \subseteq \{\alpha, \dots, \alpha^n\}$ *is denoted by* $\mathsf{poly}_k(\mathcal{R})$.

Since this set is a linear space over $\mathbb{F}_q$, we state its dimension.

Lemma 2.1. *The dimension of* $\mathsf{poly}_k(\mathcal{R})$ *over* $\mathbb{F}_q$ *is* $\mathsf{dim}(\mathsf{poly}_k(\mathcal{R})) = (k - |\mathcal{R}|)^+$.

Proof. Indeed, when $|\mathcal{R}| \geq k$, the only element in $\mathsf{poly}_k(\mathcal{R})$ is the zero polynomial. Otherwise, observe that any $p(x) \in \mathcal{P}_i$ can be written as $c(x)g_j(x)$ where $c(x)$ is the minimal polynomial of $\mathcal{R}_i$. Then, deg $(g_i(x)) < k - |\mathcal{R}_i|$. Any set of polynomials $\{b_1(x), \dots, b_r(x)\}$ is linearly independent if and only if $\{g_1(x), \dots, g_r(x)\}$ is linearly independent. When considered as vectors of $k - |\mathcal{R}_i|$ entries, the maximum number of linearly independent such vectors is then precisely $k - |\mathcal{R}_i|$, and in this case achieved by taking $g_i(x) = x^i$, for $i \in \{0, \dots, k - |\mathcal{R}_i| - 1\}$. □

Given an SMAN defined by $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$, we can associate every $v_j \in \mathcal{V}$ with the α^j. With this association, we require a polynomial corresponding to a row of $\mathbf{G}_i$ (see (2.4)) to vanish on all those α^j's for which S_i is not connected to v_j, i.e. with every S_i we associate a set of roots $\mathcal{R}_i$, corresponding to those relay nodes that are not connected to S_i. Note that C_i can now be expressed as

$$C_i = |\{v_j : (i,j) \in \mathcal{E}\}| = |\mathcal{V}| - |\{v_j : (i,j) \notin \mathcal{E}\}| = n - |\mathcal{R}_i|.$$

Similarly, we have

$$C_{\mathcal{S}'} = n - |\{v_j : \forall S_i \in \mathcal{S}', (i,j) \notin \mathcal{E}\}| = n - |\cap_{S_i \in \mathcal{S}'} \mathcal{R}_i|.$$

As a result, the capacity region of an SMAN is given by

$$\sum_{S_i \in \mathcal{S}'} r_i \leq k - | \cap_{S_i \in \mathcal{S}'} \mathcal{R}_i |, \forall \mathcal{S}' \subseteq \mathcal{S}, \tag{2.5}$$

where we have used the fact that $k = n - 2z$. We can now formulate the problem of finding a capacity-achieving code for an SMAN with z erroneous links to one where we require m subsets of polynomials $\mathcal{T}_1, \ldots, \mathcal{T}_m$ where $|\mathcal{T}_i| = r_i$ and $\cup_{i=1}^m \mathcal{T}_i$ is a linearly independent set over $\mathbb{F}_q$, and the cardinalities obey (2.5). Furthermore, each polynomial in $\mathcal{T}_i$ vanishes on $\mathcal{R}_i = \{\alpha^j : [\mathbf{A}]_{i,j} = 0\}$ and is of degree less than k. We will study how the linear spaces generated by the $\mathcal{T}_i$'s interact and then provide the main results. To this end, we define $\mathcal{P}_i := \mathsf{poly}_k\left(\mathcal{R}_i\right)$.

Fact 2.1. *Given $\mathcal{P}_i, \mathcal{P}_j$, we have $\mathcal{P}_i \cap \mathcal{P}_j = \mathsf{poly}_k\left(\mathcal{R}_i \cup \mathcal{R}_j\right)$.*

Proposition 2.1. *Let $k \geq | \cup_{i=1}^n \mathcal{R}_i |$. Then $\langle \mathcal{P}_1, \ldots, \mathcal{P}_n \rangle = \mathsf{poly}_k\left(\mathcal{R}_1 \cap \cdots \cap \mathcal{R}_n\right)$.*

Proof. We demonstrate the proof using induction. For the base case, fix $n = 2$. Clearly, we have $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle \subseteq \mathsf{poly}_k\left(\mathcal{R}_1 \cap \mathcal{R}_2\right)$. Let $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2|$ and express $\mathsf{dim}\left(\langle \mathcal{P}_1, \mathcal{P}_2 \rangle\right)$ as follows,

$$\begin{aligned}
\mathsf{dim}\left(\langle \mathcal{P}_1, \mathcal{P}_2 \rangle\right) &= \mathsf{dim}\left(\mathcal{P}_1\right) + \mathsf{dim}\left(\mathcal{P}_2\right) - \mathsf{dim}\left(\mathcal{P}_1 \cap \mathcal{P}_2\right) \\
&= k - |\mathcal{R}_1| + k - |\mathcal{R}_2| - \left(k - |\mathcal{R}_1 \cup \mathcal{R}_2|\right)^+ \\
&= k - \left(|\mathcal{R}_1| + |\mathcal{R}_2| - |\mathcal{R}_1 \cup \mathcal{R}_2|\right) \\
&= k - |\mathcal{R}_1 \cap \mathcal{R}_2| \\
&= \mathsf{dim}\left(\mathsf{poly}_k\left(\mathcal{R}_1 \cap \mathcal{R}_2\right)\right).
\end{aligned}$$

Thus, we assert that $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle = \mathsf{poly}_k\left(\mathcal{R}_1 \cap \mathcal{R}_2\right)$. For the inductive step, let $k \geq | \cup_{i=1}^{n+1} \mathcal{R}_i |$ and assume that $\langle \mathcal{P}_1, \ldots, \mathcal{P}_n \rangle = \mathsf{poly}_k\left(\mathcal{R}_1 \cap \cdots \cap \mathcal{R}_n\right)$. We know that

$$\mathsf{dim}\left(\langle \langle \mathcal{P}_1, \ldots, \mathcal{P}_n \rangle, \mathcal{P}_{n+1} \rangle\right) = \mathsf{dim}\left(\mathsf{poly}_k\left(\left(\mathcal{R}_1 \cap \cdots \cap \mathcal{R}_n\right) \cap \mathcal{R}_{n+1}\right)\right),$$

so we conclude that $\langle \mathcal{P}_1, \ldots, \mathcal{P}_{n+1} \rangle = \mathsf{poly}_k\left(\mathcal{R}_1 \cap \cdots \cap \mathcal{R}_n + 1\right)$. □

Corollary 2.1. *Suppose $k \geq | \cup_{i=1}^n \mathcal{R}_i |$. Then, we have that*

$$\mathcal{P}_n \cap \langle \mathcal{P}_1, \ldots, \mathcal{P}_{n-1} \rangle = \langle \mathcal{P}_1 \cap \mathcal{P}_n, \ldots, \mathcal{P}_{n-1} \cap \mathcal{P}_n \rangle.$$

Proof. Fact 2.1 implies $\mathcal{P}_n \cap \langle \mathcal{P}_1, \ldots, \mathcal{P}_{n-1} \rangle = \mathsf{poly}_k\left(\mathcal{R}_n \cup \left(\cap_{i=1}^{n-1} \mathcal{R}_i\right)\right)$. This is exactly $\langle \mathcal{P}_1 \cap \mathcal{P}_n, \ldots, \mathcal{P}_{n-1} \cap \mathcal{P}_n \rangle$ after interpreting it as a polynomial space using Proposition 2.1. □

Definition 2.3. *Let $\mathcal{P}_1, \mathcal{P}_2$ be subspaces. Define $\mathcal{P}_1 \backslash \mathcal{P}_2$ as the unique subspace that is a direct sum complement of $\mathcal{P}_1 \cap \mathcal{P}_2$ in $\mathcal{P}_1$, i.e. $\mathcal{P}_1 = (\mathcal{P}_1 \backslash \mathcal{P}_2) \oplus (\mathcal{P}_1 \cap \mathcal{P}_2)$.*

The following two facts follow easily from the definition.

Fact 2.2. *Let $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle$ denote the Minkowski sum of $\mathcal{P}_1$ and $\mathcal{P}_2$. Then, $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle = (\mathcal{P}_1 \backslash \mathcal{P}_2) \oplus \mathcal{P}_2$.*

Fact 2.3. $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle = (\mathcal{P}_1 \backslash \mathcal{P}_2) \oplus (\mathcal{P}_2 \backslash \mathcal{P}_1) \oplus (\mathcal{P}_1 \cap \mathcal{P}_2)$.

Let us see how these observations allow us to construct a distributed Reed–Solomon code for an SMAN with two sources.

Code Construction for SMAN with Two Source Nodes

Proposition 2.2 (two-source distributed Reed–Solomon code)**.** *Let $G(\mathcal{S}, \mathcal{V}, \mathcal{E})$ define an SMAN with two source nodes in which z relay links are erroneous. Let $\mathcal{R}_i$ correspond to the relay nodes in $\mathcal{V}$ not connected to S_i. Then, for any rate pair (r_1, r_2) such that*

$$\begin{aligned} r_1 &\leq k - |\mathcal{R}_1|, \\ r_2 &\leq k - |\mathcal{R}_2|, \\ r_1 + r_2 &\leq k - |\mathcal{R}_1 \cap \mathcal{R}_2|, \end{aligned}$$

there exists a distributed Reed–Solomon code capable of correcting up to z errors.

Proof. Let $k = n - 2z$. First, assume that $k < |\mathcal{R}_1 \cup \mathcal{R}_2|$. By Lemma 2.1 and Fact 2.1, we have that $\mathcal{P}_1 \cap \mathcal{P}_2$ is trivial. Furthermore, we have $\mathsf{dim}\,(\mathcal{P}_i) = k - |\mathcal{R}_i|$ so $r_i \leq \mathsf{dim}\,(\mathcal{P}_i)$. Let $\mathcal{T}_i$ be a set of r_i linearly independent polynomials chosen from $\mathcal{P}_i$. Let $\mathbb{F}_q$ be such that $q \geq n$. Let $\mathcal{A} = \{\alpha, \ldots \alpha^n\}$ where $\alpha \in \mathbb{F}_q$ is primitive. Consider the set of vectors $\mathcal{G}_i = \{(p(\alpha), \ldots, p(\alpha^n)) : p(x) \in \mathcal{T}_i\} \subseteq \mathsf{RS}[n, k]$. By Observation 2.1, for each vector in $\mathcal{G}_i$, the entries corresponding to $\mathcal{R}_i$ are equal to 0. As a result, when organized as the rows $\mathbf{G}_i$, each row of this matrix has the support of $\mathbf{A}_i$. Since $\mathcal{P}_1 \cap \mathcal{P}_2$ is trivial, we know that $\mathcal{T}_1 \cup \mathcal{T}_2$ span an $\mathbb{F}_q$-linear space of dimension $r_1 + r_2$ and so $\mathsf{rank}\,(\mathbf{G}) = r_1 + r_2$.

Now consider the case when $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2|$. Here, we can utilize Fact 2.1 to infer that $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle = \mathsf{poly}_k\,(\mathcal{R}_1 \cap \mathcal{R}_2)$, so $\mathsf{dim}\,(\langle \mathcal{P}_1, \mathcal{P}_2 \rangle) = k - |\mathcal{R}_1 \cup \mathcal{R}_2|$. We can write the cut-set bounds as

$$\begin{aligned} r_1 &\leq \mathsf{dim}\left(\mathcal{P}_1 \backslash \mathcal{P}_2\right) + \mathsf{dim}\left(\mathcal{P}_1 \cap \mathcal{P}_2\right), \\ r_2 &\leq \mathsf{dim}\left(\mathcal{P}_2 \backslash \mathcal{P}_1\right) + \mathsf{dim}\left(\mathcal{P}_1 \cap \mathcal{P}_2\right), \\ r_1 + r_2 &\leq \mathsf{dim}\left(\mathcal{P}_1 \backslash \mathcal{P}_2\right) + \mathsf{dim}\left(\mathcal{P}_2 \backslash \mathcal{P}_1\right) + \mathsf{dim}\left(\mathcal{P}_1 \cap \mathcal{P}_2\right). \end{aligned}$$

For the first two bounds, we have utilized Definition 2.3 whereas the last bound follows from Fact 2.3. Assume that both $r_1 > \mathsf{dim}\left(\mathcal{P}_1 \backslash \mathcal{P}_2\right)$ and $r_2 > \mathsf{dim}\left(\mathcal{P}_2 \backslash \mathcal{P}_1\right)$. Otherwise, if for example $r_1 \leq \mathsf{dim}\left(\mathcal{P}_1 \backslash \mathcal{P}_2\right)$ holds, then we can choose $\mathcal{T}_1$ as a set of r_1 linearly independent polynomials from $\mathcal{P}_1 \backslash \mathcal{P}_2$ and $\mathcal{T}_2$ as a set of r_2 polynomials from $\mathcal{P}_2$ and guaranteeing that $\mathcal{T}_1 \cup \mathcal{T}_2$ span a linear space of dimension $r_1 + r_2$. Now let $r_1 = \mathsf{dim}\left(\mathcal{P}_1 \backslash \mathcal{P}_2\right) + \delta_1$ and $r_2 = \mathsf{dim}\left(\mathcal{P}_2 \backslash \mathcal{P}_1\right) + \delta_2$. It follows from the third cut-set bound that $\delta_1 + \delta_2 \leq \mathsf{dim}\left(\mathcal{P}_1 \cap \mathcal{P}_2\right)$. As a result, we let $\mathcal{T}_1$ be a set of polynomials comprising a basis for $\mathcal{P}_1 \backslash \mathcal{P}_2$ and δ_1 linearly independent polynomials from $\mathcal{P}_1 \cap \mathcal{P}_2$. Similarly, the set $\mathcal{T}_2$ comprises a basis for $\mathcal{P}_2 \backslash \mathcal{P}_1$ and set of δ_2 linearly independent polynomials from $\mathsf{dim}\left(\mathcal{P}_1 \cap \mathcal{P}_2\right)$ not chosen in $\mathcal{T}_2$. By Fact 2.3, it follows that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a linearly independent set of polynomials. The generator matrix $\mathbf{G}$ is built analogously as earlier and is full rank, so the proof is complete. □

While it might seem that method used to construct a distributed Reed–Solomon code for two-source SMANs is more complicated that possibly necessary, the technique used to handle the three-source case heavily relies on it.

Code Construction for SMANs Where $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$

We prelude by presenting key results that generalize Facts 2.1 and 2.3. In particular, we will show how to decompose all possible Minkowski sums of $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ in a way that allows us to formulate the code construction problem into one based on resource allocation. The decompositions will further allow us to express the capacity region 2.5 in a form that is directly related to the dimensions of the derived spaces.

Lemma 2.2. *Let $\mathcal{P}_i = \mathsf{poly}_k\left(\mathcal{R}_i\right)$ and suppose $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$. Then,*

$$\begin{aligned} \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle = &\mathcal{P}_1 \backslash \langle \mathcal{P}_2, \mathcal{P}_3 \rangle \oplus \mathcal{P}_1 \backslash \langle \mathcal{P}_2, \mathcal{P}_3 \rangle \oplus \mathcal{P}_3 \backslash \langle \mathcal{P}_1, \mathcal{P}_2 \rangle \\ &\oplus (\mathcal{P}_1 \cap \mathcal{P}_2) \backslash \mathcal{P}_3 \oplus (\mathcal{P}_1 \cap \mathcal{P}_3) \backslash \mathcal{P}_2 \oplus (\mathcal{P}_2 \cap \mathcal{P}_3) \backslash \mathcal{P}_3 \\ &\oplus (\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3). \end{aligned}$$

Proof. By Fact 2.3, we have $\mathcal{P}_1 = \mathcal{P}_1\backslash\left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle \oplus \mathcal{P}_1 \cap \left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle$. Furthermore, Proposition 2.1 implies $\mathcal{P}_1 \cap \left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle = \left\langle \mathcal{P}_1 \cap \mathcal{P}_2, \mathcal{P}_1 \cap \mathcal{P}_3\right\rangle$, where another application of Fact 2.3 yields

$$\left\langle \mathcal{P}_1 \cap \mathcal{P}_2, \mathcal{P}_1 \cap \mathcal{P}_3\right\rangle = (\mathcal{P}_1 \cap \mathcal{P}_2)\backslash(\mathcal{P}_1 \cap \mathcal{P}_3) \oplus (\mathcal{P}_1 \cap \mathcal{P}_3)\backslash(\mathcal{P}_1 \cap \mathcal{P}_2) \oplus \mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3.$$

Furthermore, note that $\mathcal{P}_1 \cap \mathcal{P}_2 = (\mathcal{P}_1 \cap \mathcal{P}_2)\backslash(\mathcal{P}_1 \cap \mathcal{P}_3) \oplus (\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3)$ and $\mathcal{P}_1 \cap \mathcal{P}_2 = (\mathcal{P}_1 \cap \mathcal{P}_2)\backslash\mathcal{P}_3 \oplus (\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3)$ which implies that $(\mathcal{P}_1 \cap \mathcal{P}_2)\backslash(\mathcal{P}_1 \cap \mathcal{P}_3) = (\mathcal{P}_1 \cap \mathcal{P}_2)\backslash\mathcal{P}_3$. Likewise, we have $(\mathcal{P}_1 \cap \mathcal{P}_3)\backslash(\mathcal{P}_1 \cap \mathcal{P}_2) = (\mathcal{P}_1 \cap \mathcal{P}_3)\backslash\mathcal{P}_2$. This allows us to conclude that

$$\mathcal{P}_1 = \mathcal{P}_1\backslash\left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle \oplus (\mathcal{P}_1 \cap \mathcal{P}_2)\backslash\mathcal{P}_3 \oplus (\mathcal{P}_1 \cap \mathcal{P}_3)\backslash\mathcal{P}_2 \oplus (\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3).$$

Similarly, one obtains

$$\begin{aligned}\mathcal{P}_2 &= \mathcal{P}_2\backslash\left\langle \mathcal{P}_1, \mathcal{P}_3\right\rangle \oplus (\mathcal{P}_1 \cap \mathcal{P}_2)\backslash\mathcal{P}_3 \oplus (\mathcal{P}_2 \cap \mathcal{P}_3)\backslash\mathcal{P}_1 \oplus (\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3);\\ \mathcal{P}_3 &= \mathcal{P}_3\backslash\left\langle \mathcal{P}_1, \mathcal{P}_2\right\rangle \oplus (\mathcal{P}_1 \cap \mathcal{P}_3)\backslash\mathcal{P}_2 \oplus (\mathcal{P}_2 \cap \mathcal{P}_3)\backslash\mathcal{P}_1 \oplus (\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3).\end{aligned}$$

It now follows that

$$\begin{aligned}\left\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\right\rangle =& \mathcal{P}_1\backslash\left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle + \mathcal{P}_1\backslash\left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle + \mathcal{P}_3\backslash\left\langle \mathcal{P}_1, \mathcal{P}_2\right\rangle\\ &+(\mathcal{P}_1 \cap \mathcal{P}_2)\backslash\mathcal{P}_3 + (\mathcal{P}_1 \cap \mathcal{P}_3)\backslash\mathcal{P}_2 + (\mathcal{P}_2 \cap \mathcal{P}_3)\backslash\mathcal{P}_3 \qquad (2.6)\\ &+(\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3),\end{aligned}$$

where it remains to prove that this decomposition is a direct sum. Let $p_{\mathcal{I}}$ denote an element in $(\cap_{i\in\mathcal{I}}\mathcal{P}_i)\backslash\left\langle \{\mathcal{P}_{i'}\}_{i'\notin\mathcal{I}}\right\rangle$. For example, $p_1 \in \mathcal{P}_1\backslash\left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle$ and $p_{1,2,3} \in (\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3)$ Suppose that the polynomial $p(x) \in \left\langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\right\rangle$ can be expressed in two different ways. This implies that $\sum_{\mathcal{I}\subseteq\{1,2,3\}} p_{\mathcal{I}} = 0$ where not all $p_{\mathcal{I}}$ are zero. Without loss of generality, suppose $p_1 \neq 0$. Note that all the remaining $p_{\mathcal{I}}$'s are such that $\mathcal{I}\cap\{2,3\} \neq \emptyset$. As a result, we conclude that $p_1 = -\sum_{\mathcal{I}\cap\{2,3\}\neq\emptyset} p_{\mathcal{I}} \in \left\langle \mathcal{P}_1, \mathcal{P}_2\right\rangle$. Thus we obtain a contradiction since $p_1 \in \mathcal{P}_1\backslash\left\langle \mathcal{P}_2, \mathcal{P}_3\right\rangle$, so it must be that $p_1 = 0$. Similarly, we have $p_2 = 0$ and $p_3 = 0$. Now suppose, again without of generality, that $p_{1,2} \neq 0$. Then, we have $-p_{1,2} = p_{1,3} + p_{2,3} + p_{1,2,3} \in \mathcal{P}_3$, again a contradiction since $p_{1,2} \notin \mathcal{P}_3$. Theremore, we deduce $p_{1,2} = p_{1,3} = p_{2,3} = 0$. Lastly, it follows that $p_{1,2,3} = 0$ so we conclude that the decomposition in (2.3) is indeed a direct sum. □

This lemma allows us to express the polynomial spaces of interest as decompositions of subspaces that will aid us in constructing distributed Reed–Solomon codes.

Proposition 2.3. *Let* $\mathcal{P}_i = \mathsf{poly}_k\left(\mathcal{R}_i\right)$ *and suppose* $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$. *For each* $\mathcal{I} \subseteq \{1,2,3\}$, *define* $\mathcal{D}_{\mathcal{I}} = (\cap_{i\in\mathcal{I}}\mathcal{P}_i)\backslash \left\langle \{\mathcal{P}_{i'}\}_{i'\notin\mathcal{I}}\right\rangle$ *with* $d_{\mathcal{I}} = \mathsf{dim}\left(\mathcal{D}_{\mathcal{I}}\right)$. *Then,*

$$\begin{aligned}
\mathcal{P}_1 &= \mathcal{D}_1 \oplus \mathcal{D}_{1,2} \oplus \mathcal{D}_{1,3} \oplus \mathcal{D}_{1,2,3},\\
\mathcal{P}_2 &= \mathcal{D}_2 \oplus \mathcal{D}_{1,2} \oplus \mathcal{D}_{2,3} \oplus \mathcal{D}_{1,2,3},\\
\mathcal{P}_3 &= \mathcal{D}_3 \oplus \mathcal{D}_{1,3} \oplus \mathcal{D}_{2,3} \oplus \mathcal{D}_{1,2,3},\\
\langle\mathcal{P}_1, \mathcal{P}_2\rangle &= \mathcal{D}_1 \oplus \mathcal{D}_2 \oplus \mathcal{D}_{1,2} \oplus \mathcal{D}_{1,3} \oplus \mathcal{D}_{2,3} \oplus \mathcal{D}_{1,2,3},\\
\langle\mathcal{P}_1, \mathcal{P}_3\rangle &= \mathcal{D}_1 \oplus \mathcal{D}_3 \oplus \mathcal{D}_{1,2} \oplus \mathcal{D}_{1,3} \oplus \mathcal{D}_{2,3} \oplus \mathcal{D}_{1,2,3},\\
\langle\mathcal{P}_2, \mathcal{P}_3\rangle &= \mathcal{D}_2 \oplus \mathcal{D}_3 \oplus \mathcal{D}_{1,2} \oplus \mathcal{D}_{1,3} \oplus \mathcal{D}_{2,3} \oplus \mathcal{D}_{1,2,3},\\
\langle\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\rangle &= \mathcal{D}_1 \oplus \mathcal{D}_2 \oplus \mathcal{D}_3 \oplus \mathcal{D}_{1,2} \oplus \mathcal{D}_{1,3} \oplus \mathcal{D}_{2,3} \oplus \mathcal{D}_{1,2,3}.
\end{aligned}$$

Furthermore, the capacity region (2.5) *can be written as*

$$\begin{aligned}
r_1 &\leq d_1 + d_{1,2} + d_{1,3} + d_{1,2,3}, && (2.7)\\
r_2 &\leq d_2 + d_{1,2} + d_{2,3} + d_{1,2,3}, && (2.8)\\
r_3 &\leq d_3 + d_{1,3} + d_{2,3} + d_{1,2,3}, && (2.9)\\
r_1 + r_2 &\leq d_1 + d_2 + d_{1,2} + d_{1,3} + d_{2,3} + d_{1,2,3}, && (2.10)\\
r_1 + r_3 &\leq d_1 + d_3 + d_{1,2} + d_{1,3} + d_{2,3} + d_{1,2,3}, && (2.11)\\
r_2 + r_3 &\leq d_2 + d_3 + d_{1,2} + d_{1,3} + d_{2,3} + d_{1,2,3}, && (2.12)\\
r_1 + r_2 + r_3 &\leq d_1 + d_2 + d_3 + d_{1,2} + d_{1,3} + d_{2,3} + d_{1,2,3}. && (2.13)
\end{aligned}$$

Proof. Proving the validity of each listed subspace decomposition follows directly from Lemma 2.2. As for the the new form of the capacity region, the first three bounds follow by definition since $d_i = k - |\mathcal{R}_i|$. Since we have assumed that $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$, the remaining bounds follow by invoking Proposition 2.1. □

The statement of Proposition 2.3 provides intuition for choosing the polynomials $\mathcal{T}_i$ (and hence the codewords) for each $\mathbf{G}_i$ to construct a distributed Reed–Solomon code. The main idea is to start by allocating a basis for $\mathcal{D}_i$ to each $\mathcal{T}_i$, then linearly independent polynomials from $\mathcal{D}_{i,j}$ are shared by $\mathcal{T}_i$ and $\mathcal{T}_j$, if necessary, and so on.

The rest of this section is devoted to proving the main result of this chapter. First, we will use the machinery developed thus far to provide a proof for the case where $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$. Later on, a separate technique will be used to handle the complementary case.

Proposition 2.4. *Let $G = (\mathcal{S}, \mathcal{V}, \mathcal{E})$ be an SMAN with three sources, in which z relay links are erroneous. Let $\mathcal{R}_i$ be the set of nodes not connected to S_i, and assume that $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$. Then, a distributed Reed–Solomon code exists capable of correcting z errors.*

Proof. First, assume without loss of generality that $r_1 + r_2 \leq d_1 + d_2 + d_{1,2}$. Using Proposition 2.2, we can choose linearly independent sets $\mathcal{T}_1$, $\mathcal{T}_2$ from $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_{1,2}$, so that $|\mathcal{T}_i| = r_i$ and the $\mathbb{F}_q$-span of $\mathcal{T}_1 \cup \mathcal{T}_2$ is of dimension $r_1 + r_2$. What remains is to select a set of r_3 linearly independent polynomials $\mathcal{T}_3$ from $\mathcal{P}_3$. Note that the previous allocations intersect trivially with $\mathcal{P}_3$ so we can select from it r_3 linearly independent polynomials to form $\mathcal{T}_3$. Furthermore, the fact that $r_1 + r_2 \leq d_1 + d_2 + d_{1,2}$ and (2.9) imply (2.11), (2.12) and (2.13) ensuring that the bounds are not violated. We conclude that the $\mathbb{F}_q$-span of $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ is of dimension $r_1 + r_2 + r_3$ and so the the corresponding codewords form a generator matrix for a distributed Reed–Solomon code.

Henceforth, we assume $r_i + r_j > d_i + d_j + d_{i,j}$ for $i, j \in \{1, 2, 3\}$. The allocation now is straightforward. Let $\rho_{1,2} = r_1 + r_2 - (d_1 + d_2 + d_{i,j})$ be the number of polynomials still needed for $\mathcal{T}_1$ and $\mathcal{T}_2$ after allocating a basis of $\mathcal{D}_1$ to $\mathcal{T}_1$, a basis of $\mathcal{D}_2$ to $\mathcal{T}_2$, and splitting a basis of $\mathcal{D}_{1,2}$ (arbitrarily) amongst both $\mathcal{T}_1$ and $\mathcal{T}_2$. Furthermore, let $\rho_3 = r_3 - d_3$ be the number of polynomials still needed for $\mathcal{T}_3$ after allocating to it a basis of $\mathcal{D}_3$. We can express the relevant cut-set bounds (2.9), (2.12) and (2.13) now as

$$\begin{aligned}
\rho_3 &\leq d_{1,3} + d_{2,3} + d_{1,2,3}, \\
\rho_{1,2} &\leq d_{1,3} + d_{2,3} + d_{1,2,3}, \\
\rho_{1,2} + \rho_3 &\leq d_{1,3} + d_{2,3} + d_{1,2,3}.
\end{aligned}$$

After allocating enough polynomials from $\mathcal{D}_{1,3}$, $\mathcal{D}_{2,3}$ and $\mathcal{D}_{1,2,3}$ to $\mathcal{T}_3$ so that $|\mathcal{T}_3| = r_3$, the third inequality ensures that both $\mathcal{T}_1$ and $\mathcal{T}_2$ can be completed to bases of sizes r_i and r_2, respectively. As previously seen, this can be done while ensuring that the dimension of the $\mathbb{F}_q$-span of $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ is $r_1 + r_2 + r_3$. This concludes the proof for this special case. □

A remark is in order. The assumption that $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$ is critical for this part of the proof. Indeed, it allowed us to establish Proposition 2.1 and relate the dimensions of the constituent spaces in each decomposition to the capacity

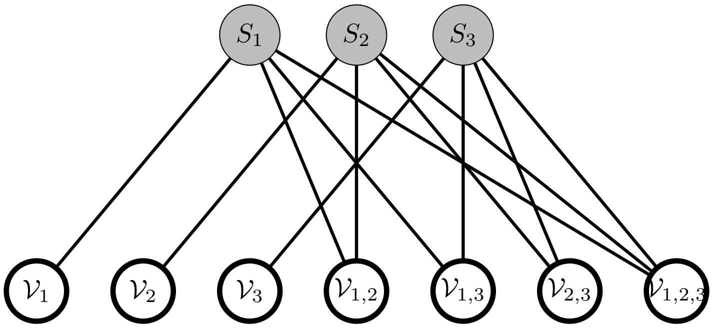


Figure 2.2: A condensed representation of an SMAN with three source nodes. Here, relay nodes connected to the same subset of source nodes are coalesced together and indexed by that subset. For example, the vertex $\mathcal{V}_{1,3}$ includes all relay nodes connected to S_1 and S_3 only.

region (2.5). We resort to different techniques that enable us to construct a distributed Reed–Solomon code for an SMAN in which $k < |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$.

Code Construction for SMANs Where $k < |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$

When restricted to those with three sources, any SMAN can be represented by a graph of the form in Figure 2.2. Here, the relay nodes are considered as subsets of $\mathcal{V}$, where they are grouping according to how they are connected to the sources. In particular, each relay node in $\mathcal{V}_1$ is connected only to S_1, each relay node in $\mathcal{V}_{2,3}$ is connected only to both S_2 and S_3, and so on. With this view in mind, we can express the generator matrix of *any* SMAN with three sources as follows:

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \mathbf{G}_3 \end{bmatrix} = \left[\begin{array}{c|c|c|c|c|c|c} \times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \times & \mathbf{0} & \times & \mathbf{0} & \times & \times \\ \mathbf{0} & \mathbf{0} & \times & \times & \times & \mathbf{0} & \times \end{array}\right]. \tag{2.14}$$

The various subsets of the relays correspond naturally to the columns of $\mathbf{G}$. The relays in $\mathcal{V}_1$ are indentified with the first block of columns. Indeed, these are the relays that have access to S_1's message only. The fifth block of columns identifies $\mathcal{V}_{1,3}$ as these are the relays that have access to both S_1 and S_3, only.

Remember that each $\mathbf{G}_i$ comprises r_i rows. The form of the matrix in (2.14) allows us to make the following observation. If, without loss of generality, we have that $r_1 \leq |\mathcal{V}_1|$, then one can construct a distributed Reed–Solomon code by placing an identity matrix (of size r_1) in the columns of $\mathbf{G}_1$ correponding to $\mathcal{V}_1$. In this way, the matrix $\mathbf{G}_2$ and $\mathbf{G}_3$ can be populated with appropriate codewords using Proposition 2.2.

Proposition 2.5. *Let $G = (\mathcal{S}, \mathcal{V}, \mathcal{E})$ be an SMAN with three sources, in which z relay links are erroneous. Let $\mathcal{V}_i \subseteq \mathcal{V}$ be the set of relay nodes connected to S_i only. If for some $i \in \{1, 2, 3\}$ we have $r_i \leq |\mathcal{V}_i|$, then a distributed Reed–Solomon code exists capable of correcting z errors.*

Proof. For ease of exposition, let $n_{\mathcal{I}} = |\mathcal{V}_{\mathcal{I}}|$. Without loss of generality, suppose $r_1 \leq n_1$. Identify the j^{th} column of $\mathbf{G}$ with α^j. Now, let $\mathcal{R}_1 = \{j : [\mathbf{A}]_{1,j} = 0\}$ and define $p(x) = \prod_{j \in \mathcal{R}_1}(x - \alpha^j)$ where α is primitive in $\mathbb{F}_q$. In words, the polynomial $p(x)$ vanishes on those powers of α that correspond to the relay nodes not connected to S_1. Finally, consider the set of polynomials

$$\mathcal{T}_{1,b} = \left\{ t_j(x) := p(x) \prod_{\substack{i=1 \\ i \neq j}}^{r_1} (x - \alpha^i) : j = 1, \ldots, r_1 \right\}. \tag{2.15}$$

When the polynomials in $\mathcal{T}_{1,b}$ are evaluated on $\mathcal{A} = \{\alpha, \ldots, \alpha^n\}$, the resulting codewords can be organized in a matrix to form

$$\mathbf{G}_1 = \begin{bmatrix} t_1(\alpha) & \cdots & t_1(\alpha^n) \\ t_2(\alpha) & \cdots & t_2(\alpha^n) \\ \vdots & \ddots & \vdots \\ t_{r_1}(\alpha) & \cdots & t_{r_1}(\alpha^n) \end{bmatrix} = \left[\begin{array}{ccccc|c|c|c|c|c|c} \star & 0 & \cdots & 0 & \times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ 0 & \star & \cdots & 0 & \times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \vdots & \vdots & \ddots & \vdots & \boldsymbol{\vdots} & \boldsymbol{\vdots} & \boldsymbol{\vdots} & \boldsymbol{\vdots} & \boldsymbol{\vdots} & \boldsymbol{\vdots} & \boldsymbol{\vdots} \\ 0 & 0 & \cdots & \star & \times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \end{array}\right]. \tag{2.16}$$

Here, the symbol $\star$ represents some nonzero element from $\mathbb{F}_q$, and the symbol $\times$ represents a vector of such elements. When confined to $\{\alpha, \ldots, \alpha_{r_1}\}$, the polynomial $t_j(x)$, by construction, is nonzero only when evaluated at α^j. Hence, the first r_1 columns of $\mathbf{G}_1$ form a diagonal matrix of the presented form and so it is immediate that $\mathbf{G}_1$ is full rank. Now the columns indexed by $\mathcal{R}_1$ are those corresponding to $\mathcal{V}_2 \cup \mathcal{V}_3 \cup \mathcal{V}_{2,3}$. These zeros are put in place since $p(x)$, a factor of each $t_j(x)$, was constructed to vanish on $\mathcal{R}_1$. As a result, every row of $\mathbf{G}_1$ adheres to the constraints imposed by the SMAN. Next, we verify that the polynomials are at most of degree $k - 1$. This is established by the following reasoning:

$$\begin{aligned} \mathsf{deg}\,(t_j(x)) &= |\mathcal{R}_1| + r_1 - 1 \\ &= n - C_1 + r_1 - 1 \\ &\leq n - 2z - 1 \\ &= k - 1. \end{aligned}$$

Here, we have used the fact that $C_1 = |\mathcal{V}| - |\mathcal{R}_1| = n - |\mathcal{R}_1|$. Furthemore, the cut-set bounds (2.2) imply the inequality and k is defined as $k = n - 2z$.

Next, we argue that $\mathbf{G}_2$ and $\mathbf{G}_3$ can be treated using Proposition 2.2. It suffices to show a global solution for $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ constitutes a solution for $\mathbf{G}_1$ as just described, along with any valid solution for $\mathbf{G}_2$ and $\mathbf{G}_3$ subject to

$$\begin{aligned} r_2 &\le C_2 - 2z, \\ r_3 &\le C_3 - 2z, \\ r_2 + r_3 &\le C_{2,3} - 2z, \end{aligned}$$

not violating the global cut-set bound

$$r_1 + r_2 + r_3 \le n - 2z.$$

Indeed, it is easily verified that $n - n_1 = n - |\mathcal{R}_2 \cap \mathcal{R}_3|$, which by definition is equal to $C_{2,3}$. As a result, the fact that $r_1 \le n_1$ and $r_2 + r_3 \le C_{2,3} - 2z$ hold for any valid solution for $\mathbf{G}_2$ and $\mathbf{G}_3$ proves the claim. □

Given this proposition, it remains to prove the existence of distributed Reed–Solomon codes for the case where $r_i > n_i$. With an eye toward that goal, we restate, for the reader's convenience, the classical BCH bound.

Fact 2.4 (BCH bound)**.** *Let $p(x)$ be a polynomial in $\mathbb{F}_q[x]$ (not divisible by $x^{q-1} - 1$ and suppose $p(x)$ vanishes at $\alpha^{b+1}, \dots, \alpha^{b+t}$, where $\alpha \in \mathbb{F}_q$ is primitive. Then, all the coefficients of $p(x)$ are nonzero.*

A proof can be found in [McE86, p.238] The roots of $p(x)$ are *consecutive* powers of α which allows us to make the following observation.

Lemma 2.3. *Suppose $p(x) = \prod_{j=1}^{t}(x - \alpha^j)$. Then, the roots of $p(\alpha^{-l}x)$ are precisely $\{\alpha^{l+1}, \dots, \alpha^{l+t}\}$.*

Proof. Note that $p(\alpha^{-l}x) = \prod_{j=1}^{t}(\alpha^{-l}x - \alpha^j) = \alpha^{-lt}\prod_{j=1}^{t}(x - \alpha^{l+j})$, and $\alpha^{-it} \neq 0$, as we are working in a field. □

A very useful application of the BCH bound is to construct a set of linearly independent polynomials from one whose roots are consecutive powers of α.

Lemma 2.4. *Let $p(x)$ be the annihilator of $\{\alpha^{b+1}, \dots, \alpha^{b+t}\}$ and construct the set of polynomials $\mathcal{P} = \{p(\alpha^j x) : j \in \mathcal{J}\}$, where all the elements of $\mathcal{J}$ are distinct, and $r := |\mathcal{J}| \leq t+1$. Then, the polynomials in $\mathcal{P}$ are linearly independent over $\mathbb{F}_q$.*

Proof. Write $p(x) = \sum_{l=0}^{t} p_l x^l$ and $p(\alpha^j x) = \sum_{l=0}^{t} p_l \alpha^{jl} x^l$ and consider the matrix $\mathbf{P}$ formed by the coefficients of the $p(\alpha^j x)$'s,

$$\mathbf{P} = \begin{bmatrix} p_0 & p_1\alpha^{j_1} & \dots & p_t\alpha^{j_1 t} \\ p_0 & p_1\alpha^{j_2} & \dots & p_t\alpha^{j_2 t} \\ \vdots & \vdots & \ddots & \vdots \\ p_0 & p_1\alpha^{j_r} & \dots & p_t\alpha^{j_r t} \end{bmatrix}.$$

This matrix is never tall, by assumption, and so we consider the matrix $\hat{\mathbf{P}}$ formed from the first r columns of $\mathbf{P}$. We have

$$\begin{aligned} \mathsf{det}(\hat{\mathbf{P}}) &= \begin{vmatrix} p_0 & p_1\alpha^{j_1} & \dots & p_{r-1}\alpha^{j_1(r-1)} \\ p_0 & p_1\alpha^{j_2} & \dots & p_{r-1}\alpha^{j_2(r-1)} \\ \vdots & \vdots & \ddots & \vdots \\ p_0 & p_1\alpha^{j_r} & \dots & p_{r-1}\alpha^{j_r(r-1)} \end{vmatrix} \\ &= \begin{vmatrix} 1 & \alpha^{j_1} & \dots & \alpha^{j_1(r-1)} \\ 1 & \alpha^{j_2} & \dots & \alpha^{j_2(r-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{j_r} & \dots & \alpha^{j_r(r-1)} \end{vmatrix} \prod_{i=0}^{r-1} p_i. \end{aligned}$$

Using the BCH bound from Fact 2.4, we establish that all p_i's are nonzero. Therefore $\mathsf{det}(\hat{\mathbf{P}})$ is equal to the determinant of the Vandermonde matrix with defining set $\{\alpha^{j_1}, \dots, \alpha^{j_r}\}$, multiplied by a non-zero scalar. The elements $\{\alpha^{j_1}, \dots, \alpha^{j_r}\}$ are all distinct in $\mathbb{F}_q$, by assumption, which implies that the matrix $\hat{\mathbf{P}}$ is full rank which, in turn, proves that the polynomials $\mathcal{P}$ are linearly independent. □

We now show the existence of distributed Reed–Solomon codes for SMANs with three sources under the assumption $r_i > n_i$ for all i.

Proposition 2.6. *Let $G = (\mathcal{S}, \mathcal{V}, \mathcal{E})$ be an SMAN with three sources, in which z relay links are erroneous. Let $\mathcal{V}_i \subseteq \mathcal{V}$ be the set of relay nodes connected to S_i only. Assume that $r_i > |\mathcal{V}_i|$ for $i = 1, 2, 3$. Then, a distributed Reed–Solomon code exists capable of correcting z errors.*

Proof. As was done in the proof of Proposition 2.5, we place a diagonal matrix $\mathbf{D}_i$ of size n_i in the submatrix that corresponds to the columns represented by $\mathcal{V}_i$ and the block $\mathbf{G}_i$, and the remaining $\rho_i := r_i - n_i$ rows have all zeros in those columns. As an example, the matrix $\mathbf{G}_1$ is given by

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{G}_{1,a} \\ \mathbf{G}_{1,b} \end{bmatrix} = \left[\begin{array}{cccc|c|c|c|c|c|c} \star & 0 & \cdots & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ 0 & \star & \cdots & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \star & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \hline 0 & 0 & \cdots & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ 0 & 0 & \cdots & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \end{array}\right]. \tag{2.17}$$

Here $\mathbf{G}_{i,a} \in \mathbb{F}_q^{n_i \times n}$ and $\mathbf{G}_{i,b} \in \mathbb{F}_q^{\rho_i \times n}$. The polynomials $\mathcal{T}_{i,a}$ corresponding to $\mathbf{G}_{i,a}$ are chosen according to (2.15), where now r_i is replaced with n_i. The remainder of the proof is to show how to select the polynomials $\mathcal{T}_{i,b}$ corresponding to $\mathbf{G}_{i,b}$.

First, we permute the rows of of $\mathbf{G}$ so that it is the form given below.

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_a \\ \mathbf{G}_b \end{bmatrix} = \left[\begin{array}{ccc|ccc|c} \mathbf{D}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \mathbf{D}_2 & \mathbf{0} & \times & \mathbf{0} & \times & \times \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_3 & \times & \times & \mathbf{0} & \times \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \mathbf{0} & \times & \times \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \times & \times & \mathbf{0} & \times \end{array}\right]. \tag{2.18}$$

The blocks of columns of $\mathbf{G}$ (in the given order) have sizes $n_1, n_2, n_3, n_{2,3}, n_{1,3}, n_{1,2}, n_{1,2,3}$, while the blocks of rows have sizes $n_1, n_2, n_3, \rho_1, \rho_2, \rho_3$. The matrix $\mathbf{G}_a$ is taken care of as just described, so we focus on $\mathbf{G}_b$. We need to pin down some relations between the various parameters in order to customize the construction appropriately.

First, it is useful to note that the condition $k < |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$ is equivalent to $n_{1,2,3} - 2z < 0$. This is readily seen by noting that $n_{1,2,3} = n - |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$. Before proceeding, let us consider the inequalities

$$r_1 \leq n_1 + n_{1,3}, \tag{2.19}$$

$$r_2 \leq n_2 + n_{1,2}, \tag{2.20}$$

$$r_3 \leq n_3 + n_{2,3}, \tag{2.21}$$

and suppose that at least two of them hold. In particular, and without loss of generality[2], we assume that

$$r_1 \leq n_1 + n_{1,3}, \tag{2.22}$$

$$r_2 \leq n_2 + n_{1,2}. \tag{2.23}$$

We will first present a construction under those assumptions and then for the complementary case. As usual, let the j^{th} column of $\mathbf{G}$ be associated with α^j. Furthermore, define $\bar{n} = n_1 + n_2 + n_3$ and let $c(x) = \prod_{i=1}^{\bar{n}}(x - \alpha^i)$. This polynomial will produce the all-zero columns of $\mathbf{G}_a$, namely those corresponding to $\mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$. Now define $p(x)$ that vanishes on a set of powers of α^j with cardinality $t := k - \bar{n} - 1$, i.e.,

$$p(x) = \prod_{j=1}^{t}(x - \alpha^j). \tag{2.24}$$

This polynomial will be used to appropriately place zeros in the columns corresponding to $\mathcal{V}_{2,3} \cup \mathcal{V}_{1,3} \cup \mathcal{V}_{1,2}$. Lemma 2.3 enables us to construct the polynomials for $\mathbf{G}_b$ from shifts of $p(x)$. To do so, let us define the following sets,

$$\mathcal{J}_1 = \left\{\bar{n} + n_{2,3} - t, \ldots, \bar{n} + n_{2,3} - t + \rho_1 - 1\right\}, \tag{2.25}$$

$$\mathcal{J}_2 = \left\{\bar{n} + n_{2,3} + n_{1,3} - t, \ldots, \bar{n} + n_{2,3} + n_{1,3} + \rho_2 - 1\right\}, \tag{2.26}$$

$$\mathcal{J}_3 = \left\{\bar{n} + n_{2,3} + n_{1,3} + n_{1,2} - t, \ldots, \bar{n} + n_{2,3} + n_{1,3} + n_{1,2} - t + \rho_3 - 1\right\}. \tag{2.27}$$

The polynomials corresponding to $\mathbf{G}_{i,2}$ are now given by

$$\mathcal{T}_{i,b} = \left\{c(x)p(\alpha^{-j}x) : j \in \mathcal{J}_i\right\}. \tag{2.28}$$

Let us see why this assignment is valid. First, the polynomial $c(x)$ produces the zeros required in the columns of $\mathbf{G}_2$ corresponding to $\mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$. Now, consider the polynomials in $\mathcal{T}_{1,b}$. By Lemma 2.3, the polynomial $p(\alpha^{-(\bar{n}+n_{2,3}-t)}x)$ vanishes on $\{\alpha^j : j = \bar{n} + n_{2,3} - t + 1, \ldots, \bar{n} + n_{2,3}\}$, which clearly contains $\{\alpha^j : j = \bar{n} + 1, \ldots, \bar{n} + n_{2,3}\}$. As a result, this codeword resulting from evaluating $c(x)p(\alpha^{-(\bar{n}+n_{2,3}-t)}x)$ on $\mathcal{A}$ will have zero coordinates; amongst others, those corresponding to $\mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3 \cup \mathcal{V}_{2,3}$. Now we have to show that this is true for any polynomial in $\mathcal{T}_{1,b}$. In particular, we want to show that for any $0 \leq j \leq \rho_1 - 1$, one has

$$\{\bar{n} + 1, \ldots, \bar{n} + n_{2,3}\} \subseteq \{\bar{n} + n_{2,3} - t + 1 + j, \ldots, \bar{n} + n_{2,3} + j\}.$$

[2] No loss of generality is incurred when one interprets the labels 1, 2 and 3 as *some* labeling of the three sources S_1, S_2 and S_3.

Assuming the contrary means that there is a smallest j in the prescribed range for which $\bar{n} + n_{2,3} - t + 1 + j > \bar{n} + 1$. Using the cut-set bound $r_1 \leq C_1 - 2z$, one can deduce

$$\begin{aligned} \rho_1 - 1 &\geq k - \bar{n} - n_{2,3} - 1 \\ &= n - 2z - \bar{n} - n_{2,3} - 1 \\ &= n_{1,2} + n_{2,3} + n_{1,2,3} - 2z - 1. \end{aligned}$$

Furthermore, we note that $\rho = r_1 - n_1$ yields

$$r_1 > n_1 + n_{1,2} + n_{2,3} + n_{1,2,3} - 2z = C_1 - 2z,$$

a contradiction. As a result, we establish every polynomial in $\mathcal{T}_{1,b}$ evaluates to a valid codeword for $\mathbf{G}_1$. The same claim holds for both $\mathcal{T}_{2,b}$ and $\mathcal{T}_{3,b}$ and the proof is identical.

What remains now is to prove that $\mathcal{T}_{1,b} \cup \mathcal{T}_{2,b} \cup \mathcal{T}_{3,b}$ is a linearly independent set of polynomials over $\mathbb{F}_q$. To this end, we start by showing that the sets $\mathcal{J}_1, \mathcal{J}_2$ and $\mathcal{J}_3$ are pairwise disjoint, to rule out the possibility that a polynomial amongst the $\mathcal{T}_{i,b}$'s is repeated. Working toward a contradiction, suppose $\mathcal{J}_1 \cap \mathcal{J}_2 \neq \emptyset$. Given that the elements in the $\mathcal{J}_i$'s are consecutive, there is a smallest j in the range $0 \leq j \leq \rho_1 - 1$ such that $\bar{n} + n_{2,3} - t + j = \bar{n} + n_{2,3} + n_{1,3} - t$. This implies that $n_{1,3} = j \leq \rho_1 - 1$. Similarly, if $\mathcal{J}_2 \cap \mathcal{J}_3 \neq \emptyset$, then $\rho_2 - 1 \geq n_{1,2}$. Both these cases contradict the assumptions (2.22), (2.23). By the ordering of the elements in the sets, it follows now that $\mathcal{J}_1 \cap \mathcal{J}_3 = \emptyset$. As a result, the elements in $\mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3$ are all distinct.

This distinctness will help us show that the polynomials in $\mathcal{T}_{1,b} \cup \mathcal{T}_{2,b} \cup \mathcal{T}_{3,b}$ are linearly independent. In view of (2.28), this is true if and only if the polynomials $p(\alpha^j x)$ are linearly independent, for $j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3$. Therefore, we focus on the latter. Note that the total number of polynomials under consideration is $\rho_1 + \rho_2 + \rho_3$, which is less that $t := k - \bar{n} - 1$, by the cut-set bound on $r_1 + r_2 + r_3$. Applying the BCH bound to the coefficients of $p(x)$, followed by invoking Lemma 2.4, implies that $\mathcal{T}_{1,b} \cup \mathcal{T}_{2,b} \cup \mathcal{T}_{3,b}$ are linearly independent, and so $\mathsf{rank}\,(\mathbf{G}_b) = \rho_1 + \rho_2 + \rho_3$ and as a result the matrix $\mathbf{G}$ is full rank.

Let us revisit the inequalities (2.19), (2.20) and (2.21). Now suppose that exactly one holds[3]. Without loss of generality, suppose that the first two are false, i.e.

$$r_1 > n_1 + n_{1,3}, \tag{2.29}$$

$$r_2 > n_2 + n_{1,2}. \tag{2.30}$$

[3] It can be shown that at least one of the inequalities has to hold.

In this case, we mimic the previous construction but choose the sets $\mathcal{J}_1, \mathcal{J}_2$ and $\mathcal{J}_3$ in a different way. The reason is that (2.22) and (2.23) are necessary for the sets $\mathcal{J}_1, \mathcal{J}_2$ and $\mathcal{J}_3$ defined earlier to be pairwise distinct. Nonetheless, we can mend the construction by choosing the sets $\mathcal{J}_1, \mathcal{J}_2$ and $\mathcal{J}_3$ as

$$\mathcal{J}_1 = \{\bar{n} + n_{23} - t, \ldots, \bar{n} + n_{23} - t + \rho_1 - 1\}, \tag{2.31}$$

$$\mathcal{J}_2 = \{\bar{n} + n_{23} - t + \rho_1, \ldots, \bar{n} + n_{23} - t + \rho_1 + \rho_2 - 1\}, \tag{2.32}$$

$$\mathcal{J}_3 = \{\bar{n} + n_{23} - t + \rho_1 + \rho_2, \ldots, \bar{n} + n_{23} - t + \rho_1 + \rho_2 + \rho_3 - 1\}. \tag{2.33}$$

Note that these three sets form the integer interval $\{\bar{n} + n_{23} - t, \ldots, \bar{n} + n_{23} - t + \rho_1 + \rho_2 + \rho_3 - 1\}$. As a result, pairwise distinctness is immediate. It remains to show that $\mathcal{T}_{i,b}$ in (2.28), defined using the new $\mathcal{J}_i$, vanishes on the powers of α corresponding to $\mathcal{V}_{\{1,2,3\}\setminus i}$. Note that the set $\mathcal{J}_1$ is unchanged from the previous case so its evaluations are valid for $\mathbf{G}_1$. Now consider $p(\alpha^{-j}x)$ for an arbitrary $j \in \mathcal{J}_2$. We want this polynomial to vanish on $\{\alpha^l : l \in \mathcal{V}_{1,3}\}$, or equivalently, on the powers of α indexed by

$$\{\bar{n} + n_{2,3} + 1, \ldots, \bar{n} + n_{2,3} + n_{1,3}\}. \tag{2.34}$$

Consider $p(\alpha^{-(\bar{n}+n_{23}-t+\rho_1+j)}x)$, where $0 \le j \le \rho_2 - 1$. This polynomial vanishes on the powers of α indexed by

$$\{\bar{n} + n_{2,3} - t + 1 + \rho_1 + j, \ldots, \bar{n} + n_{2,3} + \rho_1 + j\}. \tag{2.35}$$

Suppose that the set (2.34) is not contained in the set (2.35). First, let us assume that $\bar{n} + n_{2,3} - t + 1 + \rho_1 + j > \bar{n} + n_{2,3} + 1$. This is equivalent to $\rho_1 + j > t$. By definition, we have

$$r_1 - n_1 + r_2 - n_2 - 1 \ge r_1 - n_1 + j > k - \bar{n} - 1.$$

This implies that $r_1 + r_2 > k - n_3 = C_{1,2} - 2z$, which a contradiction to the cut-set bound on $r_1 + r_2$. Now suppose that $\bar{n} + n_{2,3} + n_{1,3} > \bar{n} + n_{2,3} + \rho_1 + j$, which is equivalent to

$$r_1 < n_1 + n_{1,3}. \tag{2.36}$$

This is in contradiction with (2.29). Thus every polynomial in $\mathcal{T}_{2,b}$ vanishes on $\mathcal{V}_{1,3}$ and so it is valid for $\mathbf{G}_2$.

What is left is to prove a similar claim for $\mathcal{T}_{3,b}$. Any polynomial in this set must vanish on the powers of α indexed by

$$\{\bar{n} + n_{2,3} + n_{1,3} + 1, \ldots, \bar{n} + n_{2,3} + n_{1,3} + n_{1,2}\}, \tag{2.37}$$

whereas an arbitrary polynomial in $\mathcal{T}_{3,b}$ vanishes on the powers of α indexed by

$$\{\bar{n} + n_{2,3} - t + 1 + \rho_1 + \rho_2 + j, \ldots, \bar{n} + n_{2,3} + \rho_1 + \rho_2 + j\}. \qquad (2.38)$$

Similar to the earlier case, assuming that $\bar{n} + n_{2,3} + n_{1,3} - t + 1 + \rho_1 + \rho_2 + j < \bar{n} + n_{2,3} + n_{1,3} + 1$ yields a contradiction to the cut-set bound on $r_1 + r_2 + r_3$. On the other hand, suppose that

$$\bar{n} + n_{2,3} + \rho_1 + \rho_2 + j < \bar{n} + n_{2,3} + n_{1,3} + n_{1,2}.$$

This supposition along with $j \geq 0$ implies that

$$r_1 + r_2 < n_1 + n_{1,3} + n_2 + n_{1,2}.$$

The sum of the inequalities (2.29) and (2.23) contradicts this statement. Hence, the polynomials $\mathcal{T}_{i,b}$ are valid for the respective $\mathbf{G}_i$, and by construction, are all distinct. An application of the BCH bound from Fact 2.4 proves that the polynomials in $\mathcal{T}_{1,b} \cup \mathcal{T}_{2,b} \cup \mathcal{T}_{3,b}$ are linearly independent.

Having handled the two separate cases, one in which at least two of the three inequalities in (2.19), (2.20) and (2.21) are true, and the other in which exactly one is true, and having also ruled out the possibility of all three being false, as a result, the proof of this proposition is complete. □

Propositions 2.4, 2.5 and 2.6 together form the fabric of the main result. Indeed, any SMAN falls under the assumptions of one of the three propositions for which a corresponding distributed Reed–Solomon code can be constructed. The main result of this chapter, Theorem 2.2 is proved.

Alternative Constructive for Two-Source SMANs

In this section, we present an alternate construction to the one given in Proposition 2.2.

Proposition 2.7. *Let $G(\mathcal{S}, \mathcal{V}, \mathcal{E})$ represent an SMAN where $|\mathcal{S}| = 2$. Let $\mathcal{R}_i = \{j : (i,j) \notin \mathcal{E}\}$ (both not empty), i.e. the set of relay nodes not connected to S_i. Suppose, without loss of generality, that $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$. Now define $z_i(x) = \prod_{j \in \mathcal{R}_i}(x - \alpha^j)$ and let $s_1(x), s_2(x)$ be polynomials such that either $\mathsf{deg}\,(s_1(x)) = C_1 - 2z - r_1$ or $\mathsf{deg}\,(s_2(x)) = C_2 - 2z - r_2$ and $\mathsf{gcd}\,(s_1(x), z_2(x)) = \mathsf{gcd}\,(s_2(x), z_1(x)) = 1$. Finally, for $i = 1, 2$, define the transformation polynomials*

$$\mathcal{T}_i = \{x^j s_i(x) z_i(x), j = 0, \ldots, r_i - 1\}.$$

We have that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a linearly independent set of polynomials over $\mathbb{F}_q$.

Proof. Working towards a contradiction, suppose that for a particular choice of $\lambda_i^{(j)}$, where some are non-zero, it holds that

$$\sum_{j=0}^{r_1-1} \lambda_1^{(j)} x^j s_1(x) z_1(x) + \sum_{j=0}^{r_2-1} \lambda_2^{(j)} x^j s_2(x) z_2(x) = 0. \tag{2.39}$$

Let $\sum_{j=0}^{r_1-1} \lambda_i^{(j)} x^j = q_i(x)$, where $\deg{(q)}_i\,(x) \leq r_i - 1$. Then we can express (2.39) as

$$q_1(x) s_1(x) z_1(x) = q_2(x) s_2(x) z_2(x). \tag{2.40}$$

Since $\gcd{(s_i(x), z_l(x))} = 1$, it must be the case that $q_1(x) = s_2(x) z_2(x) c(x)$. By the degree constraint on $q_1(x)$, we have

$$\begin{aligned} \deg{(c(x))} &= \deg{(q_1(x))} - \deg{(s_2(x))} - \deg{(z_2(x))} && (2.41)\\ &\leq r_1 - 1 - (C_2 - 2z) + r_2 - |\mathcal{R}_2| && (2.42)\\ &= r_1 + r_2 - (C_{1,2} - 2z) - 1 && (2.43)\\ &\leq -1 && (2.44) \end{aligned}$$

since $C_2 + |\mathcal{R}_2| = C_{1,2}$ and the last inequality follows from (2.2). Thus, $c(x) = q_1(x) = 0$ and so we arrive at a contradiction. □

The proposition provides an easy way to construct solutions for SMANS with two sources. Indeed, the polynomials $s_1(x)$ and $s_2(x)$ can be constructed easily over $\mathbb{F}_q[x]$ where $q > n$. First, it must be that either $C_1 - 2z - r_1 \leq C_2$ or $C_2 - 2z - r_2 \leq C_1$ hold since otherwise we arrive at $r_1 + r_2 < 0$. Suppose without loss of generality that $C_2 - 2z - r_2 \leq C_1$. Out of the n code coordinates, $z_1(x)$ does not vanish on C_1 of them. Thus, $s_2(x)$ can be chosen such that its $C_2 - 2z - r_2$ roots are contained in these C_1 coordinates. Now set $s_1(x) = 1$. By construction, both $s_1(x)$ and $z_2(x)$ are separable but do not share a common root and so are coprime as required.

2.4 Examples

We offer in this section a series of examples to portray the different constructions presented in this chapter.

Example 2.1. *Consider the SMAN in Figure 2.1, with adjacency matrix*

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Assume $\mathbf{r} = (2, 2, 2)$ *and* $z = 1$. *It follows that the construction from Proposition 2.1 should be used. The constituent code is a* $[10, 8, 3]$ *RS code over* $\mathbb{F}_{11}$ *with primitive element* $\alpha = 2$. *The polynomials and sets of interest, chosen according to* (2.25),(2.26) *and* (2.27), *are tabulated below:*

$$\begin{aligned} c(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^3), \\ p(x) &= (x-\alpha^1)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4), \\ \mathcal{J}_1 &= \{1\}, \\ \mathcal{J}_2 &= \{3\}, \\ \mathcal{J}_3 &= \{5\}. \end{aligned}$$

We have $n_i = \rho_i = 1$ *for all* i, *so the sets* $\mathbf{T}_{1,a}$,$\mathbf{T}_{1,b}$ *and* $\mathbf{T}_{3,a}$ *each consist of a single polynomial.*

$$\begin{aligned} t_{1,a}(x) &= (x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5), \\ t_{2,a}(x) &= (x-\alpha)(x-\alpha^3)(x-\alpha^6)(x-\alpha^7), \\ t_{3,a}(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^8)(x-\alpha^9). \end{aligned}$$

The remaining polynomials are given by

$$\begin{aligned} t_{1,b}(x) &= c(x)p(\alpha^{-1}x) = \alpha^{-4}c(x)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5), \\ t_{2,b}(x) &= c(x)p(\alpha^{-3}x) = \alpha^{-12}c(x)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)(x-\alpha^7), \\ t_{3,b}(x) &= c(x)p(\alpha^{-5}x) = \alpha^{-20}c(x)(x-\alpha^6)(x-\alpha^7)(x-\alpha^8)(x-\alpha^9). \end{aligned}$$

Note how the polynomial $t_{1,b}(x)$ *has repeated roots at* α^2 *and* α^3. *The resulting generator matrix in the form of* (2.18) *is given by*

$$\mathbf{G} = \left[\begin{array}{ccc|ccccccc} 2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 2 & 7 & 4 & 5 & 8 \\ \mathbf{0} & 1 & \mathbf{0} & 5 & 4 & \mathbf{0} & \mathbf{0} & 1 & 9 & 6 \\ \mathbf{0} & \mathbf{0} & 9 & 5 & 2 & 3 & 5 & \mathbf{0} & \mathbf{0} & 8 \\ \hline 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 3 & 1 & 4 & 6 & 6 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & 0 & \mathbf{0} & \mathbf{0} & 2 & 4 & 3 \\ \mathbf{0} & \mathbf{0} & 0 & 1 & 1 & 0 & 0 & \mathbf{0} & \mathbf{0} & 7 \end{array}\right].$$

The required zeros are depicted in boldface.

The second example is pertinent to the second variant that can arise in Proposition 2.6.

Example 2.2. *Consider an SMAN whose adjacency matrix is given by*

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Suppose that $(r_1, r_2, r_3) = (2, 3, 1)$ *and* $z = 1$*. Here, the assumptions* (2.29) *and* (2.30) *hold. As a result, one has to choose the index sets chosen according* (2.31)*,* (2.32) *and* (2.33)*,*

$$\begin{aligned} \mathcal{J}_1 &= \{-1, 0\}, \\ \mathcal{J}_2 &= \{1, 2, 3\}, \\ \mathcal{J}_3 &= \{4\}. \end{aligned}$$

The resulting generator matrix is given by

$$\mathbf{G} = \left[\begin{array}{cccccccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 2 & 6 & 7 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 2 & 6 & 7 \\ \hline 7 & 0 & 0 & 0 & \mathbf{0} & 0 & 2 & 6 \\ 6 & 7 & 0 & 0 & \mathbf{0} & 0 & 0 & 2 \\ 2 & 6 & 7 & 0 & \mathbf{0} & 0 & 0 & 0 \\ \hline 0 & 2 & 6 & 7 & 0 & \mathbf{0} & \mathbf{0} & 0 \end{array}\right].$$

2.5 Discussion

The main contribution of this chapter is a constructive proof of the existence of a distributed Reed–Solomon code for any point in the capacity region (2.2) of a simple multiple access network with three sources. The proof of existence relies on a methodical way of partitioning the problem instance space. In particular, if we let $\mathcal{R}_i$ be the set of relay nodes not connected to S_i, then the main technical condition ($k \lesseqgtr |\cup_{i=1}^{m} \mathcal{R}_i|$) has an operational meaning. If $k \geq |\cup_{i=1}^{m} \mathcal{R}_i|$ holds, then the generator matrix of the desired distributed Reed–Solomon code can be constructed as a systematic one. Indeed, the techniques presented in Section 2.3 can be generalized to handle any number of sources m. Nonetheless, we present a cleaner approach in Chapter 4 to handle this scenario that utilizes results from graph theory.

On the other hand, if $k < |\cup_{i=1}^{m} \mathcal{R}_i|$ then one has to rely on the construction from Section 2.3. Whilst the approach presented here might seem complex, the basic idea is quite intuitive. First, we ensure than any relay nodes connected to a single source node S_i are fully utilized and, together, they carry linearly independent symbols. This effectively decouples those nodes from the rest of the network. Once that is done, the special case of networks with three sources allow for utilizing techniques pertinent to cyclic codes. Indeed, this assumption allowed us to construct the codewords from a single polynomial. At this point, this technique does not appear to be generalizable to SMANs with more than three source nodes. However, let us emphasize that meaningful networks with $k < |\cup_{i=1}^{m} \mathcal{R}_i|$ can be constructed. The most straightforward example is one in which a cyclic structure is enforced, i.e.

$$\mathbf{G} = \begin{bmatrix} \times & \times & \cdots & \cdots & \times & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \cdots & \times & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \times & \times & \cdots & \cdots & \times & 0 \end{bmatrix}.$$

One can verify that the network corresponding to such matrix falls under this case. The fact that the zeros are consecutive to one another allows us to use the technique based on Lemma 2.4 to pepper this matrix in a way that renders is full rank. Nonetheless, it is when more complex structures are considered that this technique fails. The BCH bound was critical in proving that the polynomials chosen using this approach were linearly independent and so more complicated scenarios might require more sophisticated bounds such as the ones due to Roos [Roo82; Roo83] and Hartmann & Tzeng [HT72].

It is also worth mentioning the special structure of the coordinates of the underlying Reed–Solomon code was heavily used in Proposition 2.6. The following example shows why the choice cannot be arbitrary.

Example 2.3. *Consider a SMAN with $z = 1$ and adjacency matrix given by*

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}. \tag{2.45}$$

Fix the underlying Reed–Solomon to be length 6 and dimension 3 over $\mathbb{F}_7$. Suppose we associate the columns of the required generator matrix with $\mathcal{A} = \{1, 6, 2, 5, 3, 4\}$. The polynomials corresponding to the rows of the generator matrix are

$$t_1(x) = (x-1)(x-6) = x^2 + 6,$$
$$t_2(x) = (x-2)(x-5) = x^2 + 3,$$
$$t_3(x) = (x-3)(x-4) = x^2 + 5.$$

Given that the underlying code is of dimension 3*, any other set of polynomials valid for* $\mathbf{A}$ *are scalar multiples of the ones provided. Note that the particular ordering of* $\mathcal{A}$ *results in a set of polynomials that spans a space of dimension 2.*

Chapter 3

DISTRIBUTED GABIDULIN CODES

3.1 Introduction

In this chapter, we develop the results presented in Chapter 2 to codes designed for multi-source multicast networks employing linear network coding. The pioneering work of Kötter and Kschischang in [KK08] introduced subspace coding for error control in single-source multicast networks employing random linear network coding. The source node encodes its information in a choice of a subspace and then transmits packets corresponding to basis vectors of that subspace. The packets are transmitted over the network, using random linear network coding, and a number of packets being transmitted in the network may be arbitrarily corrupted. If the received packets span a subspace that intersects the transmitted subspace sufficiently, the destination node can decode correctly. Silva et al. [SKK08] further showed that a constant dimension subspace code can be built by *lifting* an appropriate rank-metric code such as a Gabidulin code [Gab85], where lifting preserves the code distance properties under the rank metric.

In the multi-source multicast scenario, two or more source nodes seek to transmit a set of messages to one or more destinations, over a network of arbitrary topology. Each destination is interested in receiving all the messages, and an omniscient adversary can arbitrarily corrupt a bounded number of packets/links in the network. The capacity region for the case where the source nodes have independent messages was established in [Dik+10], and is given by the cut-set bounds for each subset of messages. In this chapter, we consider the scenario in which each source node has access to a subset of messages. The capacity region follows from [Dik+10] by considering the cut-set bounds from each subset of messages to the destination nodes. The results of this chapter show that the actual source nodes can encode independently in a distributed manner such that the overall subspace codewords form a subcode of a single-source lifted Gabidulin code, with no reduction in the capacity region for up to three messages. The special case in which the destination is directly connected to each source node was considered in the previous chapter; where it suffices to employ Reed–Solomon codes. In this chapter, we consider the more general setting and construct distributed Gabidulin codes that can be employed in an arbitrary topology. It is worth mentioning that a distributed Gabidulin code can be designed for those networks whose structure doesn't introduce deletions, i.e. the total number of degrees of freedom injected into the network is preserved at the destination node. More details on this scenario will be given in the discussion.

Related Work

Network error-correcting codes for single-source multicast networks were presented in [Jag+07], where a rate-optimal (given an adversary's limits) polynomial time construction was presented. The paper [KK08] introduced *subspace coding* as a paradigm that suffices to achieve the capacity of a network employing non-coherent network coding.

The multisource multicast problem was first fully studied by Dikaliotis et al. in [Dik+10]. Under the assumption of a bounded number of adverserial errors, the capacity region for such networks was established and a capacity-achieving error-correction scheme based on Gabidulin codes [Gab85] was constructed. The main drawback of this construction is the reliance on a technique that requires a series of nested finite fields, which resulted in a coding complexity that is exponential in the number of sources.

The papers [JFD09; SFD08; Moh+] consider a similar case albeit in the absence of adversarial errors. The network is modeled as a time-varying channel problem and its capacity is derived. Several code constructions based on subspace coding are presented. In particular, the one given in [JFD09] extends the result of [KK08] to the multisource multicast setting.

3.2 Preliminaries

We begin by giving a quick overview of error correction in network coded communication networks. We review the basics of rank-metric codes and focus on Gabidulin codes. We then described the ring of linearized polynomials, which is crucial to the code construction presented in this chapter. Lastly, we close this section with some useful facts and terminology that will be used in the subsequent sections.

Network model

We consider a general network with n source nodes $\mathcal{S} = \{S_1, \dots, S_n\}$ each of which has access to a subset of a set of messages $\mathcal{M} = \{M_1, M_2, \dots, M_L\}$. We consider a multicast scenario in which a destination node is interested in retrieving all available messages. Figure 3.1 provides a pictorial representation of this setup. Message M_i

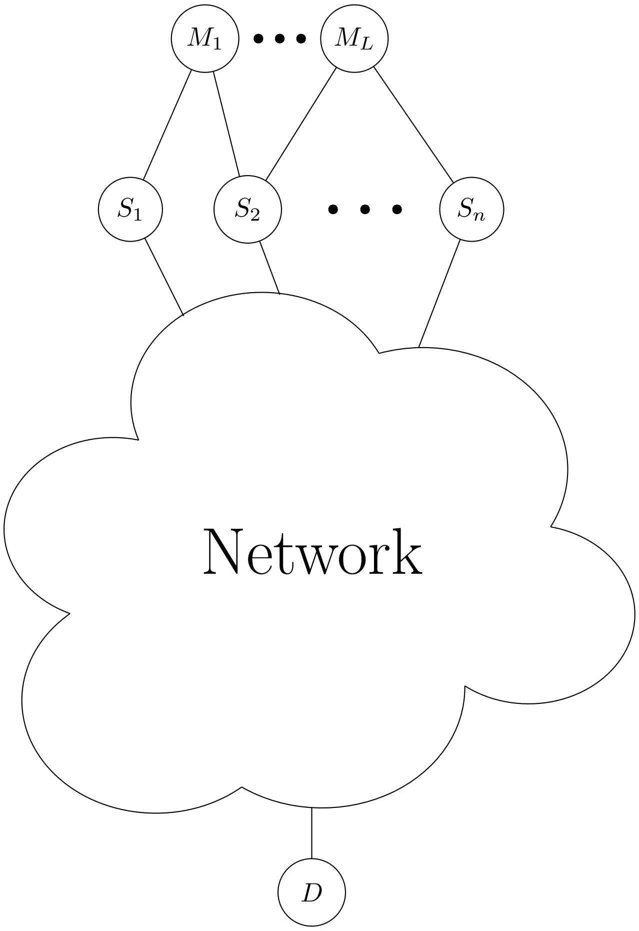


Figure 3.1: A multisource multicast network consists of a set of L messages jointly held by a set of sources. Each vertex $S_{\mathcal{J}}$ represents the set of sources nodes that can access messages $\{M_j : j \in \mathcal{J}\}$.

is of rate r_i symbols of $\mathbb{F}_{q^m}$. An omniscient adversary injects erroneous packets on up to z links in the network.

Without loss of generality, we assume that each source node has one outgoing edge. Let $\mathcal{S}_{\mathcal{J}}$ be the set of source nodes with access to messages $\{M_j : j \in \mathcal{J}\}$, and define $n_{\mathcal{J}} := |\mathcal{S}_{\mathcal{J}}|$. Each source node in $\mathcal{S}_{\mathcal{J}}$ injects a packet into the network on its outgoing edge, where each packet is a linear combination of the messages indexed by $\mathcal{J} \subseteq \{1, \ldots, L\}$, i.e. the messages it has access to. Throughout this chapter, we will consider the case when $L = 3$.

Let $\mathcal{I}(\mathcal{M}')$ denote the index set of elements in $\mathcal{M}'$, i.e. $\mathcal{I}(\mathcal{M}') = \{i : M_i \in \mathcal{M}'\}$. Also define $\mathcal{I} := \mathcal{I}(\mathcal{M})$ and $r_{\mathcal{I}(\mathcal{M}')} := \sum_{i \in \mathcal{I}(\mathcal{M}')} r_i$. In particular, we define $R := r_{\mathcal{I}}$. The minimum cut capacity (min-cut) from $\mathcal{M}'$ to destination D is denoted by $m_{\mathcal{I}(\mathcal{M}')}$, $\forall \mathcal{M}' \subseteq \mathcal{M}$. From [Dik+10], the capacity region $\mathcal{R}$ is given by cut set bounds for each subset of messages, i.e. the capacity region is the set of all vectors $\mathbf{r} = (r_1, r_2, \ldots, r_L)$ such that

$$r_{\mathcal{I}(\mathcal{M}')} \leq m_{\mathcal{I}(\mathcal{M}')} - 2z, \forall \mathcal{M}' \subseteq \mathcal{M}. \tag{3.1}$$

We will assume that all min-cuts defining the capacity region of a multiple source multicast network can be assumed to be in the layer between the messages and the source nodes[1]. In particular, we will assume that these quantities can be computed from the bipartite graph describing the relationship between the messages and source nodes. Indeed, the capacity region can now be expressed using the various quantities $n_{\mathcal{J}}$ similar to what was done in Chapter 2.

Single-source subspace codes

Let $\mathbb{F}_q$ be the finite field with q elements, where q is a power of a prime. In single-source subspace coding, the source node generates a batch of n packets, each of length m, which are treated as vectors over some finite field $\mathbb{F}_q$, and arranged as the rows of a matrix $\mathbf{X} \in \mathbb{F}_q^{n \times m}$. The source node then injects the packets into the network. In the presence of linear network coding [Ho+06], the destination node collects a set of N packets that constitute linear combinations of the rows of $\mathbf{X}$. The overall network transformation from the source node to a destination node is represented by a matrix $\mathbf{H} \in \mathbb{F}_q^{N \times n}$, meaning that the destination node receives the packets corresponding to rows of matrix $\mathbf{Y} = \mathbf{HX} \in \mathbb{F}_q^{N \times m}$. Thus, the network can be thought of as a matrix-valued channel in which the input alphabet is the set of matrices $\mathbb{F}_q^{n \times m}$ and the output alphabet is the set $\mathbb{F}_q^{N \times m}$. To quantify the impact of erroneous packets being injected into the network, a suitable metric has to be introduced. The rank metric is a natural candidate [SKK08; KK08] for such scenarios[2].

The rank distance between two matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ is given by $d_{\mathrm{R}}(\mathbf{X}_1, \mathbf{X}_2) := \mathsf{rank}\,(\mathbf{X}_2 - \mathbf{X}_1)$. If we assume that erroneous packets are injected into up to z links in the network, then the destination node receives

$$\mathbf{Y} = \mathbf{HX} + \mathbf{Z}, \tag{3.2}$$

where $\mathsf{rank}\,(\mathbf{Z}) \leq z$.

We begin by presenting a useful fact that will be heavily relied on.

[1] In particular, we require that the sum of outgoing edges from $\mathcal{S}$ to the network is equal to $m_{\mathcal{I}}$. For more details, please refer to Section 3.6.

[2] Indeed, this metric was considered long before by Delsarte in [Del78]

Fact 3.1. *Let $\mathbb{F}_{q^m}$ be an m^{th} degree extension of $\mathbb{F}_q$ with a fixed basis $\beta_1, \ldots \beta_m$. The field $\mathbb{F}_{q^m}$ is isomorphic to the vector space $\mathbb{F}_q^m$ via the mapping φ, where for a fixed $\gamma \in \mathbb{F}_{q^m}$ given by $\gamma = \sum_{i=1}^m c_i\beta_i$ for c_i's $\in \mathbb{F}_q$, the evaluation is given by*

$$\varphi : \gamma \mapsto (c_1, \ldots, c_m).$$

The two representations will be interchanged frequently, and the one chosen in each instance of appearance will be specified clearly. This notion is useful for considering vector data packets as symbols over a larger finite field, which will be the defining field for the error-correcting code being used.

Gabidulin Codes

A useful observation is one that allows us to extend the isomorphism from Fact 3.1 to one that handles vectors over $\mathbb{F}_{q^m}$.

Fact 3.2. *Let $\Gamma = (\gamma_1, \ldots, \gamma_n)^{\mathsf{t}} \in \mathbb{F}_{q^m}^n$, where $\gamma_i = \sum_{j=1}^m c_{i,j}\beta_j$, for $c_{i,j}$'s $\in \mathbb{F}_q$. The vector spaces $\mathbb{F}_{q^m}^n$ and $\mathbb{F}_q^{n\times m}$ are isomorphic by the mapping Φ;*

$$\Phi : \Gamma \mapsto \begin{bmatrix} c_{1,1} & \cdots & c_{1,m} \\ \vdots & \ddots & \vdots \\ c_{n,1} & \cdots & c_{n,m} \end{bmatrix} = \begin{bmatrix} \varphi(\gamma_1) \\ \vdots \\ \varphi(\gamma_n) \end{bmatrix}.$$

Indeed, the mapping Φ is just the mapping φ applied component-wise.

A Gabidulin code of length n and dimension k over $\mathbb{F}_{q^m}$ is a linear space of column vectors $\mathcal{C}_{\text{GC}} \subseteq \mathbb{F}_{q^m}^n$. The previous fact allows us to regard any codeword $\mathbf{c} \in \mathcal{C}_{\text{GC}}$ as a matrix $\mathbf{C} \in \mathbb{F}_q^{n\times m}$. Unless otherwise stated, a boldface symbol in lowercase will denote a column vector in $\mathbb{F}_{q^m}^n$, while the same boldface symbol in uppercase will denote the same element when represented as a matrix in $\mathbb{F}_q^{n\times m}$. Furthermore, the rank of a codeword $\mathbf{c}$ will be defined as $\mathsf{rank}\,(\mathbf{C})$. Gabidulin codes are maximum rank distance (MRD) codes, i.e. $d_{\text{R}} = n - k + 1$. The generator matrix of a Gabidulin code resembles that of a Reed–Solomon code quite closely. Choose the coordinates of the code $g_1, \ldots, g_n \in \mathbb{F}_{q^m}$ to be linearly independent over $\mathbb{F}_q$, so that $n \leq m$. For ease of notation, let $[i] = q^i$. The generator matrix of a Gabidulin code of length n, dimension k and minimum rank distance $d_{\text{R}} = n - k + 1$ is given by:

$$\mathbf{G}_{\text{GC}} = \begin{bmatrix} g_1^{[0]} & g_1^{[1]} & \cdots & g_1^{[k-1]} \\ g_2^{[0]} & g_2^{[1]} & \cdots & g_2^{[k-1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_n^{[0]} & g_n^{[1]} & \cdots & g_n^{[k-1]} \end{bmatrix}. \tag{3.3}$$

The code $\mathcal{C}_{\text{GC}}$ is given by the right-image of this matrix. In particular, a message $\mathbf{m}$ is encoded as $\mathbf{G}_{\text{GC}}\mathbf{m}$. To use this code in a multicast setting, a source node arranges its information packets in a matrix $\mathbf{M} \in \mathbb{F}_q^{k\times m}$, and then computes $\mathbf{c} = \mathbf{G}_{\text{GC}}\mathbf{m}$, where $\mathbf{m}$ is obtained via the inverse mapping Γ^{-1}. The transmitted (coded) packets are the rows of $\mathbf{C}$, obtained by applying Γ to $\mathbf{c}$.

Linearized Polynomials

A set of polynomials intimately related to Gabidulin codes is the set of linearized polynomials.

Definition 3.1. *A linearized polynomial $P(x)$ over $\mathbb{F}_{q^m}$ with q-degree d is one that can be expressed as $P(x) = \sum_{i=0}^{d} p_i x^{q^i}$.*

Analogous to Reed–Solomon codes, Gabidulin codes can be viewed as the image of a special set of polynomials when evaluated at linearly independent elements of a field $\mathbb{F}_{q^m}$.

Definition 3.2. *Let $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$ be a set of elements in $\mathbb{F}_{q^m}$ that are linearly independent over $\mathbb{F}_q$. A Gabidulin code of length n and dimension k is the set of linearized polynomials with q-degree less than k evaluated at $\mathcal{A}$.*

$$\mathcal{C} = \left\{ (m(\alpha_1), \dots, m(\alpha_n)) : m(x) = \sum_{i=0}^{d} m_i x^{q^i}, d < k \right\}. \tag{3.4}$$

Furthermore, the set of linearized polynomials equipped with conventional polynomial addition along with the composition operation $C(x) = A(x) \otimes B(x) := A(B(x))$ form a non-commutative ring, with no zero-divisors. It can be shown that the roots of a linearized polynomial $P(x)$ form a vector space over $\mathbb{F}_q$.

Fact 3.3. *Let $P(x)$ be a linearized polynomial over $\mathbb{F}_{q^m}$ and suppose α, β are two roots of $P(x)$, then for any $\gamma \in \mathbb{F}_q$, one has $P(\gamma\alpha + \beta) = 0$.*

Using this fact, one can characterize the minimal linearized polynomial with a prescribed root space.

Fact 3.4. *Let $\langle \mathcal{T} \rangle \subseteq \mathbb{F}_{q^m}$ be spanned by linearly independent $\mathcal{T} = \{\alpha_1, \dots, \alpha_t\}$. The minimal polynomial of $\langle \mathcal{T} \rangle$, given by $M_{\mathcal{T}}(x) = \prod_{\beta \in \langle \mathcal{T} \rangle}(x - \beta)$, is a linearized polynomial with q-degree $\mathsf{deg}_q\, M_{\mathcal{T}}(x) = t$.*

We heavily rely on this characterization. Indeed, we will design the target generator matrix by constructing linearized polynomials that vanish on particular subsets of the code's coordinates. Another useful result is one that deals with factoring linearized polynomials. Note that, however, the non-commutative nature of the composition operation allows us to make a one-sided claim.

Fact 3.5. *Any linearized polynomial $P(x)$ whose root space contains $\langle \mathcal{T} \rangle$ can be written as $P(x) = Q(x) \otimes M_{\mathcal{T}}(x)$, for some linearized polynomial $Q(x)$.*

Interestingly, one can show that the reverse factorization $P(x) = M_{\mathcal{T}}(x) \otimes Q(x)$ holds when the coefficients of $P(x)$ lie in $\mathbb{F}_q$. Nonetheless, the standard factorization over the ring $\mathbb{F}_{q^m}[x]$ clearly holds,

Fact 3.6. *Any linearized polynomial $P(x)$ whose root space contains $\langle \mathcal{T} \rangle$ can be written as $P(x) = V(x) M_{\mathcal{T}}(x)$, for some polynomial $V(x)$.*

An expected consequence of the composition of two linearized polynomials is given below.

Fact 3.7. *Let $\mathsf{deg}_q\, A(x) = a$, $\mathsf{deg}_q\, B(x) = b$ and $C(x) = A(x) \otimes B(x)$. Then $\mathsf{deg}_q\, C(x) = a + b$.*

A standard reference on linearized polynomials is [LN97], where proofs for all facts presented in this subsection are given.

Distributed Gabidulin Codes

We have restricted ourselves to the networks with three messages and assumed that the set of source nodes is given by

$$\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_{2,3}, \mathcal{S}_{1,3}, \mathcal{S}_{1,2}, \mathcal{S}_{1,2,3}\},$$

where each of the source nodes in $\mathcal{S}_{\mathcal{J}}$ can inject a total of $n_{\mathcal{J}}$ packets into the network. Since the source nodes in $\mathcal{S}_{\mathcal{J}}$ can code across the same set of messages $\{M_j : j \in \mathcal{J}\}$, these coded symbols can be organized into a length $n_{\mathcal{J}}$ column vector given by

$$\mathbf{c}_{\mathcal{J}} = \sum_{j \in \mathcal{J}} \mathbf{G}^j_{\mathcal{J}} \mathbf{m}_j.$$

Here, the vector $\mathbf{m}_j \in \mathbb{F}_{q^m}^{r_j \times 1}$ is the vector representation of the message M_j, which has rate r_j. Furthermore, the matrix $\mathbf{G}^j_{\mathcal{J}}$ is the coding matrix that $\mathcal{S}_{\mathcal{J}}$ employs to encode $\mathbf{m}_j$. As a result, the overall linear transformation is represented by,

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 & \mathbf{G}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{G}_1^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_2^{(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_3^{(3)} \\ \mathbf{0} & \mathbf{G}_{2,3}^{(2)} & \mathbf{G}_{2,3}^{(3)} \\ \mathbf{G}_{1,3}^{(1)} & \mathbf{0} & \mathbf{G}_{1,3}^{(3)} \\ \mathbf{G}_{1,2}^{(1)} & \mathbf{G}_{1,2}^{(2)} & \mathbf{0} \\ \mathbf{G}_{1,2,3}^{(1)} & \mathbf{G}_{1,2,3}^{(2)} & \mathbf{G}_{1,2,3}^{(3)} \end{bmatrix}. \tag{3.5}$$

Each message $\mathbf{m}_i$, $i = 1, 2, 3$ is a length r_i column vector over $\mathbb{F}_{q^m}$, i.e., the overall codeword is computed by the source nodes in a distributed fashion as $\mathbf{c} = \mathbf{G}\mathbf{m}$, where $\mathbf{m} = (\mathbf{m}_1^{\mathsf{t}}, \mathbf{m}_2^{\mathsf{t}}, \mathbf{m}_3^{\mathsf{t}})^{\mathsf{t}}$. The transmitted packets are the rows of $\mathbf{C}$, as obtained by applying Γ from Fact 3.2 to $\mathbf{c}$.

Following [SKK08], we *lift* the overall codeword, which preserves the distance of the underlying code and provides side information to the decoder at the destination.

Definition 3.3. *A codeword* $\mathbf{C} \in \mathbb{F}_q^{n \times m}$ *is lifted to* $\bar{\mathbf{C}}$ *by appending to its left an identity matrix of size* n*, i.e.*

$$\bar{\mathbf{C}} = [\mathbf{I} \quad \mathbf{C}].$$

To emulate this operation at the global level of the network, the source nodes in $\mathcal{S}_{\mathcal{J}}$ will lift its portion of the overall codeword by appending its codeword with an identity matrix along with additional zeros as necessary. To facilitate this process, let us fix an ordering of the the power set of $\{1, 2, 3\}$ (excluding the empty set) as $\mathcal{A} = \{\{1\}, \{2\}, \{3\}, \{2, 3\}, \{1, 3\}, \{1, 2\}, \{1, 2, 3\}\}$. For $\mathcal{J} \in \mathcal{A}$, $\mathcal{J} - 1$ denotes the element less than $\mathcal{J}$ while $\mathcal{J} + 1$ denotes the element greater than $\mathcal{J}$, with respect to the ordering on $\mathcal{A}$. A source $\mathcal{S}_{\mathcal{J}}$ will lift its codeword according to the following mapping:

$$\mathbf{C}_{\mathcal{J}} \mapsto \left[\begin{array}{c|c} \mathbf{0}_{n_1} \cdots \mathbf{0}_{n_{\mathcal{J}-1}} \quad \mathbf{I}_{n_{\mathcal{J}}} \quad \mathbf{0}_{n_{\mathcal{J}+1}} \cdots \mathbf{0}_{n_{1,2,3}} & \mathbf{C}_{\mathcal{J}} \end{array} \right]. \tag{3.6}$$

Let $\mathbf{H}_{\mathcal{J}} \in \mathbb{F}_q^{N \times n_{\mathcal{J}}}$ encapsulate the effect of the random linear network code on the packets $\bar{\mathbf{C}}_{\mathcal{J}}$ transmitted by $\mathcal{S}_{\mathcal{J}}$. The overall linear transformation, along with the injected error packets, can now be described in terms of individual lifted codewords as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{H}_1 & \cdots & \mathbf{H}_{1,2,3} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{C}}_1 \\ \vdots \\ \bar{\mathbf{C}}_{1,2,3} \end{bmatrix} + \mathbf{Z} = \mathbf{H}\bar{\mathbf{C}} + \mathbf{Z}. \tag{3.7}$$

We would like the destination to decode received packets using a low-complexity algorithm. Our approach is to let $\mathbf{G}$ be the generator matrix of a subcode of a Gabidulin code. As mentioned earlier, Gabidulin codes are well-studied and a variety of low-complexity decoders exist [Loi06; WAS13; SK09a]. We construct such a subcode, which we call a distributed Gabidulin code, using the techniques of Chapter 2. The main result of this chapter is given by the following theorem.

Theorem 3.1 (Main Result)**.** *Let $\mathcal{N}$ be a multiple-source multicast network of arbitrary topology with source nodes $\mathcal{S}$, and messages $\mathcal{M} = \{M_1, M_2, M_3\}$. Let an adversary corrupt up to z links of this network. For any rate vector $\mathbf{r}$ in the capacity region $\mathcal{R}$ given by* (3.1)*, a distributed network error-correcting code can be constucted as subcode of a suitable Gabidulin code.*

To prove the theorem, we derive linearized polynomial analogs of Propositions 2.4, 2.5 and 2.6 and show how, with a few extra technical steps, any point in the capacity region of a multiple-source multicast network can be achieved. The next section is devoted to constructing distributed Gabidulin codes and proving Theorem 3.1.

3.3 Construction

The set of linearized polynomials of q-degree less than k can be viewed as a k-dimensional vector space over $\mathbb{F}_{q^m}$. With this view in mind, we start by deriving analogs of various facts and propositions presented in Section 2.3.

Definition 3.4. *Let $\mathcal{R} \subset \mathbb{F}_{q^m}$ be a set of linearly independent elements. The set of linearized polynomials of q-degree less than k whose root space contains the subspace $\langle \mathcal{R} \rangle$ is denoted by $\mathsf{poly}_k\left(\langle \mathcal{R} \rangle\right)$.*

Fact 3.8. *The dimension of $\mathsf{poly}_k\left(\langle \mathcal{R} \rangle\right)$ is $(k - |\mathcal{R}|)^+$.*

Proof. Note that the claim on the dimension follows from the fact that any polynomial $P(x) \in \mathsf{poly}_k\left(\langle \mathcal{R} \rangle\right)$ can be written as $P(x) = V(x) \otimes M_{\mathcal{R}}(x)$ using Fact 3.5. From there, it follows that the q-degree of $V(x)$ is at most $k - |\mathcal{R}|$. □

Let us consider the intersection of two spaces $\mathsf{poly}_k\left(\langle \mathcal{R}_1 \rangle\right)$ and $\mathsf{poly}_k\left(\langle \mathcal{R}_2 \rangle\right)$. For the sake of clarity, define $\mathcal{P}_i := \mathsf{poly}_k\left(\langle \mathcal{R}_i \rangle\right)$.

Fact 3.9. $\mathcal{P}_1 \cap \mathcal{P}_2 = \mathsf{poly}_k\left(\langle \mathcal{R}_1 \cup \mathcal{R}_2 \rangle\right)$.

Proof. Let $\langle \mathcal{R} \rangle$ denote the root space of $p(x) \in \mathcal{P}_1 \cap \mathcal{P}_2$. It follows that $\langle \mathcal{R}_i \rangle \subseteq \langle \mathcal{R} \rangle$ for $i = 1, 2$, and being a linear space, the root space $\langle \mathcal{R} \rangle$ contains the Minkowski sum $\langle \mathcal{R}_1 \rangle + \langle \mathcal{R}_2 \rangle = \langle \mathcal{R}_1 \cup \mathcal{R}_2 \rangle$. The reverse inclusion is immediate. □

Now let us consider the Minkowski sum of two spaces $\mathcal{P}_1$ and $\mathcal{P}_2$. We have the following lemma.

Lemma 3.1. *Let the Minkowski sum of the spaces $\mathcal{P}_1$ and $\mathcal{P}_2$ be defined as*

$$\langle \mathcal{P}_1, \mathcal{P}_2 \rangle = \{\delta_1 P_1(x) + \delta_2 P_2(x) : P_1(x) \in \mathcal{P}_1, P_2(x) \in \mathcal{P}_2, \delta_1, \delta_2 \in \mathbb{F}_{q^m}\}.$$

Suppose that $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2|$. Then, we have that $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle = \mathsf{poly}_k\left(\mathcal{R}_1 \cap \mathcal{R}_2\right)$.

Proof. Let us first show that $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle \subseteq \mathsf{poly}_k\left(\mathcal{R}_1 \cap \mathcal{R}_2\right)$. Denote by $\mathcal{B}$ the root space $\langle \mathcal{R}_1 \cap \mathcal{R}_2 \rangle$. For any $P_1(x) \in \mathcal{P}_1$ and $P_2(x) \in \mathcal{P}_2$, Fact 3.5 tells us that $P_1(x) = Q_1(x) \otimes M_{\mathcal{B}}(x)$ and $P_2(x) = Q_2(x) \otimes M_{\mathcal{B}}(x)$, where the $Q_i(x)$'s are linearized polynomials of q-degree less than k. Since the set of linearized polynomials is a ring, we conclude that

$$\begin{aligned} \delta_1 P_1(x) + \delta_2 P_2(x) &= \delta_1 Q_1(x) \otimes M_{\mathcal{B}}(x) + \delta_2 Q_2(x) \otimes M_{\mathcal{B}}(x) \\ &= (\delta_1 Q_1(x) + \delta_2 Q_2(x)) \otimes M_{\mathcal{B}}(x) \\ &= Q(x) \otimes M_{\mathcal{B}}(x), \end{aligned}$$

where we have defined $Q(x) := (\delta_1 Q_1(x) + \delta_2 Q_2(x))$. By definition, the polynomial $\delta_1 P_1(x) + \delta_2 P_2(x)$ lives in $\mathsf{poly}_k\left(\mathcal{B}\right) = \mathsf{poly}_k\left(\langle \mathcal{R}_1 \cap \mathcal{R}_2 \rangle\right)$. Furthermore, we can express $\mathsf{dim}(\langle \mathcal{P}_1, \mathcal{P}_2 \rangle)$ as follows,

$$\begin{aligned} \mathsf{dim}(\langle \mathcal{P}_1, \mathcal{P}_2 \rangle) &= \mathsf{dim}(\mathcal{P}_1) + \mathsf{dim}(\mathcal{P}_2) - \mathsf{dim}(\mathcal{P}_1 \cap \mathcal{P}_2) \\ &= k - |\mathcal{R}_1| + k - |\mathcal{R}_2| - (k - |\mathcal{R}_1 \cup \mathcal{R}_2|)^+ \\ &= k - (|\mathcal{R}_1| + |\mathcal{R}_2| - |\mathcal{R}_1 \cup \mathcal{R}_2|) \\ &= k - |\mathcal{R}_1 \cap \mathcal{R}_2| \\ &= \mathsf{dim}(\mathsf{poly}_k\left(\langle \mathcal{R}_1 \cap \mathcal{R}_2 \rangle\right)). \end{aligned}$$

Thus, we assert that $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle = \mathsf{poly}_k\left(\mathcal{R}_1 \cap \mathcal{R}_2\right)$. □

An inductive argument leads to the following proposition:

Proposition 3.1. *Let $k \geq |\cup_{i=1}^{n} \mathcal{R}_i|$. Then $\langle \mathcal{P}_1, \ldots, \mathcal{P}_n \rangle = \mathsf{poly}_k\left(\langle \mathcal{R}_1 \cap \cdots \cap \mathcal{R}_n \rangle\right)$.*

Up until now, the $\mathbb{F}_{q^m}$-subspaces of linearized polynomials defined as $\mathsf{poly}_k\left(\langle \mathcal{R}_i \rangle\right)$ behave very similarly to classical sets of polynomials. Indeed, a comparison between Proposition 2.1 and the one presented in this chapter reveals this intimacy. This last corollary will endow us with the final building block required to use Proposition 2.1.

Corollary 3.1. *Suppose $k \geq | \cup_{i=1}^{n} \mathcal{R}_i|$. Then, we have that*

$$\mathcal{P}_n \cap \langle \mathcal{P}_1, \dots, \mathcal{P}_{n-1} \rangle = \langle \mathcal{P}_1 \cap \mathcal{P}_n, \dots, \mathcal{P}_{n-1} \cap \mathcal{P}_n \rangle.$$

Proof. Using Fact 3.9 and Proposition 3.1, we know that $\mathcal{P}_n \cap \langle \mathcal{P}_1, \dots, \mathcal{P}_{n-1} \rangle = \mathsf{poly}_k\left(\left\langle \mathcal{R}_n \cup (\cap_{i=1}^{n-1} \mathcal{R}_i)\right\rangle\right)$. Let $\mathcal{R}_i' = \mathcal{R}_1 \cup \mathcal{R}_n$. Since $\mathcal{R}_n \cup (\cap_{i=1}^{n-1} \mathcal{R}_i) = \cap_{i=1}^{n-1} \mathcal{R}_i'$, we define $\mathcal{P}_i' = \mathcal{P}_1 \cap \mathcal{P}_n = \mathsf{poly}_k\left(\mathcal{R}_i'\right)$ and apply the same proposition again to obtain the result. □

As argued earlier, we assume that the min-cuts dictating the capacity region (3.1) are due to the bipartite graph describing the relationship between the messages and the source nodes. As a result, the problem of constructing a capacity-achieving distributed Gabidulin code becomes completely analogous, in principle, to that of a distributed Reed–Solomon code from Chapter 2. Hence, we identify with each network a bipartite graph $G = (\mathcal{M}, \mathcal{S}, \mathcal{E})$ in which $(i, j) \in \mathcal{E}$ if and only if S_j can access message M_i. The capacity region can then be immediately read the corresponding adjacency matrix or equivalent from (3.5). We can now state the first result.

3.4 Code Construction for Networks with Two Messages

Proposition 3.2 (Two-Source Distributed Gabidulin Code)**.** *Let $G = (\mathcal{M}, \mathcal{S}, \mathcal{E})$ be a multiple source multicast network with two messages, in which z links are corrupted by an omniscient adversary. Let $\mathcal{R}_i$ be the set of source nodes with no access to M_i. Then, for any rate pair (r_1, r_2) such that*

$$\begin{aligned} r_1 &\leq k - |\mathcal{R}_1|, \\ r_2 &\leq k - |\mathcal{R}_2|, \\ r_1 + r_2 &\leq k - |\mathcal{R}_1 \cap \mathcal{R}_2|, \end{aligned}$$

there exists a distributed Gabidulin code capable of correcting up to z rank errors.

Proof. The proof of this proposition is identical to that of Proposition 2.2. The polynomial spaces considered are replaced with linearized polynomial spaces for which Definition 2.3 and Fact 2.3 apply. □

3.5 Code Construction for Networks with Three Messages

Having established the existence of distributed Gabidulin codes for multiple source multicast networks with two messages, we now present the first result pertinent to networks with three messages. On a high level, we consider two different cases. The first is one in which a purely algebraic solution is possible similar to that of Proposition 2.4.

Code Construction: $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$**.**

Proposition 3.3. *Let $G = (\mathcal{M}, \mathcal{S}, \mathcal{E})$ be a multiple source multicast network with three message, in which z links are corrupted by an omniscient adversary. Let $\mathcal{R}_i$ be the set of source nodes with no access to M_i, and assume that $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$, where $k = |\mathcal{S}| - 2z$. Then, a distributed Gabidulin code exists for any point in the capacity region of G, capable of correcting z rank errors.*

Proof. First, let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_{q^m}$ be a set of linearly independent field elements. Furthermore, identify α_i with the i^{th} row of $\mathbf{G}$ in (3.5). Since $k \geq |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$, Fact 3.9, Proposition 3.1 and Corollary 3.1 render Lemma 2.2 and Proposition 2.3 from Chapter 2 valid for linearized polynomials, when defining $\mathcal{P}_i := \mathsf{poly}_k(\langle \mathcal{R}_i \rangle)$. This allows us to express the capacity region of the network in terms of the bounds from the Proposition since the capacity region is completely determined by G. With this observation, the allocation of transformation polynomials can be done as required and so the proof is complete. □

Code Construction: $k < |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$ **and** $r_i \leq n_i$**.**

Remember the definition $n_i = |\mathcal{S}_i|$, the number of source nodes with access to message M_i. Let us now present the counterpart of Proposition 2.5. In this case, we are able to construct distributed Gabidulin codes by reducing the problem in hand to two subproblems and then relying on Proposition 3.2.

Proposition 3.4. *Let $G = (\mathcal{M}, \mathcal{S}, \mathcal{E})$ be a multiple source multicast network with three messages, in which z links are corrupted by an omniscient adversary. Let $\mathcal{S}_i$ be the set of source nodes with access to M_i only. If for some $i \in \{1, 2, 3\}$ we have $r_i \leq |\mathcal{S}_i|$, a distributed Gabidulin code exists for any point in the capacity region of G, capable of correcting z rank errors.*

Proof. Remember that $n_{\mathcal{I}} = |\mathcal{S}_{\mathcal{I}}|$. Without loss of generality, suppose $r_1 \leq n_1$. As usual identify the j^{th} row of $\mathbf{G}$ with α_j. Choose $\alpha_1, \ldots, \alpha_n$ so that they are linearly independent in $\mathbb{F}_{q^m}$. Let $\mathcal{R}_1 = \{j : [\mathbf{A}]_{1,j} = 0\}$ and define for each $j = 1, \ldots, r_1$

$$\mathcal{D}_j = \{\alpha^j : j \in \mathcal{R}_1\} \cup \{\alpha_1, \ldots, \alpha_{r_1}\} \setminus \{\alpha_j\}.$$

Finally, let $T_j(x)$ be the minimal linearized polynomial of $\langle \mathcal{D}_j \rangle$;

$$T_j(x) = \prod_{\gamma \in \langle \mathcal{D}_j \rangle} (x - \gamma).$$

The evaluation of the $T_j(x)$'s forms a set of codewords which, when organized as a matrix, have the following form.

$$\mathbf{G}_1 = \begin{bmatrix} T_1(\alpha_1) & \cdots & T_{r_1}(\alpha_1) \\ T_1(\alpha_2) & \cdots & T_{r_1}(\alpha_2) \\ \vdots & \ddots & \vdots \\ T_1(\alpha_n) & \cdots & T_{r_1}(\alpha_n) \end{bmatrix} = \left[\begin{array}{cccc} \star & 0 & \cdots & 0 \\ 0 & \star & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \star \\ \times & \times & \cdots & \times \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \times & \times & \cdots & \times \\ \hline \times & \times & \cdots & \times \\ \hline \times & \times & \cdots & \times \end{array}\right]. \tag{3.8}$$

As before, the symbol $\star$ represents a nonzero element from $\mathbb{F}_{q^m}$, and the symbol $\times$ represents a column vector of such elements. The blocks of rows correspond to $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_{2,3}, \mathcal{S}_{1,3}, \mathcal{S}_{1,2}, \mathcal{S}_{1,2,3}\}$. We have made it pictorially evident that $r_1 \leq |\mathcal{S}_1|$. Since the α_i's are linearly independent, we are guaranteed that α_j is not a root of $P_j(x)$, ensuring that the first diagonal entries are indeed nonzero and so $\mathbf{G}_1$ has full column rank.

From here, the allocation for $\mathbf{G}_2$ and $\mathbf{G}_3$ follows immediately from Proposition 3.2 and the concatenation of $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ as the columns of the matrix $\mathbf{G}$ forms a full rank generator matrix , as in (3.5), for a distributed Gabidulin code. □

Code Construction: $k < |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$ **and** $r_i > n_i$**.**

The last outstanding case is that of $k < |\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3|$ in which a decomposition like the one used in Proposition 3.4 is not possible. It is at this point that an analog of Proposition 2.6 requires slightly more effort.

In this case, we assume that $r_i > n_i$ for all i. Similar to the approach of Chapter 2, each matrix $\mathbf{G}_i$ is partitioned into two blocks, where for example

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{G}_{1,a} & \mathbf{G}_{1,b} \end{bmatrix} = \left[\begin{array}{cccc|cccc} \star & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \star & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \star & 0 & 0 & \cdots & 0 \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \hline \times & \times & \cdots & \times & \times & \times & \cdots & \times \\ \hline \times & \times & \cdots & \times & \times & \times & \cdots & \times \\ \hline \times & \times & \cdots & \times & \times & \times & \cdots & \times \end{array}\right]. \tag{3.9}$$

The blocks $\mathbf{G}_{i,a}$ can be chosen as described earlier in the proof of Proposition 3.4. At this point, it might be tempting to mimic the construction of Propositions 2.5 and 2.6 when construction the $\mathbf{G}_{i,b}$'s . We argue now that this method does not carry through immediately and use this opportunity to motivate the correct construction. Define $\bar{n} := n_1 + n_2 + n_3$ and $t := k - \bar{n} - 1$.

Depending on whether assumptions (2.22),(2.23) or assumptions (2.29),(2.30) hold, define the sets $\mathcal{J}_1, \mathcal{J}_2$ and $\mathcal{J}_3$ according to (2.25),(2.26) and (2.27) or (2.31), (2.32) and (2.33), respectively. The techniques presented henceforth do not rely on which case is true. For the sake of clarity, we list these sets when, without loss of generality, the assumptions (2.29),(2.30) hold.

$$\mathcal{J}_1 = \{n_{23} - t, \ldots, n_{23} - t + \rho_1 - 1\}, \tag{3.10}$$
$$\mathcal{J}_2 = \{n_{23} - t + \rho_1, \ldots, n_{23} - t + \rho_1 + \rho_2 - 1\}, \tag{3.11}$$
$$\mathcal{J}_3 = \{n_{23} - t + \rho_1 + \rho_2, \ldots, n_{23} - t + \rho_1 + \rho_2 + \rho_3 - 1\}, \tag{3.12}$$

where $\rho_i = r_i - n_i$. Next, define the sets

$$\mathcal{N} := \{\gamma_1, \ldots, \gamma_{\bar{n}}\}, \tag{3.13}$$
$$\mathcal{B}_j := \{\beta_{j+1}, \ldots, \beta_{j+t}\}. \tag{3.14}$$

We associate the first $\bar{n}$ rows of the desired $\mathbf{G}$ with $\mathcal{N} = \{\gamma_1, \ldots, \gamma_{\bar{n}}\}$ and the rest with $\mathcal{B} = \{\beta_1, \ldots, \beta_{n-\bar{n}}\}$. The sets $\mathcal{N}$ and $\mathcal{B}$ are chosen to be jointly linearly independent. Furthermore, the subscripts of the β_i's are taken modulo $n - \bar{n}$, after which we add one. One potentially choice for the set of polynomials that define $\mathbf{G}_{i,b}$ is

$$\mathcal{T}_{i,b} = \{M_{\mathcal{N}\cup\mathcal{B}_j}(x) : j \in \mathcal{J}_i\}. \tag{3.15}$$

The polynomial $M_{\mathcal{N}\cup\mathcal{B}_j}(x)$ ensures that the entries of $\mathbf{G}_{i,b}$ corresponding to $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$ all evaluate to zero as required by (3.9). Furthermore, if we consider $\mathbf{G}_{1,b}$, the entries corresponding to $\mathcal{S}_{2,3}$ also evaluate to zero since, given the way $\mathcal{J}_1$ was chosen, the set $\mathcal{B}_j$ ensures that as guaranteed by the proof of Propositions 2.5 and 2.6. The same claim holds for $\mathbf{G}_{2,b}$ and $\mathbf{G}_{3,b}$.

By Fact 3.5, we can express the polynomials in $\mathcal{T}_b$ as

$$\mathcal{T}_b = \{Q_j(x) \otimes M_{\mathcal{N}}(x) : j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3\}. \tag{3.16}$$

Lemma 3.2. *The polynomials $\mathcal{T}_b$ from* (3.16) *are linearly independent if and only if the $Q_j(x)$'s are so.*

Proof. The polynomials are linearly dependent if and only if $\sum_j c_j Q_j(x) \otimes M_{\mathcal{N}}(x) = 0$ for some non zero $c_j \in \mathbb{F}_{q^m}$. This is equivalent to

$$\left(\sum_j c_j Q_j(x)\right) \otimes M_{\mathcal{N}}(x) = 0.$$

Given that the ring of linearized polynomials has no zero-divisors, this holds when $\sum_j c_j Q_j(x) = 0$, i.e. if and only if the $Q_j(x)$'s are linearly dependent. □

Unlike the case of regular polynomials, it is not immediately clear what the $Q_j(x)$'s are. In the case of Reed–Solomon codes, they were precisely the counterparts of the $M_{\mathcal{B}_j}(x)$'s, which were constructed to be linearly independent. Since we cannot make the same claim here, we will choose the sets in (3.13) and (3.14) carefully so that we can assert their linear independence. Toward this end, we present a series of technical results that culminate in a choice of $\mathcal{N}$ and $\mathcal{B}$ so that the $Q_j(x)$'s are linearly independent.

Let $P(x)$ be the minimal linearized polynomial of $\{\alpha^{b+1}, \dots, \alpha^{b+t}\}$, where α is primitive in $\mathbb{F}_{q^m}$. It turns out that even though $P(x)$ is linearized, its coefficients[3] are all nonzero.

[3] The coefficients of a linearized polynomials are those corresponding to monomials of the the form x^{q^i}.

Lemma 3.3. *Let $P(x)$ be the minimal linearized polynomial of $\{\alpha^{b+1}, \ldots, \alpha^{b+t}\}$, where α is primitive in $\mathbb{F}_{q^m}$. Then, all coefficients of $P(x)$ are nonzero.*

Proof. The polynomial $P(x)$ is of q-degree t since its root space is spanned by t linearly independent elements in $\mathbb{F}_{q^m}$. Furthermore, it has at most $t+1$ coefficients since it is linearized. An application of the BCH bound 2.4 tells us all these coefficients are nonzero. □

Proposition 3.5. *Let $P(x)$ be the minimal linearized polynomial of $\{\alpha^{b+1}, \ldots, \alpha^{b+t}\}$ and consider the set of polynomials $\{P(\alpha^{-j}x) : j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3\}$. The polynomials are linearly independent over $\mathbb{F}_{q^m}$ if and only if the elements in $\{\alpha^{-j} : j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3\}$ are linearly independent over $\mathbb{F}_q$.*

Proof. Define $s := |\mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3| = \rho_1 + \rho_2 + \rho_3$. Write $P(\alpha^j x) = \sum_{i=0}^{t} p_i \alpha^{j[i]} x^{[i]}$ and consider the matrix $\mathbf{P}$ whose columns are the coefficients of $P(\alpha^{-j}x)$ for $j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3$,

$$\mathbf{P} = \begin{bmatrix} p_0\alpha^{j_1} & p_0\alpha^{j_2} & \cdots & p_0\alpha^{j_s} \\ p_1\alpha^{j_1[1]} & p_1\alpha^{j_2[1]} & \cdots & p_1\alpha^{j_s[1]} \\ \vdots & \vdots & \ddots & \vdots \\ p_t\alpha^{j_1[t]} & p_t\alpha^{j_2[t]} & \cdots & p_t\alpha^{j_s[t]} \end{bmatrix}.$$

By the cut-set bounds we know that $s \leq k$, which enables us to form $\hat{\mathbf{P}}$ from the first s rows of $\mathbf{P}$. Writing out the determinant yields

$$\begin{aligned} \mathsf{det}(\hat{\mathbf{P}}) &= \begin{vmatrix} p_0\alpha^{j_1} & p_0\alpha^{j_2} & \cdots & p_0\alpha^{j_s} \\ p_1\alpha^{j_1[1]} & p_1\alpha^{j_2[1]} & \cdots & p_1\alpha^{j_s[1]} \\ \vdots & \vdots & \ddots & \vdots \\ p_{s-1}\alpha^{j_1[s-1]} & p_{s-1}\alpha^{j_2[s-1]} & \cdots & p_{s-1}\alpha^{j_s[s-1]} \end{vmatrix} \\ &= \begin{vmatrix} \alpha^{j_1} & \alpha^{j_2} & \cdots & \alpha^{j_s} \\ \alpha^{j_1[1]} & \alpha^{j_2[1]} & \cdots & \alpha^{j_s[1]} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{j_1[s-1]} & \alpha^{j_2[s-1]} & \cdots & \alpha^{j_s[s-1]} \end{vmatrix} \prod_{i=0}^{s-1} p_i. \end{aligned} \tag{3.17}$$

The polynomial $P(x)$ has t cyclically consecutive roots and so by the BCH bound all $t+1$ p_i's are non-zero. The Vandermonde-like matrix in (3.17) has non-zero determinant if and only if $\{\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_s}\}$ are linearly independent over $\mathbb{F}_q$ [LN97, p.109]. Hence, $\mathsf{det}(\hat{\mathbf{P}}) \neq 0$ and $\mathbf{P}$ has full column rank and in turn the polynomials $P(\alpha^{-j}x)$, $j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3$ are linearly independent over $\mathbb{F}_{q^m}$ if and only if the elements in $\{\alpha^{-j} : j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3\}$ are pairwise distinct. □

With regards to their evaluation , the linearized polynomial $P(\alpha^{-l}x)$ behaves the same way as in the case for regular polynomials described in Lemma 2.3.

Lemma 3.4. *Let $P(x) = \prod_{\beta \in \langle \mathcal{U} \rangle}(x - \beta)$, where $\mathcal{U} = \{\alpha, \ldots, \alpha^t\}$. Fix a number l, then $P(\alpha^{-l}x) = \alpha^{-lq^t} M_{\mathcal{U}'}(x)$, where $\mathcal{U}' = \{\alpha^{l+1}, \ldots, \alpha^{l+t}\}$.*

Proof. By definition, we have

$$\begin{aligned} P(\alpha^{-l}x) &= \prod_{c_1,\ldots,c_t \in \mathbb{F}_q} \left(\alpha^{-l}x - \sum_{i=1}^{t} c_i \alpha^i \right) \\ &= \prod_{c_1,\ldots,c_t \in \mathbb{F}_q} \alpha^{-l} \left(x - \sum_{i=1}^{t} c_i \alpha^{i+l} \right) \\ &= \alpha^{-lq^t} \prod_{\gamma \in \left\langle \alpha^{l+1},\ldots,\alpha^{l+t} \right\rangle} (x - \gamma). \end{aligned}$$

□

Lemma 3.5. *Let $\mathcal{B}_j = \{\beta_{j+1}, \ldots, \beta_{j+t}\}$, $\mathcal{N} = \{\gamma_1, \ldots, \gamma_{\bar{n}}\}$ where $\beta_i, \gamma_i \in \mathbb{F}_{q^m}$. Now define $\delta_j = M_{\mathcal{N}}(\beta_j)$ and let $\mathcal{B}'_j = \{\delta_1, \ldots \delta_t\}$. Then the elements in $\mathcal{B}'_j$ are linearly independent over $\mathbb{F}_q$ if and only if the elements in $\mathcal{B}_j$ are linearly independent over $\mathbb{F}_q$ and $\langle \mathcal{B}_j \rangle \cap \langle \mathcal{N} \rangle = 0$.*

Proof. By definition, if the elements in $\mathcal{B}'_j$ are linearly independent then for $c_1, \ldots, c_t \in \mathbb{F}_q$ not all equal to zero, we have

$$\sum_{i=1}^{t} c_i M_{\mathcal{N}}(\beta_{j+i}) = M_{\mathcal{N}} \left(\sum_{i=1}^{t} c_i \beta_{j+i} \right) \neq 0.$$

This implies that $\sum_{i=1}^{t} c_i \beta_{j+i} \notin \langle \mathcal{N} \rangle$, the root space of $M_{\mathcal{N}}(x)$, implying that $\langle \mathcal{B}_j \rangle \cap \langle \mathcal{N} \rangle = 0$. In particular, we also have $\sum_{i=1}^{t} c_i \beta_{j+i} \neq 0$, so the elements in $\mathcal{B}_j$ are linearly independent. On the other hand, if any element $\sum_{i=1}^{t} c_i \beta_{j+i}$ does not lie in $\langle \mathcal{N} \rangle$, and is not equal to zero, we have

$$0 \neq M_{\mathcal{N}} \left(\sum_{i=1}^{t} c_i \beta_{j+i} \right) = \sum_{i=1}^{t} c_i M_{\mathcal{N}}(\beta_{j+i}) = \sum_{i=1}^{t} c_i \delta_i.$$

As a consequence, the elements of $\mathcal{B}'_j$ are linearly independent thus the proof is complete. □

The lemma asserts that $M_{\mathcal{N}}(x)$ defines an isomorphism between $\mathcal{B}_j$ and $\mathcal{B}'_j$. This is readily seen when one views the polynomial as an $\mathbb{F}_q$-linear map from $\mathbb{F}_{q^m}$ to $\mathbb{F}_{q^m}$ defined by its kernel $\langle \mathcal{N} \rangle$ and restricts its domain to a space that intersects trivially with this kernel. Lemma 3.5 allows us to characterize the $Q_j(x)$'s from (3.16).

Lemma 3.6. *Consider an arbitrary polynomial in* (3.16) *given by* $P_j(x) = Q_j(x) \otimes M_{\mathcal{N}}(x)$*. Let* $\mathcal{B}_j = \{\beta_{j+1}, \ldots, \beta_{j+t}\}$ *and* $\mathcal{B}'_j = \{\delta_1, \ldots, \delta_t\}$*, where* $\delta_i = M_{\mathcal{N}}(\beta_{j+i})$*. If* $\langle \mathcal{B}_j \rangle \cap \langle \mathcal{N} \rangle = 0$*, then one has* $Q_j(x) = M_{\mathcal{B}'_j}(x)$*.*

Proof. An application of Lemma 3.5 reveals that $\mathsf{dim}(\langle \mathcal{B}_j \rangle) = \mathsf{dim}(\langle \mathcal{B}'_j \rangle)$. Furthermore, we have

$$\mathsf{deg}_q\, Q_j(x) = \mathsf{deg}_q\, M_{\mathcal{N} \cup \mathcal{B}_j}(x) - \mathsf{deg}_q\, M_{\mathcal{N}}(x) = \mathsf{dim}(\langle \mathcal{B}_j \rangle),$$

by Fact 3.7 and the fact that $\langle \mathcal{B}_j \rangle \cap \langle \mathcal{N} \rangle = 0$. Lastly, for any element $\gamma \in \langle \mathcal{B}'_j \rangle$, one has $Q_j(\gamma) = 0$. Hence, the root space of $\langle \mathcal{B}'_j \rangle$ is contained in the root space of $Q_j(x)$ and as a consequence, they are in fact equal. □

Suppose now that we can choose the sets $\mathcal{N}$ and $\mathcal{B}_j$ in (3.13) and (3.14) so that for $j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3$, the polynomial $M_{\mathcal{N}}(x)$ maps $\mathcal{B}_j$ to $\{\alpha^{j+1}, \ldots, \alpha^{j+t}\}$. This will ensure that $Q_j(x)$ from (3.16) is nothing but $P(\alpha^{-j}x)$ (up to scaling), where $P(x)$ is the minimal linearized polynomial of $\{\alpha, \ldots, \alpha^t\}$. By Proposition 3.5, we can conclude that the $Q_j(x)$'s are linearly independent. We are in a position that allows to state the main result of this section.

Proposition 3.6. *Let* $G = (\mathcal{M}, \mathcal{S}, \mathcal{E})$ *be a multiple source multicast network with three messages, in which* z *links are corrupted by an omniscient adversary. Let* $\mathcal{S}_i \subseteq \mathcal{S}$ *be the set of source nodes with access to* M_i *only. Suppose that for* $i \in \{1, 2, 3\}$ *we have* $r_i > |\mathcal{S}_i|$*. Then, for any point in the capacity region of G, a distributed Gabidulin code exists, for a special choice of code coordinates, capable of correcting* z *rank errors.*

Allow us to emphasize the fact that the choice of code coordinates is not arbitrary. Indeed, a major component of the proof of this proposition is devoted to crafting the defining elements of the constituent Gabidulin code.

Proof. Identify the first $\bar{n}$ rows of $\mathbf{G}$ in (3.9) with $\mathcal{N} = \{\gamma_1, \ldots, \gamma_{\bar{n}}\}$ and the remaining rows with $\mathcal{B} = \{\beta_1, \ldots, \beta_\nu\}$, where $\nu := n - \bar{n}$. Let $\mathcal{A} = \{\alpha, \ldots, \alpha^\nu\}$.

Furthermore, define $\mathcal{B}_j$ as in (3.14). For $j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3$ from (3.10), (3.11) and (3.12), the polynomials $M_{\mathcal{N} \cup \mathcal{B}_j}(x)$, when evaluated on $\mathcal{N} \cup \mathcal{B}$, form a generator matrix $\mathbf{G}$ that obeys the mask given in (3.9). This is immediate from Proposition 2.6. Now by Lemma 3.5, the polynomial $M_{\mathcal{B}_j \cup \mathcal{N}}(x)$ factors as

$$M_{\mathcal{B}_j \cup \mathcal{N}}(x) = Q_j(x) \otimes M_{\mathcal{N}}(x).$$

Suppose that there is a choice of jointly linearly independent $\mathcal{N}$ and $\mathcal{B}$ so that $M_{\mathcal{N}}(x)$ maps $\mathcal{B}_j$ to $\mathcal{A}_j = \{\alpha^{j+1}, \ldots, \alpha^{j+t}\}$. By Lemma 3.6, we have $Q_j(x) = M_{\mathcal{A}_j}(x)$ which, by construction, is equal (up to a constant) to $M_{\mathcal{A}}(\alpha^{-j}x)$. We conclude by Proposition 3.5 that the $M_{\mathcal{A}}(\alpha^{-j}x)$'s, and hence the $Q_j(x)$'s, are linearly independent over $\mathbb{F}_{q^m}$. As a result, the the set of polynomials

$$\mathcal{T}_b = \{M_{\mathcal{B}_j \cup \mathcal{N}}(x) : j \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3\}$$

corresponds to a full rank generator matrix of a distributed Gabidulin code designed for G. □

In the next section, we provide a recipe that constructs the sets $\mathcal{N}$ and $\mathcal{B}$ with the prescribed properties.

Choosing the Coordinates of the Constituent Code

For simplicity, let $\nu := n - \bar{n}$. From Lemma 3.5, we are to simultaneously find $\mathcal{N} = \{\gamma_1, \ldots, \gamma_{\bar{n}}\}$ and $\mathcal{B} = \{\beta_1, \ldots, \beta_\nu\}$, jointly linearly independent over $\mathbb{F}_q$, so that $M_{\mathcal{N}}(x)$ maps β_i to α^i. In this section, we provide a recipe that does exactly this using standard techniques from abstract algebra. First, let us introduce the trace map.

Definition 3.5. *The trace map* $\mathsf{Tr} : \mathbb{F}_{q^m} \to \mathbb{F}_q$ *is defined by* $\mathsf{Tr}\,(a) = \sum_{i=0}^{m-1} a^{[i]}$.

The following lemma shows how to find a member of the dual basis of a set of linearly independent elements in $\mathbb{F}_{q^m}$.

Lemma 3.7. *Let* $= \{u_1, \ldots, u_t\}$ *be linearly independent elements over* $\mathbb{F}_q$ *and* $t < m$*. Then there exists* $\lambda \in \mathbb{F}_{q^m}$ *such that* $\mathsf{Tr}\,(\lambda u_i) = 0$ *for all* $u_i \in \mathcal{U}$*.*

Proof. The equations $\mathsf{Tr}\left(\lambda u_i\right) = 0$ can be expressed as the following linear system over $\mathbb{F}_{q^m}$.

$$\begin{bmatrix} u_1 & u_1^{[1]} & \cdots & u_1^{[m-1]} \\ \vdots & \vdots & \ddots & \vdots \\ u_t & u_t^{[1]} & \cdots & u_t^{[m-1]} \end{bmatrix} \begin{bmatrix} \lambda \\ \lambda^{[1]} \\ \vdots \\ \lambda^{[m-1]} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Let $b_1, \ldots, b_m$ be a basis for $\mathbb{F}_{q^m}$, and set $\lambda = \sum_{i=1}^{m} a_i b_i$, where $a_i \in \mathbb{F}_q$. Also, note that $\lambda^{[j]} = \sum_{i=1}^{m} a_i b_i^{[j]}$. The linear system can now be expressed as

$$\mathbf{UBa} = \begin{bmatrix} u_1 & u_1^{[1]} & \cdots & u_1^{[m-1]} \\ \vdots & \vdots & \ddots & \vdots \\ u_t & u_t^{[1]} & \cdots & u_t^{[m-1]} \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_m \\ b_1^{[1]} & b_2^{[1]} & \cdots & b_m^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{[m-1]} & b_2^{[m-1]} & \cdots & b_m^{[m-1]} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

This can be further developed to

$$\mathbf{UBa} = \begin{bmatrix} \mathsf{Tr}\left(u_1 b_1\right) & \mathsf{Tr}\left(u_1 b_2\right) & \cdots & \mathsf{Tr}\left(u_1 b_m\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{Tr}\left(u_t b_1\right) & \mathsf{Tr}\left(u_t b_2\right) & \cdots & \mathsf{Tr}\left(u_t b_m\right) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Note that matrix $\mathbf{U}$ has rank t and matrix $\mathbf{B}$ is invertible and so $\mathbf{UB}$ is of rank t over $\mathbb{F}_{q^m}$. By properties of the trace map, the matrix lives in the space of matrices with entries in $\mathbb{F}_q$; the solution to this system is a linear space over $\mathbb{F}_q$ of dimension $m - t$. □

Lemma 3.8. $\mathsf{Tr}\left(a\right) = 0$ *if and only if there exists* $b \in \mathbb{F}_{q^m}$ *such that* $a = b^q - b$.

Proof. Suppose $a = b^q - b$, then by definition,

$$\begin{aligned} \mathsf{Tr}\left(a\right) &= \sum_{i=0}^{m-1} a^{[i]} \\ &= \sum_{i=0}^{m-1} (b^q - b)^{[i]} \\ &= \sum_{i=0}^{m-1} b^{[i+1]} - y^{[i]} \\ &= b^{[m]} - b = 0. \end{aligned}$$

On the other hand, let b be a root of $x^q - x - a$ in some extension of $\mathbb{F}_q$. Then $\mathsf{Tr}\left(a\right) = \mathsf{Tr}\left(\beta^q - \beta\right) = \beta^{[m]} - \beta = 0$ and $\beta^{[m]} = \beta$ implying that $\beta \in \mathbb{F}_{q^m}$. □

The last building block is a proposition that enlarges a set of linearly independent elements from $\mathbb{F}_{q^m}$.

Lemma 3.9. *Let $\mathcal{U} = \{u_1, \ldots u_t\}$ be a linearly independent set in $\mathbb{F}_{q^m}$ and suppose λ is such that $\mathsf{Tr}\,(\lambda u_i) = 0$ for all i. Furthermore, put y_i such that $\lambda u_i = y_i^q - y_i$. The elements in $\mathcal{Y} = \{y_1, \ldots, y_t, 1\}$ are linearly independent over $\mathbb{F}_q$. Furthermore, the factorization $M_{\mathcal{Y}}(x) = M_{\mathcal{U}}(x) \otimes (x^q - x)/\lambda$ holds.*

Proof. Let $c_0 + \sum_{i=1}^{\tau} c_i y_i = 0$. Then $c_0 + \sum_{i=1}^{\tau} c_i y_i^q = 0$. Subtracting the first expression from the second yields $\sum_{i=1}^{\tau} c_i(y_i^q - y_i) = \sum_{i=1}^{\tau} c_i(\lambda u_i) = 0$. Thus $c_1 = \ldots = c_\tau = 0$ since the u_i are linearly independent by assumption, and consequently one has $c_0 = 0$. To prove the factorization of $M_{\mathcal{Y}}(x)$, note that $M_{\mathcal{Y}}(a) = 0$ for all $a \in \mathbb{F}_q$. Furthermore, one has $\prod_{a \in \mathbb{F}_q}(x - a) = x^q - x$ and so $M_{\mathcal{Y}} = P(x) \otimes (x^q - x)$. Let $P(x) = \sum_{i=0}^{t} p_i x^{[i]}$. It follows that

$$\begin{aligned} M_{\mathcal{Y}}(x) &= \sum_{i=0}^{t} p_i (x^q - x)^{[i]} \\ &= \sum_{i=0}^{t} p_i \lambda^{[i]} \left(\frac{x^q - x}{\lambda} \right)^{[i]} \\ &= \left(\sum_{i=0}^{t} p_i \lambda^{[i]} x^{[i]} \right) \otimes \left(\frac{x^q - x}{\lambda} \right). \end{aligned}$$

Let $Q(x) = \sum_{i=0}^{t} p_i \lambda^{[i]} x^{[i]}$. Then, for all y_i, we have

$$0 = M_{\mathcal{Y}}(y_i) = Q(x) \otimes \left(\frac{y_i^q - y_i}{\lambda} \right) = Q(u_i).$$

By Fact 3.7, we conclude that $Q(x) = M_{\mathcal{U}}(x)$. □

This proposition suggests an iterative approach to constructing $\mathcal{N}$ and $\mathcal{B}$.

Proposition 3.7. *Let $\mathcal{U}_0 = \{u_{0,1}, \ldots u_{0,t}\}$ be a linearly independent set. For $i = 0, \ldots, \bar{n} - 1$, choose λ_i such that $\mathsf{Tr}\,(\lambda_i u_{i,j}) = 0$ for all $j = 1, \ldots, t + i$ and define the set $\mathcal{U}_{i+1}$ recursively via*

$$\mathcal{U}_{i+1} = \{b_{i,1}, \ldots, b_{i,t+i}, 1\}, \tag{3.18}$$

where $b_{i,j}$ is such that $b_{i,j}^q - b_{i,j} = \lambda_i u_{i,j}$. Then, it holds that

$$M_{\mathcal{U}_{\bar{n}}}(x) = M_{\mathcal{U}_0}(x) \otimes \left(\frac{x^q - x}{\lambda_0} \right) \otimes \cdots \otimes \left(\frac{x^q - x}{\lambda_{\bar{n}-1}} \right). \tag{3.19}$$

Proof. We demonstrate the proof by induction. For $i = 0$, the result for $M_{\mathcal{U}_1}(x)$ follows from Lemma 3.9. Now suppose that for $j < \bar{n} - 1$, we have

$$M_{\mathcal{U}_j}(x) = M_{\mathcal{U}_0}(x) \otimes \left(\frac{x^q - x}{\lambda_0}\right) \otimes \cdots \otimes \left(\frac{x^q - x}{\lambda_{j-1}}\right). \tag{3.20}$$

By Lemma 3.9, the following factorization holds.

$$M_{\mathcal{U}_{j+1}}(x) = M_{\mathcal{U}_j}(x) \otimes \frac{x^q - x}{\lambda_j}. \tag{3.21}$$

Plugging in 3.20 into 3.21 completes the proof. □

We can now build $\mathcal{N}$ by applying Proposition 3.7 to $\mathcal{U}_0 = \{\alpha, \ldots, \alpha^\nu\}$. The resulting set is

$$\mathcal{U}_{\bar{n}} = \{u_{\bar{n},1}, \ldots, u_{\bar{n},\nu}, u_{\bar{n},\nu+1}, \ldots, u_{\bar{n},\nu+\bar{n}-1}, 1\},$$

which can be partitioned into $\mathcal{N}$ and $\mathcal{B}$ as

$$\begin{aligned} \mathcal{N} &= \{u_{\bar{n},\nu+1}, \ldots, u_{\bar{n},\nu+\bar{n}-1}, 1\}, \\ \mathcal{B} &= \{u_{\bar{n},1}, \ldots, u_{\bar{n},\nu}\}. \end{aligned}$$

For clarity, the procedure for building $\mathcal{B}$ and $\mathcal{N}$ is outlined in Algorithm 1.

Algorithm 1 Computing coordinates of constituent Gabidulin code

```
procedure COORDINATES($\nu, \bar{n}$)
    $u_{0,j} \leftarrow \alpha^j, \forall j \in \{1, \ldots, \nu\}$
    $\mathcal{U}_0 \leftarrow \{u_{0,1}, \ldots, u_{0,\nu}\}$
    for $i \leftarrow 0, \bar{n} - 1$ do
        $\lambda_i \leftarrow \lambda$ such that $\mathsf{Tr}\left(\lambda u_{i,j}\right) = 0, \forall u_{i,j} \in \mathcal{U}_i$
        $u_{i+1,j} \leftarrow a_j$ such that $\lambda_i u_{i,j} = a_j^q - a_j, \forall u_{i,j} \in \mathcal{U}_i$
        $\mathcal{U}_{i+1} \leftarrow \{u_{i+1,1}, \ldots, u_{i+1,\nu+i}, 1\}$
    end for
    return $\mathcal{U}_{\bar{n}}$
end procedure
```

As a result, the two sets $\mathcal{N}$ and $\mathcal{B}$ now define the coordinates of the constituent Gabidulin code from which a distributed code for a multisource multicast network can be extracted.

3.6 Discussion

In this section, we allude to a variety of topics pertinent to the topic of the current chapter. First, we discuss decoding aspects of distributed Gabidulin codes.

Decoding Distributed Gabidulin Codes

In [SKK08], Silva et al. showed how the concept of lifting a rank-metric code (Definition 3.3) provides the destination node with side information when decoding the received packets. The method of [SKK08] for decoding lifted Gabidulin codes can be used to decode our construction, which is essentially a subcode of a lifted Gabidulin code. The complexity of the decoding process is $O(dm)$ operations in $\mathbb{F}_{q^m}$, where d is the minimum rank distance of the constituent code.

Rank Deficiency of the Network

While designing distributed Gabidulin subject to a mask constraint on the generator matrix is interesting on its own, it is essential that we revisit the structure of the multiple-source multicast network that dictates this constraint. In particular, we justify our assumption that $m_{\mathcal{I}} = n$, where n is the total number of source nodes in the network. Indeed, under our assumption, we ensure that the network is one where the matrix $\mathbf{H}$ from 3.2 has full column rank. Let us now justify why this assumption is critical to our framework. Suppose that $\mathsf{rank}\,(\mathbf{H}) = n - \rho$ and let z be the total number of adverserial errors introduced in the network. The result [SK09a, Theorem 11] deems it necessary and sufficient to use a Gabidulin code of minimum distance $d > 2z + \rho$ to account for the rank-deficiency of $\mathbf{H}$. As a result, there are examples of networks for which designing a distributed Gabidulin code that is capacity achieving might not be possible. Note that in our setup, we are requiring that the symbols being transmitted by the source nodes, when viewed together as a signal codeword, live in a Gabidulin code.

Example 3.1. *Consider the cartoon depicted in Figure 3.2. Suppose $r_1 = r_2 = 1$ and the adversary injects a single erroneous packet at the outgoing link of V_1.*

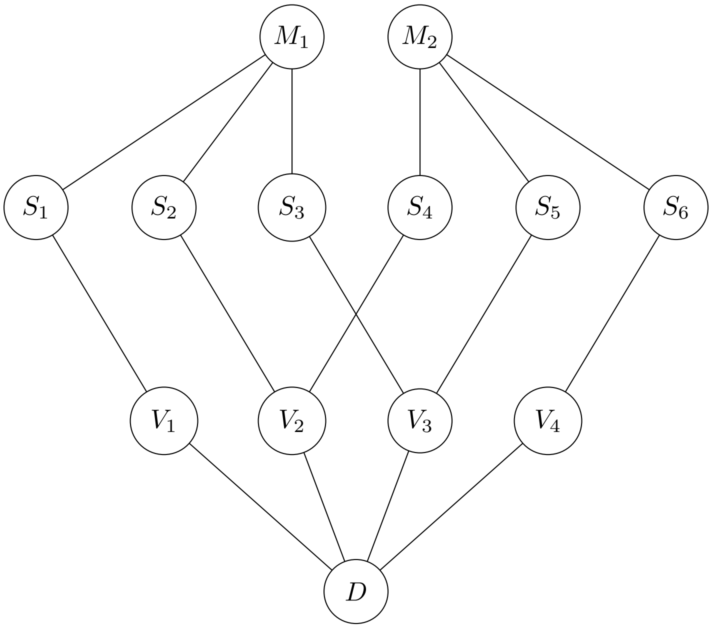


Figure 3.2: A multi-source multicast network that inherently introduces rank-deficiency into $\mathbf{H}$, the transformation describing the random linear network code. For each M_i, the min-cut between M_i and D is equal to 3 while the mincut between $\{M_1, M_2\}$ is equal to 4. Under the presence of a single erroneous link, the rate pair (r_1, r_2) is achievable.

The rank deficiency here is $\rho = 2$. By [SK09a, Theorem 11], a code of minimum distance 5 is necessary. The code length is $n = 6$, the total number of sources. Any distributed linear network error-correcting code must have the following generator matrix.

$$\mathbf{G} = \begin{bmatrix} a_1 & 0 \\ a_2 & 0 \\ a_3 & 0 \\ 0 & b_1 \\ 0 & b_2 \\ 0 & b_3 \end{bmatrix}. \tag{3.22}$$

The first three rows correspond to the outgoing links of S_1, S_2, S_3 and the rest correspond to S_4, S_5, S_6. Since every column of $\mathbf{G}$ is of weight at most 3, it is impossible to construct the distributed Gabidulin code. The example could potentially be motivating to consider other metrics, and in particular the injection metric from [SK09b], for the construction. Indeed, the achievability scheme of Dikaliotis et al. in [Dik+10] heavily depends on the injection metric.

The Constituent Code's Coordinates

Note that the construction of Proposition3.6 heavily relies on a particular choice of coordinates that was constructed in Section 3.5. On the other hand, all other constructs were valid for a general choice of linearly independent elements. For the sake of elegance, it is worth exploring a unified choice of coordinates that is suitable for all given constructions. In Chapter 2, it was possible to naturally choose the coordinates as powers of a primtive element. It is worth mentioning that elements of a normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ could potentially be used in the setting considered in this chapter.

With regards to the finite field $\mathbb{F}_{q^m}$ over which our constructions operate, we have that $m > n$ is sufficient. Propositions 3.3, 3.6 require n linearly independent elements in $\mathbb{F}_{q^m}$. Furthermore, Proposition 3.7 extends a basis of size $n - \bar{n}$ to one of size n. The bound $m > n$ is optimal for Gabidulin codes when the min-cut of the network n is interpreted as the blocklength of the code.

Chapter 4

CODING WITH CONSTRAINTS: SYSTEMATIC CONSTRUCTIONS

4.1 Introduction

We examine in this chapter an error-correcting coding framework in which we must encode s message symbols using a length n error-correcting code subject to a set of encoding constraints. Specifically, each coded symbol is a function of only a subset of the message symbols. This setup generalizes the one considered in Chapter 2 and arises in various situations. One such case is that of a sensor network in which each sensor can measure a certain subset of a set of parameters. The sensors would like to collectively encode the readings to allow for the possibility of measurement errors. Another scenario is one in which a client wishes to download data files from a set of servers, each of which stores information about a subset of the data files. The user should be able to recover all of the data even in the case when some of the file servers fail. Ideally, the user should also be able to download the files faster in the absence of server failures. To protect against errors, we would like the coded symbols to form an error-correcting code with reasonably high minimum distance. On the other hand, efficient download of data is permitted when the error-correcting code is of systematic form. We present an upper bound on the minimum distance of an error-correcting code when subjected to encoding constraints, reminiscent of the cut-set bounds of [Dik+10]. Under a certain technical condition, we provide a code construction that achieves this bound. It turns out that this condition is sufficient for a systematic construction based on Reed–Solomon codes. Furthermore, we refine our bound in the case that we demand a systematic linear error-correcting code, and present a construction that achieves the bound. In both cases, the codes can be decoded efficiently due to the fact that our construction utilizes Reed-Solomon codes.

Related Work

The problem of constructing error-correcting codes with constrained encoding has been addressed by a variety of authors. Dau et al. [Dau+13; DSY14; DSY15] considered the problem of finding linear MDS codes with constrained generator matrices. They have shown that, under certain assumptions, such codes exist over large enough finite fields, as well as over small fields in a special case. A similar problem, known as the weakly secure data exchange problem, was studied in [YS11],[YSZ14]. The problem deals with a set of users, each with a subset of messages, who are interested in broadcasting their information securely when an eavesdropper is present. In particular, the authors of [YSZ14] conjecture the existence of secure codes based on Reed-Solomon codes and present a randomized algorithm to produce them. In the context of multi-source multicast network coding, we cite the relevant papers [Dik+10; Hal+14; HHD14].

4.2 Problem Setup

Consider a bipartite graph $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ with $s = |\mathcal{M}| \leq |\mathcal{V}| = n$. The set $\mathcal{E}$ is the set of edges of the graph, with $(m_i, c_j) \in \mathcal{E}$ if and only if $m_i \in \mathcal{M}$ is connected to $c_j \in \mathcal{V}$. This graph defines a code where the vertices $\mathcal{M}$ correspond to message symbols and the vertices $\mathcal{V}$ correspond to codeword symbols. A bipartite graph with $s = 3$ and $n = 7$ is depicted in figure 4.1. Thus, if each m_i and c_j are assigned values in the finite field $\mathbb{F}_q$ with q elements, then our messages are the vectors $\mathbf{m} = (m_1, \ldots, m_s) \in \mathbb{F}_q^s$ and our codewords are the vectors $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$. Each codeword symbol c_j will be a function of the message symbols to which it is connected, as we will now formalize.

Henceforth, $[\mathbf{c}]_{\mathcal{I}}$ is the subvector of $\mathbf{c}$ with elements indexed by $\mathcal{I} \subseteq \{1, ..., n\}$, and $[\mathbf{A}]_{i,j}$ is the $(i, j)^{\text{th}}$ element of a matrix $\mathbf{A}$. Let $\mathcal{N}(c_j)$ denote the neighborhood of $c_j \in \mathcal{V}$, i.e. $\mathcal{N}(c_j) = \{m_i \in \mathcal{M} : (m_i, c_j) \in \mathcal{E}\}$. Similarly, define $\mathcal{N}(m_i) = \{c_j : (m_i, c_j) \in \mathcal{E}\}$. We will also consider neighborhoods of subsets of the vertex sets, i.e. for $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{N}(\mathcal{V}') = \cup_{c_j \in \mathcal{V}'} \mathcal{N}(c_j)$. The neighborhood of a subset of $\mathcal{M}$ is defined in a similar manner. Let m_i take values in $\mathbb{F}_q$ and associate with each $c_j \in \mathcal{V}$ a function $f_j : \mathbb{F}_q^s \longrightarrow \mathbb{F}_q$. We restrict each f_j to be a function of $\mathcal{N}(c_j)$ only. Now consider the set $\mathcal{C} = \{(c_1, \ldots, c_n) : c_j = f_j(\mathbf{m}), \mathbf{m} \in \mathbb{F}_q^s\}$. The set $\mathcal{C}$ is an error-correcting code of length n and size at most q^s. We will denote the minimum distance of $\mathcal{C}$ as $d(\mathcal{C})$. If we restrict f_j to be *linear*, then we obtain a linear code with dimension at most s.

The structure of the code's generator matrix can be deduced from the graph G. Let $\mathbf{g}_j \in \mathbb{F}_q^{s\times 1}$ be a column vector such that the i^{th} entry is zero if $m_i \notin \mathcal{N}(c_j)$. Defining $f_j(\mathcal{N}(c_j)) = \mathbf{m}\mathbf{g}_j$ yields a linear function in which c_j is a function of $\mathcal{N}(c_j)$ only, as required. A concatenation of the vectors $\mathbf{g}_j$ forms the following matrix:

$$\mathbf{G} = \begin{bmatrix} | & & | \\ \mathbf{g}_1 & \cdots & \mathbf{g}_n \\ | & & | \end{bmatrix}, \tag{4.1}$$

where $\mathbf{G} \in \mathbb{F}_q^{s \times n}$ is the generator matrix of the code $\mathcal{C}$.

We associate with the bipartite graph $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ an adjacency matrix $\mathbf{A} \in \{0,1\}^{s \times n}$, where $[\mathbf{A}]_{i,j} = 1$ if and only if $(m_i, c_j) \in \mathcal{E}$. For the example in figure 4.1, this matrix is equal to

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.2}$$

A *valid* generator matrix $\mathbf{G}$ (in generic form) is built from $\mathbf{A}$ by replacing non-zero entries with indeterminates. The choice of indeterminates (from a suitably-sized finite field $\mathbb{F}_q$) determines the dimension of the code and its minimum distance. For general linear codes, the Singleton bound (on minimum distance) is tight over large alphabets. In the presence of encoding constraints, the Singleton bound can be rather loose. In the next section, we derive an upper bound on the minimum distance of any code (linear or non-linear) associated with a bipartite graph. This bound is reminiscent of the cut-set bounds of Dikaliotis et al. in [Dik+10].

Subcodes of Reed-Solomon Codes

We will utilize Reed–Solomon codes as described in Section 1.2. Contrary to the some of the results in Chapter 2, we will not restrict the coordinates of $\mathsf{RS}[n,k]$ to be powers of a primitive element in $\mathbb{F}_q$. This process is very similar to that presented in Chapter 2. For the reader's convenience, we restate it here. First, let $\mathbb{F}_q$ be a finite field with cardinality $q \geq n$. Associate to each $c_j \in \mathcal{V}$ a distinct element $\alpha_j \in \mathbb{F}_q$. Consider the i^{th} row of the adjacency matrix $\mathbf{A}$ of G, and let $t_i(x) = \prod_{j:[\mathbf{A}]_{i,j}=0}(x - \alpha_j)$. For example, $t_3(x) = (x - \alpha_1)(x - \alpha_2)$ corresponds to the the third row of $\mathbf{A}$ in (4.2). Choose k such that $k > \mathsf{deg}\,(t_i(x))$, $\forall i$. If $\mathbf{t}_i \in \mathbb{F}_q^k$ is the (row) vector of coefficients of $t_i(x)$ and $\mathbf{G}_{\text{RS}}$ is the generator matrix of a Reed-Solomon code with defining set $\{\alpha_1, \ldots, \alpha_n\}$ and dimension k, then $\mathbf{t}_i\mathbf{G}_{\text{RS}} = (t_i(\alpha_1), \ldots, t_i(\alpha_n))$ is a vector that is valid for the i^{th} row of $\mathbf{G}$, i.e. if $[\mathbf{A}]_{i,j} = 0$ then $[\mathbf{t}_i\mathbf{G}_{\text{RS}}]_j = 0$. A horizontal stacking of the vectors $\mathbf{t}_i$ results in a

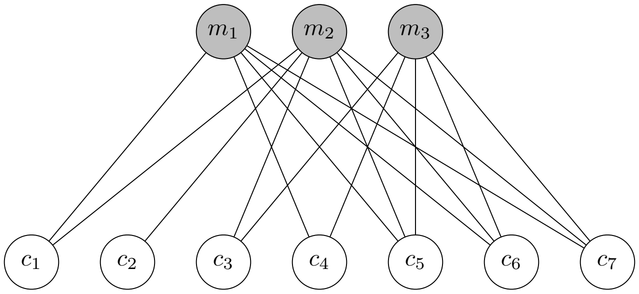


Figure 4.1: A bipartite graph representing an error-correcting code subject to encoding constraints. The code comprises seven code symbols that encode three message symbols.

transformation matrix $\mathbf{T}$ that will produce a valid generator matrix $\mathbf{G}$ from $\mathbf{G}_{\text{RS}}$:

$$\mathbf{G} = \mathbf{T}\mathbf{G}_{\text{RS}} = \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_s \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_n \\ \vdots & \ddots & \vdots \\ \alpha_1^{(k-1)} & \cdots & \alpha_n^{(k-1)} \end{bmatrix}. \tag{4.3}$$

The rank of $\mathbf{G}$ will be equal to the rank of $\mathbf{T}$, and the resulting code $\mathcal{C}$ will have a minimum distance $d(\mathcal{C})$ that is determined by $\mathcal{C}_{\text{RS}}$. Indeed, $d(\mathcal{C}) \geq d(\mathcal{C}_{\text{RS}})$.

4.3 Minimum Distance

In this section, an upper bound on the minimum distance of a code defined by a bipartite graph $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ is derived. The bound closely resembles the cut-set bounds of [Dik+10]. In most cases, this bound is tighter than the Singleton bound for a code of length n and dimension s. For each $\mathcal{M}' \subseteq \mathcal{M}$ define $n_{\mathcal{M}'} := |\mathcal{N}(\mathcal{M}')|$. This is the number of code symbols c_j in $\mathcal{V}$ that are a function of the information symbols $\mathcal{M}'$. The following proposition characterizes the minimum distance of any code defined by G.

Proposition 4.1. *Fix a field $\mathbb{F}_q$. For any code $\mathcal{C}$ with $|\mathcal{C}| = q^s$ defined by a fixed graph $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$, the minimum distance $d(\mathcal{C})$ obeys*

$$d(\mathcal{C}) \leq n_{\mathcal{M}'} - |\mathcal{M}'| + 1, \quad \forall \mathcal{M}' \subseteq \mathcal{M}. \tag{4.4}$$

Proof. Working toward a contradiction, suppose $d(\mathcal{C}) > n_{\mathcal{I}} - |\mathcal{I}| + 1$ for some $\mathcal{I} \subseteq \mathcal{M}$. Let $\mathcal{C}'$ be the encoding of all message vectors $\mathbf{m}$ where $[\mathbf{m}]_{\mathcal{I}^{\mathsf{c}}} \in \mathbb{F}_q^{|\mathcal{I}^{\mathsf{c}}|}$ has some arbitrary but fixed value. Note that $[\mathbf{c}]_{\mathcal{N}(\mathcal{I})^{\mathsf{c}}}$ is the same for all $\mathbf{c} \in \mathcal{C}'$, since the symbols $\mathcal{N}(\mathcal{I})^{\mathsf{c}}$ are a function of $\mathcal{I}^{\mathsf{c}}$ only. Since $|\mathcal{I}| > n_{\mathcal{I}} - d(\mathcal{C}) + 1$,

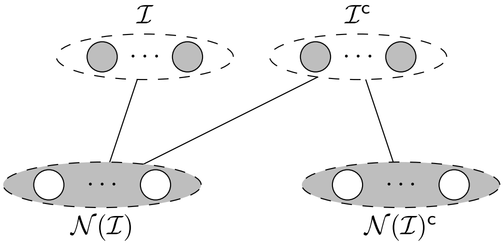

$\mathcal{I}$
$\mathcal{I}^{\mathrm{c}}$
$\mathcal{N}(\mathcal{I})$
$\mathcal{N}(\mathcal{I})^{\mathrm{c}}$

Figure 4.2: Partitions of $\mathcal{M}$ and of $\mathcal{V}$ used in the proof of Proposition 4.1. The set $\mathcal{N}(\mathcal{I})$ is a function of both $\mathcal{I}$ and $\mathcal{I}^{\mathrm{c}}$, while the set $\mathcal{N}(\mathcal{I})^{\mathrm{c}}$ is a function of $\mathcal{I}^{\mathrm{c}}$ only.

then by the pigeonhole principle there exist $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}'$ such that, without loss of generality, the first $n_{\mathcal{I}} - d(\mathcal{C}) + 1$ symbols of $[\mathbf{c}_1]_{\mathcal{N}(\mathcal{I})}$ and $[\mathbf{c}_2]_{\mathcal{N}(\mathcal{I})}$ are identical. Furthermore, $[\mathbf{c}_1]_{\mathcal{N}(\mathcal{I})^{\mathrm{c}}} = [\mathbf{c}_2]_{\mathcal{N}(\mathcal{I})^{\mathrm{c}}}$. Finally, since $\mathcal{N}(\mathcal{I})$ and $\mathcal{N}(\mathcal{I})^{\mathrm{c}}$ partition $\mathcal{V}$, we obtain $d_{\mathrm{H}}(\mathbf{c}_1, \mathbf{c}_2) \leq n - (n_{\mathcal{I}} - d(\mathcal{C}) + 1 + (n - n_{\mathcal{I}})) = d(\mathcal{C}) - 1$, a contradiction. Figure 4.2 illustrates the relation between $\mathcal{I}$ and the corresponding partition of $\mathcal{V}$. □

As a direct corollary, we obtain the following upper bound on $d(\mathcal{C})$:

Corollary 4.1.

$$d(\mathcal{C}) \leq \min_{\mathcal{M}' \subseteq \mathcal{M}} \{n_{\mathcal{M}'} - |\mathcal{M}'|\} + 1. \tag{4.5}$$

Our next task is to provide constructions of codes that achieve this bound.

4.4 Systematic Construction

In this section, we provide a code construction that achieves the minimum distance bound stated in corollary 4.1. We appeal to Hall's Theorem, a well-known result in graph theory that establishes a necessary and sufficient condition for finding a matching in a bipartite graph. Some terminology needed from graph theory is defined in the following subsection.

Graph Theory Preliminaries

Let $G = (\mathcal{S}, \mathcal{T}, \mathcal{E})$ be a bipartite graph. A *matching* is a subset $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ such that no two edges in $\tilde{\mathcal{E}}$ share a common vertex. A vertex is said to be *covered* by $\tilde{\mathcal{E}}$ if it is incident to an edge in $\tilde{\mathcal{E}}$. An $\mathcal{S}$-*covering* matching is one by which each vertex in $\mathcal{S}$ is covered. We will abuse terminology and say that an edge $e \in \tilde{\mathcal{E}}$ is *unmatched* if $e \notin \tilde{\mathcal{E}}$. We can now state Hall's Theorem.

Theorem 4.1. *Let $G = (\mathcal{S}, \mathcal{T}, \mathcal{E})$ be a bipartite graph. There exists an $\mathcal{S}$-covering matching if and only if $|\mathcal{S}'| \leq |\mathcal{N}(\mathcal{S}')|$ for all $\mathcal{S}' \subseteq \mathcal{S}$.*

For a proof of the theorem, see e.g. [VW11, p.53].

Set $d_{\mathsf{min}} = \min_{\mathcal{M}' \subseteq \mathcal{M}}\{n_{\mathcal{M}'} - |\mathcal{M}'|\} + 1$. In order to construct a generator matrix $\mathbf{G} \in \mathbb{F}_q^{s \times n}$ for a code $\mathcal{C}$ with minimum distance d_{min}, we will use an $[n, n - d_{\mathsf{min}} + 1]$ Reed-Solomon code with generator matrix $\mathbf{G}_{\mathrm{RS}}$. We will then extract $\mathcal{C}$ as a subcode using an appropriately built transformation matrix $\mathbf{T}$ to form $\mathbf{G} = \mathbf{T}\mathbf{G}_{\mathrm{RS}}$ such that $\mathbf{G}$ is in systematic form, which implies that the dimension of $\mathcal{C}$ is s. Finally, we will show that $d(\mathcal{C}) = d_{\mathsf{min}}$.

Our construction is as follows: consider a graph $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ defining $\mathcal{C}$, and define the set $\mathcal{A} = \{c_j : \mathcal{N}(c_j) = \mathcal{M}\}$, i.e. $\mathcal{A}$ is the set of code symbols that are a function of *every* message symbol. Note that $\mathcal{A} \subseteq \mathcal{N}(\mathcal{M}')$ for every $\mathcal{M}' \subseteq \mathcal{M}$. Therefore, if $a = |\mathcal{A}|$ then the size of the neighborhood of $\mathcal{N}(\mathcal{M}')$ can be expressed as $n_{\mathcal{M}'} = r_{\mathcal{M}'} + a$, where $r_{\mathcal{M}'}$ is the cardinality of the set $\mathcal{R}(\mathcal{M}') = \mathcal{N}(\mathcal{M}') \setminus \mathcal{A}$.

Theorem 4.2. *Let $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$. Set $d_{\mathsf{min}} = \min_{\mathcal{M}' \subseteq \mathcal{M}}\{n_{\mathcal{M}'} - |\mathcal{M}'|\} + 1$ and $k_{\mathsf{min}} = n - d_{\mathsf{min}} + 1$. A linear code $\mathcal{C}$ with parameters $[n, s, d_{\mathsf{min}}]$ valid for G can be constructed with a systematic-form generator matrix provided that $k_{\mathsf{min}} \geq r_{\mathcal{M}}$.*

Proof. First, we establish a bound on a, the number of code symbols that are (potentially) a function of every message symbol. Note that since $n = n_{\mathcal{M}} = r_{\mathcal{M}} + a$ and $k_{\mathsf{min}} \geq r_{\mathcal{M}}$, then we have $a \geq d_{\mathsf{min}} - 1$. Fix an arbitrary subset $\mathcal{A}^* \subseteq \mathcal{A}$ of size $a^* = a - (d_{\mathsf{min}} - 1)$, which is guaranteed to exist by virtue of the bound on a, and let $\mathcal{B} = \mathcal{A} \setminus \mathcal{A}^*$. Now, we focus on a particular subgraph of G defined by $G^* = (\mathcal{M}, \mathcal{V}^*, \mathcal{E}^*)$ where $\mathcal{V}^* = \mathcal{V} \setminus \mathcal{B}$, and $\mathcal{E}^* = \{(m_i, c_j) \in \mathcal{E} : c_j \in \mathcal{V}^*\}$ is the edge set corresponding to this subgraph. Since $n_{\mathcal{M}'} = r_{\mathcal{M}'} + a$, then from the definition of d_{min} we have

$$|\mathcal{M}'| \leq r_{\mathcal{M}'} + a - (d_{\mathsf{min}} - 1), \quad \forall \mathcal{M}' \subseteq \mathcal{M}. \tag{4.6}$$

The neighborhood of every subset $\mathcal{M}'$ when restricted to $\mathcal{V}^*$ is exactly $\mathcal{N}^*(\mathcal{M}') = \mathcal{R}(\mathcal{M}') \cup \mathcal{A}^*$, with cardinality $n^*_{\mathcal{M}'} = r_{\mathcal{M}'} + a^*$. The bounds (4.6) can now be expressed in a way suitable for the condition of Hall's theorem:

$$|\mathcal{M}'| \leq n^*_{\mathcal{M}'}, \quad \forall \mathcal{M}' \subseteq \mathcal{M}. \tag{4.7}$$

An $\mathcal{M}$-covering matching in G^* can be found by letting $\mathcal{S} = \mathcal{M}$ and $\mathcal{T} = \mathcal{V}^*$ in theorem 4.1. Let $\tilde{\mathcal{E}} = \{(m_i, c_{j(i)})\}_{i=1}^{s} \subseteq \mathcal{E}^*$ be such a matching, and $\tilde{\mathcal{V}}$ the subset of $\mathcal{V}^*$ that is covered by $\tilde{\mathcal{E}}$. Let $\mathbf{A}_{\tilde{\mathcal{E}}}$ be the adjacency matrix of G when the edge set $\{(m_i, c_j) \in \mathcal{E} : c_j \in \tilde{\mathcal{V}}, j \neq j(i)\}$ is removed. The number of zeros in any row of $\mathbf{A}_{\tilde{\mathcal{E}}}$ is at most $n - d_{\mathsf{min}}$. To see this, note that the edges in $\mathcal{E}$ incident to $\mathcal{B}$ are not removed by the matching, and every $m_i \in \mathcal{M}$ is connected to at least one vertex in $\mathcal{V}^*$. Next, we build a valid $\mathbf{G}$ for G using $\mathbf{A}_{\tilde{\mathcal{E}}}$, utilizing the method described in Section 5.3. Fix a $[n, n - d_{\mathsf{min}} + 1]$ Reed-Solomon code with generator matrix $\mathbf{G}_{\text{RS}}$ and defining set $\{\alpha_1, \ldots, \alpha_n\}$. The i^{th} transformation polynomial is $t_i(x) = \prod_{j:[\mathbf{A}_{\tilde{\mathcal{E}}}]_{i,j}=0}(x - \alpha_i)$. Since the number of zeros in any row of $\mathbf{A}_{\tilde{\mathcal{E}}}$ is at most $n - d_{\mathsf{min}}$, we have $\mathsf{deg}\,(t_i(x)) \leq n - d_{\mathsf{min}} = k - 1$ for all i. We use the $t_i(x)$, after normalizing by $t_i(\alpha_{j(i)})$, to construct a transformation matrix $\mathbf{T}$ and then $\mathbf{G} = \mathbf{T}\mathbf{G}_{\text{RS}}$ is valid for G. Note that $\mathbf{G}$ is in systematic form due the fact that the columns of $\mathbf{A}_{\tilde{\mathcal{E}}}$ indexed by $\{j(i)\}_{i=1}^{s}$ form a permutation of the identity matrix of size s. Lastly, $d(\mathcal{C}) = d_{\mathsf{min}}$ since $d(\mathcal{C}) \leq d_{\mathsf{min}}$ by (4.5), and $d(\mathcal{C}) \geq d_{\mathsf{min}}$ since $\mathcal{C}$ is a subcode of a code with minimum distance d_{min}. □

The inequality $a \geq d_{\mathsf{min}} - 1$ provides a sufficient condition for a given graph G to accommodate a system generator matrix without sacrificing the minimum distance. A natural question to ask is whether this condition is necessary. It turns out that it is not: there exist bipartite graphs for which a systematic construction achieves the upper bound on minimum distance (4.5) but $a \leq d_{\mathsf{min}}$. A simple example is one whose adjacency matrix is given below:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

It can be checked that $d_{\mathsf{min}} = 3$ whereas $a = 1$. Clearly, a systematic construction exists by using a subcode of $\mathsf{RS}[6, 4]$.

In the next section, we consider the scenario where the code designed for a particular constraint graph is required to be systematic. We will show that this requirement can potentially decrease the largest achievable minimum distance. We will characterize this quantity and call it the *systematic minimum distance* of the graph.

4.5 Minimum Distance for Systematic Linear Codes

We will restrict our attention to the case where a code valid for G is linear, so that each $c_j \in \mathcal{V}$ is a linear function of the message symbols $m_i \in \mathcal{N}(c_j)$. We seek to answer the following: What is the greatest minimum distance attainable by a *systematic* linear code valid for G?

Any systematic code must correspond to a matching $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ which identifies each message symbol $m_i \in \mathcal{M}$ with a unique codeword symbol $c_{j(i)} \in \mathcal{V}$, where $j(i) \in \{1, \dots, n\}$. Explicitly, $\tilde{\mathcal{E}}$ consists of s edges of the form $(m_i, c_{j(i)})$ for $i = 1, \dots, s$ such that $c_{j(i_1)} \neq c_{j(i_2)}$ for $i_1 \neq i_2$. As before, $\tilde{\mathcal{V}}$ is the subset of vertices in $\mathcal{V}$ which are involved in the matching: $\tilde{\mathcal{V}} = \{c_{j(i)}\}_{i=1}^{s}$. Our code becomes systematic by setting $c_{j(i)} = m_i$ for $i = 1, \dots, s$, and choosing each remaining codeword symbol $c_j \notin \tilde{\mathcal{V}}$ to be some linear function of its neighboring message symbols $m_i \in \mathcal{N}(c_j)$.

Definition 4.1. *For $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$, let $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ be an $\mathcal{M}$-covering matching so that $\tilde{\mathcal{E}} = \{(m_i, c_{j(i)})\}_{i=1}^{s}$. Let $\tilde{\mathcal{V}} = \{c_{j(i)}\}_{i=1}^{s}$ be the vertices in $\mathcal{V}$ which are covered by $\tilde{\mathcal{E}}$. Define the matched adjacency matrix $\mathbf{A}_{\tilde{\mathcal{E}}} \in \{0,1\}^{s \times n}$ so that $[\mathbf{A}_{\tilde{\mathcal{E}}}]_{i,j} = 1$ if and only if either $(m_i, c_j) \in \tilde{\mathcal{E}}$, or $c_j \notin \tilde{\mathcal{V}}$ and $(m_i, c_j) \in \mathcal{E}$. In other words, $\mathbf{A}_{\tilde{\mathcal{E}}}$ is the adjacency matrix of the bipartite graph formed by starting with G and deleting the edges $\{(m_i, c_j) \in \mathcal{E} \ : \ c_j \in \tilde{\mathcal{V}} \text{ and } j \neq j(i)\}$.*

Definition 4.2. *Let $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ be a matching for $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ which covers $\mathcal{M}$. Let $z_{\tilde{\mathcal{E}}}$ be the maximum number of zeros in any row of the corresponding matched adjacency matrix $\mathbf{A}_{\tilde{\mathcal{E}}}$, and define $k_{\tilde{\mathcal{E}}} := z_{\tilde{\mathcal{E}}} + 1$. Furthermore, define $k_{\mathsf{sys}} = \mathsf{min}_{\tilde{\mathcal{E}}} k_{\tilde{\mathcal{E}}}$ where $\tilde{\mathcal{E}}$ ranges over all matchings for G which cover $\mathcal{M}$, and $d_{\mathsf{sys}} = n - k_{\mathsf{sys}} + 1$.*

Lemma 4.1. *For a given bipartite graph $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ which merits a matching that covers $\mathcal{M}$, we have*

$$s \leq k_{\mathsf{min}} \leq k_{\mathsf{sys}} \leq n \tag{4.8}$$

and

$$d_{\mathsf{sys}} \leq d_{\mathsf{min}}. \tag{4.9}$$

Proof. Let $\mathbf{A}$ be the adjacency matrix of G.

For any subset $\mathcal{M}' \subseteq \mathcal{M}$ we have $d_{\mathsf{min}} \leq n_{\mathcal{M}'} - |\mathcal{M}'| + 1$, and likewise $k_{\mathsf{min}} = n - d_{\mathsf{min}} + 1 \geq |\mathcal{M}'| + (n - n_{\mathcal{M}'})$. Taking $\mathcal{M}' = \mathcal{M}$ (and noting that in our framework, every $c_j \in \mathcal{V}$ is connected to at least one vertex in $\mathcal{M}$, hence $n_{\mathcal{M}} = n$) we obtain $k_{\mathsf{min}} \geq s$.

Now choose a set $\mathcal{M}'$ for which the above relation holds with equality, that is, $k_{\mathsf{min}} = |\mathcal{M}'| + (n - n_{\mathcal{M}'})$. Since $\mathcal{N}(\mathcal{M}')$ is simply the union of the support sets of the rows of $\mathbf{A}$ corresponding to $\mathcal{M}'$, the quantity $n - n_{\mathcal{M}'} = n - |\mathcal{N}(\mathcal{M}')|$ is *at most* equal to the largest number of zeros in any row of $\mathbf{A}$. On the other hand, for any matching $\tilde{\mathcal{E}}$ which covers $\mathcal{M}$, the zeros of $\mathbf{A}$ are a subset of the zeros of $\mathbf{A}_{\tilde{\mathcal{E}}}$, so from its definition we must have that $k_{\tilde{\mathcal{E}}}$ is greater than or equal to k_{min}, hence

$k_{\mathsf{sys}} \geq k_{\mathsf{min}}$. It follows directly that $d_{\mathsf{sys}} \leq d_{\mathsf{min}}$. Finally, it is clear from definition that for any $\mathcal{M}$-covering matching $\tilde{\mathcal{E}}$ we must have that $k_{\tilde{\mathcal{E}}}$ is less than the length of the adjacency matrix $\mathbf{A}$, which is n, hence $k_{\mathsf{sys}} \leq n$. □

Corollary 4.2. *Let $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ be a bipartite graph which merits a systematic linear code. The largest minimum distance obtainable by a systematic linear code is d_{sys}.*

Proof. Let $\mathcal{C}$ be a systematic linear code which is valid for G. Then $\mathcal{C}$ must have a codeword containing at least $k_{\mathsf{sys}} - 1$ zeros, i.e. a codeword of Hamming weight at most $n - k_{\mathsf{sys}} + 1 = d_{\mathsf{sys}}$. Since the code is linear, this Hamming weight is an upper bound for its minimum distance, so $d(\mathcal{C}) \leq d_{\mathsf{sys}}$.

It remains to see that there are systematic linear codes which are valid for G and achieve a minimum distance of d_{sys}. Let $\tilde{\mathcal{E}}$ be an $\mathcal{M}$-covering matching for G such that $k_{\tilde{\mathcal{E}}} = k_{\mathsf{sys}}$. Then for any $k \geq k_{\mathsf{sys}}$, we claim that an $[n, k]$ Reed-Solomon code contains a systematic linear subcode that is valid for G. Indeed, choose a set of n distinct elements $\{\alpha_i\}_{i=1}^{n} \subseteq \mathbb{F}_q$ as the defining set of our Reed-Solomon code. Then, to form our subcode's generator matrix $\mathbf{G}$, note that (as mentioned before) $\mathbf{G}$ must have zero entries in the same positions as the zero entries of $\mathbf{A}_{\tilde{\mathcal{E}}}$, and indeterminate elements in the remaining positions. There are at most $k_{\mathsf{sys}} - 1$ zeros in any row of $\mathbf{A}_{\tilde{\mathcal{E}}}$ (and at *least* $s - 1$ zeros in each row, since there must be s columns that have nonzero entries in exactly one row). For each row $i \in \{1, \ldots, s\}$ of $\mathbf{A}_{\tilde{\mathcal{E}}}$, let $\mathcal{I}_i \subseteq \{1, \ldots, n\}$ be the set of column indices j such that $[\mathbf{A}_{\tilde{\mathcal{E}}}]_{i,j} = 0$. Then form the polynomial $t_i(x) = \prod_{j \in \mathcal{I}_i}(x - \alpha_j)$ and normalize by $t_i(\alpha_{j(i)})$, which accordingly has degree at most k_{sys} (and at least $s - 1$). We now set the i^{th} row of $\mathbf{G}$ to be $(t_i(\alpha_1), \ldots, t_i(\alpha_n))$, and we see that by construction this row has zeros precisely at the indices $j \in \mathcal{I}_i$ as desired.

The rows of $\mathbf{G}$ generate a code with minimum distance at least that of the original Reed-Solomon code, which is $n - k + 1$. Furthermore, by setting $k = k_{\mathsf{sys}}$ for our Reed-Solomon code, we see this new code $\mathcal{C}$ has minimum distance at least $n - k_{\mathsf{sys}} + 1 = d_{\mathsf{sys}}$. Since by our previous argument, $d(\mathcal{C}) \leq d_{\mathsf{sys}}$, the minimum distance of $\mathcal{C}$ must achieve d_{sys} with equality. □

Compared to the result of Theorem 4.2, the systematic minimum distance d_{sys} is always achievable. In the next section, we present an example in which the systematic minimum distance is strictly less than the bound provided by (4.5).

4.6 Example

In this section, we construct a systematic linear code that is valid for the graph in Figure 4.1. The bound of 4.5 asserts that $d(\mathcal{C}) \leq 5$ for any $\mathcal{C}$ valid for G. However, corollary 4.2 shows that $d(\mathcal{C}_{\mathsf{sys}}) \leq 4$ for any valid systematic linear code $\mathcal{C}_{\mathsf{sys}}$. A matching achieving this bound is given by the edges $\tilde{\mathcal{E}} = \{(m_1, v_1), (m_2, v_2), (m_3, v_3)\}$ and so the edges removed from the graph are $\{(m_2, v_1), (m_2, v_3)\}$. The new adjacency matrix $\mathbf{A}_{\tilde{\mathcal{E}}}$ is given by

$$\mathbf{A}_{\tilde{\mathcal{E}}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ \mathbf{0} & 1 & \mathbf{0} & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \tag{4.10}$$

where boldface zeros refer to those edges removed from G because of the matching $\tilde{\mathcal{E}}$.

A generator matrix which is valid for $\mathbf{A}_{\tilde{\mathcal{E}}}$ can be constructed from that of a $[7, 4]$ Reed-Solomon code over $\mathbb{F}_7$ with defining set $\{0, 1, \alpha, \ldots, \alpha^5\}$ where α is a primitive element in $\mathbb{F}_7$, using the method described in 5.3.

The polynomials corresponding to the transformation matrix are given by

$$t_1(x) = \alpha^5(x-1)(x-\alpha), \tag{4.11}$$

$$t_2(x) = \alpha^4 x(x-\alpha)(x-\alpha^2), \tag{4.12}$$

$$t_3(x) = \alpha^3 x(x-1). \tag{4.13}$$

Finally, the systematic generator matrix for $\mathcal{C}_{\mathsf{sys}}$ is

$$\mathbf{G}_{\mathsf{sys}} = \begin{bmatrix} 1 & 0 & 0 & \alpha^2 & \alpha^5 & 1 & \alpha^5 \\ 0 & 1 & 0 & 0 & 1 & \alpha^4 & 1 \\ 0 & 0 & 1 & \alpha^5 & \alpha^5 & \alpha^2 & 1 \end{bmatrix}. \tag{4.14}$$

4.7 Discussion

We comment in this section on two aspects related to the results presented. The first one addresses the issue of constructing good codes as subcodes of general MDS codes while the second relates to computational issues.

Achievability Using MDS Codes

We have utilized Reed-Solomon codes to construct systematic linear codes, valid for a particular $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$, that attain the highest possible distance. It is worth mentioning that this choice is not necessary and in fact, the Reed-Solomon code utilized can be replaced with any linear MDS code with the same parameters.

Lemma 4.2. *Fix an arbitrary $[n, k]$ linear MDS code $\mathcal{C}$. For any $\mathcal{I} \subseteq [n]$ where $|\mathcal{I}| \leq k - 1$, there exists $\mathbf{c} \in \mathcal{C}$ such that $[\mathbf{c}]_{\mathcal{I}} = \mathbf{0}$.*

Proof. Let $\mathbf{G} = [\mathbf{g}_i]_{i=1}^n$ be the generator matrix of $\mathcal{C}$ and let $\mathbf{G}_{\mathcal{I}} = [\mathbf{g}_i]_{i \in \mathcal{I}}$. Since $|\mathcal{I}| \leq k - 1$, $\mathbf{G}_{\mathcal{I}}$ has full column rank and so it has a non-trivial left nullspace of dimension $k - |\mathcal{I}|$. If $\mathbf{h}$ is any vector in that nullspace then $\mathbf{c} = \mathbf{h}\mathbf{G}$ is such that $[\mathbf{c}]_{\mathcal{I}} = \mathbf{0}$. □

Therefore, to produce a valid linear code $\mathcal{C}$ for $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$ with $d(\mathcal{C}) = d^*$, where $d^* \leq n_{m_i}$ for all $m_i \in \mathcal{M}$, we fix an arbitrary $[n, n - d^* + 1]$ MDS code and then select vectors $\mathbf{h}_1, \ldots, \mathbf{h}_s$ such that $\mathbf{h}_i$ is in the left nullspace of $\mathbf{G}_{\mathcal{I}_i}$, where $\mathcal{I}_i = \{j : \mathbf{A}_{i,j} = 0\}$. Note that the specific selection of the $\mathbf{h}_i$ determines the dimension of $\mathcal{C}$. For a systematic construction, in which the dimension of the code is guaranteed to be s, some extra care has to be taken when choosing the $\mathbf{h}_i$. We must choose each $\mathbf{h}_i$ such that its not in the nullspace of $\mathbf{g}_{j(i)}$, which the column corresponding to the systematic coordinate $c_{j(i)}$.

Computational Aspects

It is worth addressing the computational complexities of computing d_{min} and d_{sys}. Indeed, Proposition 1 in [Dau+15b] provides a polynomial-time algorithm to compute d_{min}. The authors show that for a particular $G = (\mathcal{M}, \mathcal{V}, \mathcal{E})$, the largest $d(\mathcal{C})$ satisfying

$$d(\mathcal{C}) \leq |\mathcal{N}(\mathcal{M}')| - |\mathcal{M}'| + 1, \quad \forall \mathcal{M}' \subseteq \mathcal{M} \tag{4.15}$$

can be found in polynomial time. Their strategy relies on verifying a min-cut condition for a particular graph constructed from G. They show that this graph is polynomially sized in G and thus the min-cut condition can be verified in polynomial time using the Ford-Fulkerson Algorithm [FF56].

In comparison, the upper bound on the systematic minimum distance of a graph $G(\mathcal{M}, \mathcal{V}, \mathcal{E})$ given in Section 4.5 assumes the enumeration of all $\mathcal{M}$-covering matchings, which is exponentially large in the size of the graph. An algorithm that computes d_{sys} and provides the corresponding the matching without resorting to such enumeration would be of great benefit. Indeed, such an algorithm, if it runs in polynomial time, would provide optimal systematic codes in an efficient manner. In addition, such algorithm can also be used to compute d_{min} for those graphs in which the technical condition of Theorem 4.2 holds, since the construction that achieves d_{min} in that case is a systematic one.

Chapter 5

BALANCED REED–SOLOMON AND TAMO–BARG CODES

5.1 Introduction

The ubiquity of Reed–Solomon (RS) codes in the industry is in part due to their optimality on the rate-distance curve, i.e. they can correct a fixed number of errors with optimal redundancy. Companies such as Facebook are known to utilize RS codes in their production clusters [Mur+14]. They have also been incorporated in modern distributed file systems such as HDFS by Apache Hadoop [Apaa; Clo].

The problem with RS codes (MDS codes in general) is that the complexity of decoding is independent of the number of errors/erasures present. To tackle this problem, locally recoverable codes (LRCs) were recently developed [Gop+12; Kam+14; Sil+13; PD12; HCL07; TB14; TB15; Sat+13]. The central problem these codes aim to solve is minimizing the number of code symbols needed to recover a few - with respect to the number of data symbols stored - erased ones. This number is known as the locality of the code. Naturally, the redundancy required to protect against a fixed number of errors/erasures is larger than that of RS codes.

Having considered optimal redundancy, fast decoding and low locality as metrics for efficient error-correcting codes, the encoding speed is an important metric that is worth pondering over. This is especially important when recognizing the fact that a lot of the data currently being gathered is stored but barely read again [Mur+14]. Modern data archiving services[Ama; Goo; Fac] call for schemes that are robust, low-cost and low-powered. This viewpoint allows us to optimize the encoding process at the expense of, perhaps, other benefits. Let us illustrate with an example.

Consider a group of n storage nodes that jointly encode a message vector $\mathbf{m} \in \mathbb{F}_q^k$ using an error-correcting code $\mathcal{C}$, with generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$. In particular, every storage node S_i encodes the message symbols using $\mathbf{g}_i$, the i^{th} column of $\mathbf{G}$, to produce a code symbol c_i. The time required to compute c_i is a function of the weight - the number of nonzero entries - of $\mathbf{g}_i$. If $\mathcal{C}$ is chosen as an MDS code, then it can be argued the *average* encoding time required when using a systematic $\mathbf{G}$ is minimal. If we consider the *maximal* encoding time, where the maximization is over S_i, then systematic encoding is as slow as using a generator matrix that has no zeros.

What about a solution that lives in between these two extremes? In this work, we consider *balanced* generator matrices in which the rows of are of fixed, but tunable, weight and the columns have essentially the same weight. The benefit of a balanced generator matrix $\mathbf{G}$ is that every code symbol c_i is computed in roughly the same amount of time. This ensures that the computational load is balanced across the storage system, i.e. there are no storage nodes that behave as bottlenecks. Furthermore, if we fix each row of $\mathbf{G}$ to have weight s, then updating a single message symbol impacts exactly s storage nodes. When $s = d$, where d is the minimum distance of the code, we obtain a *balanced* and *sparsest* generator matrix. In general, for $\mathbf{G}$ to be balanced, the weight of each column has to be either $\left\lfloor \frac{k}{n} s \right\rfloor$ or $\left\lceil \frac{k}{n} s \right\rceil$. This is seen from the fact the total number of nonzeros in $\mathbf{G}$ is ks, which is to be distributed equally among the n columns.

The problem of constructing balanced and sparsest generator matrices for error-correcting codes was first considered in [Dau+13], where it was shown, through a probabilistic argument, that such matrices exist for maximum distance separable (MDS) codes. While probabilistic constructions are easily to implement, decoding random codes is known to be hard. In [HLH16a; HLH16b], it was shown that any cyclic Reed–Solomon (RS) code possesses such a sparsest and balanced generator matrix. The codes are optimal in terms of the finite field size required and can be decoded using standard RS decoding algorithms. With the growing interest in LRCs, balanced and sparse Tamo–Barg (TB) codes were introduced in [Hal+17]. In that work, a cyclic TB code [TB15] with minimum distance d and locality r is shown to exhibit a balanced generator matrix where each row is a codeword of weight $d + r - 1$.

The contributions of this work are as follows. First, we generalize the results in [HLH16a; HLH16b] and present a general construction for balanced Reed–Solomon codes, where each row is a codeword of weight w, such that $d \leq w \leq n-1$. Second, we extend the results of [Hal+17] in a similar direction to accommodate a wider variety of row weights, and handle all cyclic TB codes.

While these generalizations result in denser generator matrices, the proof techniques could be of potential use when one is interested in enforcing different types of structure in a codes generator matrix. For related problems, see [HTH15; DSY15; YSZ14].

In the next section, we present an overview of TB codes, along with the key concepts necessary for the results presented in the rest of the chapter. The main results are given in Sections 5.3 and 5.4. We conclude with a discussion of the results and remark on future directions.

5.2 Preliminaries

Reed–Solomon Codes

In this chapter, we will work with Reed–Solomon codes that are cyclic, i.e. those where the defining set is chosen as $\{1, \alpha, \dots, \alpha^{n-1}\}$, where α is a primitive element in $\mathbb{F}_q$. A generator matrix for this code is given by

$$\mathbf{G}_{\text{RS}} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha & \cdots & \alpha^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{(k-1)} & \cdots & \alpha^{(n-1)(k-1)} \end{bmatrix}. \tag{5.1}$$

Viewing Reed–Solomon codes through the lense of polynomials allows us to easily characterize codewords with a prescribed set of coordinates required to be equal to 0. It is known that if we interpolate a degree $k-1$ polynomial $t(x)$ that vanishes on a prescribed set of $k-1$ points, then $t(x)$ is unique up to multiplication by a scalar. Suppose we would like to specify a minimum weight codeword $\mathbf{c} \in \mathsf{RS}[n,k]$ for which $c_{i_0} = \cdots = c_{i_{k-2}} = 0$. We let $t(x) = \prod_{j=0}^{k-1}(x - \alpha^{i_j}) = \sum_{i=0}^{k-1} t_i x^i$, and form the vector of coefficients of $t(x)$ as $\mathbf{t} = (t_0, t_1, \dots, t_{k-1})$. The codeword resulting from encoding of $\mathbf{t}$ using $\mathbf{G}_{\text{RS}}$ is a codeword $\mathbf{c}$ with zeros in the desired coordinates. Indeed, the vector $\mathbf{t}\mathbf{G}_{\text{RS}}$ is the evaluation of the polynomial $t(x)$ at $\{1, \alpha, \dots, \alpha^{n-1}\}$. Since $t(x)$ has $\{\alpha^{i_1}, \dots, \alpha^{i_l}\}$ as roots, it follows that $[\mathbf{t}\mathbf{G}_{\text{RS}}]_{i_1} = \cdots = [\mathbf{t}\mathbf{G}_{\text{RS}}]_{i_l} = 0$. This correspondence between codewords and polynomials will allow us to focus on the latter when constructing generator matrices with the prescribed structure.

A key result that is used to lower bound the minimum distance of cyclic codes is the BCH bound.

Fact 5.1 (BCH bound)**.** *Let $p(x)$ be a nonzero polynomial (not divisible by $x^{q-1} - 1$) with coefficients in $\mathbb{F}_q$. Suppose $p(x)$ has t (cyclically) consecutive roots, i.e. $p(\alpha^{j+1}) = \cdots = p(\alpha^{j+t}) = 0$, where α is primitive in $\mathbb{F}_q$. Then at least $t+1$ coefficients of $p(x)$ are nonzero.*

For a proof of this fact, see e.g. [McE86, p.238]. The BCH bound ensures that all the coefficients of a degree t polynomial with exactly t consecutive roots are nonzero, a result that we will heavily rely on.

Tamo–Barg Codes

We begin by presenting a formal definition of LRCs after which we specialize our presentation to the family of codes developed in [TB14; TB15]. An LRC $\mathcal{C} \subset \mathbb{F}_q^n$ has locality r if every symbol of a codeword $\mathbf{c} \in \mathcal{C}$ can be recovered by accessing at most r other symbols of $\mathbf{c}$. Formally, for each $i \in \{1, \dots, n\}$, there is a subset $\mathcal{I}_i \subset \{1, \dots, n\} \backslash i$, where $|\mathcal{I}_i| \leq r$ such that $c_i = f_i(\{c_j : j \in \mathcal{I}_i\})$. The coordinates $c_i \cup \{c_j : j \in \mathcal{I}_i\}$ form what is known as a local repair group. The notion of local recovery in error-correcting codes was first introduced in [HCL07] with the invention of Pyramid Codes. The work of Gopalan *et al.* [Gop+12] established a Singleton-type bound for linear LRCs, and was later generalized to all codes by Papailiopoulos & Dimakis in [PD12]. The bound states that the minimum distance of an $[n, k, r]$ LRC obeys

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{5.2}$$

Unless otherwise stated, assume henceforth that $r + 1 \mid n$ and $r \mid k$. A Tamo–Barg code $\mathsf{TB}[n, k, r]$ of length n, dimension k and locality r is the set of evaluations of a special subset of polynomials of degree at most $k + \frac{k}{r} - 2$. We will say that a polynomial $m(x)$ is *valid* for $\mathsf{TB}[n, k, r]$ whenever it satisfies (5.3). The codeword $\mathbf{c}$ corresponding to $m(x)$ is then given by $\mathbf{c} = (m(\alpha_1), \dots m(\alpha_n))$, where $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q$. A message polynomial is represented by

$$m(x) = \sum_{\substack{i=0 \\ i \neq r \bmod r+1}}^{k+\mu-2} m_i x^i, \tag{5.3}$$

where the m_i's are the k information symbols. The n evaluation points are chosen as the union of an order subgroup of $\mathbb{F}_q^\times$, the multiplicative group of $\mathbb{F}_q$, along with its cosets. Formally, let $\mathbb{H}$ be a subgroup of $\mathbb{F}_q^\times$, of order $r + 1$. Furthermore, let $\mathbb{H}$ be generated by $\beta := \alpha^\nu$, where α is a primitive element in $\mathbb{F}_q$, and $\nu := \frac{n}{r+1}$, i.e.

$$\mathbb{H} = \{1, \beta, \dots, \beta^r\}.$$

For $i = 0, 1, \dots, \nu - 1$, let $\mathbb{H}_i$ be the coset of $\mathbb{H}$ represented by α^i, i.e.

$$\mathbb{H}_i = \{\alpha^i, \alpha^i\beta, \dots, \alpha^i\beta^r\}. \tag{5.4}$$

The code can now be viewed as the evaluation of polynomials as in (5.3) on $\mathbb{H}_0, \dots, \mathbb{H}_{\nu-1}$. In particular, we have

$$m(\mathbb{H}_i) \leftrightarrow (m(\alpha^i), m(\alpha^i\beta), \dots, m(\alpha^i\beta^r)). \tag{5.5}$$

The codeword $\mathbf{c}$ corresponding to $m(x)$ is given by $\mathbf{c} = (m(\mathbb{H}_0), \dots, m(\mathbb{H}_{\nu-1}))$. Each set of symbols $m(\mathbb{H}_i)$ is a *local group*.

The code just described can can be viewed as a carefully crafted subcode of $\mathsf{RS}[n, k+\mu-1]$. To characterize its generator matrix, note that an element of the set of polynomials mapped to $\mathsf{TB}[n,k,r]$ can be expressed as

$$m(x) = \sum_{i=0}^{r-1} \left(\sum_{j=0}^{\mu-1} m_{i,j} x^{j(r+1)} \right) x^i. \tag{5.6}$$

This adopted view allows us to express the generator matrix of the code as

$$\mathbf{G}_{\mathsf{TB}} = \begin{bmatrix} \mathbf{G}_{\mathsf{TB},0} \\ \mathbf{G}_{\mathsf{TB},1} \\ \vdots \\ \mathbf{G}_{\mathsf{TB},r-1} \end{bmatrix},$$

where each submatrix $\mathbf{G}_{\mathsf{TB},i} \in \mathbb{F}_q^{\mu \times n}$ is given by

$$\mathbf{G}_{\mathsf{TB},i} = \begin{bmatrix} \alpha_1^i & \cdots & \alpha_n^i \\ \alpha_1^{(r+1)+i} & \cdots & \alpha_n^{(r+1)+i} \\ \vdots & \ddots & \vdots \\ \alpha_1^{(\mu-1)(r+1)+i} & \cdots & \alpha_n^{(\mu-1)(r+1)+i} \end{bmatrix}. \tag{5.7}$$

The rows of the required balanced matrix $\mathbf{G}$ will be codewords whose zero coordinates coincide with entire cosets. Furthermore, the corresponding polynomials will vanish on *consecutive* cosets, where by consecutive we mean that the representatives of the cosets in question are consecutive powers of α. To this end, we need to characterize the annihilator polynomial of a coset $\mathbb{H}_i$.

Fact 5.2. *Let $\mathbb{H}_i$ be the coset of $\mathbb{H}$ represented by α^i. The annihilator polynomial of $\mathbb{H}_i$ is given by*

$$z_i(x) = \prod_{h \in \mathbb{H}_i} (x - h) = x^{r+1} - \alpha^{i(r+1)}.$$

Now consider a polynomial that annihilates *consecutive* cosets, i.e.

$$p(x) = \prod_{i=0}^{t-1} z_i(x). \tag{5.8}$$

Note that $p(x)$ is a polynomial in $x^{(r+1)}$ so it is valid for $\mathsf{TB}[n,k,r]$ when $t \le \mu$. Furthermore, it has t nonzero coefficients, as demonstrated by the following proposition.

Proposition 5.1 (Coset BCH bound)**.** *Let $z_i(x)$ be the annihilator of $\mathbb{H}_i$ and define $p(x) = \prod_{i=0}^{t-1} z_i(x)$. The polynomial $p(x)$ has t nonzero coefficients.*

Proof. First, note that $p(x)$ can be written as $p(x) = \sum_{i=0}^{t-1} p_i x^{i(r+1)}$. As a result, $p(x)$ has at most t nonzero coefficients. Furthermore, the polynomial $p(x)$ vanishes on $\{1, \alpha, \ldots, \alpha^{t-2}\}$, a set of $t-1$ consecutive roots. Applying the BCH bound from Fact 5.1 on $p(x)$ yields the result. □

Next, we formalize the notion of balanced matrices and present a construction to realize them. Formally, we identify a matrix $\mathbf{A} \in \{0,1\}^{k \times n}$ in which, for a fixed row weight w, the weights of any two columns differ by at most 1, making it a *balanced* matrix. Each row $\mathbf{a}$ of $\mathbf{A}$ prescribes a mask for some $\mathbf{c}$ for the code of interst. When $\mathsf{RS}[n,k]$ is considered, the support of $\mathbf{c}$ is identical to that of $\mathbf{a}$. However, in the TB code setting, the vector $\mathbf{a}$ prescribes the cosets that the polynomial corresponding to $\mathbf{c}$ vanishes on. At this point, a referral to (5.5) is handy.

w-Balanced Matrices

We start this subsection by presenting a method that produces a w-balanced matrix which can serve our needs. Then, we show how this scheme enables us to construct a w-balanced generator matrix for Reed–Solomon codes. Later on, we develop the techniques further and present results for Tamo–Barg Codes.

Definition 5.1 (w-balanced matrix)**.** *A matrix $\mathbf{A}$ of size k by n is called w-balanced if the following conditions hold:*

P(1) *Every row of $\mathbf{A}$ has the same weight w.*

P(2) *Every column is of weight $\left\lceil \frac{wk}{n} \right\rceil$ or $\left\lfloor \frac{wk}{n} \right\rfloor$.*

A w-balanced error-correcting code is one that is generated by a matrix obeying the properties of Definition 5.1. Let us emphasize that not just any w-balanced matrix can serve as a mask for a target generator matrix. Suppose that for our choice of parameters k, w and n we have that $n \mid wk$. In this case, one can take $\mathbf{A} \in \{0,1\}^{k \times n}$ as the adjacency matrix of a $(w, \frac{wk}{n})$ biregular bipartite graph on k left vertices and n right vertices. Consider the following example.

Example 5.1. *Let $n = 8$, $k = 5$ and $w = 4$, where we are interested in finding a balanced generator matrix for* $\mathsf{RS}[8,5]$. *One possible realization of of a matrix $\mathbf{A}$*

that obeys the conditions of Definition 5.1 is

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.9}$$

Note that the first and last rows are identical, and are of weight 4*, which is the minimum distance of* $\mathsf{RS}[8,5]$*. As alluded to earlier, any two codewords of minimum weight with the same support are scalar multiples of one another. This immediately rules out the possibility of* $\mathbf{A}$ *serving as a mask for a generator matrix of* $\mathsf{RS}[8,5]$*. Indeed, the distinctness of the rows of* $\mathbf{A}$ *is necessary and sufficient.*

Having shown the necessity of carefully constructing a mask matrix for the sought-after generator matrix, the first contribution of this work is to provide a simple algorithm that does this. When the code of interest is $\mathsf{RS}[n,k]$, we present a construction of a w-balanced generator matrix where $n-k+1 \leq w \leq n-1$. For $\mathsf{TB}[n,k,r]$ with minimum distance d, we require $w = d + \delta(r+1) - 2$, where $1 \leq \delta \leq \frac{k}{r}$. In particular, choosing $\delta = 1$ results in a generator matrix where each row is of weight $d + r - 1$.

Let $\mathbf{a}$ be a vector of length n composed of w consecutive ones followed by $n-w$ consective zeros, i.e. $\mathbf{a} = (1,\ldots,1,0,\ldots 0)$. In addition, let $\mathbf{a}_j$ denote the right cyclic shift of $\mathbf{a}$ by j positions[1]. To simplify notation, we use $(x)_n$ to denote $x \,\mathsf{mod}\, n$. Furthermore, we extend this notation to sets by letting $\{x_1,\ldots,x_l\}_n$ denote $\{(x_1)_n,\ldots,(x_l)_n\}$. For example, if $n=8$ and $w=4$, then we have $\mathbf{a}_6 = (1,1,0,0,0,0,1,1)$. Roughly speaking, the desired matrix $\mathbf{A}$ is built by setting its first row to $\mathbf{a}$ and then choosing the next row by cyclically shifting $\mathbf{a}$ by w positions to the right. As mentioned earlier, duplicate rows in $\mathbf{A}$ are to be avoided, and the way to do so is outlined in Construction 5.1.

Construction 5.1. *Let both* k *and* w *be strictly less than* n*. Define the quantities* $g := \mathsf{gcd}(w,n)$*,* $\eta := \frac{n}{g}$*,* $\varphi = \left\lfloor \frac{k}{\eta} \right\rfloor$ *and* $\rho = k - \eta\varphi$*. Define the index sets*

$$\begin{aligned} \mathcal{I}_1 &= \{jw + i : 0 \leq j \leq \eta - 1, 0 \leq i \leq \varphi - 1\}, \\ \mathcal{I}_2 &= \{jw + \varphi : 0 \leq j \leq \rho - 1\}, \end{aligned}$$

[1] Clearly, a shift by $j \geq n$ is equivalent to one where j is taken modulo n.

and $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$. The matrix $\mathbf{A}$ whose rows are given by

$$\{\mathbf{a}_l : l \in \mathcal{I}\}$$

satisfies ***P(1)*** *and* ***P(2)****. Furthermore, the rows of $\mathbf{A}$ are pairwise distinct.*

Proof. First, consider the rows indexed by $\mathcal{I}_1$, and in particular, the subset corresponding to $i = 0$. For $\mathbf{a}_{jw}$, the entries equal to 1 are those indexed by $\{jw, \ldots, (j+1)w - 1\}_n$ while the rest are equal to 0. As a result, for $\{\mathbf{a}_{jw}\}_{j=0}^{\eta-1}$, the entries equal to 1 are those indexed by

$$\{0, \ldots, w-1, \ldots, (\eta-1)w, \ldots, \eta w - 1\}_n.$$

The elements in this set are all distinct before reducing modulo n and furthermore, there are ηw of them, which is a multiple of n. This implies that each residue modulo n appears in the set exactly $\gamma := \frac{w}{g}$ times. The same reasoning applies for each subset of $\mathcal{I}_1$ corresponding to each $i = 1, \ldots, \varphi - 1$ and as a result, the matrix formed by $\{\mathbf{a}_l\}_{l \in \mathcal{I}_1}$ is such that each column has weight $\varphi\gamma$.

Now consider those rows indexed by $\mathcal{I}_2$. If $\frac{wk}{n}$ is an integer then so is $\frac{k}{\eta}$, implying that $\mathcal{I}_2$ is empty, and so **P(2)** holds by the assertion made earlier since $\varphi\gamma = \frac{wk}{n}$. Otherwise, note that $|\mathcal{I}_2| \leq \eta - 1$ since $\rho = k - \eta\varphi = (k)_\eta$, which we will assume to be greater than or equal to 1. Since φ is a common additive factor to all terms in $\mathcal{I}_2$, it can be ignored for now. Thus, we focus on the set $\{0, \ldots, w-1, \ldots, (\rho-1)w, \ldots, \rho w - 1\}$. We can partition the set according to the following relabeling

$$\{0, \ldots, \left\lfloor \frac{\rho w}{n} \right\rfloor n - 1\} \cup \{\left\lfloor \frac{\rho w}{n} \right\rfloor n \ldots, \rho w - 1\},$$

and then reduce the elements modulo n. We conclude that each of the residues $0, \ldots, (\rho w - 1)_n$ appears $\left\lfloor \frac{\rho w}{n} \right\rfloor + 1$ times, whereas each of the remaining ones appears a total of $\left\lfloor \frac{\rho w}{n} \right\rfloor$ times. Combining this with $\mathcal{I}_1$ and adding φ to all indices results in the following conclusion. The columns of $\mathbf{A}$ indexed by $\{\varphi, \varphi+1, \ldots, \varphi + \rho w - 1\}_n$ have the following weight

$$1 + \left\lfloor \frac{(k - \varphi\eta)w}{n} \right\rfloor + \varphi\gamma = \left\lceil \frac{wk}{n} \right\rceil,$$

where we have used the fact that $\frac{\varphi\eta w}{n}$ is an integer. The remaining columns have weight equal to $\left\lfloor \frac{kw}{n} \right\rfloor$. This proves that **P(2)** holds. Property **P(1)** is immediate.

Let us show that the rows are pairwise distinct. Suppose that the rows indexed by $j_1w + i_1$ and $j_2w + i_2$ are identical, where $0 \leq j \leq \eta - 1$ and $0 \leq i \leq \varphi$. This occurs if and only if $j_1w + i_1 \equiv j_2w + i_2 \bmod n$, which is equivalent to $(j_1 - j_2)w \equiv i_2 - i_1 \bmod n$. The quantity g by definition divides both w and n so it must divide $i_2 - i_1$. Note that for $i_1 \leq i_2$,

$$1 \leq |i_2 - i_1| \leq \varphi = \left\lfloor \frac{k}{n} g \right\rfloor < g.$$

Thus, we must have $i_1 = i_2$ and so $(j_1 - j_2)w \equiv 0 \bmod n$. This implies that repeated rows cannot correspond to elements for which one is in $\mathcal{I}_1$ and the other is in $\mathcal{I}_2$. By definition, we have that $\eta \mid n$ but $\gcd(\eta, w) = 1$ so it must be the case that $\eta \mid (j_1 - j_2)$. For distinct j_1 and j_2, we can also bound $|j_1 - j_2|$ by

$$1 \leq |j_1 - j_2| \leq \eta - 1,$$

so we must have $j_1 = j_2$, which proves that the rows of $\mathbf{A}$ are distinct. □

The construction allows us to identify which columns are of weight $\left\lceil \frac{wk}{n} \right\rceil$.

Proposition 5.2. *The columns of* $\mathbf{A}$ *as obtained by Construction 5.1 with weight* $\left\lceil \frac{wk}{n} \right\rceil$ *are those indexed by*

$$\mathcal{H} = \{\varphi, \varphi + 1, \ldots, \varphi + wk - 1\}_n.$$

Proof. It suffices to show that $(\rho w)_n = (wk)_n$. By definition, we know that $\rho = k - \left\lfloor \frac{k}{\eta} \right\rfloor \eta$ and so $\rho w = kw - \left\lfloor \frac{kw}{\eta w} \right\rfloor \eta w$. Now consider $(\rho w)_n$,

$$\begin{aligned}
(\rho w)_n &= \rho w - \left\lfloor \frac{\rho w}{n} \right\rfloor n \\
&= kw - \left\lfloor \frac{kw}{\eta w} \right\rfloor \eta w - \left\lfloor \frac{kw}{n} - \left\lfloor \frac{kw}{\eta w} \right\rfloor \frac{\eta w}{n} \right\rfloor n \\
&= kw - \left\lfloor \frac{kw}{n} \right\rfloor n \\
&= (wk)_n.
\end{aligned}$$

To deduce the third statement from the second, we have use the fact that $\lfloor x + y \rfloor = \lfloor x \rfloor + y$ when y is an integer. Here, $x = \frac{kw}{n}$ and $y = \left\lfloor \frac{kw}{\eta w} \right\rfloor \frac{\eta w}{n}$. □

Construction 5.1 provides a remedy to the matrix in (5.9).

Example 5.2. *Let $n = 8$, $k = 5$ and $w = 4$. A 4-balanced matrix $\mathbf{A}$ is given by Construction 5.1 with parameters are given by $g = 4$, $\eta = 2$, $\varphi = 2$ and $\rho = 1$. The index sets are given by $\mathcal{I}_1 = \{0, 4, 1, 5\}$ and $\mathcal{I}_2 = \{2\}$ which give rise to*

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}. \tag{5.10}$$

It turns out that this matrix can serve as a mask matrix for a 4-balanced generator matrix of $\mathsf{RS}[8, 5]$ defined over $\mathbb{F}_9$. The rows are taken as the evaluations of the following polynomials on $\{1, \alpha, \dots, \alpha^8\}$, where α generates $\mathbb{F}_9^\times$.

$$\begin{aligned} p^{(0)}(x) &= \prod_{i=4}^{7}(x - \alpha^i), \\ p^{(4)}(x) &= 2\prod_{i=4}^{7}(x - \alpha^{i+1}), \\ p^{(1)}(x) &= \prod_{i=4}^{7}(x - \alpha^{i+4}), \\ p^{(5)}(x) &= 2\prod_{i=4}^{7}(x - \alpha^{i+5}), \\ p^{(2)}(x) &= \prod_{i=4}^{7}(x - \alpha^{i+2}). \end{aligned}$$

The resulting 4-balanced generator matrix is given by

$$\mathbf{G} = \begin{bmatrix} \alpha^3 & \alpha^2 & \alpha^4 & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^3 & \alpha^2 & \alpha^4 & \alpha \\ 0 & \alpha^3 & \alpha^2 & \alpha^4 & \alpha & 0 & 0 & 0 \\ \alpha & 0 & 0 & 0 & 0 & \alpha^3 & \alpha^2 & \alpha^4 \\ 0 & 0 & \alpha^3 & \alpha^2 & \alpha^4 & \alpha & 0 & 0 \end{bmatrix}.$$

One can check that $\mathbf{G}$ is full rank over $\mathbb{F}_9$ for α whose minimal polynomial over $\mathbb{F}_3$ is $x^2 + 2x + 2$. The way we chose the polynomials is determined by the set $\mathcal{I}$ from Construction 5.1. In the next section, we formalize this procedure and demonstrate that the polynomials are linearly independent over the code's field, implying that the corresponding codewords span the code.

5.3 Balanced Reed–Solomon Codes

We will motivate the procedure by which we select the codewords of our generating matrix by revisiting the previous example. We start of by fixing $p(x) = \prod_{i=4}^{7}(x-\alpha^i)$. Then, we form the set of polynomials

$$\mathcal{P} = \{p(\alpha^{-i_l}x) : i_l \in \mathcal{I}\}. \tag{5.11}$$

Now consider corresponding to an arbitrary $i_l \in \mathcal{I}$. This polynomial can be expressed as

$$p(\alpha^{i_l}x) = \prod_{i=4}^{7}(\alpha^{-i_l}x - \alpha^i) = \alpha^{-4i_l}\prod_{i=4}^{7}(x - \alpha^{i+i_l}).$$

When evaluated on $\{1, \alpha, \dots, \alpha^8\}$, this polynomial vanishes on and only on the subset $\{\alpha^{4+i_l}, \dots, \alpha^{7+i_l}\}$. Let us record this observation.

Fact 5.3. *The polynomial $p(\alpha^{-l}x)$ is the annihilator of $\alpha^{l+d}, \dots, \alpha^{l+n-1}$ if and only if $p(x)$ is the annihilator of $\alpha^d, \dots, \alpha^{n-1}$.*

Thus, the coordinates of the corresponding codeword that are equal to 0 are precisely those indexed by $\{4 + i_l \dots 7 + i_l\}_n$, which is in agreement with $\mathbf{a}_j$. Hence, the codewords corresponding to the polynomials in (5.11) form a 4-balanced generator matrix whose support is determined by $\mathbf{A}$ in (5.10). In turns out that these polynomials are linearly independent over the underlying field if and only if the elements of $\mathcal{I}$ are pairwise distinct and $w = n - k + 1$. To that end, we present a lemma that formalizes this claim.

Lemma 5.1. *Let $p(x) = \sum_{i=0}^{z} p_i x^i \in \mathbb{F}_q[x]$ and define $\mathcal{P} = \{p(\alpha^{j_l}x)\}_{l=0}^{z}$. The polynomials in $\mathcal{P}$ are linearly independent over $\mathbb{F}_q$ if and only if the elements of $\{\alpha^{j_l}\}_{l=0}^{z}$ are distinct in $\mathbb{F}_q$, and $p_i \neq 0$ for $i = 0, 1 \dots, z$.*

Proof. Let us express $p(\alpha^{j_l}x)$ as

$$p(\alpha^{j_l}x) = \sum_{i=0}^{z} p_i \alpha^{j_l i} x^i.$$

We collect the coefficients of $p(\alpha^{j_l}x)$ in a row vector and form the following matrix,

$$\mathbf{M} = \begin{bmatrix} p_0 & p_1\alpha^{j_0} & \dots & p_z\alpha^{j_0 z} \\ p_0 & p_1\alpha^{j_1} & \dots & p_z\alpha^{j_1 z} \\ \vdots & \vdots & \ddots & \vdots \\ p_0 & p_1\alpha^{j_z} & \dots & p_z\alpha^{j_z z} \end{bmatrix}.$$

The determinant of $\mathbf{M}$ can be written as

$$\mathsf{det}\left(\mathbf{M}\right)=\begin{vmatrix}p_0 & p_1\alpha^{j_0} & \dots & p_z\alpha^{j_0 z}\\ p_0 & p_1\alpha^{j_1} & \dots & p_z\alpha^{j_1 z}\\ \vdots & \vdots & \ddots & \vdots\\ p_0 & p_1\alpha^{j_z} & \dots & p_z\alpha^{j_z z}\end{vmatrix}=\begin{vmatrix}1 & \alpha^{j_0} & \dots & \alpha^{j_0 z}\\ 1 & \alpha^{j_1} & \dots & \alpha^{j_1 z}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & \alpha^{j_z} & \dots & \alpha^{j_z z}\end{vmatrix}\prod_{i=0}^{z} p_i.$$

The Vandermonde matrix has a nonzero determinant if and only if $\{\alpha^{j_l}\}_{l=0}^{z}$ are distinct in $\mathbb{F}_q$. Thus the determinant is nonzero if and only if that holds and all the coefficients of $p(x)$ are nonzero. □

The lemma in its current form provides a tool to constructing d-balanced Reed–Solomon codes, where d is the minimum distance of $\mathsf{RS}[n,k]$.

Theorem 5.1. *For $d = n - k + 1$, let $\mathbf{A}$ be a d-balanced matrix obtained from Construction 5.1 with index set $\mathcal{I}$. Fix $p(x) = \prod_{i=d}^{n-1}(x-\alpha^i)$ and let $\mathcal{P} = \{p(\alpha^{-i_l}x) : i_l \in \mathcal{I}\}$. Then, the matrix $\mathbf{G}$ whose l^{th} row is the codeword corresponding to $p(\alpha^{-i_l}x)$ is a d-balanced generator matrix for $\mathsf{RS}[n,k]$.*

Proof. By construction, the zero coordinates of $\mathbf{a}_{i_l}$, the l^{th} row of $\mathbf{A}$, are indexed by

$$\{i_l + d, i_l + d + 1, \dots, i_l + n - 1\}_n.$$

Furthermore, the polynomial $p(\alpha^{i_l}x) = \alpha^{-(k-1)i_l}\prod_{i=d}^{n-1}(x - \alpha^{i+i_l})$ vanishes on

$$\{\alpha^{i_l+d}, \alpha^{i_l+d+1}, \dots, \alpha^{i_l+n-1}\}.$$

Now collect the evaluations of the polynomials $\mathcal{P}$ as the rows of $\mathbf{G}$. Once we associate the j^{th} column of $\mathbf{G}$ with α^j, the support of $\mathbf{G}$ is that of $\mathbf{A}$, which is d-balanced.

To prove that $\mathbf{G}$ is full rank, it suffices to show the the polynomials in $\mathcal{P}$ are linearly independent over the underlying field. We apply Lemma 5.1 to $p(x)$ and $\{\alpha^{i_l}\}_{i_l\in\mathcal{I}}$. By the BCH bound, we are guaranteed that the coefficients of $p(x)$ are all nonzero. Construction 5.1 ensures that all the elements of $\mathcal{I}$ are distinct modulo n, which implies that all the elements of the set $\{\alpha^{-i_l}\}_{l=0}^{k-1}$ are distinct in $\mathbb{F}_q$. As a result, the polynomials $\mathcal{P}$ are linearly independent over $\mathbb{F}_q$ and so $\mathbf{G}$ spans $\mathsf{RS}[n,k]$. □

Theorem 5.1 provides a method to construct what is known as *sparsest* and *balanced* Reed–Solomon codes [HLH16a; HLH16b]. They are sparsest in the sense that each row of the generator matrix is a minimum distance codeword.

Now suppose that for the same code $\mathsf{RS}[8,5]$, we are interested in a 6-balanced generator matrix. Even though Construction 5.1 provides a 6-balanced mask matrix, it is not possible to use the set of polynomials immediately as prescribed by Theorem 5.1. In particular, Lemma 5.1 doesn't apply since the polynomial $p(x)$ is of degree 2 while the number of polynomials in $\mathcal{P}$ is 5. Luckily, the case when the desired row weight need not be d is attainable with little effort.

Theorem 5.2. *For $n-k+1 \leq w \leq n-1$, let $\mathbf{A}$ be a w-balanced matrix obtained from Construction 5.1 with index set $\mathcal{I} = \{i_0, \ldots, i_{k-1}\}$. Fix $p(x) = \prod_{i=w}^{n-1}(x-\alpha^i)$ and let*

$$\begin{aligned}\mathcal{P}_1 &= \{p(\alpha^{-i_l}x) : l = 0, 1, \ldots, n-w\},\\ \mathcal{P}_2 &= \{x^{l-n+w}p(\alpha^{-i_l}x) : l = n-w+1, \ldots k-1\}.\end{aligned}$$

Then, the matrix $\mathbf{G}$ whose l^{th} row is the codeword corresponding to $p(\alpha^{-i_l}x)$ is a w-balanced generator matrix for $\mathsf{RS}[n,k]$.

Proof. The fact that $\mathbf{G}$ is w-balanced is immediate from the proof of Theorem 5.1. The next task is to prove that $\mathbf{G}$ is full rank. First, we apply Lemma 5.1 to the set $\mathcal{P}_1$ since all $n-w+1$ coefficients of $p(x)$ are nonzero by the BCH bound and the elements $\{\alpha^{-i_l}\}_{l=0}^{n-w}$ are distinct in $\mathbb{F}_q$. Furthermore, all polynomials in $\mathcal{P}_1$ are of degree $n-w$ and so the degree of a polynomial in their $\mathbb{F}_q$-span is upper bounded by the same quantity. Now note that the polynomials in $\mathcal{P}_2$ are all of different degree and so they are linearly independent over $\mathbb{F}_q$. Finally, the fact that the degree of any polynomial in $\mathcal{P}_2$ is lower bounded by $n-w+1$ ensures that $\mathcal{P}_1$ and $\mathcal{P}_2$ are jointly linearly independent. The fact that all polynomials are of degree at most $k-1$ allows us to conclude that their evaluations live in $\mathsf{RS}[n,k]$ and so they span the code. □

Let us revisit the example that was suggested just before the statement of the theorem. It is possible to verify the following matrix does indeed generate $\mathsf{RS}[8,5]$.

$$\mathbf{G} = \begin{bmatrix} \alpha^5 & \alpha^3 & \alpha^2 & \alpha^3 & \alpha^6 & \alpha^2 & 0 & 0 \\ \alpha^2 & \alpha^3 & \alpha^6 & \alpha^2 & 0 & 0 & \alpha^5 & \alpha^3 \\ \alpha^6 & \alpha^2 & 0 & 0 & \alpha^5 & \alpha^3 & \alpha^2 & \alpha^3 \\ 0 & 0 & \alpha^7 & \alpha^6 & \alpha^6 & 1 & \alpha^4 & \alpha \\ 0 & \alpha^7 & \alpha^7 & \alpha^8 & \alpha^3 & 1 & \alpha^6 & 0 \end{bmatrix}.$$

The matrix $\mathbf{G}$ corresponds to the index set $\mathcal{I} = \{0, 6, 4, 2, 1\}$. The polynomials corresponding to the rows of $\mathbf{G}$ are derived from $p(x) = (x - \alpha^6)(x - \alpha^7)$ and are given by

$$\begin{aligned}
p^{(0)}(x) &= (x - \alpha^6)(x - \alpha^7),\\
p^{(6)}(x) &= \alpha^{-4}(x - \alpha^4)(x - \alpha^5),\\
p^{(4)}(x) &= (x - \alpha^2)(x - \alpha^3),\\
p^{(2)}(x) &= \alpha^{-4}x(x - 1)(x - \alpha),\\
p^{(1)}(x) &= \alpha^{-2}x^2(x - \alpha)(x - \alpha^7).
\end{aligned}$$

The idea behind the construction is the exploitation of the fact the we are interested in a generator matrix whose rows are high-weight codewords. The codewords correspond to low-degree polynomials, which allows us to use the extra degrees of freedom available in constructing the set $\mathcal{P}_2$. In fact, one can select $\mathcal{P}_2$ as any set of polynomials whose degrees are all different, and are between $n - w + 1$ and $k - 1$. This will still guarantee that the resulting generator matrix is full rank albeit not w-balanced. Nonetheless, one can potentially use this technique to enforce other patterns in the structure of $\mathbf{G}$.

5.4 Balanced Tamo–Barg Codes

The theme of the construction is as follows. First we will assume that $r \mid k$, We will construct a w-balanced generator matrix $\mathbf{G}_i$ for each subcode generated by a particular $\mathbf{G}_{\mathsf{TB},i}$ as defined (5.7). Arranging these $\mathbf{G}_i$'s as the rows of a matrix $\mathbf{G}$ will result in a w-balanced generator matrix for $\mathsf{TB}[n, k, r]$. We will build a mask matrix $\mathbf{A}_i \in \{0, 1\}^{\mu \times \nu}$ for each $\mathbf{G}_i$, $i = 0, 1, \ldots, r - 1$, by focusing on the cosets associated with each local group in the code. As in the construction of balanced Reed–Solomon codes, a fact along the lines of Fact 5.3 will allow us to form the rows of $\mathbf{G}_i$ from a single polynomial $p(x)$.

Fact 5.4. *The polynomial $p(\alpha^{-l}x)$ is the annihilator of $\mathbb{H}_{l+\delta}, \ldots, \mathbb{H}_{l+\nu-1}$ if and only if $p(x)$ is the annihilator of $\mathbb{H}_{\delta}, \ldots, \mathbb{H}_{\nu-1}$.*

Let us focus on building $\mathbf{G}_0$ from $\mathbf{A}_0$ and motivate the construction by the following example. The row weight of choice is $w = n - (\mu - 1)(r + 1)$.

Example 5.3. *Consider $\mathsf{TB}[15, 6, 2]$ for which we would like to construct a 9-balanced generator matrix. Note that $\nu = 5$ and $\mu = 3$, and define $\delta := \nu - \mu + 1 =$*

3. *The first step is to use Construction 5.1 to build a* 3*-balanced* $\mathbf{A}_0 \in \{0,1\}^{\mu\times\nu}$*,*

$$\mathbf{A}_0 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

This mask matrix prescribes the following set of polynomials defined using $p(x)$*, the annihilator of* $\mathbb{H}_3$ *and* $\mathbb{H}_4$*.*

$$\begin{aligned} p_{0,0}(x) &= p(x) = z_3(x)z_4(x), \\ p_{0,1}(x) &= p(\alpha^{-3}x) = \alpha^{-18}z_1(x)z_2(x), \\ p_{0,2}(x) &= p(\alpha^{-6}x) = \alpha^{-36}z_0(z)z_4(x). \end{aligned}$$

Each $p_{0,j}(x)$ *is a polynomial in* x^3 *with degree* 6*. The evaluations result in the following matrix.*

$$\mathbf{G}_0 = \begin{bmatrix} \alpha^3 & \alpha^3 & \alpha^3 & \alpha^{11} & \alpha^{11} & \alpha^{11} & \alpha^9 & \alpha^9 & \alpha^9 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha^9 & \alpha^9 & \alpha^9 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^3 & \alpha^3 & \alpha^3 & \alpha^{11} & \alpha^{11} & \alpha^{11} \\ 0 & 0 & 0 & \alpha^3 & \alpha^3 & \alpha^3 & \alpha^{11} & \alpha^{11} & \alpha^{11} & \alpha^9 & \alpha^9 & \alpha^9 & 0 & 0 & 0 \end{bmatrix}.$$

Proposition 5.3. *Fix* $\delta = \nu - \mu + 1$ *and let* $\mathbf{A}_0$ *be a* δ*-balanced matrix obtained from Construction 5.1 along with index set* $\mathcal{I}$*. Define the set of polynomials using* $p(x)$ *from* (5.8) *as*

$$\mathcal{P}_0 = \{p(\alpha^{-i_l}x) : i_l \in \mathcal{I}\}. \tag{5.12}$$

The evaluations of $\mathcal{P}_0$ *at* $\mathbb{F}_q^\times$ *result in a set of codewords from* $\mathsf{TB}[n,k,r]$ *that form a* $\delta(r+1)$*-balanced matrix.*

Proof. First, we have that $p(\alpha^{-i_l}x)$ annihilates the cosets $\{\mathbb{H}_{i_l+\delta},\ldots,\mathbb{H}_{i_l+\nu-1}\}$ as guaranteed by Fact 5.4, which is in correspondence with the l^{th} row of $\mathbf{A}_0$. After relabeling the polynomials in $\mathcal{P}_0$ as $\mathcal{P}_0 = \{p_{0,i}(x)\}_{i=0}^{\mu-1}$, consider the matrix $\mathbf{P}_0$ formed by the evaluation of $\mathcal{P}_0$ on $\mathbb{F}_q^\times$,

$$\mathbf{P}_0 = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \cdots & \mathbf{P}_{0,r} \end{bmatrix},$$

where

$$\mathbf{P}_{0,i} = \begin{bmatrix} p_{0,0}(\beta^i) & p_{0,0}(\alpha\beta^i) & \cdots & p_{0,0}(\alpha^{\nu-1}\beta^i) \\ \vdots & \vdots & \ddots & \vdots \\ p_{0,\mu-1}(\beta^i) & p_{0,\mu-1}(\alpha\beta^i) & \cdots & p_{0,\mu-1}(\alpha^{\nu-1}\beta^i) \end{bmatrix}.$$

Note that $\mathbf{P}_{0,i}$ and $\mathbf{A}_0$ have the same dimensions. Furthermore, since the polynomials in $\mathcal{P}_0$ are coset annihilators, the entry $[\mathbf{P}_{0,i}]_{j,l} = p_{0,j}(\alpha^l\beta^i)$ is equal to zero if and only if $[\mathbf{A}_0]_{j,l}$ is equal to zero. Therefore, the fact that $\mathbf{A}_0$ is δ-balanced implies that $\mathbf{P}_0$ is also so. Finally, the weight of each row of $\mathbf{P}_0$ is equal to $\delta(r+1)$ since the size of each coset is $r+1$. □

The proposition portrays the correspondence between the mask matrix $\mathbf{A}_0$ and $\mathbf{G}_0$. Indeed, the matrix $\mathbf{G}_0$ is obtained by a column permutation of $\mathbf{P}_0$, so that columns corresponding to the same local group are adjacent. As a result, the matrix $\mathbf{G}_0$ is balanced. Furthermore, similar to the Reed–Solomon case, the matrix $\mathbf{G}_0$ is full rank if and only if the elements of the index set $\mathcal{I}$ are distinct.

Lemma 5.2. *Let $p(x) = \sum_{i=0}^{z} p_i x^{i(r+1)} \in \mathbb{F}_q[x]$ and define $\mathcal{P} = \{p(\alpha^{j_l}x)\}_{l=0}^{z}$. The polynomials in $\mathcal{P}$ are linearly independent over $\mathbb{F}_q$ if and only if the elements of $\{\alpha^{j_l(r+1)}\}_{l=0}^{z}$ are distinct in $\mathbb{F}_q$, and $p_i \neq 0$ for $i = 0, 1 \ldots, z$.*

Proof. By expressing $p(\alpha^{j_l}x)$ as $\sum_{i=0}^{z} p_i \alpha^{j_l i(r+1)} x^{i(r+1)}$, we can apply the proof idea of Lemma 5.1 to obtain the result. □

Let us complete the construction intiated in Example 5.3. The mask matrix $\mathbf{A}_1$ is identical to $\mathbf{A}_0$ but the polynomials are now given by

$$\begin{aligned} p_{1,0}(x) &= x p_{0,0}(x) = x p(x), \\ p_{1,1}(x) &= x p_{0,1}(x) = x p(\alpha^{-3}x), \\ p_{1,2}(x) &= x p_{0,2}(x) = x p(\alpha^{-6}x). \end{aligned}$$

Note that each $p_{1,j}(x)$ annihilates the same cosets as $p_{0,j}(x)$. As a result, the evaluations of these polynomials form a matrix $\mathbf{G}_1$ with the same mask as $\mathbf{G}_0$. Once we put $\mathbf{G}_0$ and $\mathbf{G}_1$ in a single matrix $\mathbf{G}$, we obtain the follow 9-balanced generator matrix for $\mathsf{TB}[15, 6, 2]$.

$$\mathbf{G} = \left[\begin{array}{ccccccccccccccc}
\alpha^3 & \alpha^3 & \alpha^3 & \alpha^{11} & \alpha^{11} & \alpha^{11} & \alpha^9 & \alpha^9 & \alpha^9 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^9 & \alpha^9 & \alpha^9 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^3 & \alpha^3 & \alpha^3 & \alpha^{11} & \alpha^{11} & \alpha^{11} \\
0 & 0 & 0 & \alpha^3 & \alpha^3 & \alpha^3 & \alpha^{11} & \alpha^{11} & \alpha^{11} & \alpha^9 & \alpha^9 & \alpha^9 & 0 & 0 & 0 \\
\hline
\alpha^3 & \alpha^8 & \alpha^{13} & \alpha^{12} & \alpha^2 & \alpha^7 & \alpha^{11} & \alpha & \alpha^6 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^9 & \alpha^{14} & \alpha^4 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^6 & \alpha^{11} & \alpha & \alpha^{15} & \alpha^5 & \alpha^{10} \\
0 & 0 & 0 & \alpha^4 & \alpha^9 & \alpha^{14} & \alpha^{13} & \alpha^3 & \alpha^8 & \alpha^{12} & \alpha^2 & \alpha^7 & 0 & 0 & 0
\end{array}\right].$$

Indeed, one can verify that $\mathsf{rank}\,(\mathbf{G}) = 6$. This is guaranteed by the multiplicative factor x appearing in each $p_{1,j}(x)$.

Allow us to comment on the previous example. By the choice of parameters it turned out that the same mask matrix can be used for all $\mathbf{G}_i$'s since $\frac{\mu w}{\nu}$ happened to be an integer and so all columns of $\mathbf{A}_i$ are of the same weight. This is not the case in general, calling for the need to modify the way the mask matrices are chosen. The following example illustrates this modification.

Example 5.4. *Consider* $\mathsf{TB}[15, 8, 2]$ *so that* $\nu = 5$ *and* $\mu = 4$*, and set* $w = 6$*. Again, we require two mask matrices* $\mathbf{A}_0, \mathbf{A}_1 \in \{0,1\}^{4\times 5}$*, each of which are* 2*-balanced. We obtain* $\mathbf{A}_0$ *through Construction 5.1:*

$$\mathbf{A}_0 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Now construct $\mathbf{A}_1$ *via cyclically shifting each row of* $\mathbf{A}_0$ *by three positions, which is the number of columns of weight* 2 *in* $\mathbf{A}_0$*.*

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Consider the matrix $\mathbf{A}$ *by vertically stacking* $\mathbf{A}_0$ *and* $\mathbf{A}_1$ *and associate the* j^{th} *column with* $\mathbb{H}_j$*. Once we form the generator matrix* $\mathbf{G}$*, similar to what was done in the previous example, we notice that each column in the local group corresponding to* $\mathbb{H}_0$ *is of weight* 4*, while all the rest are of weight* 3*.*

Indeed, the process of obtaining $\mathbf{A}_1$ from $\mathbf{A}_0$ through cyclic shifts to produce a w-balanced $\mathbf{A}$ can be generalized. Before we formalize this claim, we need the following proposition.

Proposition 5.4. *Suppose* $\tau < \nu$ *and let* $\mathbf{w}$ *be the length* ν *vector where* $[\mathbf{w}]_i = b$ *for* $i \in \mathcal{B} = \{\varphi, \ldots, \varphi + \tau - 1\}_\nu$ *and* $[\mathbf{w}]_i = s$ *such that* $s \in \{b, b-1\}$*, otherwise. Let* $\mathbf{w}_j$ *be a right cyclic shift of* $\mathbf{w}$ *by* j *positions. Consider the matrix* $\mathbf{W}$ *whose rows are* $\{\mathbf{w}_l\}_{l\in\mathcal{L}}$ *where* $\mathcal{L} = \{i\tau : 0 \le i \le r-1\}$*. Let* w_i *be the sum of the entries of the* i^{th} *column of* $\mathbf{W}$*. It holds that* $|w_i - w_j| \le 1$ *for any* i *and* j*.*

Proof. First, we can subtract φ from the elements of $\mathcal{B}$ and add φ to $\mathcal{L}$ without changing $\mathbf{W}$. The way we obtain $\mathbf{W}$ now is by choosing $\mathcal{I}_1 = \emptyset$ and $\mathcal{I}_2 = \mathcal{L}$ in Construction 5.1. Following the same proof, we see that each of the columns indexed by $\{\varphi, \varphi+1, \ldots, \varphi+r\tau-1\}_\nu$ has $\lceil \frac{r\tau}{\nu} \rceil$ entries equal to b, while the remaining $r - \lceil \frac{r\tau}{\nu} \rceil$ entries are equal to s. The sum of entries in any of those columns is given by,

$$u_b = \left\lceil \frac{r\tau}{\nu} \right\rceil b + \left(r - \left\lceil \frac{r\tau}{\nu} \right\rceil\right) s.$$

In an analogous fashion, the sum across each of the remaining columns of $\mathbf{A}$ is

$$u_s = \left\lfloor \frac{r\tau}{\nu} \right\rfloor b + \left(r - \left\lfloor \frac{r\tau}{\nu} \right\rfloor\right) s.$$

Using $b - s \in \{0, 1\}$, we conclude that for any two columns,

$$|w_i - w_j| \leq u_b - u_s = \left(\left\lceil \frac{r\tau}{\nu} \right\rceil - \left\lfloor \frac{r\tau}{\nu} \right\rfloor\right)(b - s) \leq 1.$$

□

We can use this proposition to provide a flexible method of constructing balanced mask matrices for any cyclic TB code.

Lemma 5.3. *Let both δ and μ be strictly less than ν. Let $\mathbf{A}_0 \in \{0,1\}^{\mu \times \nu}$ be δ-balanced obtained using Construction 5.1 and let $\tau = (\delta\mu)_\nu$ be the number of columns of weight $\lceil \frac{\mu\delta}{\nu} \rceil$. For each $i = 1, \ldots, r-1$, construct $\mathbf{A}_i$ by cyclically shifting each row of $\mathbf{A}_0$ by $i\tau$ positions to the right. Then, the matrix $\mathbf{A}$ given by*

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \vdots \\ \mathbf{A}_{r-1} \end{bmatrix}$$

is w-balanced.

Proof. Let $\mathbf{w}$ be a weight vector associated with $\mathbf{A}_0$, where $[\mathbf{w}]_i = \lceil \frac{\delta\mu}{\nu} \rceil$ for $i \in \{\varphi, \varphi+1, \ldots, \varphi+\tau-1\}_\nu$, and $[\mathbf{w}]_i = \lfloor \frac{\delta\mu}{\nu} \rfloor$ otherwise. By construction, the weight vector of $\mathbf{A}_i$ is given by $\mathbf{w}_{i\tau}$. Collecting the weight vectors in $\mathbf{W}$ and applying Proposition 5.4 gives the required result. □

Armed with Lemma 5.3, we can now show how to construct a $(d+r-1)$-balanced generator matrix for $\mathsf{TB}[n, k, r]$.

Theorem 5.3. *Consider* $\mathsf{TB}[n,k,r]$ *defined over* $\mathbb{F}_q$ *where* $q = n+1$, $\delta = \nu - \mu + 1$. *Let* $\mathbf{A}_0$ *be a* δ*-balanced mask matrix obtained from Construction 5.1 with index set* $\mathcal{I} = \{i_0, \ldots, i_{\mu-1}\}$. *Fix* $p(x) = \prod_{i=\delta}^{\nu-1}(x^{r+1} - \alpha^{i(r+1)})$ *and define the base set of polynomials* $\mathcal{P} = \{p(\alpha^{-i_l}x) : l = 0, 1, \ldots, \mu - 1\}$. *Set* $\tau = (\delta\mu)_\nu$ *and define the transformation polynomials* $\mathcal{P}_i = \{x^i q(\alpha^{-i\tau}x) : q(x) \in \mathcal{P}\}$ *for* $i = 0, 1, \ldots, r-1$. *The evaluations of* $\cup_{i=0}^{r-1}\mathcal{P}_i$ *form a* $(d+r-1)$*-balanced matrix* $\mathbf{G}$ *that generates* $\mathsf{TB}[n,k,r]$.

Proof. Consider an arbitrary set $\mathcal{P}_i$. As per Fact 5.4, the polynomial $q(\alpha^{-i\tau}x)$ annihilates $\mathbb{H}_{i\tau+i_l+\delta}, \ldots, \mathbb{H}_{i\tau+i_l+\nu-1}$ if and only if $q(x)$ annihilates $\mathbb{H}_{i_l+\delta}, \ldots, \mathbb{H}_{i_l+\nu-1}$. Furthermore, the multiplicative factor of x^i appearing in each element of $\mathcal{P}_i$ has no effect on the nonzero roots and so the same claim holds on $x^i q(\alpha^{-i\tau}x)$. As a result, the l^{th} row of $\mathbf{G}_i$ is a right cyclic shift of the l^{th} row of $\mathbf{G}_0$ by $i\tau$ positions to the right. This implies that $\mathbf{A}_i$ is obtained from $\mathbf{A}_0$ by the same operation and so it is the mask matrix corresponding to $\mathbf{G}_i$. Lemma 5.3 now proves that the matrix $\mathbf{A}$ is $(d+r-1)$-balanced and Proposition 5.3 ensures that $\mathbf{G}$ is also so.

Let us analyze the rank of $\mathbf{G}$. Lemma 5.2 ensures that $\mathbf{G}_0$ is of rank μ. First, we have that $p(x) = \sum_{i=0}^{\mu-1} p_i x^{i(r+1)}$ and all p_i's are nonzero by virtue of Proposition 5.1. Furthermore, the elements $\{\alpha^{-i_l(r+1)} : i_l \in \mathcal{I}\}$ are all distinct in $\mathbb{F}_q$ since the i_l's are all distinct modulo ν, meaning that $-i_l(r+1) \bmod n$ is unique for each l. To see why the same claim holds for an arbitrary $\mathbf{G}_i$, note that all the elements in the corresponding set $\{\alpha^{-i_l(r+1)-i\tau} : i_l \in \mathcal{I}\}$ are also distinct in $\mathbb{F}_q$, since the added exponent is common to all of them, and the multiplicative term x^i has no effect on linear independence over $\mathbb{F}_q$. Having shown that $\mathsf{rank}\,(\mathbf{G}_i) = \mu$ for all $i = 0, \ldots, r-1$, we need to complete the proof by demonstrating $\mathsf{rank}\,(\mathbf{G}) = k$. To do so, observe that a polynomial $t_i(x)$ in the $\mathbb{F}_q$-span of $\mathcal{P}_i$ is of the form

$$t(x) = \sum_{j=0}^{\mu-1} t_j x^{j(r+1)+i}.$$

Hence, the polynomials $t_i(x)$ and $t_j(x)$ are never equal thanks to the fact that they do not have common monomials. We conclude that $\cup_{i=0}^{r-1}\mathcal{P}_i$ is a linearly independent set of polynomials and $\mathsf{rank}\,(\mathbf{G}) = \mu r = k$. □

The results presented until now rely on two important assumptions. Firstly, we required that $r \mid k$ and used this crucially when partition the generator matrix of $\mathsf{TB}[n,k,r]$ into $\mu = \frac{k}{r}$ blocks. Secondly, the fact that $w = d + r - 1 =$

$(\nu-\mu+1)(r+1)$ was used in proving the linear independence of $\cup_{i=0}^{r-1}\mathcal{P}_i$. The next result relaxes this second assumption to allow for w of the form $\delta(r+1)$, where $\delta > \nu-\mu+1$. The method used to achieve such values of w closely resembles that of Theorem 5.2. In particular, larger values of w provide more degrees of freedom that can help generate linearly independent sets of polynomials.

Theorem 5.4. *Consider* $\mathsf{TB}[n,k,r]$ *defined over* $\mathbb{F}_q$ *where* $q=n+1$. *Fix* $\delta \le \nu-1$ *and let* $\mathbf{A}_0$ *be a* δ*-balanced mask matrix obtained from Construction 5.1 with index set* $\mathcal{I}=\{i_0,\ldots,i_{\mu-1}\}$. *Fix* $p(x)=\prod_{i=\delta}^{\nu-1}(x^{r+1}-\alpha^{i(r+1)})$ *and define the base set of polynomials* $\mathcal{P}=\mathcal{P}^{(1)}\cup\mathcal{P}^{(2)}$ *where*

$$\mathcal{P}^{(1)}=\{p(\alpha^{-i_l}x): l=0,1,\ldots,\nu-\delta\},$$
$$\mathcal{P}^{(2)}=\{x^{(l-\nu+\delta)(r+1)}p(\alpha^{-i_l}x): l=\nu-\delta+1,\ldots,\mu-1\}.$$

Set $\tau=(\delta\mu)_\nu$ *and, for* $i=0,\ldots,r-1$, *define the set of transformation polynomials* $\mathcal{P}_i=\mathcal{P}_i^{(1)}\cup\mathcal{P}_i^{(2)}$ *where*

$$\mathcal{P}_i^{(1)}=\{x^i q(\alpha^{-i\tau}x): q(x)\in\mathcal{P}^{(1)}\},$$
$$\mathcal{P}_i^{(2)}=\{x^i q(\alpha^{-i\tau}x): q(x)\in\mathcal{P}^{(2)}\}.$$

The evaluations of $\cup_{i=0}^{r-1}\mathcal{P}_i$ *form a* $\delta(r+1)$*-balanced matrix* $\mathbf{G}$ *that generates* $\mathsf{TB}[n,k,r]$.

Proof. First, note that the term $x^{l-\nu+\delta}$ appearing in $\mathcal{P}^{(2)}$ has no effect on the structure of the resulting generator matrix $\mathbf{G}$. As a result, the fact that the matrix $\mathbf{G}$ is $\delta(r+1)$-balanced is a consequence of Proposition 5.4 and Lemma 5.3.

To prove that $\mathbf{G}$ is full rank, we first show that $\mathcal{P}$ is a set of linearly independent polynomials. As before, Lemma 5.1 ensures that the coefficient of each $x^{i(r+1)}$ in $p(x)$ is nonzero, and Lemma 5.2 guarantees that $\mathcal{P}^{(1)}$ is a linearly independent set of polynomials. The degree of $p(x)$ is $(\nu-\delta)(r+1)$ so this quantity is an upper bound on any polynomial in the $\mathbb{F}_q$-span of $\mathcal{P}^{(1)}$. Next, we have that the degrees of the polynomials in $\mathcal{P}^{(2)}$ are $(\nu-\delta+1)(r+1),(\nu-\delta+2)(r+1),\ldots,(\mu-1)(r+1)$, and are all different. This ensures that $\mathcal{P}^{(2)}$ is also a linearly independent set of polynomials and so is $\mathcal{P}$. Finally, the same argument used in the proof of Theorem 5.3 ensures that $\cup_{i=0}^{r-1}\mathcal{P}_i$ are linearly independent over $\mathbb{F}_q$ and so $\mathsf{rank}\,(\mathbf{G})=k$. □

As mentioned earlier, Theorems 5.3 and 5.4 assume that $r \mid k$. In the next section, we lift this assumption and present results of similar flavor.

Construction w-balanced Tamo–Barg codes with no restriction on r and k

In [TB14, p.4], the authors show how to slightly modify their construction to accomodate any r and k. Let $\bar{\mu} = \left\lfloor \frac{k}{r} \right\rfloor$ and $s = k \bmod r$. The form of a message polynomial now becomes

$$m(x) = \sum_{i=0}^{r-1} \left(\sum_{j=0}^{\bar{\mu}-1} m_{i,j} x^{j(r+1)} \right) x^i + \sum_{i=0}^{s-1} m_{i,\bar{\mu}} x^{\bar{\mu}(r+1)} x^i = m_1(x) + m_2(x). \quad (5.13)$$

Note that $m_1(x)$ is in the form of (5.6). Therefore, the subset of all polynomials (5.13) with $m_2(x) = 0$ correspond to $\mathsf{TB}[n, \bar{\mu}r, r]$, which is a subcode of $\mathsf{TB}[n, k, r]$. This motivates the following idea. For a fixed $w = \delta(r + 1)$, where $\delta \geq \nu - \bar{\mu} + 1$, the first $\bar{\mu}r$ rows of the desired $\mathbf{G}$ are obtained as a w- balanced $\mathbf{G}_1$ through either Theorem 5.3 or Theorem 5.4. The remaining s rows, collected in $\mathbf{G}_2$, are chosen according to a mask matrix $\mathbf{A}_2 \in \{0, 1\}^{s \times \nu}$ which is $(\delta + 1)$-balanced, and so $\mathbf{G}_2$ ends up being $(\delta + 1)(r + 1)$-balanced. The particular choice of $\mathbf{A}_2$ will ensure that the columns of $\mathbf{G}$ differ in weight by at most 1. Let us formally put this description in a definition.

Definition 5.2 ((w_1, w_2)-Balanced Matrix)**.** *A matrix $\mathbf{A}$ of size k by n is called (w_1, w_2)-balanced if the following conditions hold:*

- *Every row of $\mathbf{A}$ is of weight w_1 or w_2.*
- *Any two columns of $\mathbf{A}$ differ in weight by at most 1.*

Theorem 5.5. *Consider $\mathsf{TB}[n, k, r]$ and let $\bar{\mu} = \left\lfloor \frac{k}{r} \right\rfloor$.Fix, $\delta \geq \nu - \bar{\mu} + 1$. Let $\mathbf{G}_1$ be a δ-balanced generator matrix for $\mathsf{TB}[n, \bar{\mu}r, r]$, Let $p(x)$ be the annihilator of $\mathbb{H}_{\delta+1}, \ldots, \mathbb{H}_{\nu-1}$. Fix $s = k \bmod r$, $z = \bar{\mu} - (\nu - \delta - 1)$ and define the set of transformation polynomials*

$$\mathcal{P} = \{x^{z(r+1)+i} p(\alpha^{-\omega - i(\delta+1)} x) : i = 0, \ldots, s - 1\}. \quad (5.14)$$

Organize the evaluations of $\mathcal{P}$ in $\mathbf{G}_2$ and form $\mathbf{G}$ as the vertical concatenation of $\mathbf{G}_1$ and $\mathbf{G}_2$. For a suitable value of ω, the matrix $\mathbf{G}$ is $(\delta(r+1), (\delta+1)(r+1))$-balanced and generates $\mathsf{TB}[n, k, r]$.

Proof. Let us start by characterizing the support of $\mathbf{G}_2$. Let $\tilde{\mathbf{a}}$ be a mask vector for $p(x)$, where $[\tilde{\mathbf{a}}]_i = 0$ if and only if $p(\mathbb{H}_i) = 0$. Thus, the first $\delta + 1$ entries of $\mathbf{a}$ are equal to 1 and the rest are equal to zero. If we set $\mathbf{a} = \tilde{\mathbf{a}}_\omega$, then Fact 5.4 tells us that the mask vector for $x^{z(r+1)+i} p(\alpha^{-\omega - i(\delta+1)} x)$ is given by $\mathbf{a}_{i(\delta+1)}$. We can now apply Proposition 5.4 to matrix $\mathbf{A}$ with rows $\{\mathbf{a}_{i(\delta+1)}\}_{i=0}^{s-1}$ with $\varphi = \omega$ and $\tau = \delta + 1$. This implies that $\mathbf{A}$ is $(\delta+1)$-balanced which, in turn, implies that $\mathbf{G}_2$ is $(\delta+1)(r+1)$-balanced.

We now turn to $\mathbf{G}$. If all columns of $\mathbf{G}_1$ happen to be of the same weight, then we automatically have that any two columns of $\mathbf{G}$ differ in weight by 1. The same reasoning applies by considering $\mathbf{G}_2$. Therefore, let us assume that the column weights in both $\mathbf{G}_1$ and $\mathbf{G}_2$ are different. Suppose that the columns of $\mathbf{G}_1$ with the larger weight b_1 correspond to the cosets indexed by $\{\varphi, \varphi+1, \ldots, \varphi+\tau+1\}_\nu$, while the rest are of weight $s_1 := b_1 - 1$. We claim that setting $\omega = (\varphi + \tau - 1)_\nu$ ensures that $\mathbf{G}$ is as required. Without loss of generality, we can assume $\varphi = 0$ as this is equivalent to shifting each row of $\mathbf{G}$ by φ positions to the left. With this assumption in hand we can identify the “heavier" columns of $\mathbf{G}_1$ with the set

$$\mathcal{B}_1 = \{0, 1, \ldots, \tau - 1\}_\nu,$$

and the ”lighter" columns with

$$\mathcal{S}_1 = \{\tau, \tau+1, \ldots, \tau+\nu-1\}_\nu,$$

An application of Proposition 5.2 to $\mathbf{A}$ yields a similar identification for $\mathbf{G}_2$:

$$\mathcal{B}_2 = \{\tau, \tau+1, \ldots, \tau - 1 + s(\delta+1)\}_\nu.$$

where each column corresponding to an element of $\mathcal{B}_2$ is of weight b_2, whereas the rest are of weight $s_2 := b_2 - 1$. The sets $\mathcal{S}_1$ and $\mathcal{B}_2$ can interact in two different ways. First, suppose that $\mathcal{S}_1 \subseteq \mathcal{B}_2$, as depicted in the following diagram, where the vertical rule marks $(\tau)_\nu$;

$$\left[\begin{array}{c|c} \mathcal{B}_1 & \mathcal{S}_1 \\ \hline \mathcal{B}_2 \quad \mathcal{S}_2 & \mathcal{B}_2 \end{array}\right].$$

The columns of $\mathbf{G}$ corresponding to $\mathcal{S}_1$ are of weight $s_1 + b_2$, while the remaining ones are of weight $b_1 + b_2$ or $b_1 + s_2$. It is immediate that the difference between any two of those three quantities is at most 1.

The other scenario corresponds to $\mathcal{B}_2 \subseteq \mathcal{S}_1$,

$$\left[\begin{array}{c|c} \mathcal{B}_1 & \mathcal{S}_1 \\ \hline \mathcal{S}_2 & \mathcal{B}_2 \quad \mathcal{S}_2 \end{array}\right].$$

In this case, the columns of $\mathbf{G}$ corresponding to $\mathcal{B}_2$ are of weight $s_1 + b_2$, while the remaining ones are of weight $s_1 + s_2$ or $b_1 + s_2$. The claim on the column weights also holds.

The proof is incomplete unless we assert that $\mathsf{rank}\left(\mathbf{G}\right) = k$. First, we have $\mathsf{rank}\left(\mathbf{G}_1\right) = \bar{\mu}r$ as guaranteed by the theorem that provided $\mathbf{G}_1$. Keeping in mind that $\mathsf{deg}\left(p(x)\right) = (\nu - \delta - 1)(r+1)$, it is immediate that

$$\mathsf{deg}\left(x^{z(r+1)+i}p(\alpha^{-\omega-i(\delta+1)}x)\right) = \bar{\mu}(r+1)+i.$$

Furthermore, we know that the degree of any polynomial corresponding to a row of $\mathbf{G}_1$ is at most $(\bar{\mu}-1)(r+1)+r-1$, which is strictly less that $\bar{\mu}(r+1)+i$, guaranteeing that $\mathbf{G}$ is full rank. □

5.5 Discussion

The main results presented in this chapter provide a somewhat flexible method to construct generator matrices from error-correcting codes that are balanced. Theorem 5.1 provides us with a way to construct *sparsest* and *balanced* Reed–Solomon codes. The fact that each row of $\mathbf{G}$ is of minimal weight implies that when a single message symbol is updated, the least number of code symbols possible need to be modified. This feature can be appealing in the context of distributed storage systems since the number of storage nodes contacted during an update is minimal. As discussed, the balanced property ensures that all storage nodes finish computing their respective code symbols in the same amount of time. A slight drawback of sparsest and balanced generator matrix is the fact that it is non-systematic. Indeed, this could be a serious hindrance to read-intensive applications, such as data centers accommodating “hot" data. However, the target of our work is those applications that are “write-intensive" which, as mentioned, are caterers to massive amounts of data.

The fact that the row weight w is arbitrary could potentially be useful when optimizing some performance measure other than the encoding speed. Furthermore, the techniques utilized in the proofs of Theorems 5.2 and 5.4 did not use the fact that all k rows of $\mathbf{G}$ are of weight w. This could potentially shed light on generalizing techniques for the construction of Chapter 2 in which a generator matrix of a Reed–Solomon code is required to be of a particular structure. Relevant works that could be benefit from such technique are [DSY14; Dau+15a; YS11; YSZ14].

Extending the concept of balanced generator matrices to Tamo–Barg codes is natural given their target application and theoretical significance as generalizations of Reed–Solomon codes. When the locality parameter r is small, the result presented in Theorem 5.3 realizes a matrix $\mathbf{G}$ that is close to being sparsest.

A general notion of LRCs is one in which every local group corresponds to a $(r+\rho-1, r)$ MDS code, i.e. any local group can withstand $\rho - 1$ erasures. These codes are said to have (r, ρ) locality, and the construction in [TB14] is naturally extended to those codes by letting the defining subgroup $\mathbb{H}$ be of order $r+\rho-1$, with $n = \nu(r+\rho-1)$. The code $\mathsf{TB}[n, k, r, \rho]$ is obtained by modifying the polynomials in (5.6) to

$$m(x) = \sum_{i=0}^{r-1} \left(\sum_{j=0}^{\mu-1} m_{i,j} x^{j(r+\rho-1)} \right) x^i. \tag{5.15}$$

With this view in mind, we can extend the results of Section 5.4 where one can now construct w-balanced TB codes with (r, ρ) locality for $w = d + \delta(r+\rho-1) + 2$, where $1 \leq \delta \leq \frac{k}{r}$. Theorem 5.5 can also be generalized in the same manner.

Chapter 6

REED–SOLOMON CODES FOR DISTRIBUTED COMPUTATION

6.1 Introduction

The size of today's datasets have made it necessary to often perform computations in a distributed manner. For this reason, parallel and distributed computing has attracted a lot of attention in recent years from both machine learning and optimization communities [Boy+11; Rec+11; Zin+10; Gem+11].

The parallel nature of distributed computation promises significant speedups. Yet, the speedups observed in practice [DB13] fall far behind. Indeed, in the face of substantial or heterogeneous delays, distributed computing may suffer from being slow, which defeats the purpose of the exercise. There are several approaches to address this problem. One naive way to tackle this issue, especially when the task consists of many iterations, is to ignore the *straggling machines* and hope that on average the taskmaster receives enough information to be able to compute the required function accurately. However, it is clear that in this case the performance of the learning algorithm may be significantly impacted due to the lost updates. An alternative and more appropriate way to resolve this issue is to introduce some *redundancy* in the data provided to the machines, in order to efficiently trade off per-machine computation time for less overall wait time, and be able to recover the correct update using only a few machines. Redundancy on its own, however, can incur severe communication costs, as the machines not only have to compute additional tasks, but also communicate the results. The challenge then is to design a clever scheme for distributing the task among the machines, such that the computation can be recovered efficiently using a few machines, independent of which machines they are. As a result, a scheme in which the machines not only compute additional tasks, but *code* across them seems appropriate for combating stragglers, and ensuring that redundant computations are communicated efficiently; straggler mitigation schemes based on coding theory are starting to gain traction.

Related Work

Coding-theoretic straggler mitigation schemes were first introduced by Lee et al. [Lee+16] where it was shown that distributed matrix multiplication can be spedup significantly by relying on MDS codes. Dutta et al. [DCG16] proposed a coding technique for introducing redundancy in computation, instead of simple repetition, for speeding up computing linear transformations by sparsifying the matrices provided to the machines. A coded MapReduce framework was introduced by Li et al. in [LMA16b], which is used for facilitating data shuffling in distributed computing. The first work to consider schemes for distributed gradient descent is that of Tandon et al. [Tan+16], where it was shown that an optimal distribution scheme exists along with efficient encoding and decoding functions that can recover the full gradient from the least number of machines theoretically possible. It was shown that this can be accomplished by a careful distribution of the subtasks across the machines and then employing an MDS code to specify the coding coefficients. Furthermore, this allowed for the recovery of the full gradient at the task master by using an algorithm that runs in cubic time.

6.2 Problem Setup

We consider the problem of straggler mitigation in a distributed gradient descent setting. We focus on the task of fitting a vector of parameters $\beta \in \mathbb{R}^p$ to a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, where the goal is minimizing a loss function of form

$$L(\mathcal{D}; \beta) = \sum_{i=1}^{N} \ell(x_i, y_i; \beta), \tag{6.1}$$

where $\ell(x_i, y_i; \beta)$ is the loss function of the model when evaluated at x_i. Gradient descent algorithms are usually used for solving such machine learning problems. The gradient of the loss function, with respect to β is given by

$$\nabla L(\mathcal{D}; \beta) = \sum_{i=1}^{N} \nabla \ell(x_i, y_i; \beta). \tag{6.2}$$

The model β evolves as a function of time according to the gradient descent rule:

$$\beta_{t+1} = \beta_t - \eta_t \nabla L(\mathcal{D}; \beta),$$

where η_t is known as the learning rate and is allowed to depend on time.

We assume the setting where the computation is distributed equally amongst n machines and the master is to recover the gradient from any $n - s$ machines in order to update the model β_t, where s depends on the computational load. See Fig. 6.1 .

More formally, consider the partition of the dataset into $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ where the $\mathcal{D}_i$'s are disjoint subsets of $\mathcal{D}$ of size $\frac{N}{k}$ (assume k divides N). This allows us to

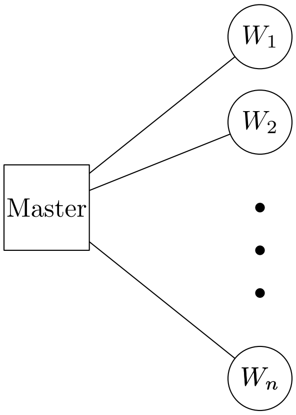


Figure 6.1: Schematic representation of the taskmaster and the n workers.

rewrite (6.2) as

$$\nabla L(\mathcal{D};\beta) = \sum_{i=1}^{k} \sum_{(x,y)\in\mathcal{D}_i} \nabla\ell(x,y;\beta). \tag{6.3}$$

Define $g_i := \sum_{(x,y)\in\mathcal{D}_i} \nabla\ell(x,y;\beta)$ and $g := [g_1^{\mathsf{t}}, g_2^{\mathsf{t}}, \ldots, g_k^{\mathsf{t}}]^{\mathsf{t}}$, where g_i is a row vector of size p. Let $\mathbf{1}$ denote the all-one column vector of length k. The goal is to recover $\nabla L(\mathcal{D};\beta) = \mathbf{1}^{\mathsf{t}} g$ in a distributed fashion. We have a set of n workers $\{W_1, W_2, \ldots, W_n\}$. Also define the taskmaster, M, as the entity responsible for computing the gradient from the (partial) result of computations done by workers.

In principle, each W_i can compute a partial gradient and send it back to M. Once M receives $\{g_1, \ldots, g_k\}$, it computes the gradient by simply adding the results. In order to avoid the impact of stragglers, designing a flexible scheme which can compute the gradient using the result of computations of some (but not all) machines is crucial.

In order to have a scheme that can tolerate any set of stragglers of size at most s, the taskmaster should be able to recover the gradient vector given the result of any $f := n - s$ machines. This set of machines can vary from iteration to iteration and so our scheme should accommodate this fact. Suppose each worker W_i is provided with w data partitions, $\{\mathcal{D}_{i_1}, \ldots, \mathcal{D}_{i_w}\}$, on which it computes the partial gradients $\{g_{i_1}, g_{i_2}, \ldots, g_{i_w}\}$. Here, we assume that the workers are doing same amount of computations, i.e., w is fixed for all workers. It is here where we see that redundancy is introduced, as the quantity wn is usually greater than (or equal to) k.

The following proposition indicates the upper bound for the number of stragglers in terms of system parameters. We refer the reader to [Tan+16] for the proof.

Proposition 6.1. *In a distributed system with n workers, if we partition the data into k disjoint sets and each worker computes the gradient on exactly w sets, the number of stragglers, s, must satisfy*

$$s \leq \left\lfloor \frac{wn}{k} \right\rfloor - 1. \tag{6.4}$$

After computing the corresponding partial gradients, $\{g_{i_1}, g_{i_2}, \ldots, g_{i_w}\}$, each worker W_i can send all these vectors to M. However, there is benefit in W_i sending a *linear combination* of $\{g_{i_1}, \ldots, g_{i_w}\}$. Indeed, the amount of data being sent (communication load) is reduced by a factor of w, and we can show that with an appropriate choice of the encoding coefficients, there is no loss in the number of stragglers that can be tolerated.

This choice amounts to a careful distribution of the data partitions among the n workers, and prescribing a linear combination to each W_i. Mathematically, this can be expressed as designing a matrix $\mathbf{B} \in \mathbb{C}^{n \times k}$ such that:

- Each row of $\mathbf{B}$ contains exactly w nonzero entries.
- The linear space generated by any f rows of $\mathbf{B}$ contains the all one vector, $\mathbf{1}^{\mathsf{t}}$.

The nonzero locations in $\mathbf{b}_i^{\mathsf{t}}$, the ith row of $\mathbf{B}$, determine the indices of data partitions assigned to worker i. The first property ensures that each worker computes partial gradients on exactly w chunks of data.

Furthermore, the values of these nonzero entries prescribe the linear combination. The coded partial gradient sent by W_i is given by

$$c_i = \sum_{j=1}^{k} \mathbf{B}_{i,j} g_j = \mathbf{b}_i^{\mathsf{t}} g, \tag{6.5}$$

where $c_i \in \mathbb{C}^{1 \times p}$ is a row vector which denotes the linear combination of the partial gradient sent from W_i to M. Since $\mathbf{b}_i$ has sparsity w each worker essentially computes a weighted sum of the gradients of w chunks of the data. Define the encoded computation matrix $\mathbf{C} \in \mathbb{C}^{n \times p}$ as

$$\mathbf{C} = [c_1^{\mathsf{t}}, c_2^{\mathsf{t}}, \ldots, c_n^{\mathsf{t}}]^{\mathsf{t}} = \mathbf{B} g, \tag{6.6}$$

where the i th row of $\mathbf{C}$ is equal to c_i. The goal is design matrix $\mathbf{B}$ in such a way that the taskmaster be able to recover the gradient $\nabla L(\mathcal{D};\beta) = \mathbf{1}^{\mathsf{t}} g$ from any f rows of $\mathbf{C}$. In particular, let $\mathcal{F} = \{i_1, \ldots, i_f\}$ be the index set of surviving machines and let $\mathbf{B}_{\mathcal{F}}$ be the sub-matrix of $\mathbf{B}$ indexed by $\mathcal{F}$. If $\mathbf{1}^{\mathsf{t}}$ is in the linear space generated by the rows of $\mathbf{B}_{\mathcal{F}}$, there is a corresponding vector, $\mathbf{a}_{\mathcal{F}}$ such that:

$$\mathbf{a}_{\mathcal{F}}^{\mathsf{t}} \mathbf{C}_{\mathcal{F}} = \mathbf{a}_{\mathcal{F}}^{\mathsf{t}} \mathbf{B}_{\mathcal{F}} g = \mathbf{1}^{\mathsf{t}} g = \nabla L(\mathcal{D};\beta). \tag{6.7}$$

This has to hold for any set of indices $\mathcal{F} \subset [n]$ of size f. Using the theory of maximum-distance separable (MDS) codes, the authors of [Tan+16], construct an encoding matrix $\mathbf{B}$, along with the set of decoding vectors $\{a_{\mathcal{F}} : \mathcal{F} \subset [n], |\mathcal{F}| = f\}$. The encoding matrix they designed must satisfy $w \geq \frac{k}{n}(s+1)$ according to Proposition 6.1.

Preliminaries

In a distributed scheme that does not employ redundancy, the taskmaster has to wait for all the workers to finish in order to compute the full gradient. However, in the scheme outlined above, the taskmaster needs to wait for the fastest f machines to recover the full gradient. Clearly, this requires more computation by each machine. Note that in the uncoded setting, the amount of computation that each worker does is $\frac{1}{n}$ of the total work, whereas in the coded setting each machine performs a $\frac{w}{k}$ fraction of the total work. From (6.4), we know that if a scheme can tolerate s stragglers, the fraction of computation that each worker does is

$$\frac{w}{k} \geq \frac{s+1}{n}. \tag{6.8}$$

Therefore, the computation load of each worker increases by a factor of $(s+1)$. At this point, it is not immediately clear what the optimal value for $\frac{w}{k}$ is. This optimization will be analyzed in detail in Section 6.5.

It is worth noting that it is often assumed [Tan+16; Lee+16; DCG16] that the decoding vectors are precomputed for all possible combinations of returning machines, and the decoding cost is not taken into account in the total computation time. In a practical system, however, it is not very reasonable to compute and store all the decoding vectors, especially as there are $\binom{n}{f}$ such vectors, and this grows quickly with n. A contribution of this chapter is the design of an online algorithm for computing the decoding vectors on the fly, for the indices of the f workers that respond first. The approach is based on the idea of inverting Vandermonde matrices, which can be done very efficiently. In the sequel, we show how to construct an encoding matrix $\mathbf{B}$ for any w, k and n, such that the system is resilient to $\left\lfloor \frac{wn}{k} \right\rfloor - 1$ stragglers, along with an efficient algorithm for computing the decoding vectors $\{\mathbf{a}_{\mathcal{F}} : \mathcal{F} \subset [n], |\mathcal{F}| = f\}$.

6.3 Construction

The basic building block of our encoding scheme is a matrix $\mathbf{M} \in \{0,1\}^{n \times k}$, where each row is of weight w, which serves as a *mask* for the matrix $\mathbf{B}$. Each column of $\mathbf{B}$ will be chosen as a codeword from a suitable Reed–Solomon Code over $\mathbb{C}$, with support dictated by the corresponding column in $\mathbf{M}$. Whereas the authors of [Tan+16] choose the *rows* of $\mathbf{B}$ as codewords from a suitable MDS code, this approach does not immediately work when k is not equal to n.

Balanced Mask Matrices

We will utilize techniques from [HLH16a; HLH16b] to construct the matrix M (and then $\mathbf{B}$). For that, we present the following definition.

Definition 6.1 (Balanced matrix)**.** *A matrix* $\mathbf{M} \in \{0,1\}^{n \times k}$ *is column (row)-balanced if for fixed row (column) weight, the weights of any two columns (rows) differ by at most 1.*

Ultimately, we are interested in a column-balanced matrix $\mathbf{M}$ with row weight w that prescribes a mask for the encoding matrix $\mathbf{B}$. As an example, let $n = 8$, $k = 4$ and $w = 3$. Then $\mathbf{M}$ is given by

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \tag{6.9}$$

where each column is of weight $\frac{nw}{k} = 6$. Note that for this choice of parameters, $\mathbf{M}$ is both row and column-balanced. Now consider another set of parameters $n = 8$, $k = 5$ and $w = 4$. The quantity $\frac{nw}{k}$ is not an integer so clearly the weights of the columns of $\mathbf{M}$ have to be different. A particular column-balanced $\mathbf{M}$ is given below.

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{6.10}$$

Note that the first two columns form a *row*-balanced matrix with column weight equal to seven, and the last three columns form another row-balanced matrix with column weight equal to six. This example serves as an inspiration for an algorithm that constructs column-balanced matrices for fixed row weight from two constituents each of which are row-balanced. The following algorithm produces a row-balanced mask matrix. For a fixed column weight d, each row has weight either $\lfloor \frac{kd}{n} \rfloor$ or $\lceil \frac{kd}{n} \rceil$. Let $(x)_n$ denote the remainder from dividing x by n so that $0 \leq (x)_n \leq n-1$.

Algorithm 2 Row-Balanced Mask Matrix $\mathbf{M}$

```
Input:
  $n$: Number of rows
  $k$: Number of columns
  $d$: Weight of each column
  $t$: Offset parameter
Output: Row-balanced $M \in \{0,1\}^{n,k}$
  function RowBalancedMaskMatrix($n$,$k$,$d$,$t$)
      $\mathbf{M} \leftarrow \mathbf{0}_{n,k}$
      for $i = 0$ to $k-1$ do
          for $j = 0$ to $d-1$ do
              $r = (id + j + t)_n$
              $\mathbf{M}_{r,i} = 1$
          end for
      end for
      return $\mathbf{M}$
  end function
```

The intuition behind Algorithm 2 is best demonstrated through an example. Consider the first two columns of the matrix in (6.10) as

$$\mathbf{M}_h = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{6.11}$$

We obtain $\mathbf{M}_h$ by calling RowBalancedMaskMatrix(8,2,7,0). The first column of the required matrix is set to be a length seven vector whose first six entries are equal to one and whose remaining entries are equal to zero. The second column is built by taking the first column and cyclically shifting its entries downwards by d positions. In general, the j^{th} column is chosen as a downward cyclic shift of the first column by $(j-1)d$ positions. The parameter t determines the starting position of the sequence of ones in the first column, which in this case is position 0. More generally, it determines which rows are of weight $\left\lfloor \frac{kd}{n} \right\rfloor$ and which are of weight $\left\lceil \frac{kd}{n} \right\rceil$. Now consider the last 3 columns of the matrix in (6.10) as

$$\mathbf{M}_l = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \tag{6.12}$$

which is obtained by running RowBalancedMaskMatrix(8,3,6,6). Choosing $t = 6$ indicates that the starting position of the sequence of ones in the first column begins at position 6 and cyclically continues for d positions. The particular value of this parameter will be selected carefully when constructing a column-balanced matrix with fixed row weight. For a proof of correctness of Algorithm 2, please refer to the appendix.

Once the row weight w is fixed, if $d = \frac{wk}{n} \in \mathbb{Z}$, then Algorithm 2 can be used to generate a matrix $\mathbf{M}$ which in turn, allows us to design the encoding matrix $\mathbf{B}$: the nonzero entries of $\mathbf{B}$ will be chosen appropriately so that the i^{th} row of the resulting matrix $\mathbf{B}$ specifies the encoding coefficients for W_i. The j^{th} column of $\mathbf{B}$ will be chosen as a Reed–Solomon codeword whose support is that of the j^{th} column of $\mathbf{M}$. This procedure will be described in detail in the upcoming section.

In the case $d = \frac{wk}{n} \notin \mathbb{Z}$, the chosen row weight w prevents the existence of $\mathbf{M}$ where each column weight is minimal. We have to resort to Algorithm 3 that yields $\mathbf{M}$ comprised of two matrices $\mathbf{M}_h$ and $\mathbf{M}_l$ according to

$$\mathbf{M} = \begin{bmatrix}\mathbf{M}_h & \mathbf{M}_l\end{bmatrix}. \tag{6.13}$$

The matrices $\mathbf{M}_h$ and $\mathbf{M}_l$ are constructed using Algorithm 2. Each column of $\mathbf{M}_h$ has weight $d_h := \left\lceil \frac{nw}{k} \right\rceil$ and each column of $\mathbf{M}_l$ has weight $d_l := \left\lfloor \frac{nw}{k} \right\rfloor$. Note that according to (6.4), we require $d_l \geq 2$ in order to tolerate a positive number of stragglers.

Algorithm 3 Column-balanced Mask Matrix $\mathbf{M}$

```
Input:
  $n$: Number of rows
  $k$: Number of columns
  $w$: Weight of each row
Output: Row-balanced $M \in \{0,1\}^{n,k}$.
  procedure MaskMatrix($n$,$k$,$w$)
      $k_h \leftarrow (nw)_k$
      $d_h \leftarrow \left\lceil \frac{nw}{k} \right\rceil$
      $k_l \leftarrow k - k_h$
      $d_l \leftarrow \left\lfloor \frac{nw}{k} \right\rfloor$
      $t \leftarrow (k_h d_h)_n$
      $\mathbf{M}_h \leftarrow$ RowBalancedMaskMatrix$(n, k_h, d_h, 0)$
      $\mathbf{M}_l \leftarrow$ RowBalancedMaskMatrix$(n, k_l, d_l, t)$
      $\mathbf{M} \leftarrow \begin{bmatrix}\mathbf{M}_h & \mathbf{M}_l\end{bmatrix}$
      return $\mathbf{M}$
  end procedure
```

A proof of the correctness of this algorithm is given in the appendix.

Complex Reed–Solomon Codes

This subsection introduces Reed–Solomon codes defined over the field of complex numbers. For an n^{th} root of unity $\alpha \in \mathbb{C}$, the Reed–Solomon code $\mathsf{RS}[n, f]$ of length

n and dimension f over $\mathbb{C}$ is given by the generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha & \cdots & \alpha^{f-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & \cdots & \alpha^{(n-1)(f-1)} \end{bmatrix}. \tag{6.14}$$

As described in Section 1.2, we can view the code as the image of a set of polynomials when evaluated at powers of α. In particular, the evaluation of the polynomial $t(x) = \sum_{i=0}^{f-1} t_i x^i$ on $\{1, \alpha, \ldots, \alpha^{n-1}\}$ corresponds to

$$\begin{bmatrix} t(1) \\ t(\alpha) \\ \vdots \\ t(\alpha^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha & \cdots & \alpha^{f-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & \cdots & \alpha^{(n-1)(f-1)} \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{f-1} \end{bmatrix}. \tag{6.15}$$

Building $\mathbf{B}$ from $\mathbf{M}$

Once a matrix $\mathbf{M}$ has been determined using Algorithm 3, the encoding matrix $\mathbf{B}$ can be built by picking appropriate codewords from $\mathsf{RS}[n, f]$. Consider $\mathbf{M}$ in (6.9) and the following polynomials

$$t_1(x) = \kappa_1(x - \alpha^6)(x - \alpha^7), \tag{6.16}$$

$$t_2(x) = \kappa_2(x - \alpha^4)(x - \alpha^5), \tag{6.17}$$

$$t_3(x) = \kappa_3(x - \alpha^2)(x - \alpha^3), \tag{6.18}$$

$$t_4(x) = \kappa_4(x - 1)(x - \alpha). \tag{6.19}$$

The constant κ_j is chosen such that the constant term of $t_j(x)$, i.e. $t_j(0)$, is equal to 1. The evaluations of $t_j(x)$ on $\{1, \alpha, \ldots, \alpha^7\}$ are collected in the vector $(t_j(1), t_j(\alpha), \ldots, t_j(\alpha^7))^{\mathsf{t}}$ which sits as the j^{th} column of $\mathbf{B}$ to form

$$\mathbf{B} = \begin{bmatrix} t_1(1) & t_2(1) & t_3(1) & t_4(1) \\ t_1(\alpha) & t_2(\alpha) & t_3(\alpha) & t_4(\alpha) \\ t_1(\alpha^2) & t_2(\alpha^2) & t_3(\alpha^2) & t_4(\alpha^2) \\ t_1(\alpha^3) & t_2(\alpha^3) & t_3(\alpha^3) & t_4(\alpha^3) \\ t_1(\alpha^4) & t_2(\alpha^4) & t_3(\alpha^4) & t_4(\alpha^4) \\ t_1(\alpha^5) & t_2(\alpha^5) & t_3(\alpha^5) & t_4(\alpha^5) \\ t_1(\alpha^6) & t_2(\alpha^6) & t_3(\alpha^6) & t_4(\alpha^6) \\ t_1(\alpha^7) & t_2(\alpha^7) & t_3(\alpha^7) & t_4(\alpha^7) \end{bmatrix} = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & 0 & * \\ * & * & 0 & * \\ * & 0 & * & * \\ * & 0 & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}, \tag{6.20}$$

where the $*$ symbol denotes a nonzero complex number. The generalization of this process now follows. First, we index the rows of $\mathbf{M}$ from 0 to $n-1$. We then fix the polynomial $t_j(x)$ as

$$t_j(x) = \kappa_j \prod_{i:[M]_{i,j}=0} (x - \alpha^i), \tag{6.21}$$

i.e., the polynomial whose roots are those powers of α for which the corresponding entries of $\mathbf{M}_j$ are equal to 0. Lastly, the j^{th} column of $\mathbf{B}$ is set as $\mathbf{B}_j = (t_j(1), t_j(\alpha), \ldots, t_j(\alpha^{n-1}))^{\mathsf{t}}$. This procedure is outlined in the algorithm below.

Algorithm 4 Encoding Matrix $\mathbf{B}$

```
Input:
  $n$: Number of rows
  $k$: Number of columns
  $w$: Row weight
  $\alpha$: $n^{\text{th}}$ root of unity
Output: Row-balanced encoding matrix $\mathbf{B}$.
  procedure EncodingMatrix($n$,$k$,$w$)
      $\mathbf{M} \leftarrow$ MaskMatrix($n, k, w$)
      $\mathbf{B} \leftarrow \mathbf{0}_{n\times k}$
      for $j = 0$ to $k-1$ do
          $t_j(x) \leftarrow \prod_{r:M_{r,i}=0}(x-\alpha^r)/(-\alpha^r)$
          for $i = 0$ to $n-1$ do
              $\mathbf{B}_{i,j} = t_j(\alpha^i)$
          end for
      end for
      return $B$
  end procedure
```

Once the matrix $\mathbf{B}$ is specified, the corresponding decoding vectors required for computing the gradient at the master server have to be characterized.

6.4 Decoding Vectors

In this section, we exploit the fact that $\mathbf{B}$ is constructed using Reed–Solomon codewords and show that each decoding vector $\mathbf{a}_{\mathcal{F}}$ can be computed in $O(f^2)$ time. Recall that the master server should be able to compute the gradient from *any* f surviving machines, indexed by $\mathcal{F} \subseteq [n]$, according to (6.7). The j^{th} column of $\mathbf{B}$

is determined polynomial $t_j(x)$, which is written as

$$t_j(x) = \sum_{i=0}^{f-1} t_{j,i} x^i, \tag{6.22}$$

where $t_{j,0} = 1$. We can write $\mathbf{B}$ as

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha & \cdots & \alpha^{f-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & \cdots & \alpha^{(n-1)(f-1)} \end{bmatrix} \begin{bmatrix} t_{1,0} & \cdots & t_{k,0} \\ t_{1,1} & \cdots & t_{k,1} \\ \vdots & \ddots & \vdots \\ t_{1,f-1} & \cdots & t_{k,f-1} \end{bmatrix}. \tag{6.23}$$

Now consider $\mathbf{C}_{\mathcal{F}}$, the matrix of coded partial gradients received from $\{W_i : i \in \mathcal{F}\}$. The rows of $\mathbf{B}$ corresponding to $\mathcal{F}$ are now given by

$$\mathbf{B}_{\mathcal{F}} = \mathbf{G}_{\mathcal{F}} \mathbf{T} = \begin{bmatrix} 1 & \alpha^{i_1} & \cdots & \alpha^{i_1(f-1)} \\ 1 & \alpha^{i_2} & \cdots & \alpha^{i_2(f-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{i_f} & \cdots & \alpha^{i_f(f-1)} \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \\ t_{1,1} & \cdots & t_{k,1} \\ \vdots & \ddots & \vdots \\ t_{1,f-1} & \cdots & t_{k,f-1}. \end{bmatrix}. \tag{6.24}$$

We require a vector $\mathbf{a}_{\mathcal{F}}$ such that $\mathbf{a}_{\mathcal{F}}^{\mathrm{t}} \mathbf{B}_{\mathcal{F}} = \mathbf{1}^t$. This equivalent to finding a vector $\mathbf{a}_{\mathcal{F}}$ such that

$$\mathbf{a}_{\mathcal{F}}^{\mathrm{t}} \mathbf{G}_{\mathcal{F}} = (1, 0, \ldots, 0). \tag{6.25}$$

Indeed, the matrix in the above product is a Vandermonde matrix is defined by f distinct elements and so it is invertible in $O(f^2)$ time [BP70]. In particular, this allows the master server to compute the required decoding vectors online, a feature that is appealing in scenarios where n is large. Thus, the gradient $\nabla L(\mathcal{D}; \beta)$ can be recovered in the presence of any s stragglers in $O(f^2)$ time. This an improvement compared to previous works [Tan+16] where the decoding time is $O(f^3)$. A careful inspection of inverses of Vandermonde matrices built from an n^{th} root of unity allows us to compute the required decoding vector in a space-efficient manner. This is demonstrated in the next subsection.

Space-Efficient Algorithm

Note that $\mathbf{a}_{\mathcal{F}}^{\mathrm{t}}$ is nothing but the first row of the inverse of $\mathbf{G}_{\mathcal{F}}$, which can be built from a set of polynomials $\{v_1(x), \ldots, v_f(x)\}$. Let the l^{th} column of $\mathbf{G}_{\mathcal{F}}^{-1}$ be $\mathbf{v}_l = (v_{l,0}, \ldots, v_{l,f-1})^{\mathrm{t}}$ and associate it with $v_l(x) = \sum_{i=0}^{f-1} v_{l,0} x^i$. The condition $\mathbf{G}_{\mathcal{F}}^{-1} \mathbf{v}_l = \mathbf{e}_l$, where $\mathbf{e}_l$ is the l^{th} elementary basis vector of length f, implies that

$v_l(x)$ should vanish on $\{\alpha^{i_1}, \ldots, \alpha^{i_f}\} \setminus \{\alpha^{i_l}\}$. Specifically,

$$v_l(x) = \prod_{\substack{j=1 \\ j \neq l}}^{f} \frac{x - \alpha^{i_j}}{\alpha^{i_l} - \alpha^{i_j}}. \tag{6.26}$$

The first row of $\mathbf{G}_{\mathcal{F}}^{-1}$ is given by $(v_{1,0}, \ldots, v_{f,0})$, where $v_{l,0}$ is the constant term of $v_l(x)$. Indeed, we have $v_{l,0} = v_l(0)$, which can be computed in closed form according to the following formula,

$$v_{l,0} = (-1)^{f-1} \prod_{\substack{j=1 \\ j \neq l}}^{f} \frac{\alpha^{i_j}}{\alpha^{i_l} - \alpha^{i_j}} = \prod_{\substack{j=1 \\ j \neq l}}^{f} (1 - \alpha^{i_l - i_j})^{-1}. \tag{6.27}$$

By choosing α as a primitive n^{th} root of unity, one is guaranteed that there are only $n-1$ distinct values of $(1 - \alpha^{i_l - i_j})^{-1}$. This observation proposes that taskmaster should precompute and store those $\{(1 - \alpha^{i})^{-1}\}_{i=1}^{n-1}$, and then compute each $v_{l,0}$ by utilizing lookup operations. The following algorithm outlines this procedure.

Algorithm 5 Compute Decoding Vector

```
Input:
  $\mathcal{F}$: Ordered set of surviving machines - $\{i_1, \ldots i_f\}$
  $\alpha$: $n^{\text{th}}$ root of unity
Output: Decoding vector $a$ associated with $\mathcal{F}$.
  procedure DecodingVector($\mathcal{F}$)
    $\mathbf{a} \leftarrow \mathbf{0}_f$
    for $l = 0$ to $f - 1$ do
      $\mathbf{a}_l \leftarrow \prod_{j=0, j \neq l}^{f-1} (1 - \alpha^{i_l - i_j})^{-1}$
    end for
    return $\mathbf{a}$
  end procedure
```

As described earlier, the assignment $\mathbf{a}_l \leftarrow \prod_{j=0, j \neq l}^{f-1} (1 - \alpha^{i_l - i_j})^{-1}$ now reduces to computing $\{(i_l - i_j)_n\}_{j=0, j \neq l}^{f-1}$ and then performing f lookups to retrieve the constituents of the required product.

6.5 Delay Model

In this section, we provide a theoretical model which can be used to optimize the choice of parameters that define the encoding scheme. For this purpose, we model the response time of a single computing machine as

$$T = T_{\text{delay}} + T_{\text{comp}}. \tag{6.28}$$

Here, the quantity T_{comp} is the time required for the machine to compute its portion of the gradient. This quantity is equal to $c_g \frac{Nw}{k}$, where $c_g = c_g(\ell, p)$ is a constant that indicates the time of computing the gradient for a single data point which depends on the dimension of data points, p, as well as the loss function, ℓ. The second term T_{delay} reflects the random delay incurred before the machine returns with the result of its computation. We model this delay as a Pareto distributed random variable with distribution function

$$F(t) = Pr(T_{\text{delay}} \leq t) = 1 - \left(\frac{t_0}{t}\right)^{\xi} \quad \text{for } \; t \geq t_0, \tag{6.29}$$

where the quantity t_0 can be thought of the fundamental delay of the machine, i.e. the minimum time required for a machine to return in perfect conditions. Previous works [Lee+16; LK14] model the return time of a machine as a shifted exponential random variable. We propose using this approach since the heavy-tailed nature of CPU job runtime has been observed in practice [LO86; Har99; HD97].

Let T_f denotes the expected time of computing the gradient using the first f machines. As a result we have

$$T_f = \mathbb{E}[T_{\text{delay}}^{(f)}] + T_{\text{comp}} + T_{\text{dec}}(f), \tag{6.30}$$

where $T_{\text{delay}}^{(f)}$ is the f^{th} ordered statistic of T_{delay}, and $T_{\text{dec}}(f)$ is the time required at the taskmaster for decoding. Here we assume n is large and define $\alpha := \frac{w}{k}$ as the fraction of the dataset assigned to each machine. The corresponding number of machines for this value of α is given by

$$f(\alpha) = \lceil (1-\alpha)n \rceil + 1. \tag{6.31}$$

We can show the following result which approximates $\mathbb{E}[T_{\text{delay}}^{(f)}]$ for large values of n. For a proof, please refer to the appendix.

Proposition 6.2. *The expected value of the f^{th} order statistic of the Pareto distribution with parameter ξ will converge as n grows, i.e.,*

$$\lim_{n\to\infty} \mathbb{E}[T_{delay}^{(f)}] = \lim_{n\to\infty} \mathbb{E}[T_{delay}^{(1-\alpha)n}] = t_0 \alpha^{-\frac{1}{\xi}}. \tag{6.32}$$

Proof. From [Vän76], the expected value of the f^{th} order statistic of the Pareto distribution is:

$$\mathbb{E}[T_{\text{delay}}^{(f)}] = t_0 \frac{\Gamma(n-f+1-1/\xi)\Gamma(n+1)}{\Gamma(n-f+1)\Gamma(n+1-1/\xi)},$$

where $\Gamma(x)$ is the gamma function given by $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$. We now assume that n is large and make the standard approximation

$$\Gamma(x) \sim \sqrt{\frac{2\pi}{x}}\left(\frac{x}{e}\right)^x .$$

Furthermore, (6.4) implies that the number of machines we wait for is $f = (1-\alpha)n$, for some $\alpha < 1$ which leads to

$$\mathbb{E}[T^{(f)}_{\text{delay}}] = t_0\left(1 - \frac{1}{\xi(\alpha n+1)}\right)^{\alpha n+\frac{1}{2}}\left(1 - \frac{1}{\xi(n+1)}\right)^{-n-\frac{1}{2}}\left(1 - \frac{(1-\alpha)n}{n+1-\frac{1}{\xi}}\right)^{-1/\xi} .$$

By letting $n \to \infty$, the first two terms in the product converge to $e^{-}\xi$ and e^{ξ}, respectively, which yields

$$\lim_{n\to\infty}\mathbb{E}[T^{(f)}_{\text{delay}}] = t_0\alpha^{-\frac{1}{\xi}} .$$

□

Using this result, we can approximate T_f, for $n \gg 1$,

$$T_f \approx t_0\alpha^{-\frac{1}{\xi}} + c_g N\alpha + c_m(1-\alpha)^2n^2, \tag{6.33}$$

where we assume that the taskmaster uses Algorithm 5 for decoding. If we assume c_m is the time required for one FLOP, the total decoding time is given by $c_m(f-1)f \approx c_m(1-\alpha)^2n^2$. Since α is bounded from above by the memory of each machine, one can find the optimal computation time, subject to memory constraints, by minimizing T_f with respect to α.

Offline Decoding

In the schemes where the decoding vectors are computed offline, the quantity T_{dec} does not appear in total computation time T_f. Therefore, for large values of n, we can write:

$$T_f = t_0\alpha^{-\frac{1}{\xi}} + c_g N\alpha. \tag{6.34}$$

This function can be minimized with respect to α by standard calculus to give

$$\alpha^* = \left(\frac{t_0}{c_g N\xi}\right)^{\frac{\xi}{1+\xi}} . \tag{6.35}$$

Note that this quantity is valid (less than one) if and only if one has $\frac{t_0}{c_g N\xi} < 1$. It has been observed in practice that the parameter ξ is close to one. Therefore, this assumption holds because N is assumed to be large. For illustrative purposes, we plot the function T_f from (6.34) for a given set of parameters and indicate the optimal point. This plot is given in Figure 6.2.

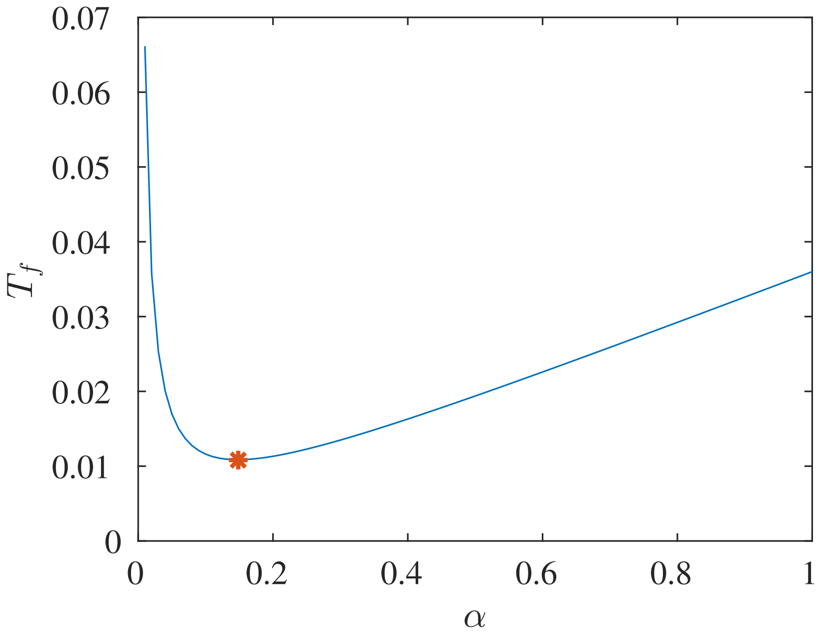


Figure 6.2: This plot shows the expected computation time in a setup where the number of training examples is $N = 12000$ and $c_g = 3 \times 10^{-6}$ to give $Nc_g = 0.035$. The parameters of the Pareto distribution corresponding to the delay are characterized by $t_0 = 0.001$ and $\xi = 1.1$. The optimizer of this function as predicted by (6.35) is $\alpha^* = 0.1477$. This point is denoted by the star symbol.

6.6 Numerical Results

To demonstrate the effectiveness of the scheme, we performed numerical simulations, using MATLAB, of a distributed gradient descent implementation for training a softmax classifier[1]. The model is trained on a cluster composed of $n = 80$ machines to classify 10000 handwritten digits from the MNIST database [LCB98]. To account for the delay, we add a random variable sampled from a Pareto distribution according to (6.29) with parameters $\xi = 1.1$ and $t_0 = 0.001$. We fix $k = n$ and find the optimal w_{RS} for our scheme by numerically optimizing (6.30). The optimal number of machines to wait for is $f_{\text{RS}} = 68$, according to (6.31). The results are aggregated and then decoded using Algorithm 5.

We compare the performance of our scheme with that proposed in [Tan+16]. We

[1] Please refer to the appendix for a quick note on the basics of softmax regression.

find the optimal w_{MDS} for this scheme by setting $T_{\text{dec}} = (1-\alpha)^3 n^3$ since the decoder requires inverting a random matrix corresponding to an MDS matrix. The optimal number of machines to wait for in this scheme is $f_{\text{MDS}} = 33$. The knowledge of the entire gradient allows us to employ accelerated gradient methods such as the one proposed by Nesterov [Nes13]. In summary, we compare the performance of the following four schemes:

- CODED - RS: Each machine gets $\frac{w_{\text{RS}}}{k}$ of the data. Wait for f_{RS} machines.
- CODED - MDS [Tan+16]: Each machine gets $\frac{w_{\text{MDS}}}{k}$ of the data. Wait for f_{MDS} machines.
- WAIT(n): Data is distributed equally amongst n machines - Wait for all n machines to return.
- WAIT(f_{RS}): Data is distributed equally amongst n machines - Wait for f_{RS} machines.
- WAIT(f_{MDS}): Data is distributed equally amongst n machines - Wait for f_{MDS} machines.

We compare several schemes by running each of them on the same dataset for a fixed amount of time (in seconds) and then measuring the test error. The results depicted in Figure 6.3 demonstrate that the scheme proposed in this chapter outperforms the four other schemes.

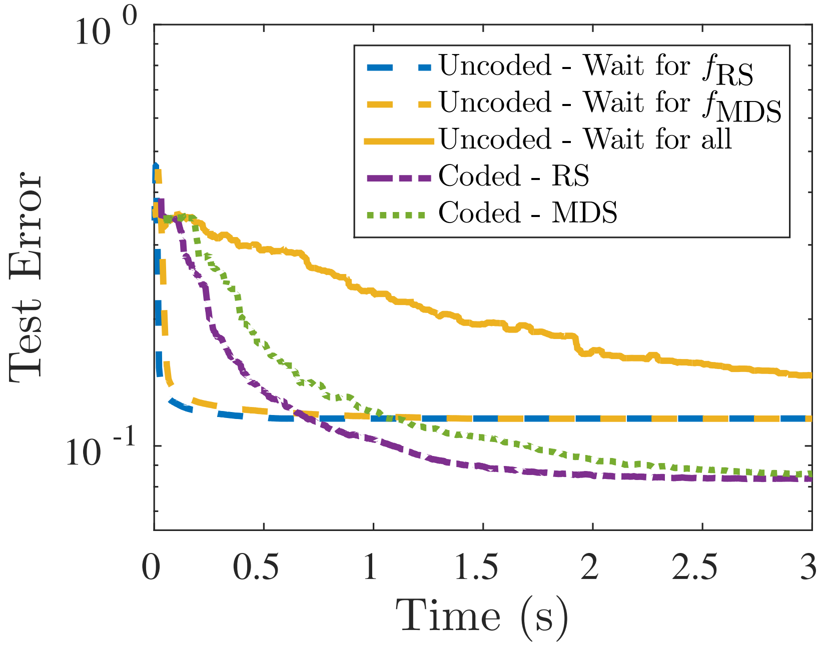


Figure 6.3: This plot shows the test error as a function of time for a softmax regression model trained using distributed gradient descent. The model was trained on $n = 80$ machines using 12000 examples from the MNIST database [LCB98] and was validated on a test set of size 10000. The Reed–Solomon based scheme (Coded - RS) waits for $f_{\text{RS}} = 68$ machines, while the one corresponding to [Tan+16] (Coded - MDS) waits for $f_{\text{MDS}} = 33$. These quantities were obtained by numerically optimizing (6.30). For comparison, we plot the performance of three uncoded schemes: one waits for all 80 machines, another waits for f_{RS} machines and the third waits for f_{MDS} of them.

6.7 Discussion

As mentioned, recent work in coding for coded distributed computation mostly focuses on performing matrix multiplication robustly. Indeed, the work of [Tan+16], and therefore our results here can also handle this case. Suppose we are interested in computing the product $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathbb{C}^{r\times c}$. By partitioning $\mathcal{A}$ along the column dimension, the product can be expressed as

$$\mathbf{y} = \begin{bmatrix} \mathbf{A}_1 & \cdots & \mathbf{A}_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_k \end{bmatrix} = \sum_{i=1}^{k} \mathbf{A}_i \mathbf{x}_i, \tag{6.36}$$

where $\mathbf{A}_i \in \mathbb{C}^{r\times \frac{c}{k}}$. The coding scheme from Algorithm 4 can now be used directly. In particular, machine i is given the data $\{(\mathbf{A}_{i_l}, \mathbf{x}_{i_l})\}_{l=1}^{w}$, where $i_1, \ldots, i_w$ is the support of the i^{th} row of the encoding matrix B. The machine computes the individual products $\mathbf{y}_{i_l} = \mathbf{A}_{i_l}\mathbf{x}_{i_l}$, for a cost of $O(\frac{rc}{k})$ and then linearly combines the $\mathbf{y}_{i_l}$ using the coding coefficients it was provided. As a result, the total computational cost per machine is $O(\frac{rc}{k}w)$. Recovery of the desired product $\mathbf{y}$ is accomplished at the master by computing the required decoding vector using Algorithm 5.

We now compare our scheme to that of Dutta el al. [DCG16], which is somewhat similar in flavor to ours but is specialized to matrix multiplication. For a fixed number of survivors $f > r$, the scheme in [DCG16] transforms the matrix of interest $\mathbf{A} \in \mathbb{R}^{r\times c}$ into a matrix $\mathbf{F} \in \mathbb{R}^{n\times c}$, with the property that any f rows of $\mathbf{F}$ can recover $\mathbf{A}$. In addition, each of its rows is of sparsity $\frac{c(n-(f-r))}{n}$. Machine i is handed the i^{th} row $\mathbf{f}_i^{\text{t}}$ so that it computes $\tilde{y}_i = \mathbf{f}_i^{\text{t}}\mathbf{x}$ for a cost of $O(c - \frac{cf}{n} + \frac{rc}{n})$. The master can now recover the desired product $\mathbf{y}$ from any f elements of $\{\tilde{y}_i\}_{i=1}^{n}$. The condition $f > r$ imposes $n > r$. In the case where this doesn't hold, the authors suggest partitioning $\mathbf{A}$ into k blocks along the row dimension, each of which is composed of $r' < f$ rows. The scheme is applied to each block, $\mathbf{A}_i$, separately and each machine is given k rows, one from each block, which in turn computes k inner products. The sparsity of each row now is $\frac{c(n-(f-r'))}{n}$, so the total computation cost per machine is $O(kc - \frac{kcf}{n} + \frac{rc}{n})$.

For the same number of survivors f, the scheme proposed in this chapter requires $O(rc - \frac{rcf}{n} + \frac{rc}{n})$ operations per machine. The complexities are asymptotically equal if k scales in r. A major advantage of our scheme is the fact that the encoding is oblivious to the the matrix $\mathbf{A}$. This provides huge savings in the encoding process especially when the application is iterative in nature, and the matrix $\mathbf{A}$ is a function of time.

6.8 Appendix

Correctness of Algorithm 2

To lighten notation, we prove correctness for $t = 0$. The general case follows immediately.

Proposition 6.3. *Let k, d and n be integers where $d < n$. The row weights of matrix $\mathbf{M} \in \{0,1\}^{n\times k}$ produced by Algorithm 2 for $t = 0$ are*

$$w_i = \begin{cases} \left\lceil \dfrac{kd}{n} \right\rceil, & i \in \{0, \ldots, (kd-1)_n\}, \\ \left\lfloor \dfrac{kd}{n} \right\rfloor, & i \in \{(kd)_n, \ldots, n-1\}. \end{cases} \tag{6.37}$$

Proof. The nonzero entries in column j of $\mathbf{M}$ are given by

$$\mathcal{S}_j = \{jd, \ldots, (j+1)d-1\}_n, \tag{6.38}$$

where the subscript n denotes reducing the elements of the set modulo n. Collectively, the nonzero indices in all columns are given by

$$\mathcal{S} = \{0, \ldots d-1, \ldots, (k-1)d, \ldots kd-1\}_n. \tag{6.39}$$

In case $n \mid kd$, each element in $\mathcal{S}$, after reducing modulo n, appears the same number of times. As a result, those indices correspond to columns of equal weight, namely $\frac{kd}{n}$. Hence, the two cases of w_i are identical along with their corresponding index sets.

In the case where $n \nmid kd$, each of the first $\left\lfloor \frac{kd}{n} \right\rfloor n$ elements, after reducing modulo n, appears the same number of times. As a result, the nonzero entries corresponding to those indices are distributed evenly amongst the n rows, each of which is of weight $\left\lfloor \frac{kd}{n} \right\rfloor$. The remaining indices $\{\left\lfloor \frac{kd}{n} \right\rfloor n, \ldots, kd-1\}_n$ contribute an additional nonzero entry to their respective rows, those indexed by $\{0, \ldots, (kd-1)_n\}$. Finally, we have that the first $(kd)_n$ rows are of weight $\left\lfloor \frac{kd}{n} \right\rfloor + 1 = \left\lceil \frac{kd}{n} \right\rceil$, while the remaining ones are of weight $\left\lfloor \frac{kd}{n} \right\rfloor$. □

Now consider the case when t is not necessarily equal to zero. This amounts to shifting (cyclically) the entries in each column by t positions downwards. As a result, the rows themselves are shifted by the same amount allowing us to conclude the following.

Corollary 6.1. *Let k, d and n be integers where $d < n$. The row weights of matrix $M \in \{0,1\}^{n \times k}$ produced by Algorithm 2 are*

$$w_i = \begin{cases} \left\lceil \dfrac{kd}{n} \right\rceil & i \in \{t, \ldots, (t+kd-1)_n\}, \\ \left\lfloor \dfrac{kd}{n} \right\rfloor & i \in \{0, t-1\} \cup \{(t+kd)_n, \ldots, n-1\}. \end{cases} \tag{6.40}$$

Correctness of Algorithm 3

According to the algorithm, the condition $k \mid nw$ implies that $k_h = 0$ leading to $M = M_l$, which is constructed using Algorithm 2.

Moving on to the general case, the matrix $\mathbf{M}$ given by

$$\begin{bmatrix} \mathbf{M}_h & \mathbf{M}_l \end{bmatrix}, \tag{6.41}$$

where each matrix is row-balanced. The particular choice of t in M_l aligns the "heavy" rows of M_h with the "light" rows of M_l, and vice-versa. The algorithm works because the choice of parameters equates the number of heavy rows n_h of M_l to the number of light rows n_l of M_h. The following lemma is useful in two ways.

Lemma 6.1. $\left\lfloor \frac{k_h d_h}{n} \right\rfloor + \left\lceil \frac{k_l d_l}{n} \right\rceil = \left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor = w.$

Proof. Note that the following holds:

$$\frac{k_h d_h}{n} + \frac{k_l d_l}{n} - 1 < \left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor < \frac{k_h d_h}{n} + \frac{k_l d_l}{n} + 1. \tag{6.42}$$

Furthermore, we have that

$$\begin{aligned} \frac{k_h d_h}{n} + \frac{k_l d_l}{n} &= \frac{k_h(d_l+1)}{n} + \frac{(k-k_h)d_l}{n} && (6.43)\\ &= \frac{k_h}{n} + \frac{kd_l}{n} && (6.44)\\ &= \frac{wn - \left\lfloor \frac{wn}{k} \right\rfloor k}{n} + \frac{kd_l}{n} && (6.45)\\ &= \frac{wn - d_l k}{n} + \frac{kd_l}{n} && (6.46)\\ &= w. && (6.47) \end{aligned}$$

We combine the two observations in one:

$$w - 1 < \left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor < w + 1, \tag{6.48}$$

and conclude that $\left\lceil \frac{k_h d_h}{n} \right\rceil + \left\lfloor \frac{k_l d_l}{n} \right\rfloor = w$. □

We have shown the concatenation of a "heavy" row of M_h along with a "light" row of M_l results in one that is of weight w. It remains to show that the concatenation of M_h and M_l results in rows of this type only.

We will assume that $n \nmid k_h d_h$ holds. From Proposition 6.3, we have $n_l = n - (k_h d_h)_n$ and $n_h = (k_l d_l)_n$. We will show that the two quantities are in fact equal. Indeed, we can express n_l as

$$\begin{aligned} n-(k_hd_h)_n &= n-k_hd_h+\left\lfloor\frac{k_hd_h}{n}\right\rfloor n && (6.49)\\ &= -k_hd_h+\left\lceil\frac{k_hd_h}{n}\right\rceil n && (6.50)\\ &= -(k-k_l)(d_l+1)+\left\lceil\frac{k_hd_h}{n}\right\rceil n && (6.51)\\ &= -(d_lk+k_h)+k_ld_l-\left\lfloor\frac{k_ld_l}{n}\right\rfloor n+nw && (6.52)\\ &= k_ld_l-\left\lfloor\frac{k_ld_l}{n}\right\rfloor n && (6.53)\\ &= (k_ld_l)_n. && (6.54) \end{aligned}$$

Hence $n_l = n_h$ and by the choice of t, the “light" rows of $\mathbf{M}_h$ align with the “heavy" rows of $\mathbf{M}_l$, and vice-versa. Furthermore, Lemma 6.1 guarantees that each row of $\mathbf{M}$ is of weight $\lceil\frac{k_hd_h}{n}\rceil+\lfloor\frac{k_ld_l}{n}\rfloor = w$. The same holds for the remaining rows, using the fact that $\lceil x\rceil+\lfloor y\rfloor = \lfloor x\rfloor+\lceil y\rceil$ when both x and y are non-integers.

Softmax Regression

Softmax regression is a technique used to train a classifier that assigns a label $y \in \{1,\ldots,K\}$ to each feature vector $x \in \mathbb{R}^p$. Th classifier accomplishes this by estimating the probability $P(y=i|x)$ for $i=1,\ldots,K$. In this case, we assume that the parameter $\beta \in \mathbb{R}^{p\times K}$ can be written as

$$\beta = \begin{bmatrix} | & & | \\ \beta_1 & \cdots & \beta_K \\ | & & | \end{bmatrix},$$

where each vector $\beta_i \in \mathbb{R}^p$ corresponds to the label i. We assume that $P(y=i|x,\beta)$ takes the form

$$P(y=i|x,\beta) \propto e^{-\beta_i^{\mathsf{t}}x},$$

which allows us to form the probability vector

$$P(x;\beta) = \begin{bmatrix} P(y=1|x,\beta) \\ \vdots \\ P(y=K|x,\beta) \end{bmatrix} = c\begin{bmatrix} e^{-\beta_1^{\mathsf{t}}x} \\ \vdots \\ e^{-\beta_K^{\mathsf{t}}x} \end{bmatrix}.$$

The quantity $c = (\sum_{j=1}^{K} e^{-\beta_j^{\mathsf{t}} x})^{-1}$ is a normalization constant. For a specific training example (x, y), the loss is given by

$$\begin{aligned} \ell(x, y; \beta) &= -\log \frac{e^{-\beta_y^{\mathsf{t}} x}}{\sum_{j=1}^{K} e^{-\beta_j^{\mathsf{t}} x}} \\ &= -\sum_{i=1}^{K} \mathbb{1}[y = i] \log \frac{e^{-\beta_i^{\mathsf{t}} x}}{\sum_{j=1}^{K} e^{-\beta_j^{\mathsf{t}} x}}, \end{aligned}$$

where $\mathbb{1}[\cdot]$ is the indicator function that returns 1 when its argument is true and 0 otherwise. Hence, for a given data set of examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, the loss function is given by,

$$L(\mathcal{D}; \beta) = -\sum_{(x,y)\in\mathcal{D}} \sum_{i=1}^{K} \mathbb{1}[y = i] \log \frac{e^{-\beta_i^{\mathsf{t}} x}}{\sum_{j=1}^{K} e^{-\beta_j^{\mathsf{t}} x}}.$$

The gradient with respect to each β_i is given by

$$\nabla_{\beta_i} L(\mathcal{D}; \beta) = -\sum_{(x,y)\in\mathcal{D}} x \left(\mathbb{1}[y = i] - \frac{e^{-\beta_i^{\mathsf{t}} x}}{\sum_{j=1}^{K} e^{-\beta_j^{\mathsf{t}} x}} \right).$$

These partial gradients are collected in a matrix as

$$\nabla L(\mathcal{D}; \beta) = \begin{bmatrix} | & & | \\ \nabla_{\beta_1} L(\mathcal{D}; \beta) & \cdots & \nabla_{\beta_K} L(\mathcal{D}; \beta) \\ | & & | \end{bmatrix},$$

which can be used to update the model β at time $t+1$ as $\beta_{t+1} = \beta_t - \eta_t \nabla L(\mathcal{D}; \beta_t)$, where η_t is the learning rate at time t.

Chapter 7

CONCLUDING REMARKS AND FUTURE DIRECTIONS

The main goal of this thesis was to present code constructions with structured generator matrices. In chapters 2, 3 and 4, the structure was imposed by the underlying network in the form of encoding constraints. In Chapter 5, it was shown that imposing certain structure results in an encoding scheme that is computationally balanced. In Chapter 6, the same balanced structure resulted in an optimal distribution of the subtasks running on a computing cluster suffering from straggling machines. In the sequel, we lay out some future directions that we think are natural extensions of the results that were presented.

Network Error Correction

Chapters 2 and 3 described distributed Reed–Solomon codes and distributed Gabidulin codes, respectively, as error-correcting codes suited for multi-source multicast networks. The codes presented are distributed in the sense that the source nodes operate independently of one another. Furthermore, they are efficient since they operate over small finite fields and can be decoded using classical polynomial-time algorithms. While they are capacity-achieving, the constructions provided are specific to networks with three messages and it is of extreme interest to generalize the results to handle an arbitrary number of messages.

Such generalization will allow us to construct an optimal code (in terms of minimum distance) when subjected to encoding constraints. In particular, Theorem 4.2 from Chapter 4 provided an upper bound on the minimum distance of any code with specified encoding constraints, and it was shown that, under a technical assumption, this bound is achievable by subcodes of Reed–Solomon codes.

While the solutions presented in chapters 2, 3 and 4 rely on different approaches, a unified framework would probably rely on the algebraic construction of Section 2.3, where it was shown that certain subcodes of Reed–Solomon codes can be decomposed into well-behaved constituents, again when certain technical assumptions hold. The problem was then shown to be equivalent to an allocation problem of basis vectors of these constituents to each of the source nodes in the network. We also remark that any unification of the construction would probably rely on existence (rather than constructive) arguments as we have seen that the choice of the underlying code matters. In particular, sections 2.3 and 3.5 show how the underlying code, through its coordinates, has to be chosen carefully; the authors of [Dau+15a] have taken steps in that direction.

Data Storage

In Chapter 4, minimum distance bounds were presented for codes that obey encoding constraints. The bounds are a function of the bipartite graph that specified the encoding constraints of every code symbol. A natural extension of that result is one that incorporates the notion of locality [Gop+12]. In particular, a locality-aware family of bounds similar in flavor to (4.4) can be of use in practical scenarios. This, in effect, leads to a characterization of the region of minimum distance - locality pairs (d, r) that can possibly be attained by an error-correcting code subject to encoding constraints determined by a given bipartite graph. From there, one would hope for a code construction that achieves every point in this region, and it would be interesting to see if the Tamo–Barg construction is one such family.

Another result presented was the systematic minimum distance bound of Section 4.5. There, a code specified by the encoding constraints is required to be systematic and it was shown that those can lead to a strict loss in the minimum distance when compared to that nonsystematic case. The construction presented to achieve the systematic bound requires the enumeration of a set of matchings for the underlying graph, whose cardinality is exponential in the code length. As a result, this construction, along with computing the systematic minimum distance of a given graph, requires exponential complexity. One would hope for an efficient algorithm that returns the optimal matching, which, in effect, results in an efficient construction of systematic codes subject to encoding constraints.

We mention two points regarding the results presented of Chapter 5, where it was shown that both Reed–Solomon codes and Tamo–Barg codes possess generator matrices that are balanced, i.e. for a fixed row weight, the number of nonzero entries is the same across the columns. The balanced construction pertinent to Reed–Solomon codes can accommodate any required row weight, in particular rows corresponding to minimum distance codewords which resulted in a sparsest and balanced generator matrix. The situation is slightly more brittle when Tamo–Barg (TB) codes are considered, where it was only possible to realize row weights that are multiples of $r + 1$, the size of each local group. From both practical and theoretical perspectives, it remains a priority to present sparsest and balanced generator matrices for these codes. We also mention constructions for codes that are not necessarily cyclic, TB codes defined using additive subgroups and locally recoverable codes with multiple recovery sets.

Distributed Computation

Mitigating stragglers in computing clusters by resorting to coding-theoretic based schemes is a young paradigm. The setup we have studied in Chapter 6 considers a distributed gradient descent implementation in which the full gradient is to be recovered from a subset of the machines that can change from iteration to iteration as long as their number f is fixed.

A natural scheme to ask for is one that provides an approximation of the gradient when the number of machines that return is strictly less than f. For any scheme which was designed for a target f, the approximation is known and can be computed as follows. Suppose that the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ is partitioned into k disjoint sets $\mathcal{D}_1, \dots, \mathcal{D}_k$, each with gradient $\mathbf{g}_i \in \mathbb{R}^{1\times p}$ that are collected in the matrix

$$\mathbf{g} = \begin{bmatrix} - & \mathbf{g}_1 & - \\ & \cdots & \\ - & \mathbf{g}_k & - \end{bmatrix}.$$

Let the encoding matrix be $\mathbf{B} \in \mathbb{C}^{n\times k}$ which is designed so that the rowspace of any f rows contains $\mathbf{1}^{\mathsf{t}}$, the all-one vector of length k, where the number of machines in the cluster is n. Now suppose that the coded data returned from $f' = f - \delta$ machines is used to approximate the gradient and the rows of $\mathbf{B}$ corresponding to these machines are given by $\mathbf{B}_\delta$. One can compute the vector $\mathbf{v}^\star$ such that it selects an element of the rowspace of $\mathbf{B}_\delta$ that is closest to $\mathbf{1}$, which approximates the gradient by $\mathbf{v}^\star\mathbf{g}$ (as opposed to $\mathbf{1}^{\mathsf{t}}\mathbf{g}$). This vector $\mathbf{v}^\star$ can be found through the following optimization problem,

$$\mathbf{v}^\star = \underset{\mathbf{v}}{\operatorname{argmin}} ||\mathbf{v}\mathbf{B}_\delta - \mathbf{1}||_2,$$

whose optimizer is given by the standard least squares solution $\mathbf{v}^\star = \mathbf{1}^{\mathsf{t}}\mathbf{B}_\delta^{\mathsf{t}}(\mathbf{B}_\delta\mathbf{B}_\delta^{\mathsf{t}})^{-1}$, when $k > f$. This formulation inspires an optimzation over the choice of encoding coefficients, i.e. the matrix $\mathbf{B}$. In principle the following optimization problem allows for the construction of a scheme that recovers the best approximation of the gradient when $f - \delta$ machines return, over all schemes $\mathcal{B}$ which allow the exact recovery of the gradient from any f machines.

$$\mathbf{B}^{\star} = \underset{\mathbf{B}\in\mathcal{B}}{\operatorname{argmin}} \quad \max_{\mathbf{B}_\delta} \quad \min_{\mathbf{v}} ||\mathbf{v}\mathbf{B}_\delta - \mathbf{1}^{\mathsf{t}}||_2.$$

We have seen that the innermost optimization can be performed analytically. The maximization characterizes a worst case behavior of the scheme over all $(f-\delta)$-sized subsets of the n machines, which one would hope is equivalent over these subsets. Note that this optimization problem is for a fixed δ, and a desirable property of an optimal scheme is that it is optimal for any δ. Natural starting points for this approach is to characterize the performance of the code construction based on random MDS codes presented in [Tan+16], and the Reed–Solomon-based scheme presented in Chapter 6 when are they used to compute approximations of the gradient.

BIBLIOGRAPHY

[Ahl+00] Rudolf Ahlswede et al. “Network information flow”. In: *IEEE Transactions on information theory* 46.4 (2000), pp. 1204–1216.

[Ama] Amazon. *Amazon Glacier*. URL: `https://aws.amazon.com/glacier/`.

[Apaa] Apache. *Apache Hadoop*. URL: `http://hadoop.apache.org/`.

[Apab] Apache. *Apache Spark*. URL: `http://spark.apache.org/`.

[BBV96] Mario Blaum, Jehoshua Bruck, and Alexander Vardy. “MDS array codes with independent parity symbols”. In: *IEEE Transactions on Information Theory* 42.2 (1996), pp. 529–542.

[Bla+95] Mario Blaum et al. “Evenodd: an efficient scheme for tolerating double disk failures in raid architectures”. In: *IEEE Transactions on Computers* 44.2 (1995), pp. 192–202.

[Boy+11] Stephen Boyd et al. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Foundations and Trends® in Machine Learning* 3.1 (2011), pp. 1–122.

[BP70] Ake Björck and Victor Pereyra. “Solution of Vandermonde systems of equations”. In: *Mathematics of Computation* 24.112 (1970), pp. 893–903.

[Clo] Cloudera. *Introduction to HDFS Erasure Coding in Apache Hadoop*. URL: `https://blog.cloudera.com/blog/2015/09/introduction-to-hdfs-erasure-coding-in-apache-hadoop/`.

[Dau+13] Son Hoang Dau et al. “Balanced Sparsest generator matrices for MDS codes”. In: *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*. 2013, pp. 1889–1893.

[Dau+15a] Son Hoang Dau et al. “Constructions of MDS Codes via Random Vandermonde and Cauchy Matrices over Small Fields”. In: *Communication, Control, and Computing (Allerton), 2015 53nd Annual Allerton Conference on*. 2015.

[Dau+15b] Son Hoang Dau et al. “Locally encodable and decodable codes for distributed storage systems”. In: *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE. 2015, pp. 1–7.

[DB13] Jeffrey Dean and Luiz André Barroso. “The Tail at Scale”. In: *Communications of the ACM* 56.2 (2013), p. 74.

[DCG16] Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover. “Short-Dot: Computing Large Linear Transforms Distributedly Using Coded Short Dot Products”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 2092–2100.

[Del78] Ph Delsarte. “Bilinear forms over a finite field, with applications to coding theory”. In: *Journal of Combinatorial Theory, Series A* 25.3 (1978), pp. 226–241.

[Dik+10] Theodoros K. Dikaliotis et al. “Multiple access network information-flow and correction codes”. In: *Information Theory, IEEE Transactions on* 57.2 (2010), pp. 1067–1079.

[DPR06] Alexandros G. Dimakis, Vinod Prabhakaran, and Kannan Ramchandran. “Decentralized Erasure Codes for Distributed Networked Storage”. In: *IEEE/ACM Transactions on Networking (TON) - Special issue on networking and information theory* 52.6 (2006), pp. 2809–2816. arXiv: `0606049 [cs.IT]`.

[DSY14] Son Hoang Dau, Wentu Song, and Chau Yuen. “On the existence of MDS codes over small fields with constrained generator matrices”. In: *Information Theory (ISIT), 2014 IEEE International Symposium on*. June 2014, pp. 1787–1791.

[DSY15] Son Hoang Dau, Wentu Song, and Chau Yuen. “On Simple Multiple Access Networks”. In: *IEEE Journal on Selected Areas in Communications* 8716.0733 (2015).

[Fac] Facebook. *Facebook’s Cold Storage System*. URL: `https://code.facebook.com/posts/1433093613662262/-under-the-hood-facebook-s-cold-storage-system-/`.

[FF56] Lester R Ford and Delbert R Fulkerson. “Maximal flow through a network”. In: *Canadian journal of Mathematics* 8.3 (1956), pp. 399–404.

[Gab85] Ernest Mukhamedovich Gabidulin. “Theory of codes with maximum rank distance”. In: *Problemy Peredachi Informatsii* 21.1 (1985), pp. 3–16.

[Gem+11] Rainer Gemulla et al. “Large-scale matrix factorization with distributed stochastic gradient descent”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, pp. 69–77.

[Goo] Google. *Google’s Archival Cloud Storage*. URL: `https://cloud.google.com/storage/archival/`.

[Gop+12] Parikshit Gopalan et al. “On the Locality of Codeword Symbols”. In: *IEEE Transactions on Information Theory* 58.11 (Nov. 2012), pp. 6925–6934.

[GS99] Venkatesan Guruswami and Madhu Sudan. "Improved decoding of Reed-Solomon and algebraic-geometry codes". In: *IEEE Transactions on Information Theory* 45.6 (1999), pp. 1757–1767.

[GW15] Venkatesan Guruswami and Mary Wootters. "Repairing Reed-Solomon Codes". In: *arXiv preprint arXiv:1509.04764* (2015).

[Hal+14] Wael Halbawi et al. "Distributed reed-solomon codes for simple multiple access networks". In: *2014 IEEE International Symposium on Information Theory*. IEEE, June 2014, pp. 651–655.

[Hal+17] Wael Halbawi et al. "Balanced and Sparse Tamo–Barg Codes". In: *2017 IEEE International Symposium on Information Theory (ISIT)*. 2017.

[Ham47] Richard Hamming. "Self-Correcting Codes - Case 20878". In: *Bell Telephone Laboratories - Memorandum 1130-RWH-MFW* (July 1947).

[Ham48] Richard Hamming. "A Theory of Self-checking and Self-correcting Codes - Case 20878". In: *Bell Telephone Laboratories - Memorandum 48-110-31* (June 1948).

[Ham50] Richard W Hamming. "Error detecting and error correcting codes". In: *Bell Labs Technical Journal* 29.2 (1950), pp. 147–160.

[Har99] Mor Harchol-Balter. "The Effect of Heavy-Tailed Job Size Distributions on Computer System Design." In: *Proc. of ASA-IMS Conf. on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics*. 1999.

[HCL07] Cheng Huang, Minghua Chen, and Jin Li. "Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems". In: *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*. IEEE, July 2007, pp. 79–86.

[HD97] Mor Harchol-Balter and Allen B Downey. "Exploiting process lifetime distributions for dynamic load balancing". In: *ACM Transactions on Computer Systems (TOCS)* 15.3 (1997), pp. 253–285.

[HHD14] Wael Halbawi, Tracey Ho, and Iwan Duursma. "Distributed gabidulin codes for multiple-source network error correction". In: *2014 International Symposium on Network Coding (NetCod)*. IEEE, June 2014, pp. 1–6.

[HLH16a] Wael Halbawi, Zihan Liu, and Babak Hassibi. "Balanced Reed-Solomon codes". In: *2016 IEEE International Symposium on Information Theory (ISIT)*. July 2016, pp. 935–939.

[HLH16b] Wael Halbawi, Zihan Liu, and Babak Hassibi. "Balanced Reed-Solomon codes for all parameters". In: *2016 IEEE Information Theory Workshop (ITW)*. Sept. 2016, pp. 409–413.

[Ho+06] T. Ho et al. "A Random Linear Network Coding Approach to Multicast". In: *IEEE Transactions on Information Theory* 52.10 (Oct. 2006), pp. 4413–4430.

[HT72] Carlos R P Hartmann and Kenneth K Tzeng. "Generalizations of the BCH bound". In: *Information and control* 20.5 (1972), pp. 489–498.

[HTH15] Wael Halbawi, Matthew Thill, and Babak Hassibi. "Coding with constraints: Minimum distance bounds and systematic constructions". In: *Information Theory (ISIT), 2015 IEEE International Symposium on*. June 2015, pp. 1302–1306.

[Jag+07] Sidharth Jaggi et al. "Resilient network coding in the presence of byzantine adversaries". In: *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE. 2007, pp. 616–624.

[JFD09] Mahdi Jafari, Christina Fragouli, and Suhas Diggavi. "Code Construction for Multiple Sources Network Coding". In: *Proceedings of the 2009 MobiHoc S3 Workshop on MobiHoc S3*. MobiHoc S3 '09. New York, NY, USA: ACM, 2009, pp. 21–24.

[Kam+14] G M Kamath et al. "Codes With Local Regeneration and Erasure Correction". In: *IEEE Transactions on Information Theory* 60.8 (Aug. 2014), pp. 4637–4660.

[KK08] Ralf Koetter and Frank R. Kschischang. "Coding for Errors and Erasures in Random Network Coding". In: *IEEE Transactions on Information Theory* 54.8 (Aug. 2008), pp. 3579–3591.

[LCB98] Yann LeCun, Corinna Cortes, and Christopher J C Burges. *The MNIST database of handwritten digits*. 1998.

[Lee+16] Kangwook Lee et al. "Speeding up distributed machine learning using codes". In: *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 1143–1147.

[LK14] Guanfeng Liang and Ulas C Kozat. "TOFEC: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes". In: *INFOCOM, 2014 Proceedings IEEE*. IEEE. 2014, pp. 826–834.

[LMA16a] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. "A unified coding framework for distributed computing with straggling servers". In: *arXiv preprint arXiv:1609.01690* (2016).

[LMA16b] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. "Fundamental tradeoff between computation and communication in distributed computing". In: *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 1814–1818.

[LN97] R Lidl and H Niederreiter. *Finite Fields*. Encyclopedia of Mathematics and its Applications v. 20, pt. 1. Cambridge University Press, 1997.

[LO86] Will Leland and Teunis J Ott. *Load-balancing heuristics and process behavior*. Vol. 14. 1. ACM, 1986.

[Loi06] Pierre Loidreau. “A Welch–Berlekamp Like Algorithm for Decoding Gabidulin Codes”. In: *Coding and Cryptography: International Workshop, WCC 2005, Bergen, Norway, March 14-18, 2005. Revised Selected Papers*. Ed. by Øyvind Ytrehus. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 36–45.

[LYC03] S-YR Li, Raymond W Yeung, and Ning Cai. “Linear network coding”. In: *IEEE transactions on information theory* 49.2 (2003), pp. 371–381.

[Mas69] J Massey. “Shift-register synthesis and BCH decoding”. In: *IEEE Transactions on Information Theory* 15.1 (1969), pp. 122–127.

[McE02] Robert J. McEliece. *The Theory of Information and Coding*. Cambridge University Press, 2002.

[McE86] Robert J. McEliece. *Finite Fields for Computer Scientists and Engineers*. The Springer International Series in Engineering and Computer Science. Springer, 1986.

[Moh+] Soheil Mohajer et al. “On the capacity of multisource non-coherent network coding”. In: *Networking and Information Theory, 2009. ITW 2009. IEEE Information Theory Workshop on*. IEEE, pp. 130–134.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*. Elsevier, 1977.

[Mur+14] Subramanian Muralidhar et al. “f4: Facebook’s Warm BLOB Storage System”. In: *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. 2014, pp. 383–398.

[Nes13] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2013.

[OD11] Frédérique Oggier and Anwitaman Datta. “Self-repairing homomorphic codes for distributed storage systems”. In: *Proceedings - IEEE INFOCOM* (2011), pp. 1215–1223. arXiv: `1008.0064`.

[PD12] Dimitris S. Papailiopoulos and Alexandros G. Dimakis. “Locally repairable codes”. In: *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE, July 2012, pp. 2771–2775.

[PGK88] David A Patterson, Garth Gibson, and Randy H Katz. “A Case for Redundant Arrays of Inexpensive Disks (RAID)”. In: *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’88. New York, NY, USA: ACM, 1988, pp. 109–116.

[Ras+09] K V Rashmi et al. "Explicit construction of optimal exact regenerating codes for distributed storage". In: *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE. 2009, pp. 1243–1249.

[Raw+12] Ankit Singh Rawat et al. "Optimal Locally Repairable and Secure Codes for Distributed Storage Systems". In: (Oct. 2012). arXiv: 1210.6954.

[Rec+11] Benjamin Recht et al. "Hogwild: A lock-free approach to parallelizing stochastic gradient descent". In: *Advances in Neural Information Processing Systems*. 2011, pp. 693–701.

[Rei+17] Amirhossein Reisizadehmobarakeh et al. "Coded computation over heterogeneous clusters". In: *arXiv preprint arXiv:1701.05973* (2017).

[Roo82] Cees Roos. "A generalization of the BCH bound for cyclic codes, including the Hartmann-Tzeng bound". In: *Journal of Combinatorial Theory, Series A* 33.2 (1982), pp. 229–232.

[Roo83] Cornelis Roos. "A new lower bound for the minimum distance of a cyclic code". In: *IEEE Transactions on Information Theory* 29.3 (1983), pp. 330–332.

[RS60] Irving Reed and Gus Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304.

[RSK11] KV Rashmi, Nihar B Shah, and P Vijay Kumar. "Enabling node repair in any erasure code for distributed storage". In: *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 1235–1239.

[Sat+13] Maheswaran Sathiamoorthy et al. "XORing Elephants: Novel Erasure Codes for Big Data". In: *39th international conference on Very Large Data Bases, VLDB* 6.5 (2013), pp. 325–336. arXiv: 1301.3791.

[SFD08] Mahdi Jafari Siavoshani, Christina Fragouli, and Suhas Diggavi. "Noncoherent multisource network coding". In: *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. IEEE. 2008, pp. 817–821.

[Sha+12a] N B Shah et al. "Distributed Storage Codes With Repair-by-Transfer and Nonachievability of Interior Points on the Storage-Bandwidth Tradeoff". In: *IEEE Transactions on Information Theory* 58.3 (2012), pp. 1837–1852.

[Sha+12b] Nihar B Shah et al. "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions". In: *IEEE Transactions on Information Theory* 58.4 (2012), pp. 2134–2158.

[Sha48] C. E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. URL: http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[Sil+13] Natalia Silberstein et al. "Optimal locally repairable codes via rank-metric codes". In: *IEEE International Symposium on Information Theory - Proceedings*. 2013, pp. 1819–1823. arXiv: 1312.3194.

[SK09a] Danilo Silva and Frank R Kschischang. "Fast encoding and decoding of Gabidulin codes". In: *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. IEEE. 2009, pp. 2858–2862.

[SK09b] Danilo Silva and Frank R Kschischang. "On metrics for error correction in network coding". In: *IEEE Transactions on Information Theory* 55.12 (2009), pp. 5479–5490.

[SKK08] Danilo Silva, Frank R. Kschischang, and Ralf Koetter. "A Rank-Metric Approach to Error Control in Random Network Coding". In: *IEEE Transactions on Information Theory* 54.9 (Sept. 2008), pp. 3951–3967.

[Tan+16] Rashish Tandon et al. "Gradient Coding". In: *arXiv preprint arXiv:1612.03301* (2016). arXiv: 1612.03301.

[TB14] Itzhak Tamo and Alexander Barg. "A family of optimal locally recoverable codes". In: *IEEE International Symposium on Information Theory - Proceedings*. Vol. 60. 8. IEEE, Aug. 2014, pp. 686–690. arXiv: 1311.3284.

[TB15] Itzhak Tamo and Alexander Barg. "Cyclic LRC codes and their subfield subcodes". In: *Information Theory (ISIT), IEEE International Symposium on* (June 2015), pp. 1262–1266. arXiv: 1502.01414v1.

[TPD13] Itzhak Tamo, Dimitris S. Papailiopoulos, and Alexandros G. Dimakis. "Optimal locally repairable codes and connections to matroid theory". In: *2013 IEEE International Symposium on Information Theory*. IEEE, July 2013, pp. 1814–1818.

[Vän76] Kerstin Vännman. "Estimators based on order statistics from a Pareto distribution". In: *Journal of the American Statistical Association* 71.355 (1976), pp. 704–708.

[VW11] Jacobus Hendricus Van Lint and Richard Michael Wilson. *A Course in Combinatorics*. Cambridge University Press, 2011.

[WAS13] Antonia Wachter-Zeh, Valentin Afanassiev, and Vladimir Sidorenko. "Fast decoding of Gabidulin codes". In: *Designs, Codes and Cryptography* 66.1 (2013), pp. 57–73.

[WB86] Lloyd Welch and Elywn Berlekamp. *Error Correction for Algebraic Codes*. 1986.

[WJW14] Da Wang, Gauri Joshi, and Gregory Wornell. “Efficient task replication for fast response times in parallel computation”. In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 42. 1. ACM. 2014, pp. 599–600.

[WJW15] Da Wang, Gauri Joshi, and Gregory Wornell. “Using straggler replication to reduce latency in large-scale parallel computing (extended version)”. In: *arXiv preprint arXiv:1503.03128* (2015).

[YC06] Raymond W Yeung and Ning Cai. “Network error correction, I: Basic concepts and upper bounds”. In: *Communications in Information & Systems* 6.1 (2006), pp. 19–35.

[YHN11] Hongyi Yao, Tracey Ho, and Cristina Nita-Rotaru. “Key agreement for wireless networks in the presence of active adversaries”. In: *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)* (Nov. 2011), pp. 792–796.

[YS11] Muxi Yan and A Sprintson. “Weakly Secure Network Coding for Wireless Cooperative Data Exchange”. In: *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. 2011, pp. 1–5.

[YSZ14] Muxi Yan, Alex Sprintson, and Igor Zelenko. “Weakly Secure Data Exchange with Generalized Reed Solomon Codes”. In: 2014, pp. 1366–1370.

[Zin+10] Martin Zinkevich et al. “Parallelized stochastic gradient descent”. In: *Advances in neural information processing systems*. 2010, pp. 2595–2603.