Rethinking the Minimum Distance: Channels With Varying Sampling Rate and Iterative Decoding of LDPC Codes

by

Lara Dolecek

B.S. (University of California, Berkeley) 1999
M.S. (University of California, Berkeley) 2004
M.A. (University of California, Berkeley) 2007

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

in

Engineering-Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor Venkat Anantharam, Chair
Professor Borivoje Nikolić
Professor David Aldous

Fall 2007

The dissertation of Lara Dolecek is approved:

Chair Date

Date

Date

University of California, Berkeley

Fall 2007

Rethinking the Minimum Distance: Channels With Varying Sampling Rate and Iterative Decoding of LDPC Codes

Copyright 2007

by

Lara Dolecek

Abstract

Rethinking the Minimum Distance: Channels With Varying Sampling Rate and Iterative Decoding of LDPC Codes

by

Lara Dolecek

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Venkat Anantharam, Chair

In this dissertation we develop novel coding theoretic approaches for two problems relevant to modern communication systems. In the first part of the thesis we address the issue of reliable communication under varying sampling rate, while in the second part we focus on the analytic understanding of the performance of low density parity check (LDPC) codes in the low bit error rate (BER) region. The underlying theme in both of these somewhat non-standard yet relevant problems is that the notion of a fundamental performance metric, typically taken to be the minimum distance of an additive error correcting code, needs to be rethought when the standard assumptions on the communication no longer hold.

In particular, in the first part of the thesis we investigate the problem of overcoming synchronization errors from a coding theoretic perspective when the timing recovery is

inadequate. This is in contrast to the traditional coding theory which typically takes the assumption of perfect synchronization for granted and is thus practically exclusively concerned with problems of additive error correction.

We study first order Reed-Muller codes as a representative example of a class of highly structured additive error correcting codes with good minimum distance properties, and investigate their behavior in the presence of both additive errors and a synchronization error. We propose a method to systematically thin Reed-Muller codes, such that the resulting thinned code is immune to additive errors as well as a synchronization error. This systematic analysis is based on first establishing several novel run-length properties of these codes.

In addition, we propose and study number theoretic constructions of sets of strings immune to multiple repetitions. These constructions are also shown to have good cardinalities. We then use these number theoretic constructions to develop a prefixing-based method to improve the immunity of an arbitrary code (a collection of binary strings) to repetition errors. This judiciously chosen prefix is shown to have length that scales logarithmically with the length of string in this collection, and thus has asymptotically negligible redundancy while providing improved immunity to repetition errors. We also provide a decoding algorithm that is a variant of the message passing decoding algorithm, but which can handle repetitions as well as additive errors without requiring additional complexity.

In the second part of the dissertation, we study the performance of iteratively decoded LDPC codes when the frequency of a decoding error is very low. These codes are commonly decoded using highly efficient iterative decoding algorithms, which provide an exponential reduction in complexity over the optimal (but highly impractical) maximum likelihood decoding. These practical iterative decoding algorithms are suboptimal on graphs that are not trees, of which LDPC codes provide a prime example. Nonetheless, LDPC codes, when equipped with these iterative decoding algorithms, are known to perform extremely well in the moderate bit error rate (BER) region.

It is also known that LDPC codes, when decoded iteratively, exhibit a so-called error floor behavior, manifested in the need for a significant increase in the signal power for only a marginal improvement in BER. Due to the limited analytical tools available to address (and predict) the low BER performance of LDPC codes, their deployment in applications requiring low BER guarantees has not quite met the original promise of these powerful codes.

In order to gain a better understanding of the low BER performance of LDPC codes, we introduce the notion of a combinatorial object that we call an absorbing set. This object is viewed as a stable point of the bit-flipping algorithm, an algorithm that can be viewed as an asymptotic 1-bit approximation to many message passing decoding algorithms. Since absorbing sets are fixed points of the message passing algorithms, the decoder can get stuck in an absorbing set that is not a codeword. In particular, if there are absorbing sets smaller that the minimum distance of the code, the decoder is likely to converge to these objects. As a result, under iterative decoding, the low BER performance will be dominated by the number and the size of dominant absorbing sets, rather that the number of minimum distance

codewords and the minimum distance itself, which is considered to be the performance metric under the maximum likelihood decoding and the key property of a code.

As a case study, we analyze the minimal absorbing sets of high-rate array-based LDPC codes. We provide a comprehensive analytic description of the minimal absorbing sets for this family of codes. In this study, we demonstrate the existence of absorbing sets whose weight is strictly smaller than the minimum distance of the code. These minimal absorbing sets, rather than minimum distance codewords, are also experimentally shown to dominate the low BER performance.

Professor Venkat Anantharam
Dissertation Committee Chair

To mom and dad.

Acknowledgments

First and foremost, I would like to thank my advisor Professor Venkat Anantharam for the guidance during my graduate studies, for teaching me the importance of rigor and preciseness, and for being a prolific source of many results spanning various disciplines of mathematics and communications theory. I thank my dissertation and qualifying exam committee members Professor Bora Nikolic for always providing a practical perspective to my work, Professor Martin Wainwright for technical assistance and Professor David Aldous for providing feedback on this thesis. I also thank Zhengya Zhang for a successful collaboration during the course of our LDPC project.

I want to also thank Ruth Gjerde, Mary Byrnes, Pat Hernan, and everyone else at the Graduate Office who always managed to sort out every kind of bureaucratic hurdle. Amy Ng with Wireless Foundations deserves very special thanks.

Thanks for technical and not so technical discussions go to my colleagues from Wireless Foundations. It was a joy working and spending time with you guys. My family and friends, here and in the old country, thank you all for your continual and unconditional love and support. Ivana, Zeljka and Danijela, thank you for your sisterhood. You never cease to inspire me. Debby ad Jim Piper, thank you for helping me cross the ocean. Without you, this thesis, and many other wonderful things, would not have been possible. Tyson, thank you for your patience. Now and always. Mom and dad, thank you for everything you have given me. This thesis is dedicated to you.

Contents

List of Figures

List of Tables

Chapter 1

Dissertation Overview

Data communication, whether over space (e.g. Ethernet, satellite broadcasting) or over time (e.s. storage applications) is inherently noisy. The goal of the channel coding theory is to provide systematic ways of introducing a controlled amount of redundancy into the transmitted data in a form of a channel code to successfully overcome the noise in the channel [60]. Channel coding theory is traditionally concerned with constructing good codes capable of correcting additive errors, since the noise in the channel is almost always taken to be additive.

A good performance metric is that of the minimum distance of a code, which is the minimum separation of the codewords in the given code, since it determines how large the additive noise in the channel can be while communication still remains reliable. Thus, a good code traditionally has large minimum distance.

In addition, good channel codes should also be equipped with simple and realizable encoding and decoding algorithms to make them useful in practice. In particular, good decoding algorithms should mimic the performance of the optimal maximum likelihood decoding, but do so with much lower complexity than the exponential complexity of the highly impractical maximum likelihood decoding algorithm.

In this dissertation we develop novel coding theoretic approaches for two somewhat non-standard yet relevant problems for modern communication systems. In the first part of the thesis we address the issue of reliable communication under varying sampling rate, while in the second part we focus on understanding the performance of low density parity check (LDPC) codes in the low bit error rate (BER) region under iterative decoding. The underlying theme of this work is that the notion of the minimum distance of a code needs to be rethought when the standard assumptions on the communication no longer hold.

In particular, when one can no longer assume perfect signal synchronization and use the code and its decoding algorithm solely to overcome additive errors, the notion of the relevant code metric needs to be redefined. A simple example is that of a code consisting of only two codewords of equal length n, the codeword c_1 being a string of alternating 0's and 1's and the codeword c_2 being a string of alternating 1's and 0's. These two strings have maximum possible additive (Hamming) distance, since they differ in every position. However, when this code is used over a channel that introduces a repetition error, their post-repetition distance drops to only 2, since, for example, a repetition in the first bit of c_1, and in the last bit of c_2, produces strings that differ in only two positions, thus making the original code not nearly as effective for communicating over a channel that introduces additive as well as a repetition error.

More generally, current trends in many emerging applications require timing recovery to be performed under increasingly stringent constraints. In digital data storage applications the continuously increasing user demand requires higher storage capacity and higher data rates while keeping the disk size the same. While this can be accomplished using advanced coding and signal processing techniques [58], this leads to the timing recovery block consuming an increasing large fraction of on-chip resources. Low power wireless applications also demand that accurate synchronization be performed under limited power and constrained chip area [2].

Sampling errors caused by poor timing recovery, such as repetitions or deletions of symbols, severely impact the decoder's performance and can undermine the benefits of other system components, since other components are traditionally not designed to deal with synchronization errors. As an alternative to developing more complex and more expensive timing recovery schemes, we adopt a coding theoretic point of view in addressing this problem. Chapter 2 elaborates on the relevant background and introduces the set-theoretic viewpoint in modelling synchronization errors. We then investigate how to modify additive error correcting codes, such as Reed-Muller codes, to ensure that they would, in addition to excellent additive error correction properties, be equipped with good synchronization error correction capabilities.

We focus on the first order Reed-Muller codes as a representative of a class of highly structured additive error correcting codes with large minimum distance. For the Reed-

Muller codes, we first establish and prove several novel properties of the run-length structure of this code. Chapter 3 contains several structural properties of these codes, which are then used in Chapter 4 in a systematic analysis of the performance of these codes in channels which in addition to additive errors also permit a repetition or a deletion of a symbol. The runlength properties of Reed-Muller codes proved in this thesis may be also of independent interest. Chapter 4 explains how to systematically thin the original Reed-Muller code to substantially improve the performance of these codes when the inadequate synchronization causes a repetition or a deletion of a coded bit, while only losing one information symbol. We also derive appropriate post-deletion and post-repetition distance and propose a modified bounded distance decoding algorithm suitable for decoding thinned Reed-Muller codes transmitted over the channels that introduce additive noise and a synchronization error. This algorithm is a variant of the Hadamard transform-based bounded distance decoding algorithm traditionally used to decode Reed-Muller codes. It also features the same complexity as this traditional algorithm, while also being able to correct for a synchronization error.

Motivated by the problem of communicating in the presence of repetition errors, in Chapter 5 we focus on explicit number-theoretic constructions of a set of strings immune to multiple repetitions. In particular, we propose an explicit number-theoretic construction of a set of strings immune to a single repetition, and then subsequently, a construction of a set of strings immune to multiple repetitions. The former construction is asymptotically optimal, whereas the latter is within a constant of the upper bound on the cardinality of such a set, and improves on the previously best known construction due to Levenshtein [34].

Using the number-theoretic construction of Chapter 5, we then in Chapter 6 propose a general method for transforming a code, as an arbitrary collection of binary strings of equal length, by prepending each string in this collection with a prefix, carefully chosen to improve the immunity to synchronization errors. The prefix itself is chosen based on the number-theoretic methods of Chapter 5 and is shown to have length that is only logarithmic in the length of the strings in the original collection. The proposed method therefore provides a general way to reach guaranteed immunity to repetition errors with asymptotically negligible redundancy. We also provide a decoding algorithm, as a variant of a message passing algorithm, capable of decoding additive as well as repetition errors. The proposed algorithm does so without the increase in complexity over the traditional message passing decoding algorithm designed to handle additive errors only. This result concludes the first part of the dissertation.

In the second part of the dissertation we turn to a different problem where we study the performance of low density parity check (LDPC) codes in the low BER regime under iterative decoding. LDPC codes are a class of additive error correcting codes that operate close to the Shannon limit, which is the absolute limit on the rate that can be achieved over a noisy channel. LDPC codes were invented by Gallager [26] in the 1960's, but then were largely forgotten until early 1990's. Their rediscovery [39], [32] sparked research interest in LDPC codes, as well as their wide consideration for many modern applications.

LDPC codes have a convenient graphical representation which makes them particularly suitable for low complexity, iterative decoding. Finite length LDPC codes and their iterative decoding algorithms have found tremendous success when used in the moderate BER region, of say 10^{-6} and above, and vast empirical evidence supports this finding [37]. However, the suboptimal nature of these highly efficient decoding algorithms on non-tree graphs (as arise from the graphical representation of most LDPC codes) has also been troublesome when LDPC codes are considered for applications that need to operate at very low bit error rates.

Many LDPC codes that have excellent performance in the moderate BER region exhibit a change in the slope of the BER vs. signal to noise ratio (SNR) curve in the very low BER region, a region which is of interest for many practical applications, implying that a significant increase in the signal power is needed for only a marginal improvement in the bit error rate. This “error-floor” phenomenon is particularly worrisome for low BER applications since this region is out of reach of pure software simulations and there is a lack of appropriate analytic tools needed to address, predict and improve the low BER performance of LDPC codes. As a consequence, the deployment of LDPC codes in applications requiring low BER guarantees has not quite met the original promise of these powerful codes. Nonetheless, with their unprecedented coding gains, LDPC codes remain strong contenders for many applications, witnessed by their recent adoption into the digital broadcasting DVB-S2 standard [65] for satellite communication as well as 10Gb/s standard for Ethernet [66]. Chapter 8 contains relevant background on LDPC codes and surveys existing work related to the “error floor” problem.

Motivated by earlier experimental observations that certain structures intrinsic to the parity check matrix of a given code are the dominant causes of the errors in the very low BER region [48],[63], we introduce the notion of a combinatorial structure in the graphical representation of an LDPC code, which we call an absorbing set. Chapter 8 also contains a formal definition of absorbing and fully absorbing sets. Absorbing sets (fully absorbing sets) are combinatorial objects that exist in the Tanner graph associated with the parity check matrix of a given code and have the property that bits in the absorbing set (and bits outside the absorbing set) have strictly more satisfied than unsatisfied checks. In particular, fully absorbing sets are stable under the bit-flipping algorithm, which is the simplest form of the message passing based decoding of LDPC codes. If there are absorbing sets smaller that the minimum distance of the code, the decoder is likely to converge to these objects. As a result, under iterative decoding, the low BER performance will be dominated by the number and the size of dominant absorbing sets, rather that the number of minimum distance codewords and the minimum distance itself, which is considered to be the performance metric under the maximum likelihood decoding and the key property of a code.

To demonstrate the importance of studying absorbing sets, we then provide a detailed analysis of the minimal absorbing sets and minimal fully absorbing sets of the high rate array-based LDPC codes in Chapter 9. Minimal (fully) absorbing sets are the sets of the smallest size and are shown experimentally to dominate low BER performance for several LDPC codes. Array-based LDPC codes, as an instance of regular LDPC codes, were chosen for this analysis due to their high performance in the moderate BER regime, and the structure of the parity check matrix that is amenable for many high-throughput applications. In particular, we analytically describe minimal absorbing sets and minimal fully absorbing sets for $\gamma = 2, 3, 4$ where γ refers to the column weight of the array-based LDPC code. We also compute the number of (fully) absorbing sets and show how it scales with the codeword length. For $\gamma = 2$, the smallest (fully) absorbing sets are actually minimum distance codewords whereas for $\gamma = 3, 4$ and large enough parity check matrix, the smallest (fully) absorbing sets have the Hamming weight that is strictly smaller than the minimum distance of the code. In the latter case, the minimal absorbing sets, rather than the minimum distance codewords, dominate the low BER performance. This claim is also supported by the experiments carried out on an emulation platform that demonstrate complete agreement with the theoretical prediction.

Lastly, Part III summarizes main contributions and proposes potential future extensions of the work presented here.

Part I

Communication Over Channels With Varying Sampling Rate

Chapter 2

Introduction

In a typical communication system a binary input message $\mathbf{x}$ is encoded at the transmitter, using a substitution-error correcting code C, into a coded sequence $\mathbf{c}$ = $C(\mathbf{x})$, which we will assume is also a binary sequence. The modulated version of this sequence may be modeled as being corrupted by additive noise, so the received waveform after matched filtering can be written as

$$r(t) = \sum_i c_i h(t - iT) + n(t), \tag{2.1}$$

where c_i is the i^{th}bit of $\mathbf{c}$, $h(t)$ is convolution of the modulating pulse and the matched filter, and $n(t)$ represents the noise introduced by the channel.

The receiver samples $r(t)$ at time instances $\{kT_s + \tau_k\}$, and the sequence of samples is fed into the decoder which decides on the most likely input message. Accurate synchronization of the sampling instants, i.e that T_s be equal to T and that each τ_k be ideal, is critical for the full utilization of the coding gain of the substitution-error correcting code.

As the operating requirements under which timing recovery must be performed become more stringent, because of higher data rates and/or longer delays in the decision feedback loop that adjusts the sampling instants, such synchronization is becoming harder to achieve. Several authors have studied the problem of accurate timing recovery. Proposed solutions include building a more sophisticated timing recovery block [38], multiple hypothesis analysis of the sampling instances [31], and for the intersymbol interference (ISI) channels in particular, a soft-output detector for both ISI and timing errors [62], and an iterative timing recovery approach that incorporates timing recovery in turbo equalization [4].

As an alternative to more complex and more expensive timing recovery schemes, we propose to shift the emphasis away from the timing recovery block and instead modify the decoding procedure and the code itself to compensate for inadequate synchronization. By analyzing the robustness of a substitution-error correction code to synchronization errors, one could use a subcode of the original code that would have good minimum distance under both substitution as well as sampling errors. The trade-off would be between the incurred rate loss associated with the code modification versus the increased complexity and latency associated with the existing approaches mentioned above. The challenge of the proposed approach lies in determining the synchronization error correction capabilities of individual codes of interest, and in determining as large as possible a subcode with the desired properties. The proposed approach can be considered for any system, and it is especially relevant for practical pilotless communication systems.

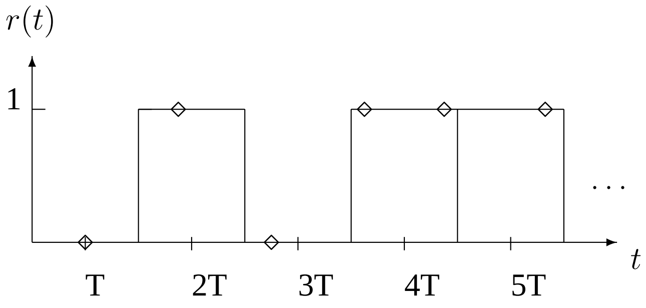

Figure 2.1: An example of oversampling.

To illustrate the issues that arise when adequate timing recovery is missing, assume (for purposes of argument) that $h(t)$ is a rectangular pulse of duration T and unit amplitude and that we are operating in the infinite signal-to-noise (SNR) regime where the effect of $n(t)$ is negligible. Then $r(t)$ simply becomes

$$r(t) = \sum_i c_i 1(iT \leq t < (i+1)T) \ . \tag{2.2}$$

If samples were taken in the middle of each pulse the sampled version of $r(t)$ would be precisely $\mathbf{c}$. Now suppose that inadequate timing recovery causes the sampling to occur at time instants $kT_s + \tau_k$.

As an example, consider a sequence $\mathbf{c}$ = (0,1,0,1,1,. . .) that results in the waveform $r(t)$ shown in Figure 2. The sampling points $kT_s + \tau_k$ are marked in the figure by $\diamond$. In this example, $T_s < T$ causes oversampling, and the sampled version of $r(t)$ contains a repeated bit (here the fourth bit is sampled twice). Analogously, when $T_s > T$, undersampling can cause the separation between two consecutive samples to be so large that some bit is not sampled at all. Therefore without adequate timing recovery the sampled version of $r(t)$ results in a sequence obtained by repeating or deleting some bits in $\mathbf{c}$.

In this work we adopt a set-theoretic model for the synchronization errors in which a codeword gives rise to a set of possible received sampled sequences which depends on how many bits are allowed to be repeated or deleted. A codeword $\mathbf{c}$ can in general give rise to a whole set of received sampled versions of $r(t)$. The possible set of such sequences depends on how good the timing recovery scheme is. When two distinct codewords $\mathbf{c_1}$ and $\mathbf{c_2}$ can result in the same sampled sequence, it is no longer possible to uniquely determine the coded sequence or its pre-image $\mathbf{x}$ from the received sequence, even in the noise-free environment. We then say that the substitution-error correcting code C has an *identification problem*. We also say that the pair of codewords $\mathbf{c_1}$ and $\mathbf{c_2}$ has an identification problem.

More generally, two distinct codewords $\mathbf{c_1}$ and $\mathbf{c_2}$ could result in sampled sequences with poor Hamming distance. This would result in poor performance over a channel that permits substitution errors. In this case we say that the substitution-error correcting code C has *poor identification*. We also say that the pair of codewords $\mathbf{c_1}$ and $\mathbf{c_2}$ has poor identification.

It should be mentioned that several authors have studied codes immune to insertions and deletions of bits. For example, the so-called Varshamov-Tenengolts code proposed in [57] and popularized by Levenshtein in [35] has been further studied by Ferreira et al., [21], Levenshtein [36], Sloane [51], and Tenengolts [55]. Related constructions were proposed in [6], [7], [12], [29] and [56]. Even though these constructions result in codes that are immune to a given number of insertions and deletions of bits, they have a limited guarantee for other desirable properties of standard substitution-error correcting codes (such as linearity and a good minimum Hamming distance). Several other authors have proposed concatenated codes that correct synchronization errors, such as in [8], [11], and [13]. In contrast to these works, our approach is to start with known substitution-error correcting codes and propose ways to modify them with only a small loss in the rate in order to continue to provide good performance under synchronization errors, which are themselves modeled as a certain number of repetitions or deletions of bits. A related problem of a code construction for frame synchronization was studied in [53] and [5].

The next two Chapters focus on the analysis of the first order Reed-Muller codes under the synchronization and substitution errors. We first prove several new structural properties of these codes in Chapter 3, which are then subsequently used in Chapter 4 where we discuss how to systematically thin these codes to improve their performance under synchronization and substitution errors, and how to efficiently decode thinned codes when both types of errors are present. Chapter 5 discusses explicit number-theoretic constructions of sets of strings capable of overcoming repetition errors. The focus of this chapter is on deriving cardinality results of these constructions using combinatorial and number-theoretic methods. Lastly, Chapter 6 discusses how to improve repetition error correcting capability of a collection of binary strings by judiciously appending a prefix to a string belonging to this collection using the number-theoretic construction from Chapter 5.

Chapter 3

Structural Properties of Reed-Muller Codes

In this Chapter we establish several new structural properties of the first order Reed-Muller codes. These properties, while also of independent interest, will be used in the subsequent chapter that discusses the performance of Reed-Muller codes under synchronization and substitution errors. Having provided a definition of the first order Reed-Muller codes in Section 3.1, we prove several runlength properties of these codes in Section 3.2. In particular, we establish several properties regarding the runlength distribution (Subsection 3.2.1), runs of runs of codewords (Subsection 3.2.2), as well as the relationship between the message vector and the runs of its codeword (Subsection 3.2.3). Section 3.3 concludes this Chapter.

3.1 First Order Reed-Muller Codes

The first order Reed-Muller codes RM(1,m) are linear $(2^m, m+1)$ substitution-error correcting codes [41]. They have good minimum distance, equal to 2^{m-1}, simple encoding, and a relatively low complexity maximum likelihood decoding algorithm ($O(n \log n)$ for $n = 2^m$). On the negative side, they have low rate.

From now on, let $C(m)$ denote the RM(1,m) code. The code $C(m)$ may be described by an $(m+1) \times 2^m$ generator matrix $\mathbf{G_m}$ given by

$$\mathbf{G_m} = \begin{bmatrix} \underline{1} \\ \mathbf{M_m} \end{bmatrix}$$

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

were $\underline{1}$ denotes the binary string of length 2^m with all entries equal to 1, and the m by 2^m submatrix $\mathbf{M_m}$ consists of lexicographically decreasing binary columns of length m. Observe that the i^{th} row of $\mathbf{G_m}$, for $1 < i \leq m+1$, consists of 2^{i-1} alternating runs of ones and zeros, and that each run is of length 2^{m-i+1}.

For future reference, we recall that every codeword in $C(m+1)$ is either the concatenation of a codeword in $C(m)$ with itself or the concatenation of a codeword in $C(m)$ with its bitwise complement [41, Thm. 2, pg. 374]. The concatenation of two binary strings a and b will be written as $[a|b]$. If c is a codeword in $C(m)$ it is straightforward to check that its bitwise complement, denoted $\bar{c}$, is also a codeword in $C(m)$. Further, its reversal, i.e. the binary string got by reading c from its end to its beginning, denoted $\overset{c}{\leftarrow}$, is also a codeword in $C(m)$. Since the operations of bitwise complementation and reversal commute, we may unambiguously denote the complement of the reversal of c as $\overline{\overset{c}{\leftarrow}}$.

We now establish several runlength properties of Reed-Muller codes.

3.2 Runlength Properties of the RM(1,m) Codes

3.2.1 Run-length distribution

Lemma 1 *The codewords in $C(m)$ can be partitioned into $2^{m-1}+1$ distinct non-empty groups G_j^m, for $0 \leq j \leq 2^{m-1}$. Here G_j^m is comprised of those codewords in $C(m)$ that have j runs of ones. G_0^m is comprised of exactly one codeword, namely the all-zero codeword. This codeword will be denoted $c_0^m(00)$. There are 4 distinct codewords in each group G_j^m, for $1 \leq j < 2^{m-1}$. These codewords may be uniquely identified by their first and last bit. They may thus be unambiguously denoted as $c_j^m(11)$, $c_j^m(10)$, $c_j^m(01)$, and $c_j^m(00)$ respectively. There are 3 distinct codewords in the group $G_{2^{m-1}}^m$. These codewords may also be uniquely identified by their first and last bit and may be unambiguously denoted as $c_{2^{m-1}}^m(11)$, $c_{2^{m-1}}^m(10)$, and $c_{2^{m-1}}^m(01)$ respectively.*

Proof: The proof is by induction on m. For $m = 1$ and $m = 2$ the statement can be verified by inspection. Suppose the assertion holds for all $1 \leq m \leq m_0$.

Let us first consider the group $G_j^{m_0}$ for $1 \leq j < 2^{m_0-1}$. By assumption, it contains 4 codewords, unambiguously denoted as $c_j^{m_0}(11)$, $c_j^{m_0}(01)$, $c_j^{m_0}(10)$, and $c_j^{m_0}(00)$ respectively. Out of the eight possible concatenations of each such codeword with either itself or its complement, 3 result in codewords in $G_{2j-1}^{m_0+1}$ (these are $[c_j^{m_0}(11)|c_j^{m_0}(11)]$, $[c_j^{m_0}(11)|\overline{c_j^{m_0}(11)}]$, and $[c_j^{m_0}(01)|\overline{c_j^{m_0}(01)}]$), 4 result in codewords in $G_{2j}^{m_0+1}$ (these are $[c_j^{m_0}(01)|c_j^{m_0}(01)]$, $[c_j^{m_0}(10)|\ \overline{c_j^{m_0}(10)}]$, $[c_j^{m_0}(10)|c_j^{m_0}(10)]$, and $[c_j^{m_0}(00)|c_j^{m_0}(00)]$), and 1 results in the codeword $[c_j^{m_0}(00)|\overline{c_j^{m_0}(00)}]$ in $G_{2j+1}^{m_0+1}$. By varying j from 1 to $2^{m_0-1}-1$, inclusive, we thus describe 3 codewords in $G_1^{m_0+1}$, 4 codewords in each $G_{j'}^{m_0+1}$ for $2 \leq j' \leq 2^{m_0}-2$ and 1 codeword in $G_{2^{m_0}-1}^{m_0+1}$ such that no two codewords that belong to the same group $G_{j'}^{m_0+1}$ agree in both the first and the last bit.

Now consider the group $G_{2^{m_0-1}}^{m_0}$. By assumption it has three codewords unambiguously denoted as $c_{2^{m_0-1}}^{m_0}(11)$, $c_{2^{m_0-1}}^{m_0}(01)$, and $c_{2^{m_0-1}}^{m_0}(10)$ respectively. There are six possibilities arising from concatenations of such a codeword with itself or its complement. Of these, 3 result in codewords in $G_{2^{m_0}-1}^{m_0+1}$ (these are $[c_{2^{m_0-1}}^{m_0}(01)|\overline{c_{2^{m_0-1}}^{m_0}(01)}]$ $[c_{2^{m_0-1}}^{m_0}(11)|c_{2^{m_0-1}}^{m_0}(11)]$, and $[c_{2^{m_0-1}}^{m_0}(11)|\overline{c_{2^{m_0-1}}^{m_0}(11)}]$) and the remaining 3 result in the codewords of $G_{2^{m_0}}^{m_0+1}$. Note that none of the latter three concatenations has both outer bits equal to '0'. Note that we have now described a total of 4 codewords in the group $G_{2^{m_0}-1}^{m_0+1}$, no two agree in both first and last bit, and we have also described 3 codewords in the the group $G_{2^{m_0}}^{m_0+1}$ of the desired form.

The concatenation of the all-zero codeword in $C(m_0)$ with the all-ones codeword yields the fourth codeword in $G_1^{m_0+1}$, and its concatenation with itself yields the only codeword in $G_0^{m_0+1}$.

We have therefore described $1 + 4 \times (2^{m_0} - 1) + 3 = 2^{m_0+2}$ codewords in $C(m_0+1)$, which is precisely the cardinality of this code, and we showed that the proposed statement holds for it. ■

By exploiting the result in Lemma 1, it is easy to verify the following, which may also of course be seen more directly.

Lemma 2 *For each $1 \leq k \leq 2^m$, in $C(m)$ there are exactly 2 codewords which have a total of k runs, and they are bitwise complements of each other.*

Proof: The complementary codewords $c_{j-1}^m(00)$ and $c_j^m(11)$ each have $2j-1$ runs. Letting j run from 1 through 2^{m-1} gives 2^{m-1} such complementary pairs of codewords. The complementary codewords $c_j^m(10)$ and $c_j^m(01)$ each have $2j$ runs. Letting j run from 1 to 2^{m-1} gives another 2^{m-1} such complementary pairs of codewords. This completes the proof. ■

Lemma 3 *Consider a codeword $\mathbf{c}$ in $C(m)$. Either $\mathbf{c}$ has all its runs of the same length, which is a power of 2, or the runs in $\mathbf{c}$ are of two different lengths, and these two lengths are consecutive powers of 2. In addition, if there are runs of two different lengths in $\mathbf{c}$, the outer runs (i.e. the leftmost run and the rightmost run) in $\mathbf{c}$ are of the smaller length.*

Proof: The proof is by induction on m. It is straightforward to check the truth of the statement for $m = 1$ and $m = 2$. Suppose now that the given statement is true for all

$1 \leq m \leq m_0$. For a codeword $\mathbf{c}$ in $C(m_0)$ let $[\mathbf{c}|\mathbf{c}]$ and $[\mathbf{c}|\overline{\mathbf{c}}]$ denote the codewords in $C(m_0+1)$ that are the concatenation of $\mathbf{c}$ with itself, and the concatenation of $\mathbf{c}$ with its complement, respectively.

Suppose first that $\mathbf{c}$ has all its runs of the same length, equal to 2^s for some $s \geq 0$. If $\mathbf{c}$ has the same starting and ending bits then in the concatenation $[\mathbf{c}|\overline{\mathbf{c}}]$ all runs have the same length 2^s, so the statement of the lemma holds. In the concatenation $[\mathbf{c}|\mathbf{c}]$ all runs except the run at the point of concatenation (if there are any such runs) have length 2^s and the run at the point of concatenation has length 2^{s+1}. The proposed statement continues to be true both in the case in which there are some runs other than the one at point of concatenation and in the case when there are no such runs. If $\mathbf{c}$ starts and ends with different bits, we may repeat the previous argument mutatis mutandis.

Now suppose that $\mathbf{c}$ has runs of different lengths, which are two consecutive powers of 2, say 2^s and 2^{s+1}. By assumption, the outer runs are of length 2^s each, and there is at least one run of length 2^{s+1}. As before, if $\mathbf{c}$ starts and ends in the same bit, the concatenation $[\mathbf{c}|\overline{\mathbf{c}}]$ will have all its runs of lengths either 2^s or 2^{s+1}. Further, the outer runs in $[\mathbf{c}|\overline{\mathbf{c}}]$ have the same length as the ones in $\mathbf{c}$, i.e. they are of length 2^s each, so the statement of the lemma is valid. In the concatenation $[\mathbf{c}|\mathbf{c}]$, the last run in the left copy of $\mathbf{c}$ and the first run in the right copy of $\mathbf{c}$ are merged together, and all other runs are unchanged in length. By assumption, the outer runs in $\mathbf{c}$ have length 2^s each, so their merger results in a run in $[\mathbf{c}|\mathbf{c}]$ of length $2 \times 2^s = 2^{s+1}$. Thus all runs in $[\mathbf{c}|\mathbf{c}]$ have length either 2^s or 2^{s+1}. Since the outer runs in $[\mathbf{c}|\mathbf{c}]$ are of the same length as the outer runs in $\mathbf{c}$, they have length 2^s, as required.

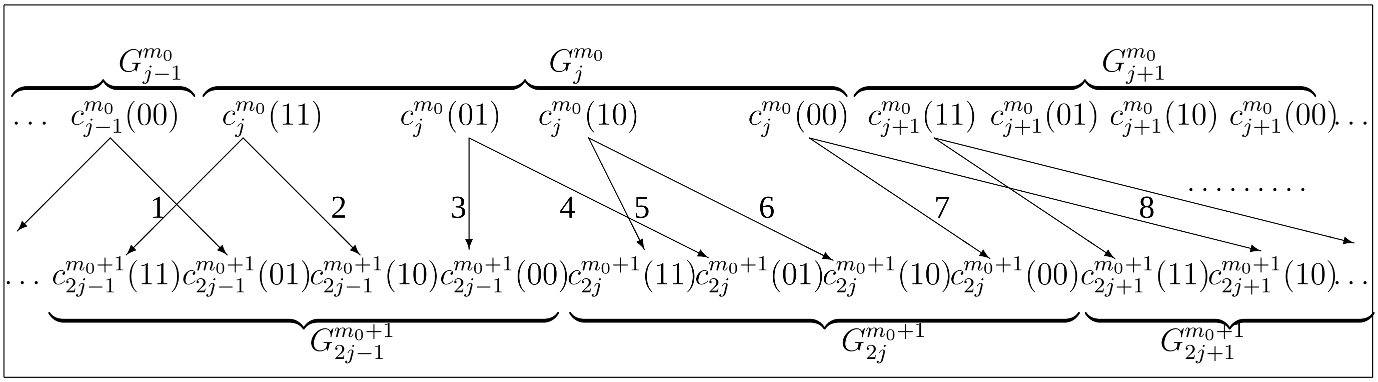

Figure 3.1: Construction of codewords in $C(m_0+1)$ from codewords in $C(m_0)$.

For **c** starting and ending in different bits, we repeat this argument mutatis mutandis.

Since each codeword in $C(m_0+1)$ can be written as a concatenation of a codeword in $C(m_0)$ either with itself or with its complement, the proof of the Lemma is complete. ■

Lemma 4 *With the exception of the all-ones codeword, all codewords belonging to the group G_j^m for $2^{p-1} < j \leq 2^p$ for some p, $0 \leq p \leq m-1$ have all runs of ones either of length 2^{m-p-1} or of length 2^{m-p}. Moreover, $(j-2^{p-1}) \times 2$ runs out of these j runs have length 2^{m-p-1}, and the remaining $2^p - j$ runs have length 2^{m-p}.*

Proof: To prove the statement we use induction on m. For small values of m, the proposed statement can be verified directly. Suppose now that the assertion holds for some $m = m_0$.

By Lemma 9, the group $G_{j'}^{m_0}$ for $2^{p-1} < j' \leq 2^p$ for some p, $0 \leq p \leq m_0 - 1$ contains codewords $c_{j'}^{m_0}(10)$, $c_{j'}^{m_0}(01)$, and $c_{j'}^{m_0}(11)$. If $j' \neq 2^{m_0-1}$ it also contains $c_{j'}^{m_0}(00)$. There is a single codeword in $G_0^{m_0}$ (the all-zeros codeword). Let us now analyze all the possible concatenations of the codewords belonging to the group $G_j^{m_0}, 0 \leq j \leq 2^{m_0-1}$ i.e. of each codeword with itself and with its complement. By Lemma 9 there are at most 4 codewords in $G_j^{m_0}$ so we have to consider at most 8 different concatenations. In doing so, the similar cases will be presented together.

- The concatenation of $c_j^{m_0}(11)$, if it exists, with itself produces a codeword in $G_{2j-1}^{m_0+1}$ (see Arrow 1 in Figure 3.1).

  If $j = 1$, $c_j^{m_0}(11)$ is the all-ones codeword in $C(m_0)$, and the concatenation with itself produces the all-ones codeword in $C(m_0 + 1)$. If $j > 1$, the outer runs in $c_j^{m_0}(11)$ must be of size 2^{m_0-p-1}. (To see this not that if j is a complete power of 2, i.e. $j = 2^p$ then all runs of ones, including the outer runs, are of size 2^{m_0-p-1} by assumption, and if j is not a complete power of 2, i.e. $2^{p-1} < j < 2^p$ then the outer runs must have size 2^{m_0-p-1} by Lemma 3). In the process of concatenation, two outer, smaller runs merge into one larger run and all other runs of ones are unaltered. Therefore, in the resulting codeword in $G_{2j-1}^{m_0+1}$, where $j > 1$, and $2^p < 2j - 1 < 2^{p+1}$, there are $2 \times (j - 2^{p-1}) \times 2 - 2 = ((2j - 1) - 2^p) \times 2$ runs of ones of size $2^{m_0-p-1} = 2^{(m_0+1)-(p+1)-1}$, and $2 \times (2^p - j) + 1 = 2^{p+1} - (2j - 1)$ runs of ones of size $2^{m_0-p} = 2^{(m_0+1)-(p+1)}$.

- The concatenation of $c_j^{m_0}(11)$, if it exists, with its complement produces a codeword in $G_{2j-1}^{m_0+1}$ (see Arrow 2 in Figure 3.1).

  The complement of $c_j^{m_0}(11)$ is $c_{j-1}^{m_0}(00)$. By assumption, $c_j^{m_0}(11)$ has $(j - 2^{p-1}) \times 2$ runs of ones of size 2^{m_0-p-1}, and $2^p - j$ runs of ones of size 2^{m_0-p}, for $j > 1$. If

$j = 1$, then $p = 0$, and the complement is the all-zero codeword, so the result of the concatenation has a single run of ones, of size $2^{m_0} = 2^{(m_0+1)-1}$.

Suppose now that $j > 1$. Then there is a corresponding p such that $2^{p-1} < j \leq 2^p$ and $0 < p \leq m_0 - 1$. Note that $2^{p-1} \leq j - 1 < 2^p$.

Case 1: $j - 1 = 2^{p-1}$

Under this condition, the codeword $c_{j-1}^{m_0}(00)$ has all $j - 1$ runs of ones of size $2^{m_0-(p-1)-1}$ each. The concatenation of $c_j^{m_0}(11)$ and $c_{j-1}^{m_0}(00)$ then has $(j - 2^{p-1}) \times 2 = 2$ runs of ones of size 2^{m_0-p-1}, and $2^p - j + j - 1 = 2^p - 1$ runs of ones of size 2^{m_0-p}. Using the fact that $2 = ((2j-1) - 2^p) \times 2$, $2^p - 1 = 2^{p+1} - (2j - 1)$ and that $2^p < 2j - 1 < 2^{p+1}$, we conclude that the resulting codeword satisfies the proposed assertion.

Case 2: $j - 1 > 2^{p-1}$

The codeword $c_{j-1}^{m_0}(00)$ has $((j-1) - 2^{p-1}) \times 2$ runs of ones of size 2^{m_0-p-1} and $2^p - (j-1)$ runs of ones of size 2^{m_0-p}. The result of the concatenation has $(j - 2^{p-1}) \times 2 + ((j-1) - 2^{p-1}) \times 2 = ((2j-1) - 2^p) \times 2$ runs of ones of size 2^{m_0-p-1}, and $2^p - j + 2^p - (j-1) = 2^{p+1} - (2j-1)$ runs of ones of size 2^{m_0-p}. Since $2^p < 2j - 1 < 2^{p+1}$, the proposed assertion holds for this choice of $j - 1$ as well.

- The concatenation of $c_j^{m_0}(01)$, if it exists, with its complement produces a codeword in $G_{2j-1}^{m_0+1}$ (see Arrow 3 in Figure 3.1).

  First note that the complement of $c_j^{m_0}(01)$ is $c_j^{m_0}(10)$, and since they both belong to the same group $G_j^{m_0}$, by assumption they both have $(j - 2^{p-1}) \times 2$ runs of ones of size 2^{m_0-p-1}, and $2^p - j$ runs of ones of size 2^{m_0-p}.

As established in Lemma 3, the outer runs are of the smaller size (here 2^{m_0-p-1}), so in the process of concatenating $c_j^{m_0}(01)$ and $c_j^{m_0}(10)$, the rightmost run of ones in $c_j^{m_0}(01)$ merges with the leftmost run of ones in $c_j^{m_0}(10)$, resulting in a run of ones of size 2^{m_0-p}. All other runs of ones are unaltered. We will treat the cases $j = 1$ and $j > 1$ separately.

If $j = 1$, both $c_j^{m_0}(01)$ and $c_j^{m_0}(10)$ have one run of ones of size 2^{m_0-1}, so their concatenation results in a codeword in $G_1^{m_0+1}$ whose sole run of ones is of size 2^{m_0}, which is consistent with the proposed assertion.

For $j > 1$, the concatenation of $c_j^{m_0}(01)$ with its complement has $(2j - 2^p) \times 2 - 2 = ((2j - 1) - 2^p) \times 2$ runs of ones of size $2^{(m_0+1)-(p+1)-1}$, and $2 \times (2^p - j) + 1 = 2^{p+1} - (2j - 1)$ runs of ones of size $2^{(m_0+1)-(p+1)}$. Since $j > 1$, $2^p < 2j - 1 < 2^{p+1}$ holds, and we can conclude that the codeword in $G_{2j-1}^{m_0+1}$ obtained by concatenating $c_j^{m_0}(01)$ with its complement satisfies the proposed assertion.

- The concatenation of $c_j^{m_0}(10)$, if it exists, with its complement produces a codeword in $G_{2j}^{m_0+1}$ (see Arrow 5 in Figure 3.1).

  Note that both $c_j^{m_0}(01)$ and its complement $c_j^{m_0}(10)$ have $(j - 2^{p-1}) \times 2$ runs of ones of size 2^{m_0-p-1}, and $2^p - j$ runs of ones of size 2^{m_0-p}. Consequently, the result of the concatenation has $(j - 2^{p-1}) \times 2 \times 2 = (2j - 2^p) \times 2$ runs of ones of size

$2^{(m_0+1)-(p+1)-1}$, and $(2^p - j) \times 2 = 2^{p+1} - 2j$ runs of ones of size 2^{m_0-p}. Since $2^p < 2j \leq 2^{p+1}$, we can conclude that the proposed assertion holds for a codeword in $G_{2j}^{m_0+1}$ obtained by concatenating $c_j^{m_0}(10)$ with its complement.

- The concatenation of $c_j^{m_0}(10)$, if it exists, with itself produces a codeword in $G_{2j}^{m_0+1}$ (see Arrow 6 in Figure 3.1).

  Now, for $2^p < 2j \leq 2^{p+1}$) the resulting codeword has $2 \times (j - 2^{p-1}) \times 2 = (2j - 2^p) \times 2$ runs of ones of size $2^{m_0-p-1} = 2^{(m_0+1)-(p+1)-1}$, and $2 \times (2^p - j) = 2^{p+1} - 2j$ runs of ones of size $2^{m_0-p} = 2^{(m_0+1)-(p+1)}$. No runs of ones are altered, they are merely duplicated. This same argument applies to the concatenation of $c_j^{m_0}(01)$ with itself (Arrow 4 in Figure 3.1), and to the concatenation of $c_j^{m_0}(00)$ with itself (Arrow 7 in Figure 3.1).

- The concatenation of $c_j^{m_0}(00)$, if it exists, with its complement produces a codeword in $G_{2j+1}^{m_0+1}$ (see Arrow 8 in Figure 3.1). If $j = 0$, $c_j^{m_0}(00)$ is the all-zeros codeword. The concatenation with its complement (the all-ones codeword) produces a codeword in $G_1^{m_0+1}$ that has a single run of ones of size $2^{(m_0+1)-1}$.

  By assumption, for $j > 0$, the codeword $c_j^{m_0}(00)$ has $(j - 2^{p-1}) \times 2$ runs of ones of size 2^{m_0-p-1}, and $2^p - j$ runs of ones of size 2^{m_0-p}. The complement of $c_j^{m_0}(00)$ is $c_{j+1}^{m_0}(11)$. We will analyze the cases when $2^{p-1} < j < 2^p$ and $j = 2^p$ separately.

  Case 1: $2^{p-1} < j < 2^p$.

  Here we have that $2^{p-1} < j + 1 \leq 2^p$, and $2^p < 2j + 1 < 2^{p+1}$. By assumption,

$c^{m_0}_{j+1}(11)$ has $((j+1)-2^{p-1})\times 2$ runs of ones of size 2^{m_0-p-1} and $2^p-(j+1)$ runs of ones of size 2^{m_0-p}. Consequently, the concatenation has $(j-2^{p-1})\times 2+((j+1)-2^{p-1})\times 2$=$((2j+1)-2^p)\times 2$ runs of ones of size $2^{m_0-p-1}=2^{(m_0+1)-(p+1)-1}$, and $2^p-j+2^p-(j+1)=2^{p+1}-(2j+1)$ runs of ones of size $2^{m_0-p}=2^{(m_0+1)-(p+1)}$. The assertion therefore holds for the codeword in $G^{m_0+1}_{2j+1}$, obtained by concatenating $c^{m_0}_j(00)$ with its complement, when $2^{p-1}<j<2^p$.

Case 2: $j=2^p$.

Now we have that $2^p<j+1\leq 2^{p+1}$ and $2^{p+1}<2j+1<2^{p+2}$. In this case, $c^{m_0}_j(00)$ has all $j=2^p$ runs of ones of size 2^{m_0-p-1}. Its complement $c^{m_0}_{j+1}(11)$ has $((j+1)-2^p)\times 2$ runs of ones of size $2^{m_0-(p+1)-1}=2^{m_0-p-2}$, and $2^{p+1}-(j+1)$ runs of ones of size $2^{m_0-(p+1)}=2^{m_0-p-1}$. The result of the concatenation has $2^p+2^{p+1}-(j+1)=2^{p+1}-1$ runs of ones of size 2^{m_0-p-1}, and $((j+1)-2^p)\times 2$ runs of ones of size 2^{m_0-p-2}. Since $j=2^p$, we can replace $2^{p+1}-1$ with $2^{p+2}-(2j+1)$ and $((j+1)-2^p)\times 2$ with $((2j+1)-2^{p+1})\times 2$. Thus, for $j=2^p$, the result of the concatenation of $c^{m_0}_j(00)$ with its complement is a codeword in $G^{m_0+1}_{2j+1}$ that has $2^{p+2}-(2j+1)$ runs of ones of size $2^{(m_0+1)-(p+2)}$, and $((2j+1)-2^{p+1})\times 2$ runs of ones of size $2^{(m_0+1)-(p+2)-1}$, where $2^{p+1}<2j+1<2^{p+2}$.

Combining the results stated so far in the proof, we conclude that Lemma 4 holds for $C(m_0+1)$. ■

For the subsequent analysis we also need to record some properties of the runs of runs in the codewords of RM(1, m).

3.2.2 Properties of the run of runs of the RM(1, m) codewords

Definition 1 *For a codeword $\mathbf{c} \in C(m)$ let $\mathbf{d} = d(\mathbf{c})$ be the string whose entries are the lengths of consecutive runs in $\mathbf{c}$, read from left to right. Let $\mathcal{D}_m = \{\mathbf{d}|\mathbf{d} = d(\mathbf{c}), \mathbf{c} \in C(m)\}$, so that $\mathcal{D}_m$ represents the collection of all possible sequences of run lengths associated with the codewords of $C(m)$.* ■

As an example, consider a codeword $\mathbf{c}$=‘10010110’, where $\mathbf{c} \in C(3)$. Then, the associated $\mathbf{d} = d(\mathbf{c})$ is $\mathbf{d}$=‘121121’.

We now state several results about such sequences of run lengths, which we will prove together.

Lemma 5 *[mirror-symmetry] $\forall \mathbf{c} \in C(m)$, the string $\mathbf{d} = d(\mathbf{c})$ possesses the mirror-symmetry property, i.e. the entry in position p in $\mathbf{d}$, denoted by $\mathbf{d}(p)$, is the same as the entry in position $l - p + 1$, denoted by $\mathbf{d}(l - p + 1)$, where l represents the length of string $\mathbf{d}$.*

Lemma 6 *If all entries in $\mathbf{d} = d(\mathbf{c})$ are either 1 or 2, with at least one entry being 1 and one being 2, then the following holds:*

1. *The leftmost entry equal to 2 must be in position 2^p, for some $p \geq 1$.*

2. *Each run of 2’s in $\mathbf{d}$ is of length $2^q - 1$, for some $q \geq 1$.*

3. *Each inner run of 1’s (where the inner run denotes a run with neighboring runs on each side) in $\mathbf{d}$ is of length $2^r - 2$, for some $r \geq 1$.*

Proof: We prove these statements by induction. We first directly verify them for small values of m. The codewords in $C(1)$ are ‘00’,‘11’,‘01’, and ‘10’, so $\mathcal{D}_1 = \{$‘2’,‘11’$\}$. The truth of the statements can be directly verified in this case. The codewords in $C(2)$ are ‘0000’, ‘1111’, ‘1100’, ‘0011’, ‘0110’, ‘1001’, ‘1010’, and ‘0101’, so

$\mathcal{D}_2 = \{$‘4’,‘22’,‘121’,‘1111’$\}$, and again the proposed statements can be verified. Similarly, the set associated with $C(3)$ is

$$\begin{aligned}\mathcal{D}_3 = \{ & \text{‘8’,‘44’,‘242’,‘2222’,‘12221’,} \\ & \text{‘121121’,‘1112111’,‘11111111’}\} ,\end{aligned}$$

and the statements hold. In particular, Lemmas 6.1 and 6.2 are applicable for the strings ‘12221’, ‘121121’, and ‘1112111’, and Lemma 6.3 is applicable for the string ‘121121’.

Suppose now that the proposed Lemmas hold for all elements of $\mathcal{D}_m$ for $1 \leq m \leq m_0$. For a codeword $\mathbf{c}$ in $C(m_0)$ let $\mathbf{c}' = [\mathbf{c}|\mathbf{c}]$ and $\mathbf{c}'' = [\mathbf{c}|\overline{\mathbf{c}}]$, and let $\mathbf{d} = d(\mathbf{c})$, $\mathbf{d}' = d(\mathbf{c}')$, and $\mathbf{d}'' = d(\mathbf{c}'')$.

First consider the case when the outermost bits in $\mathbf{c}$ are complements of each other. Then, in constructing $\mathbf{c}'$ from $\mathbf{c}$, no runs are altered and the statements in Lemmas 24 and 6 which by assumption hold for $\mathbf{d}$, continue to hold for $\mathbf{d}' = [\mathbf{d}|\mathbf{d}]$. In particular, if $\mathbf{d}$ has length l_0, $\mathbf{d}'$ has length $2l_0$. The entry $\mathbf{d}'(p)$, for $1 \leq p \leq l_0$ is the same as $\mathbf{d}'(l_0 - p + 1)$, by assumption, which is the same as $\mathbf{d}'(l_0 - p + 1 + l_0)$ = $\mathbf{d}'(2l_0 - p + 1)$. Thus, the mirror-symmetry property is preserved. The leftmost entry equal to 2 in $\mathbf{d}'$, if there is one, is in the same position as the leftmost entry equal to 2 in $\mathbf{d}$ and Lemma 6.1 holds trivially. If $\mathbf{d}'$ has only entries equal to 1 or 2, and has at least one entry of each kind, the outermost runs in $\mathbf{c}'$ and therefore in $\mathbf{c}$ must be 1-bit runs by Lemma 3. As an easy consequence, Lemma 6.2 continues to hold for $\mathbf{c}'$. By assumption, the leftmost 2 in $\mathbf{c}$ is in position 2^p for some p, so that the leftmost run of 1's in $\mathbf{d}$ is of length $2^p - 1$. The rightmost run of 1's in $\mathbf{d}$ is also $2^p - 1$ by the mirror symmetry assumption. At the point of concatenation of $\mathbf{c}$ with itself, two sequences of 1-bit runs each of length $2^p - 1$ are concatenated, and as a result, an inner run of 1's in $\mathbf{d}'$ of length $2(2^p - 1)$ = $2^{p+1} - 2$ is created. All other runs in $\mathbf{d}'$ are of the same length as the runs in $\mathbf{d}$, and Lemma 6.3 follows.

We now focus on $\mathbf{c}''$ and its $\mathbf{d}''$. All runs in $\mathbf{d}''$ remain the same as in $\mathbf{d}' = [\mathbf{d}|\mathbf{d}]$, except that the two innermost entries (which are the same by the mirror-symmetry property of $\mathbf{d}$) are replaced by a single entry of their sum. For $\mathbf{d}$ of length l_0, $\mathbf{d}''$ has length $2l_0 - 1$. The entry $\mathbf{d}''(p)$ for $1 < p \leq l_0 - 1$ is the same as $\mathbf{d}''(l_0 - p + 1)$, which is also the same as $\mathbf{d}''(l_0 - p + 1 + l_0 - 1)$ = $\mathbf{d}''((2l_0 - 1) - p + 1)$. For $p = 1$, the entry in the first position in $\mathbf{d}''$ is the same as both the first and the last entry in $\mathbf{d}$, which is itself equal to the last entry in $\mathbf{d}''$. Therefore, the mirror-symmetry property (Lemma 24) continues to hold for $\mathbf{d}''$.

If $\mathbf{d}$ has at least one entry equal to 2, its leftmost 2 is in the same position as the leftmost 2 in $\mathbf{d}''$, and Lemma 6.1 remains to hold . If $\mathbf{d}$ has all entries equal to 1, then the length of $\mathbf{d}$ is 2^{m_0} and $\mathbf{d}''$ has a single 2 in the middle position, which is then a power of 2, and both Lemma 6.1 and 6.2 hold.

By Lemma 3, if $\mathbf{c}$ has both 1-bit and 2-bit runs, the outermost runs must be 1-bit runs. If the outermost 1-bit runs in $\mathbf{c}$ are neighbored by another 1-bit runs, the innermost run of 2's in $\mathbf{d}''$ is then of length 1. If the outermost 1-bit runs in $\mathbf{c}'$ are neighbored by a sequence of consecutive 2-bit runs, which each by assumption and the symmetry property of $\mathbf{c}$ must contain $2^{q_0} - 1$ consecutive 2-bit runs, then the innermost run of 2's (at the point of concatenation in $\mathbf{c}''$) in $\mathbf{d}''$ is of length $2(2^{q_0} - 1) + 1 = 2^{q_0+1} - 1$. Since all other runs in $\mathbf{c}''$ remain unaltered we can conclude that Lemma 6.2 holds as well. Finally, Lemma 6.3 continues to hold trivially since all inner runs of 1's in $\mathbf{d}''$ already existed as inner runs of 1's in two copies of $\mathbf{d}$.

If the outermost bits in $\mathbf{c}$ are the same, we can mimic the above proof by simply exchanging $\mathbf{c}'$ and $\mathbf{c}''$. As discussed before, since each codeword in $C(m_0 + 1)$ is either a concatenation of a codeword in $C(m_0)$ with itself or with its complement, we can conclude that Lemmas 24 and 6 continue to hold for $C(m_0 + 1)$. ■

Another useful observation is given in the following:

Lemma 7 *If $\mathbf{d_a} = d(\mathbf{c_a})$ and $\mathbf{d_b} = d(\mathbf{c_b})$, for $\mathbf{c_a}$, $\mathbf{c_b} \in C(m)$ ($\mathbf{d_a}$, $\mathbf{d_b} \in \mathcal{D}_m$) and $m > 2$, are such that they have $2k + 1$ and $2k$ entries respectively, and all their entries are 1 or 2, then in the first leftmost position in which they differ, call it p, the entry is 1 in $\mathbf{d_a}$ and is 2 in $\mathbf{d_b}$, and $p < k$.*

Proof: Let s be the largest power of 2 that divides $2k$. By assumption $s \geq 1$. By Lemma 2, there exists a codeword in $C(m - s)$, call it $\mathbf{c_b^*}$, that has $r_1 = 2k/2^s$ runs and has the same leftmost bit as $\mathbf{c_b}$. In particular, if $2k$ is itself a power of 2, $\mathbf{c_b^*}$ has a single run of length $2^m/2k$. By the existence of $\mathbf{c_a}$ in $C(m)$ with $2k + 1$ runs, $2k$ is strictly less than 2^m, and thus $m - s \geq 1$. Consider a codeword in $C(m - s)$ that has $r_1 + 1$ runs, and the same leftmost bit as $\mathbf{c_a}$, and call it $\mathbf{c_a^*}$. Since r_1 is odd, $r_1 + 1 \leq 2^{m-s}$ and $\mathbf{c_a^*}$ exists by Lemma 2.

Let $\mathbf{c_e}$ be a codeword in $C(m-s-1)$ that has $(r_1+1)/2$ runs and the same leftmost bit as $\mathbf{c_a}$ (since $m-s \geq 1$, the code $C(m-s-1)$ and its codeword $\mathbf{c_e}$ exist). If $\mathbf{c_e}$ starts and ends in the same bit, which corresponds to odd $(r_1+1)/2$, we consider the codewords $\mathbf{c'_e}$ = $[\mathbf{c_e}|\overline{\mathbf{c_e}}]$ and $\mathbf{c''_e}$ = $[\mathbf{c_e}|\mathbf{c_e}]$ in $C(m-s)$, and associate $\mathbf{d'_e} = d(\mathbf{c'_e})$ and $\mathbf{d''_e} = d(\mathbf{c''_e})$ to them. Note that $|\mathbf{d'_e}|$ = $|\mathbf{d''_e}|+1$, where $|\mathbf{d'_e}|$ indicates the length of string $\mathbf{d'_e}$. Moreover, the middle entry (in position $(r_1+1)/2$) in $\mathbf{d''_e}$ is the sum of two innermost entries in $\mathbf{d'_e}$ (which span positions $(r_1+1)/2$ and $(r_1+1)/2+1$, and are equal to each other by Lemma 24), and all other entries in these two strings are the same.

If $\mathbf{c_e}$ starts and ends in complementary bits, which happens for even $(r_1+1)/2$, instead let $\mathbf{c'_e}$ = $[\mathbf{c_e}|\mathbf{c_e}]$ and $\mathbf{c''_e}$ = $[\mathbf{c_e}|\overline{\mathbf{c_e}}]$, and associate $\mathbf{d'_e} = d(\mathbf{c'_e})$ and $\mathbf{d''_e} = d(\mathbf{c''_e})$ with them. Observe that $|\mathbf{d'_e}|$ = $|\mathbf{d''_e}|+1$ as well as that $\mathbf{d''_e}$ is the same as $\mathbf{d'_e}$ except for the two innermost entries in $\mathbf{d'_e}$, which are replaced by their sum to yield the middle entry of $\mathbf{d''_e}$. By the uniqueness of a codeword in $C(m-s)$ having $|\mathbf{d'_e}|$ runs and starting with a particular bit (that being the leftmost bit of $\mathbf{c_a}$), established in Lemma 2, we conclude that $\mathbf{c^*_a}$ = $\mathbf{c'_e}$, and similarly $\mathbf{c^*_b}$ = $\mathbf{c''_e}$.

Therefore, the first leftmost position in which $\mathbf{d^*_b} = d(\mathbf{c^*_b})$ (same as $\mathbf{d''_e}$) and $\mathbf{d^*_a} = d(\mathbf{c^*_a})$ (same as $\mathbf{d'_e}$) differ is their $(r_1+1)/2^{\text{th}}$ position, such that the entry in that position in $\mathbf{d^*_b}$ is twice its counterpart in $\mathbf{d^*_a}$. By assumption on the entries of $\mathbf{d_a}$ and $\mathbf{d_b}$ being at most 2, it further follows that the entry is 1 in $\mathbf{d^*_a}$ and 2 in $\mathbf{d^*_b}$.

By constructing a sequence of codewords $\{\mathbf{c_{b,i}}\}$, for $1 \leq i \leq s+1$, starting from $\mathbf{c_{b,1}}$ = $\mathbf{c^*_b}$, and where $\mathbf{c_{b,i}} \in C(m-s-1+i)$ is the result of concatenation of $\mathbf{c_{b,i-1}}$ either with itself or with its complement (former if the outermost bits in $\mathbf{c_{b,i-1}}$ are different and latter if they are the same), we arrive at $\mathbf{c_b}$. In particular, the associated $\mathbf{d_{b,i}} = d(\mathbf{c_{b,i}})$ have length $2^{i-1}r_1$, and for the last term in the sequence $\mathbf{d_{b,s+1}}$ is of length $2^s r_1 = 2k$, which is precisely the length of $d(\mathbf{c_b})$.

Similarly, we construct a sequence of codewords $\{\mathbf{c_{a,i}}\}$, for $1 \leq i \leq s+1$, starting from $\mathbf{c_{a,1}}$ = $\mathbf{c^*_a}$. Now $\mathbf{c_{a,i}} \in C(m-s-1+i)$ is the result of concatenation of $\mathbf{c_{a,i-1}}$ with itself if the outermost bits in $\mathbf{c_{a,i-1}}$ are the same, otherwise it is the result of concatenation of $\mathbf{c_{a,i-1}}$ with its complement. The associated $\mathbf{d_{a,i}} = d(\mathbf{c_{a,i}})$ have length $2^{i-1}r_1 + 1$, so that the last term in the sequence has $2^s r_1 + 1 = 2k + 1$ runs, which is precisely the length of $\mathbf{d_a} = d(\mathbf{c_a})$. Thus, in starting from $\mathbf{c^*_a}$, by a series of concatenations in which the runs at the point of concatenation are always merged, we arrive at $\mathbf{c_a}$. Since the first leftmost entry in which $\mathbf{d^*_b}$ and $\mathbf{d^*_a}$ differ are in their $(r_1+1)/2^{\text{th}}$ leftmost positions, the first position in which $\mathbf{d_b}$ and $\mathbf{d_a}$ differ are still in their $(r_1+1)/2^{\text{th}}$ leftmost positions. Since s is at least 1, $(r_1+1)/2 \leq (k+1)/2 < k$, for $k > 1$. If $k = 1$, $\mathbf{d_a}$ is '$2^{m-1}2^{m-1}$', and $\mathbf{d_b}$ is '$2^{m-2}2^{m-1}2^{m-2}$'. For $m > 2$, $2^{m-2} > 1$, which exceeds the requirement on the entries of $\mathbf{d_b}$ being at most 2. ■

It is sometimes useful to determine the number of runs of a particular codeword based on its input message and vice versa. In the final subsection of this chapter we provide an explicit relationship between these two quantities.

3.2.3 Relationship between the input message and the run-lengths of its codeword

Let $\mathbf{a_m} = (a_0, a_m, a_{m-1}, ..., a_2, a_1)$ be a binary string of length $m+1$ and let $\mathbf{c}$ be a codeword in $C(m)$ such that $\mathbf{c} = \mathbf{a_m G_m}$. The bit a_0 multiplies the all-ones row of $\mathbf{G_m}$ and therefore does not affect the number of runs of the resulting codeword, i.e. $\mathbf{a_m} = (a_0, a_m, a_{m-1}, ..., a_2, a_1)$ and $\mathbf{a_m}' = (\overline{a_0}, a_m, a_{m-1}, ..., a_2, a_1)$ result in complement codewords (with the same number of runs). In the following we replace a_0 by x to indicate that the value of a_0 does not matter.

We denote by $R_m(a_0, a_1, ..., a_{m-1}, a_m)$ the total number of runs in $\mathbf{c}$. The following result provides a closed-form expression for $R_m(a_0, a_1, ..., a_{m-1}, a_m)$ in terms of $\mathbf{a_m}$.

Lemma 8 *The number of runs in the codeword* $\mathbf{c}$ *given by* $\mathbf{c} = \mathbf{a_m G_m}$ *where* $\mathbf{a_m} = (a_0, a_m, a_{m-1}, ..., a_2, a_1)$ *is* $R_m(a_0, a_1, ..., a_{m-1}, a_m) = 2^{m-1}a_1 + 2^{m-2} + 1/2 - \sum_{k=2}^{m} 2^{m-k-1}(-1)^{\sum_{i=1}^{k} a_i}$.

Proof: In proving this result we adopt the following viewpoint. Consider the set of m combs along the sequence of 2^m bits, which itself corresponds to a codeword in $C(m)$. Here the ith comb, $1 \leq i \leq m$, corresponds to a_i in the input message, and has teeth exactly where the row of $\mathbf{G_m}$ that multiplies a_i has a change of runs. In particular, the last, mth, comb has a single tooth that is positioned right between the left and the right half of this 2^m sequence. The penultimate comb has three teeth, immediately following the 2^{m-2}th, 2^{m-1}th and $3 \times 2^{m-2}$th bit in the sequence, and so on. The ith comb, for

$1 \leq i \leq m$, has $2^{m-i+1}-1$ teeth, each positioned immediately after the $k \times 2^{i-1}$th bit, for $1 \leq k \leq 2^{m-i+1}-1$. To determine the total number of runs in the resulting codeword we look at the total parity of teeth of those combs whose a_i's are 1, in all possible teeth locations. In particular, odd parity indicates a change of run while the even parity indicates no change of run. The total number of runs is then 1 plus the number of places where the parity of the teeth of the selected combs is odd. This can be written as

$$\begin{aligned} R_m(a_0, a_1, ..., a_{m-1}, a_m) &= 1 + 2^{m-1} \times 1(a_1 \text{ is odd }) + 2^{m-2} \times 1(a_1 + a_2 \text{ is odd })+ \\ 2^{m-i} \times 1(a_1 + \cdots + a_i \text{ is odd }) &+ \cdots + 2^{m-m} \times 1(a_1 + \cdots + a_m \text{ is odd }) \;. \end{aligned} \tag{3.1}$$

Rewrite (3.1) as

$$\begin{aligned} &R_m(a_0, a_1, ..., a_{m-1}, a_m) = 1 + 2^{m-1} \times 1(a_1 \text{ is odd })+ \\ &2^{m-3} + 2^{m-3} \times 1(a_1 + a_2 \text{ is odd }) - 2^{m-3} + \times 1(a_1 + a_2 \text{ is even })+ \\ &\vdots \\ &2^{m-i-1} + 2^{m-i-1} \times 1(a_1 + \cdots + a_i \text{ is odd }) - 2^{m-i-1} + \times 1(a_1 + \cdots + a_i \text{ is even })+ \\ &\vdots \\ &\tfrac{1}{2} + \tfrac{1}{2} \times 1(a_1 + \ldots a_m \text{ is odd }) - \tfrac{1}{2} \times 1(a_1 + \ldots a_m \text{ is even }) \;. \end{aligned} \tag{3.2}$$

Collecting the free terms and reexpressing the indicators in terms of powers of (-1) in (3.2), it follows that

$$\begin{aligned} R_m(a_0, a_1, ..., a_{m-1}, a_m) &= 1 + 2^{m-1}a_1 + (\tfrac{1}{2} + 1 + 2 + \cdots + 2^{m-3}) - \textstyle\sum_{k=2}^{m} 2^{m-k-1}(-1)^{\sum_{i=1}^{k} a_i} \\ &= 2^{m-1}a_1 + \tfrac{1}{2} + 2^{m-2} - \textstyle\sum_{k=2}^{m} 2^{m-k-1}(-1)^{\sum_{i=1}^{k} a_i}, \end{aligned} \tag{3.3}$$

which completes the proof. ■

It is also useful to know how to quickly determine the input message based on the number of runs in the codeword it generates. Let $N_{1,m}$ be the integer denoting the number of runs of a codeword in $C(m)$, and let $\mathbf{a_m}(N_{1,m}) = (a_0, a_m, ..., a_2, a_1)$ be the input message whose codeword has $N_{1,m}$ runs.

First observe from (3.1) that for $a_1 = 1$, $R_m(x, 1, ..., a_{m-1}, a_m)$ is in the interval $[2^{m-1}+ 1, 2^m]$ and for $a_1 = 0$, $R_m(x, 0, ..., a_{m-1}, a_m)$ is in the interval $[1, 2^{m-1}]$. Thus, for the given m, if $N_{1,m} \geq 2^{m-1} + 1$, a_1 must be 1, otherwise it must be zero. By substituting $a_1 = 0$ and $a_1 = 1$ in (3.1) it follows immediately that

$$R_m(x, 1, ..., a_{m-1}, a_m) + R_m(x, 0, ..., a_{m-1}, a_m) = 2^m + 1 \ .$$

To evaluate the remaining a_2 through a_m, we determine the contribution of a_2 through a_m to $N_{1,m}$. This contribution $N_{2,m}$ is $N_{1,m}$ for $a_1 = 0$ and is $2^m+1-N_{1,m}$ for $a_1 = 1$. Having determined a_1, observe that $R_m(x, 0, a_2, ..., a_{m-1}, a_m)$ = $R_{m-1}(x, a_2, ..., a_{m-1}, a_m)$, since the i^{th} row of $\mathbf{G_m}$ for $1 \leq i \leq m$ is constructed from the i^{th} row of $\mathbf{G_{m-1}}$ by repeating each entry twice. Thus, a codeword constructed from the linear combination of a subset of these particular rows of $\mathbf{G_m}$ has the same number of runs as the codeword in $C(m-1)$ constructed from the counterpart rows of $\mathbf{G_{m-1}}$.

We now view a_2 as the value that multiplies the last row of $\mathbf{G_{m-1}}$, just like a_1 did for $\mathbf{G_m}$. By using the same line of arguments as for a_1, conclude that if $N_{2,m} \geq 2^{(m-1)-1} + 1$, a_2 is 1, otherwise it is 0. The contribution $N_{3,m}$ of a_3 through a_m is $N_{2,m}$ if $a_2 = 0$ and is $2^{m-1} + 1 - N_{2,m}$ for $a_2 = 1$. Compare $N_{3,m}$ to $2^{(m-2)-1} + 1$, and if below, set $a_3 = 0$, else

$a_3 = 1$. Repeat evaluating $N_{i,m}$ and a_i until a_m is determined.

Recall that input messages $(1, a_m, ..., a_2, a_1)$ and $(0, a_m, ..., a_2, a_1)$ result in complement codewords which thus have the same number of runs.

The steps for determining the input message $\mathbf{a_m}(N_{1,m}) = (x, a_m, ...a_2, a_1)$ for the given integer $N_{1,m}$ can be outlined as follows:

1. Set $i = 1$.

2. Set $a_i = 1(N_{i,m} \geq 2^{m-i} + 1)$.

3. Set $N_{i+1,m} = (2^{m-i+1} + 1 - N_{i,m})1(a_i = 1) + N_{i,m}1(a_i = 0)$.

4. If $i = m$ return strings $(1, a_m, ..., a_2, a_1)$ and $(0, a_m, ..., a_2, a_1)$, else go back to Step 2 with $i \rightarrow i + 1$.

3.3 Summary and Concluding Remarks

Motivated by the model presented in the previous chapter, in this chapter we developed several structural properties of the RM(1,m) codes. These structural properties concerning runlength distribution, properties of these runs and the connection between the input message and the runs of its codeword may be of interest in their own right. In the next chapter we will exploit the properties established here in discussing the performance of the RM(1,m) codes under substitution and synchronization errors.

Chapter 4

Reed-Muller(1,m) Codes Under Synchronization and Substitution Errors

In this chapter we study the performance of a Reed-Muller RM(1,m) code, as an instance of a substitution-error correcting code, over channels in which, in addition to substitution errors, a sampling error can cause synchronization errors. In particular, we study the cases where the synchronization error results in the deletion of a single bit and where it results in the repetition of a single bit. In Section 4.1 we revisit the previously discussed model of synchronization errors. Section 4.2 discusses the identification problem for the RM(1,m) code under repetition and deletion errors, and provides a method to modify this code to eliminate the identification problem. The pruned code also has good identification under synchronization errors. The proofs heavily rely on the structural properties proved earlier in Chapter 3. In Section 4.3 we study the modified RM code under synchronization and substitution errors. In particular, we establish the post-repetition and the post-deletion distance of this code (Subsection 4.3.1) and provide bounded distance decoding algorithms suitable for the channels of present interest (Subsection 4.3.2). Section 4.4 provides a summary and concluding remarks.

4.1 Transmission Model Revisited

We recall the discussion of synchronization errors from Chapter 2. We adopt the following model in the infinite SNR limit. Suppose C is a (n, k) linear block code. A codeword $\mathbf{c} \in C$ is modulated using pulse-amplitude modulation (PAM), and the received waveform $r(t)$ is sampled noise-free. Let $\mathbf{r}$ be the sampled version of $r(t)$ of length l bits. We assume that the location of the first and the last bit of $\mathbf{r}$ in the received string of data is known, so that the codewords can be analyzed in isolation. Then, from l we would know the difference between the number of repetitions and the number of deletions that occurred over the channel. For instance, if the channel model permits one repetition, then if $l = n$ we know that the the sampled version of $r(t)$ equals $\mathbf{c}$, while if $l = n + 1$ the sampled version of $r(t)$ is $\mathbf{c}$ but with one bit repeated. Similarly, if the channel model permits one deletion, then if $l = n$ we know that the sampled version of $r(t)$ equals $\mathbf{c}$, while if $l = n - 1$ the sampled version of $r(t)$ is $\mathbf{c}$ with one bit deleted. These are the two channel models that we consider in this Chapter. Note that in these examples the location of the repeated (respectively deleted) bit is not known.

In general, in the infinite SNR limit a channel with synchronization errors could be modelled as introducing a certain number of repetitions and deletions in the transmitted codeword. Assuming, as above, that the location of the first and the last bit in the received string of data is known codewords could be analyzed in isolation, and we would learn the difference, $l - n$, between the number of repetitions and the number of deletions that occurred over the channel. However, we would not know the location of the repetitions and/or the deletions. This more general kind of model is not analyzed here.

This chapter is concerned with use of RM(1,m) codes over channels permitting substitution and synchronization errors under the two kinds of synchronization error models discussed in the first paragraph: the single repetition model and the single deletion model.

4.2 Identification Problem

In this section we analyze the identification problem for codewords of the RM(1,m) codes over channels permitting a single deletion. Before doing so, we first deal with the much simpler case of channels permitting only (an arbitrary number of) repetition errors.

4.2.1 The case of repetition errors

We have the following simple result:

Theorem 1 *In $C(m)$, no two codewords can result in the same string when they experience repetitions.*

Proof: For the case of one, or any number of repetitions, two codewords in $C(m)$ resulting in the same string must have the same number of runs, and the same sequence of runs. By Lemma 2 there are exactly two codewords with the same number of runs. However these two codewords are also complements of each other and therefore cannot have the same sequence of runs. We can conclude that $C(m)$ is immune to repetition errors. ■

It should be noted, nevertheless, that even single repetitions can result in pairs of codewords of the RM(1,m) code having poor identification. For instance, the codeword $c_{2^{m-1}}(01)$ and its complement $c_{2^{m-1}}(10)$ have a post-repetition Hamming distance of 2.

4.2.2 The case of a single deletion

The analysis of the identification problem for RM(1,m) codes over channels permitting a single deletion is considerably more interesting, see Theorem 2. Before proceeding to the main theorem, we first make a couple of simple remarks.

Remark 1 *[Complementarity] Consider two distinct codewords $\mathbf{c_a}$ and $\mathbf{c_b}$ in $C(m)$. If $\mathbf{c_a}$ and $\mathbf{c_b}$ can give rise to the same string after experiencing one deletion each, the same is true for their bitwise complements $\overline{\mathbf{c_a}}$ and $\overline{\mathbf{c_b}}$.*

Remark 2 *[Reversibility] Consider two distinct codewords $\mathbf{c_a}$ and $\mathbf{c_b}$ in $C(m)$, If $\mathbf{c_a}$ and $\mathbf{c_b}$ can give rise to the same string after experiencing one deletion each the same is true for their reversals $\overset{\mathbf{c_a}}{\leftarrow}$ and $\overset{\mathbf{c_b}}{\leftarrow}$.*

Here is a description of the pairs of codewords in RM(1,m) which suffer from the identification problem over channels with a single deletion, for small values of m:

Remark 3 *For $m = 0, 1, 2$ we can show by inspection the following.*

$m = 0$: *The only codewords are '0' and '1' and they can both result in an empty string.*

$m = 1$: *The codewords are '00', '11', '01', and '10'. The codewords '00', '01', and '10' can all result in '0', and the codewords '11','10', and '01' can all result in '1'.*

$m = 2$: *The codewords are '0000', '1100', '0011', '0110', '1111', '1010', '0101', and '1001'. The codeword '0011' and any one of '0110', '0101', and '1001' can result in the same string. Similarly, the codeword '1100' and any one of '1001', '1010', and '0110' can result in the same string. The same is true for '0110', and any one of '1010' and '0101' as well as for '1001' and any one of '0101' and '1010'. Also, '1010' and '0101' can result in the same string.* ■

We may now complete the analysis of the identification problem for RM(1,m) codes over channels permitting a single deletion:

Theorem 2 *Let $j = 2^{m-1}$ and $k = 2^{m-2}$. For $m \geq 3$, there is a total of 11 pairs of distinct codewords in $C(m)$ that result in the same string when each experiences a deletion. These are:*

$$\text{1.}\quad c_j^m(10) \text{ and } c_j^m(01) \left.\vphantom{c_j^m}\right\} \text{Group 1}$$

$$\left.\begin{array}{ll}
\textit{2.} & c_j^m(10) \textit{ and } c_j^m(11)\\
\textit{3.} & c_j^m(10) \textit{ and } c_{j-1}^m(00)\\
\textit{4.} & c_j^m(01) \textit{ and } c_j^m(11)\\
\textit{5.} & c_j^m(01) \textit{ and } c_{j-1}^m(00)
\end{array}\right\}\textit{Group 2}$$

$$\left.\begin{array}{ll}
\textit{6.} & c_k^m(01) \textit{ and } c_k^m(00)\\
\textit{7.} & c_k^m(01) \textit{ and } c_{k+1}^m(11)\\
\textit{8.} & c_k^m(10) \textit{ and } c_k^m(00)\\
\textit{9.} & c_k^m(10) \textit{ and } c_{k+1}^m(11)
\end{array}\right\}\textit{Group 3}$$

$$\left.\begin{array}{ll}
\textit{10.} & c_j^m(01) \textit{ and } c_{j-1}^m(01)\\
\textit{11.} & c_j^m(10) \textit{ and } c_{j-1}^m(10)
\end{array}\right\}\textit{Group 4}$$

Proof: Observe that we have already established this result for $m = 2$ in the previous remark. In the rest of the proof we will assume that $m \geq 3$.

Note that it is sufficient to assume that the deletion occurs at the end of a run, since the string resulting from a deletion of a bit in some codeword is the same irrespective of where the deleted bit was located within the run it belonged to.

Suppose $\mathbf{c_a}$ and $\mathbf{c_b}$ are distinct codewords in $C(m)$ which result in the same string when each experiences one deletion. Let $\mathbf{d_a} = d(\mathbf{c_a})$ and $\mathbf{d_b} = d(\mathbf{c_b})$ be as defined in Definition 1. We first observe that during a deletion, the total number of runs in the codeword stays the same, decreases by one, or by two. Suppose a codeword $\mathbf{c_a}$ experiences a deletion in a run of length at least 2. Then the length of $\mathbf{d_a}$ remains unchanged. If

$\mathbf{c_a}$ experiences a deletion in a run of length 1, the neighboring runs (if any) will merge and the total number of runs will decrease. In particular, if this deleted run of length 1 is an outermost run, the length of $\mathbf{d_a}$ decreases by 1. If this deleted run of length 1 is located somewhere else in $\mathbf{c_a}$, the length of $\mathbf{d_a}$ decreases by 2. It is therefore sufficient to consider the cases when the lengths of $\mathbf{d_a}$ and $\mathbf{d_b}$ differ by 0, 1, and 2. Without loss of generality assume that $|\mathbf{d_a}| \geq |\mathbf{d_b}|$. We treat the cases $|\mathbf{d_a}|$=$|\mathbf{d_b}|$, $|\mathbf{d_a}|$=$|\mathbf{d_b}|$+1, and $|\mathbf{d_a}|$=$|\mathbf{d_b}|$+2 separately.

Case 1: $|\mathbf{d_a}|$=$|\mathbf{d_b}|$

By Lemma 2, it must be that $\mathbf{c_a}$ and $\mathbf{c_b}$ are complements of each other, and consequently $\mathbf{d_a}$ = $\mathbf{d_b}$. Either both $\mathbf{c_a}$ and $\mathbf{c_b}$ experience deletions in runs of length at least 2 each, or both experience deletions in different outermost runs of length 1 each or in inner runs of length 1 each.

Since $\mathbf{c_a}$ and $\mathbf{c_b}$ differ in their leftmost bits, a deletion must occur in the leftmost bits in either $\mathbf{c_a}$ or $\mathbf{c_b}$. Without loss of generality we can assume that the leftmost bit in $\mathbf{c_a}$ is deleted. If this bit belonged to a run of length at least 2, $\mathbf{c_b}$ itself would start with a run of length at least 2, but then it would be impossible to obtain the same string from $\mathbf{c_a}$ and $\mathbf{c_b}$ when each experiences exactly one deletion. Therefore, the leftmost run in $\mathbf{c_a}$ is a run of length 1, and by Lemma 3, all runs in $\mathbf{c_a}$ (and $\mathbf{c_b}$) must be of length 1 or 2. Since $\mathbf{d_a}$ decreases by 1, the same must be true for $\mathbf{d_b}$, so that $\mathbf{c_b}$ experiences a deletion in its outermost bit, which then must be its rightmost bit. Then $\mathbf{c_a}(p)$ = $\mathbf{c_b}(p-1)$ for $1 < p \leq 2^m$ (here and in the remainder $\mathbf{c_a}(p)$ denotes the bit in the p^{th} leftmost position of $\mathbf{c_a}$), and by using the fact that $\mathbf{c_a}$ and $\mathbf{c_b}$ are complements of each other, it follows that $\mathbf{c_a}$ and $\mathbf{c_b}$ consist of alternating bits. Thus $\mathbf{c_a}$ is either $c_j^m(10)$ or $c_j^m(01)$ for $j = 2^{m-1}$, and $\mathbf{c_b}$ is its complement. This codeword pair is listed under 1 and is labeled Group 1.

Case 2: $|\mathbf{d_a}|$=$|\mathbf{d_b}|$+1

Suppose a deletion occurs in position p_a in $\mathbf{c_a}$, and in position p_b in $\mathbf{c_b}$ (we assume that the deletion occurs at the end of a run), where we index the bits in the codewords with 1 through 2^m, from left to right. It must be that either: a) $\mathbf{c_a}$ experiences a deletion in an outermost run of length 1, while $\mathbf{c_b}$ experiences a deletion in a run of length at least 2, or b) $\mathbf{c_a}$ experiences a deletion in an inner run of length 1 and $\mathbf{c_b}$ experiences a deletion in an outermost run of length 1.

Subcase 2-1: $|\mathbf{d_a}|$ is even

We view $\mathbf{c_a}$ as the result of concatenation applied to the same codeword $\mathbf{c}' \in C(m-1)$, whereby $\mathbf{c_a} = [\mathbf{c}'|\mathbf{c}']$ if $\mathbf{c}'$ has opposite outermost bits, and $\mathbf{c_a} = [\mathbf{c}'|\overline{\mathbf{c}'}]$ if the outermost bits in $\mathbf{c}'$ are the same.

In either case a) or b) there exists at least one entry in $\mathbf{d_a}$ equal to 1. Then, by Lemma 3, the outermost runs in $\mathbf{c_a}$ and $\mathbf{c}'$ are all of length 1. By mirror-symmetry (Lemma 24) we can express $\mathbf{d_a}$ and $\mathbf{d_b}$ as $\mathbf{d_a} = [A11A^R]$ and $\mathbf{d_b} = [A2A^R]$, where $A = [A_1A_2...A_l]$ is a substring of $\mathbf{d_a}$, A^R is its reverse, and $A_1 = 1$.

For the situation described in a), by the reversibility property, we may as well assume that the leftmost bit in $\mathbf{c_a}$ is deleted. Then the entry in position p in $\mathbf{d_b}$ must correspond to the entry in position $p+1$ in $\mathbf{d_a}$, in the sense that $\mathbf{d_a}(p+1)$ = $\mathbf{d_b}(p)$ $\forall p$ except for exactly one, call it p^*, for which $\mathbf{d_a}(p^*+1)$ = $\mathbf{d_b}(p^*)+1$. In particular if this entry in $\mathbf{d_b}$ is bigger than 2, by Lemma 3, it would have to be at least 4, further implying the existence of a run in $\mathbf{c_a}$ of length at least 3, which is impossible by Lemma 3 and the fact that there is at least one run of length 1 in $\mathbf{c_a}$.

Therefore $\mathbf{d_b}(p^*)$ = 2 and $\mathbf{d_a}(p^*+1)$ = 1. Since $\mathbf{d_b}(l+1)$ = 2 and $\mathbf{d_a}(l+2)$ = 1 by construction, it follows that $p^* = l+1$. Furthermore, $A_2 = A_1$, $A_3 = A_2$, ...$A_l = A_{l-1}$, so that $\mathbf{d_a}$ consists of all 1's and $\mathbf{d_b}$ has all 1's except for its innermost entry which is 2. Consequently $\mathbf{c_a}$ is either $c_j^m(10)$ or $c_j^m(01)$, and $\mathbf{c_b}$ is either $c_j^m(11)$ or $c_{j-1}^m(00)$for $j = 2^{m-1}$. One can check that all four pairs of candidate codewords suffer from the identification problem. This is the set of pairs listed under Group 2. This group of codeword pairs is closed under complementation and reversal.

Now, for the situation described in b), by the reversibility property, we may as well assume that the rightmost bit in $\mathbf{c_b}$ is deleted.

The first leftmost entries where $\mathbf{d_a}$ and $\mathbf{d_b}$ differ are their $(l+1)^{\text{st}}$ entries so the deletion in $\mathbf{c_a}$ must be in its $(l+2)^{\text{nd}}$ run, which then disappears altogether. Moreover, both $(l+1)^{\text{st}}$ and $(l+3)^{\text{rd}}$ runs in $\mathbf{c_a}$ must be of length 1 each because the $(l+1)^{\text{st}}$ run of $\mathbf{c_b}$ is of length 2. Therefore $A^R(1) = A_l = 1$.

The entry in position $l+2$ in $\mathbf{d_b}$ (which is $A^R(1)$) must be the same as the entry in position $l+4$ in $\mathbf{d_a}$, which is $A^R(2) = A_{l-1}$. The entry in position $l+3$ in $\mathbf{d_b}$, which is itself $A^R(2)$, is the same as the entry in $\mathbf{d_a}$ in position $l+5$, which is $A^R(3)$.

By continuing forward until the end of A^R, we conclude that A^R consists of all 1's, thereby making $\mathbf{d_a}$ be all 1's as well, and $\mathbf{d_b}$ be all 1's except for 2 in the middle. These two $\mathbf{d_a}$ and $\mathbf{d_b}$ have already been encountered in the situation described in a), and yield the codeword pairs listed under Group 2.

Subcase 2-2: $|\mathbf{d_a}|$ is odd

In either case a) or b) $\mathbf{d_a}$ has at least one entry equal to 1, so all its entries are either 1 or 2 by Lemma 3. If $\mathbf{d_b}$ had an entry larger than 3, by Lemma 3 case b) would not be even possible. For case a) it would require an existence of a run in $\mathbf{c_a}$ of length at least 3, which is also impossible by the same Lemma. Since all entries in $\mathbf{d_a}$ and $\mathbf{d_b}$ are then precisely 1 or 2, we can use their mirror symmetry and apply Lemma 7 to conclude that $\mathbf{d_a}$ and $\mathbf{d_b}$ have the following formats:

$\mathbf{d_a} = [A1B1A^R]$ and $\mathbf{d_b} = [A2C2A^R]$,

where $|B| = |C| + 1$ and A and C are possibly empty.

Let $|A| = p - 1$. Further, note that $|C|$ is even.

For the situation described in a) we may as well assume, by the reversibility property, that the rightmost bit in $\mathbf{c_a}$ is deleted, and that it belonged to a 1-bit run. Then the deletion in $\mathbf{c_b}$ must be in its p^{th} leftmost run (of length 2).

Since $A^R(p-1) = 1$ in $\mathbf{d_a}$, by mirror symmetry, $A(1) = 1$ (or by Lemma 3). Since the rightmost entry in $\mathbf{d_b}$ is the same as the second rightmost entry in $\mathbf{d_a}$, it further follows that $A^R(p-2) = 1$, which in turn implies that $A(2) = 1$, and so on until the end of A, thereby requiring that A consists of all 1's. Similarly, the entry in $\mathbf{d_b}$ in position $|\mathbf{d_b}| - (p-1)$, which is 2 by assumption, is the same as the entry in $\mathbf{d_a}$ in position $|\mathbf{d_a}| - p$, which is itself the last entry in B. Thus B ends in 2 and by mirror symmetry it also starts with 2. This in turn implies that C starts and ends with 2, which then implies that the next to the last entry in B is also 2. By continuing on until all entries in B and C have been encountered we can conclude that B and C consist only of 2's. Then $\mathbf{d_a}$ = '1.12.21.1' and $\mathbf{d_b}$='1.12.21.1' (if A nonempty) or $\mathbf{d_b}$='2.2' (if A empty), where '1.1' ('2.2') indicates a non-empty run of 1's (2's). For $|\mathbf{d_b}|$ even, the run of 2's in $\mathbf{d_b}$ would have to have even length (since the neighboring '1.1' runs are of the same length by the mirror-symmetry property) which is impossible by Lemma 6.2. Thus $\mathbf{d_b}$ ='2.2', A is empty, and then $\mathbf{d_a}$='12.21'. Consequently, $\mathbf{c_b}$ itself is either $c_k^m(01)$ or $c_k^m(10)$ for $k = 2^{m-2}$, and $\mathbf{c_a}$ is either $c_k^m(00)$ or $c_{k+1}^m(11)$. It can be checked that all four codeword pairs suffer from the identification problem. These are the pairs listed in Group 3. This group of codeword pairs is also closed under complementation and reversal.

For b) we may as well assume, by the reversibility property, that the rightmost bit in $\mathbf{c_b}$ is deleted, so that $\mathbf{d_b}$ ends in a 1. Note that this implies that A^R (and A) cannot be empty, and therefore $p > 1$. Then the first leftmost entry in which $\mathbf{d_a}$ and $\mathbf{d_b}$ differ is compensated for by the deletion in a 1-bit run in $\mathbf{c_a}$. Since all runs in $\mathbf{c_b}$ are of length at most 2, the deleted run in $\mathbf{c_a}$ must be bordered by two 1-bit runs. Therefore, the $(p+1)^{\text{st}}$ run (of length 1) in $\mathbf{c_a}$ is deleted, and both $(p)^{\text{th}}$ and $(p+2)^{\text{nd}}$ run in $\mathbf{c_a}$ are also of length 1. Furthermore, the entry in position t for $p + 1 \leq t \leq |\mathbf{d_b}| - 1$ in $\mathbf{d_b}$ is the same as the entry in position $t + 2$ in $\mathbf{d_a}$.

In particular, the entry in $\mathbf{d_b}$ in position $|\mathbf{d_b}|-p+1$, which is 2, is the same as the entry in position $|\mathbf{d_a}|-p+2$ in $\mathbf{d_a}$, which is $A^R(1)$. By mirror symmetry entries in positions $p-1$ in both $\mathbf{d_a}$ and $\mathbf{d_b}$ are equal to 2. Then the entry in $\mathbf{d_b}$ in position $|\mathbf{d_b}|-p+2$ is also 2, as is the entry in $\mathbf{d_a}$ in position $|\mathbf{d_a}|-p+3$. By continuing onwards until $t=|\mathbf{d_b}|-1$, and by using the mirror symmetry, we conclude that A (and A^R) consists of all 2's, which is in contradiction with the earlier requirement that the deletion in $\mathbf{c_b}$ occurs in its outermost run of length 1.

Case 3: $|\mathbf{d_a}|$=$|\mathbf{d_b}|$+2

We now consider the remaining case where the deletion in $\mathbf{c_a}$ occurs in an inner run of length 1 and in $\mathbf{c_b}$ in a run of length at least 2. This deletion in a 1-bit run of $\mathbf{c_a}$ causes its neighboring runs to merge. By Lemma 3, these runs are of length 1 or 2 each. If they were both of length 2 each, there would exist an inner run of 1's in $\mathbf{d_a}$ of length 1, which is impossible by Lemma 6.3. If one neighboring run was of length 1 and the other of length 2, the merging would require an existence of a 3-bit run in the post-deletion $\mathbf{c_b}$. By Lemma 3, the deletion in $\mathbf{c_b}$ would then have to be in a 4-bit run, and by the same Lemma, the outermost runs in $\mathbf{c_b}$ would be of length at least 2. These would have to correspond to the outermost runs in $\mathbf{c_a}$, which are themselves of length 1 each. Therefore, the deletion in $\mathbf{c_a}$ must occur in an inner 1-bit run neighbored by two 1-bit runs, and all entries in both $\mathbf{d_a}$ and $\mathbf{d_b}$ can be only 1 or 2.

Consider $\mathbf{c_c} \in C(m)$ which has $|\mathbf{d_b}|+1$ runs. For $|\mathbf{d_a}|$ even, we can think of $\mathbf{c_a}$ as being the result of concatenating a codeword $\mathbf{c_d} \in C(m-1)$ with itself if $|\mathbf{d_a}|/2$ is even, and with its complement if $|\mathbf{d_a}|/2$ is odd, such that $\mathbf{c_d}$ and $\mathbf{c_a}$ have the same leftmost bits (the existence of such codeword in $C(m-1)$ follows from Lemma 2). Furthermore, in the former case we can view $\mathbf{c_c}$ as the result of concatenating $\mathbf{c_d}$ with its complement, and in the latter case as the result of concatenating $\mathbf{c_d}$ with itself. Then $\mathbf{d_a} = [\mathbf{d_d}|\mathbf{d_d}]$, and $\mathbf{d_c} = [\mathbf{d_d}(1, l-1)|(\mathbf{d_d}(l)+\mathbf{d_d}(1))|\mathbf{d_d}(2, l)]$, where $\mathbf{d_d} = d(\mathbf{c_d})$ and $l = |\mathbf{d_d}|$. The leftmost entry in which $\mathbf{d_a}$ and $\mathbf{d_c}$ differ is their $(|\mathbf{d_c}|+1)/2^{\text{th}}$ leftmost entry. By mirror symmetry of $\mathbf{d_d}$, this entry in $\mathbf{d_c}$ is twice its counterpart in $\mathbf{d_a}$. Since all entries in $\mathbf{d_a}$ are 1 or 2, and its outermost entries are 1, it follows that all entries in $\mathbf{d_c}$ are also at most 2. Then the first leftmost entry in which $\mathbf{d_c}$ and $\mathbf{d_b}$ differ is say in position p, for $p < |\mathbf{d_b}|/2$ and $\mathbf{d_c}(p) = 1$ and $\mathbf{d_b}(p) = 2$, by Lemma 7. Since $|\mathbf{d_b}| < |\mathbf{d_c}| + 1$, the first leftmost entries in which $\mathbf{d_a}$ and $\mathbf{d_b}$ differ is in the p^{th} position, where $p < |\mathbf{d_b}|/2$.

A similar argument holds for $|\mathbf{d_a}|$ odd when the first leftmost entry in which $\mathbf{d_a}$ and $\mathbf{d_c}$ differ is then in some position p, for $p < |\mathbf{d_c}|/2$, and the first leftmost entry in which $\mathbf{d_c}$ and $\mathbf{d_b}$ differ is in their $(|\mathbf{d_b}|+1)/2^{\text{th}}$ entry. Then the first leftmost entry in which $\mathbf{d_a}$ and $\mathbf{d_b}$ differ is still in position p.

As a result and by mirror symmetry, we can then express $\mathbf{d_a}$ and $\mathbf{d_b}$ as $\mathbf{d_a} = [A1B1A^R]$ and $\mathbf{d_b} = [A2C2A^R]$, where $|B| = |C| + 2$, $|A| = p - 1$, and A and C are possibly empty.

By the reversibility property, we can assume that the leftmost error is a deletion in $\mathbf{c_a}$, which then must be in the $(p+1)^{\text{st}}$ run in $\mathbf{c_a}$ (of length 1), neighbored by 1-bit runs on each side, such that the substring ‘111’ starts at position p in $\mathbf{d_a}$ and the substring ‘2’ in $\mathbf{d_b}$ is at position p.

From $t = p + 1$ onwards, the entry in position t in $\mathbf{d_b}$ must be the same as the entry in position $t + 2$ in $\mathbf{d_a}$, except for one pair of entries. In this exception, the entry is 2 in $\mathbf{d_b}$ and 1 in $\mathbf{d_a}$. By mirror symmetry, the entry in $\mathbf{d_b}$ in position $|\mathbf{d_b}| - p + 1$ is 2 and the entry in $\mathbf{d_a}$ in position $|\mathbf{d_a}| - p + 1 = |\mathbf{d_b}| - p + 1 + 2$ is 1.

We now re-express $\mathbf{d_a}$ as $[A111D1A^R]$ and $\mathbf{d_b}$ as $[A2D2A^R]$, such that $B = 11D$. In particular, D is non-empty as otherwise $\mathbf{d_b}$ would have a run of 2's of even length which by Lemma 6.2 would imply that $\mathbf{d_b}$ consists of all 2's. As a consequence, $\mathbf{d_a}$ would have an inner run of 1's of length 4, which is impossible by Lemma 6.3.

We suppose that $|D| = l$, $l > 0$. By mirror symmetry of $\mathbf{d_a}$, $D(l) = D(l-1) = 1$, and then by mirror symmetry of $\mathbf{d_b}$, $D(1) = D(2) = 1$ as well. By mirror symmetry of $\mathbf{d_a}$, $D(l-2) = D(l-3) = 1$. By continuing on with matching up the appropriate entries in $\mathbf{d_a}$ and $\mathbf{d_b}$, and by utilizing mirror symmetry we conclude that D consists of all 1's. Then, $\mathbf{d_a} = [A1.1A^R]$ and $\mathbf{d_b} = [A21.12A^R]$, and by Lemma 6.3 $|\mathbf{d_b}|$ is even, as is then $|\mathbf{d_a}|$.

Consider $\mathbf{d'_b} = \mathbf{d_b}(1, |\mathbf{d_b}|/2)$, and $\mathbf{d'_a} = \mathbf{d_a}(1, |\mathbf{d_a}|/2)$. Since $|\mathbf{d_a}|$ and $|\mathbf{d_b}|$ are even, there exist codewords $\mathbf{c'_a}, \mathbf{c'_b} \in C(m-1)$ for which $\mathbf{d'_a} = d(\mathbf{c'_a})$ and $\mathbf{d'_b} = d(\mathbf{c'_b})$. Then $\mathbf{d'_a}$ = $[A1.1]$ and $\mathbf{d'_b}$ = $[A21.1]$. If 2 following A in $\mathbf{d'_b}$ is not in its innermost position, then it would have a mirror image in A in $\mathbf{d'_b}$ (it cannot have a mirror image in the run of 1's) but such 2 in A in $\mathbf{d'_a}$ would not have 2 as its mirror image. Thus $|A| = |\mathbf{d'_b}|/2 - 1$ and A has all 1's. Then $\mathbf{d_a}$ itself has all 1's, and $\mathbf{d_b}$ is '1.121.121.1', so that $\mathbf{c_a}$ is $c_j^m(10)$ or $c_j^m(01)$, and $\mathbf{c_b}$ is $c_{j-1}^m(10)$ or $c_{j-1}^m(01)$, for $j = 2^{m-1}$. By the current assumption on the deletion locations, it follows that $\mathbf{c_a}$ and $\mathbf{c_b}$ must have the same leftmost bit. The resulting two pairs of codewords are listed in Group 4. By reversibility and complementarity these are the only such pairs. ■

Remark 4 *It is well known that a code capable of correcting a deletion is also capable of correcting an insertion [35]. Moreover, the codeword pairs that cause the identification problem under a single insertion are the same as the codeword pairs that cause the identification problem under a single deletion, and thus Theorem 2 also gives the identification error causing codeword pairs under a single insertion.* ■

Having identified all pairs of codewords in RM(1,m) that have an identification problem, our next goal is to construct a linear subcode that has good identification under single deletion errors. It turns out this is possible to do with the loss of only one information bit, and furthermore, this subcode also has good identification for single repetition errors.

4.2.3 Pruned RM Code

Let us first recall that the i^{th} row of $\mathbf{G_m}$, for $1 < i \leq m+1$ consists of 2^{i-1} alternating runs of ones and zeros, and that each run is of length 2^{m-i+1} (see Section 3.1). Observe that the i^{th} row is then precisely $c^m_{2^{i-2}}(10)$. In particular, the last two rows of $\mathbf{G_m}$ are $c^m_{2^{m-2}}(10)$ for i = m and $c^m_{2^{m-1}}(10)$ for i = $m+1$.

We write $\mathbf{c} \in C(m)$ as $\mathbf{x}\mathbf{G_m}$, where $\mathbf{x}$ is a $(m+1)$-dimensional message vector so that $c^m_{2^{m-1}}(10)$ = $[0, 0, \ldots, 0, 1]\mathbf{G_m}$ and $c^m_{2^{m-1}}(01)$ = $[1, 0, \ldots, 0, 1]\mathbf{G_m}$. Similarly, $c^m_{2^{m-2}}(10)$ is $[0, 0, \ldots, 0, 1, 0]\mathbf{G_m}$ and $c^m_{2^{m-2}}(01)$ is $[1, 0, \ldots, 0, 1, 0]\mathbf{G_m}$.

Observe that $c^m_{2^m-1}(10)$ appears in pairs 1). through 3). and the pair 11). in Theorem 2. Its complement, the codeword $c^m_{2^m-1}(01)$ appears in pair 1)., 4)., 5). and 10). For both these codewords, there is a non-zero component in the last, i.e. $(m+1)^{\text{st}}$ position in the corresponding message vectors. Note that $c^m_{2^m-2}(10)$ appears in pairs 8). and 9). and that its complement $c^m_{2^m-2}(01)$ appears in pairs 6). and 7). Furthermore, the sum of the last two entries in the message vectors corresponding to these two codewords is 1.

We may now try to find as large as possible a linear subcode of $C(m)$, in which no two codewords cause the identification problem under one deletion. The generator matrix $\hat{G}$ of this subcode can have at most m rows. Consider a matrix consisting of the top $m-1$ rows of $\mathbf{G_m}$, followed by a binary sum of the last two rows of $\mathbf{G_m}$. Now, $\hat{G}$ has m rows and no linear combinations of its rows give rise to codewords causing the identification problem.

Therefore, if instead of using $C(m)$ of rate $\frac{m+1}{2^m}$ we use its linear subcode $\hat{C}(m)$ of rate $\frac{m}{2^m}$, generated by the top $m-1$ rows of $\mathbf{G_m}$ and the binary sum of the last two rows of $\mathbf{G_m}$, we are able to eliminate the identification problem under a single deletion while preserving the linearity of the code and suffering a very small loss in the overall rate.

Remark 5 *Since a code is immune to a single insertion if and only if it is immune to a single deletion [35] it immediately follows that in the subcode $\hat{C}(m)$ no two codewords cause the identification problem under a single insertion.* ■

In the next section, we will see that the subcode we have constructed is not just immune to single deletions; it also has good identification under the single deletion model and under the single repetition model.

In principle, one can utilize the run-length structure of the RM(1,m) code to determine large subcodes immune to any number of deletions, or even to combinations of repetitions and deletions. Such analysis quickly becomes very complicated. A detailed analysis of the identification problem for the RM(1,m) codes under the infinite SNR channel model which permits *both* one repetition and one deletion is contained in [16].

4.3 Decoding the modified RM(1,m) code over a channel with synchronization and substitution errors

In the previous section we described how to extract a linear subcode of the RM(1,m) code that is immune to a single deletion. We now consider the behavior of such a subcode over channels in which, in addition to substitution errors, synchronization errors can occur as well. We consider two kinds of channel models for synchronization errors: channels where the deletion of a single bit can occur, and channels where the repetition of a single bit can occur. As in subsection 4.1 we assume in each case that the receiver learns from the sampled output whether a deletion (respectively, a repetition) has occurred or not.

In this section, we first determine the minimum distance between the sets of strings obtained by applying a deletion of a single bit to codewords of the modified RM(1,m) code. We then compute the minimum distance between the sets of strings obtained by applying the repetition of a single bit to codewords of the modified RM(1,m) code. Finally, in each case, we propose a bounded distance decoding algorithm for up to half the corresponding minimum distance over a channel where, in addition to substitution errors, the synchronization error can occur as well. The complexity of the decoding algorithm is of the same order as that of the usual fast Hadamard transform based decoding for RM(1,m) codes.

4.3.1 Minimum distance

In this subsection we first determine the minimum Hamming distance between the elements of sets associated with distinct codewords of the modified code that result from the deletion of a single bit. Let $\hat{C}(m)$ denote the code whose generator matrix consists of the top $m-1$ rows of $\mathbf{G_m}$ and the binary sum of the last two rows of $\mathbf{G_m}$. The code $\hat{C}(m)$ is immune to one deletion by construction.

We first make the following observation:

Remark 6 *For $m \geq 2$ a codeword $\mathbf{c}$ of $C(m)$ belongs to $\hat{C}(m)$ if and only if all its quadruplets starting at position i for i mod $4 \equiv 1$ are all '1111' or '0000' or all are '0110' or '1001'.*

In the remainder we will call quadruplets of $\mathbf{c}$ starting at position i for i mod $4 \equiv 1$ *constituent* quadruplets.

For $\mathbf{c} \in \hat{C}(m)$, let $S_d(\mathbf{c})$ denote the set of strings obtained by applying the deletion of a single bit to $\mathbf{c}$.

Lemma 9 *For $\mathbf{c_a}, \mathbf{c_b}$ distinct codewords in $\hat{C}(m)$, let $D(\mathbf{c_a}, \mathbf{c_b})$ be the smallest Hamming distance between $\mathbf{s_a}$ and $\mathbf{s_b}$ where $\mathbf{s_a}$ ranges over all elements in the set $S_d(\mathbf{c_a})$ and $\mathbf{s_b}$*

ranges over all elements in the set $S_d(\mathbf{c_b})$*. Let* $D^m_{min} = \min_{\mathbf{c_a},\mathbf{c_b} \in \hat{C}(m), \mathbf{c_a} \neq \mathbf{c_b}} D(\mathbf{c_a}, \mathbf{c_b})$*. Then for* $m > 2$*,* $D^m_{min} = 2^{m-3}$*. Further, for* $m \geq 3$*,* $D(\mathbf{c_a}, \mathbf{c_b}) = 2^{m-3}$ *only for* $\mathbf{c_a} = c^m_j(01)$ *or* $c^m_j(10)$ *for* $j = 3 \times 2^{m-3}$*, and* $\mathbf{c_b}$ *either* $\mathbf{c_a} + c^m_1(10)$*, or* $\mathbf{c_a} + c^m_1(01)$*, or vice versa, and in addition for* $m \geq 4$*,* $\mathbf{c_b}$ *is also* $\mathbf{c_a} + c^m_1(00)$*, or vice versa.*

Proof: Suppose that $\mathbf{c_a}$ experiences a deletion in position p_1 and $\mathbf{c_b}$ experiences a deletion in position p_2. Without loss of generality we can assume that $p_1 < p_2$. Let $p'_1 = \lfloor (p_1 - 1)/4 \rfloor 4 + 1$ and let $p'_2 = \lfloor (p_2 - 1)/4 \rfloor 4 + 4$, so that p'_1 denotes the first position of the constituent quadruplet p_1 belongs to, and p'_2 denotes the last position of the constituent quadruplet p_2 belongs to. We also let l_1 be the Hamming distance between the strings $\mathbf{c_a}(1, p'_1 - 1)$ and $\mathbf{c_b}(1, p'_1 - 1)$, l_2 be the Hamming distance between the strings $\mathbf{c_a}(p'_1, p'_2)$ and $\mathbf{c_b}(p'_1, p'_2)$, and l_3 be the Hamming distance between the strings $\mathbf{c_a}(p'_2 + 1, 2^m)$ and $\mathbf{c_b}(p'_2 + 1, 2^m)$, where the notation $\mathbf{c_i}(p, q)$ indicates the substring of the codeword $\mathbf{c_i}$ starting at position p and ending at position q. In addition, let $n_c = (p'_2 - p'_1 + 1)/4$ be the total number of quadruplets spanned by positions p'_1 and p'_2. By the standard properties of a Reed-Muller(1,m) code, $l_1 + l_2 + l_3$ is either 2^{m-1} or 2^m. Let $\tilde{\mathbf{c}}_\mathbf{a} = [\mathbf{c_a}(1, p_1 - 1) | \mathbf{c_a}(p_1 + 1, 2^m)]$, and $\tilde{\mathbf{c}}_\mathbf{b} = [\mathbf{c_b}(1, p_2 - 1) | \mathbf{c_b}(p_2 + 1, 2^m)]$. Then the Hamming distance $d_H(\tilde{\mathbf{c}}_\mathbf{a}, \tilde{\mathbf{c}}_\mathbf{b})$ between $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ is

$$\begin{aligned} d_H(\tilde{\mathbf{c}}_\mathbf{a}, \tilde{\mathbf{c}}_\mathbf{b}) &= d_H\left(\tilde{\mathbf{c}}_\mathbf{a}(1, p'_1 - 1), \tilde{\mathbf{c}}_\mathbf{b}(1, p'_1 - 1)\right) \\ &+ d_H\left(\tilde{\mathbf{c}}_\mathbf{a}(p'_1, p'_2 - 1), \tilde{\mathbf{c}}_\mathbf{b}(p'_1, p'_2 - 1)\right) \\ &+ d_H\left(\tilde{\mathbf{c}}_\mathbf{a}(p'_2, 2^m - 1), \tilde{\mathbf{c}}_\mathbf{b}(p'_2, 2^m - 1)\right). \end{aligned}$$

Observe that the first term in the sum is simply l_1 and that the last term is l_3. We let $\tilde{l}_2$ denote the middle term, $d_H\left(\tilde{\mathbf{c}}_{\mathbf{a}}(p_1', p_2'-1), \tilde{\mathbf{c}}_{\mathbf{b}}(p_1', p_2'-1)\right)$, and we establish the relationship between $\tilde{l}_2$ and l_2 for all choices of $\mathbf{c}_{\mathbf{a}}$ and $\mathbf{c}_{\mathbf{b}}$, from which the bound on the overall distance will follow.

1) Let us first consider the case when the constituent quadruplets in $\mathbf{c}_{\mathbf{a}}$ are '0110' and '1001' and in $\mathbf{c}_{\mathbf{b}}$ are '0000' and '1111', or vice versa. In this case, the Hamming distance between $\mathbf{c}_{\mathbf{a}}$ and $\mathbf{c}_{\mathbf{b}}$ is 2^{m-1}, and the constituent quadruplet pairs starting at the same positions in $\mathbf{c}_{\mathbf{a}}$ and $\mathbf{c}_{\mathbf{b}}$ each contribute 2 to the overall Hamming distance. Therefore, $l_2 = 2n_c$.

If $p_2' - p_1' = 3$, then the deletions occur in the same quadruplet, l_2 is 2 to begin with, and the Hamming distance between $\tilde{\mathbf{c}}_{\mathbf{a}}(p_1', p_2'-1)$ and $\tilde{\mathbf{c}}_{\mathbf{b}}(p_1', p_2'-1)$ is at least 1, which can be verified by checking all cases. Hence the Hamming distance between $\tilde{\mathbf{c}}_{\mathbf{a}}$ and $\tilde{\mathbf{c}}_{\mathbf{b}}$ is at least $2^{m-1}-1$, which is strictly greater than 2^{m-3}.

Now suppose that $p_2' - p_1' > 3$. Then $n_c > 1$. After the deletions, the Hamming distance between $\tilde{\mathbf{c}}_{\mathbf{a}}(p_1'+4i, p_1'+3+4i)$ and $\tilde{\mathbf{c}}_{\mathbf{b}}(p_1'+4i, p_1'+3+4i)$, for $1 \leq i \leq n_c - 2$ is at least 1, as is the distance between the substrings $\tilde{\mathbf{c}}_{\mathbf{a}}(p_2'-3, p_2'-1)$ and $\tilde{\mathbf{c}}_{\mathbf{b}}(p_2'-3, p_2'-1)$, and between the substrings $\tilde{\mathbf{c}}_{\mathbf{a}}(p_1', p_1'+3)$ and $\tilde{\mathbf{c}}_{\mathbf{b}}(p_1', p_1'+3)$ (which again can be verified by checking all cases). Then, $\tilde{l}_2 \geq (n_c - 2) \times 1 + 1 \times 1 + 1 \times 1 = l_2/2$.

Since the Hamming distance between $\tilde{\mathbf{c}}_{\mathbf{a}}(p_1', p_2'-1)$ and $\tilde{\mathbf{c}}_{\mathbf{b}}(p_1', p_2'-1)$ is at least $l_2/2$, the Hamming distance between $\tilde{\mathbf{c}}_{\mathbf{a}}$ and $\tilde{\mathbf{c}}_{\mathbf{b}}$ is then at least $l_1 + l_2/2 + l_3$, which is lower bounded by 2^{m-2}, and thus strictly greater than 2^{m-3}.

2) Suppose now that the constituent quadruplets are '0000' and '1111' in both $\mathbf{c}_{\mathbf{a}}$ and

$\mathbf{c_b}$. The Hamming distance between $\mathbf{c_a}$ and $\mathbf{c_b}$ is either 2^{m-1} or 2^m, and the constituent quadruplet pairs starting at the same positions in $\mathbf{c_a}$ and $\mathbf{c_b}$ each contribute either 0 or 4 to the overall Hamming distance. In the segment spanning positions p_1' and p_2' in $\mathbf{c_a}$ and $\mathbf{c_b}$, $l_2/4$ of the constituent quadruplet pairs each contribute 4 to the overall Hamming distance between $\mathbf{c_a}$ and $\mathbf{c_b}$.

If $p_2' - p_1' = 3$, the deletions occur in the same quadruplet, and l_2 is either 0 or 4. Then $d_H\left(\tilde{\mathbf{c}}_\mathbf{a}(p_1', p_1'+2), \tilde{\mathbf{c}}_\mathbf{b}(p_1', p_1'+2)\right)$ is either 0 (if $l_2 = 0$) or 3 (if $l_2 = 4$). The overall distance between $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ is thus at least $2^{m-1} - 1$, which is bigger than 2^{m-3} for all $m \geq 3$.

If $p_2' - p_1' > 3$ we consider constituent quadruplets contained within positions $p_1' + 4$ and $p_2' - 4$ in $\mathbf{c_a}$ and $\mathbf{c_b}$ that start at the same positions and we denote the set of their starting positions by Tot (the set Tot is non empty as long as p_1' and p_2' belong to non-adjacent quadruplets). Let *Com* be the subset of Tot whose elements index complementary quadruplets in $\mathbf{c_a}$ and $\mathbf{c_b}$. Then the Hamming distance between the quadruplets $\mathbf{c_a}(i+1, i+4)$ and $\mathbf{c_b}(i, i+3)$, and consequently between $\tilde{\mathbf{c}}_\mathbf{a}(i, i+3)$ and $\tilde{\mathbf{c}}_\mathbf{b}(i, i+3)$ for $i \in$ *Com* is at least 3. In addition, if p_2' belongs to the constituent quadruplets of $\mathbf{c_a}$ and $\mathbf{c_b}$ that are complements of each other, the distance between $\tilde{\mathbf{c}}_\mathbf{a}(p_2'-3, p_2'-1)$ and $\tilde{\mathbf{c}}_\mathbf{b}(p_2'-3, p_2'-1)$ is at least 3. Similarly, if p_1' belongs to the constituent quadruplets of $\mathbf{c_a}$ and $\mathbf{c_b}$ that are complements of each other, the distance between $\tilde{\mathbf{c}}_\mathbf{a}(p_1', p_1'+3)$ and $\tilde{\mathbf{c}}_\mathbf{b}(p_1', p_1'+3)$ is also at least 3. Therefore, the Hamming distance between $\tilde{\mathbf{c}}_\mathbf{a}(p_1', p_2'-1)$ and $\tilde{\mathbf{c}}_\mathbf{b}(p_1', p_2'-1)$ is at least $3 \times l_2/4$, thereby making the overall distance between $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ be at least

$l_1 + 3l_2/4 + l_3$, which is again strictly greater than 2^{m-3}.

3) Finally, consider $\mathbf{c_a}$ and $\mathbf{c_b}$ with constituent quadruplets ‘0110’ and ‘1001’. Again, the Hamming distance between $\mathbf{c_a}$ and $\mathbf{c_b}$ is either 2^{m-1} or 2^m, and the constituent quadruplet pairs starting at the same positions in $\mathbf{c_a}$ and $\mathbf{c_b}$ each contribute either 0 or 4 to it.

For the case when $p_2' - p_1' = 3$, l_2 is either 0 or 4, so that the Hamming distance between $\tilde{\mathbf{c}}_\mathbf{a}(p_1', p_2' - 1)$ and $\tilde{\mathbf{c}}_\mathbf{b}(p_1', p_2' - 1)$ is either at least 0 for l_2 = 0 or at least 1 for l_2 = 4. In the former case, the Hamming distance between $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ is at least 2^{m-1}, and in latter case it is least $2^{m-1} - 3$. In particular, for $m \geq 4$, $2^{m-1} - 3$ is strictly bigger than 2^{m-3}. For $m = 3$, $2^{m-1} - 3 = 2^{m-3}$. Then $\mathbf{c_a}$ and $\mathbf{c_b}$ would have to be complements of each other in the quadruplets experiencing deletions, and would have to be the same in their other quadruplet. For $\mathbf{c_a} + \mathbf{c_b}$ being either ‘00001111’ or ‘11110000’, $\mathbf{c_a}$ is then either $c_3^3(10)$ or $c_3^3(01)$ and $\mathbf{c_b}$ is either $\mathbf{c_a} + c_1^3(10)$ or $\mathbf{c_a} + c_1^3(01)$ or vice versa. Observe that these are precisely the codeword pairs listed at the beginning of the proof for $j = 3$ and $m = 3$.

If $p_2' - p_1' > 3$ we again consider constituent quadruplets contained within positions $p_1' + 4$ and $p_2' - 4$ in $\mathbf{c_a}$ and $\mathbf{c_b}$ that start at the same positions and we denote the set of their starting positions by Tot (the set Tot is non empty as long as p_1' and p_2' belong to non-adjacent quadruplets). Let *Com* be the subset of Tot whose elements index complement quadruplets in $\mathbf{c_a}$ and $\mathbf{c_b}$, and let $Sam = Tot - Com$.

Then the Hamming distance between $\mathbf{c_a}(i + 1, i + 4)$ and $\mathbf{c_b}(i, i + 3)$ for $i \in$ *Com* is either 1 or 2, and we denote their total number by s_1^1 and s_2^1, respectively such that $s_1^1 + s_2^1 = |\mathit{Com}|$. The Hamming distance between $\mathbf{c_a}(i + 1, i + 4)$ and $\mathbf{c_b}(i, i + 3)$ for $i \in$

Sam is either 2 or 3, and we similarly denote their total number by s_2^0 and s_3^0, respectively, where $s_2^0 + s_3^0 = |\mathit{Sam}|$. In addition, if p_2' belongs to the constituent quadruplets of $\mathbf{c_a}$ and $\mathbf{c_b}$ that are complements of each other, the distance between $\tilde{\mathbf{c}}_\mathbf{a}(p_2' - 3, p_2' - 1)$ and $\tilde{\mathbf{c}}_\mathbf{b}(p_2' - 3, p_2' - 1)$ is either 1, 2, or 3, which we denote by t_2^1, and is either 0, 1, or 2 if those two quadruplets in $\mathbf{c_a}$ and $\mathbf{c_b}$ are the same, in which case we denote it by t_2^0. Let $J_2 = 1$ if these two quadruplets are complements and let $J_2 = 0$ otherwise. Finally, the distance between $\tilde{\mathbf{c}}_\mathbf{a}(p_1', p_1' + 3)$ and $\tilde{\mathbf{c}}_\mathbf{b}(p_1', p_1' + 3)$ is 0, 1, 2, or 3 if the corresponding quadruplets in $\mathbf{c_a}$ and $\mathbf{c_b}$ are the same, when is denoted by t_1^0, and is 1, 2, 3, or 4 if these quadruplets are complements, when is denoted by t_1^1. Let $J_1 = 1$ for complement quadruplets and $J_1 = 0$ for the same.

The overall Hamming distance between $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ is then

$$l_1 + J_1 t_1^1 + (1 - J_1) t_1^0 + s_1^1 + 2s_2^1 + 2s_2^0 + 3s_3^0 + J_2 t_2^1 + (1 - J_2) t_2^0 + l_3.$$

Observe that $s_1^1 + s_2^1 + J_1 + J_2 = l_2/4$.

Since $t_1^1 \geq 1$ and $t_2^1 \geq 1$ we have

$$\begin{aligned} d_H(\tilde{\mathbf{c}}_\mathbf{a}, \tilde{\mathbf{c}}_\mathbf{b}) &\geq l_1 + J_1 + s_1^1 + s_2^1 + J_2 + l_3 \\ &= l_1 + \frac{l_2}{4} + l_3 \\ &\geq \frac{1}{4} d_H(\mathbf{c}_a, \mathbf{c}_b) \geq 2^{m-3}\ . \end{aligned}$$

Equality holds in this sequence of inequalities if and only if $d_H(\mathbf{c}_a, \mathbf{c}_b) = 2^{m-1}$ (i.e. $\mathbf{c}_a$ and $\mathbf{c}_b$ are not complements of each other), $l_1 = 0$, $l_3 = 0$, $s_2^0 = 0$, $s_3^0 = 0$, $s_2^1 = 0$, and one of the following four cases holds:

(a) $(J_1, J_2) = (1, 1)$ and $(t_1^1, t_2^1) = (1, 1)$,

(b) $(J_1, J_2) = (0, 1)$ and $(t_1^0, t_2^1) = (0, 1)$,

(c) $(J_1, J_2) = (1, 0)$ and $(t_1^1, t_2^0) = (1, 0)$,

(d) $(J_1, J_2) = (0, 0)$ and $(t_1^0, t_2^0) = (0, 0)$.

Since $l_1 = 0$ and $l_3 = 0$, all constituent quadruplets in $\mathbf{c_a}(1, p'_1 - 1)$ and $\mathbf{c_b}(1, p'_1 - 1)$ as well as in $\mathbf{c_a}(p'_2 + 1, 2^m)$ and $\mathbf{c_b}(p'_2 + 1, 2^m)$ are pairwise the same. Since s_2^0 = s_3^0 = 0, the constituent quadruplets spanning positions $p'_1 + 4$ through $p'_2 - 4$ in $\mathbf{c_a}$ and $\mathbf{c_b}$ are pairwise complements of each other. Moreover, since $s_2^1 = 0$ they are actually alternating ‘1001’ and ‘0110’ in $\mathbf{c_a}$ and are alternating ‘0110’ or ‘1001’ in $\mathbf{c_b}$, or vice versa. The quadruplets in $\mathbf{c_a}$ and $\mathbf{c_b}$ to which p'_1 (p'_2) belongs are the same if $J_1 = 0$ ($J_2 = 0$), and otherwise they are complements. Therefore, for all four cases, $\mathbf{c_a} + \mathbf{c_b}$ is of the type ‘0.01.10.0’, with possibly one run of zeros empty (but not both as then $\mathbf{c_a}$ and $\mathbf{c_b}$ would be complements), and is such that it belongs to $\hat{C}(m)$. Specifically, $\mathbf{c_a} + \mathbf{c_b}$ is either $c_1^m(10)$, $c_1^m(01)$, or $c_1^m(00)$, and by $p'_2 - p'_1 > 3$, m is at least 3. Since $c_1^m(00) \notin \hat{C}(m)$ for $m = 3$, no new pairs can result from this analysis in this case, so we may assume from now on that $m \geq 4$.

Let p_l and p_r be the positions of the leftmost and the rightmost 1 in $\mathbf{c_a} + \mathbf{c_b}$. Then p'_1 is either p_l or $p_l - 4$, depending on the value of J_1 and on the format of $\mathbf{c_a} + \mathbf{c_b}$, and likewise p'_2 is either p_r or $p_r + 4$, depending on the value of J_2 and $\mathbf{c_a} + \mathbf{c_b}$. In particular, for $\mathbf{c_a} + \mathbf{c_b}$ equal to $c_1^m(10)$, J_1 must be 1 and $p'_1 = p_l$, and for $\mathbf{c_a} + \mathbf{c_b}$ equal to $c_1^m(01)$, J_2 must be 1 and $p'_2 = p_r$.

For $m > 5$, since there are at least $1/2(2^{m-2}) - 2$ contiguous alternating ‘0110’ and

‘1001’ (or vice versa) spanning positions $p'_1 + 4$ and $p'_2 - 4$ in $\mathbf{c_a}$, and since (p_l, p_r) is either $(1, 2^{m-1})$ or $(2^{m-1}+1, 2^m)$ or $(2^{m-2}+1, 3 \times 2^{m-2})$, by the concatenation principle it follows that all quadruplets spanning positions p_l and p_r in $\mathbf{c_a}$ are alternating ‘0110’ and ‘1001’, or vice versa. It then follows that under the set of constraints $(l_1 = 0, l_3 = 0, s_2^0 = 0, s_3^0 = 0, s_2^1 = 0)$, $\mathbf{c_a}$ can only be $c_j^m(01)$ or $c_j^m(10)$ for $j = 3 \times 2^{m-3}$, and $\mathbf{c_b}$ is then $\mathbf{c_a} + c_1^m(10)$, $\mathbf{c_a} + c_1^m(01)$, or $\mathbf{c_a} + c_1^m(00)$, or vice versa. It remains to determine whether these candidate codeword pairs satisfy one of the (a) through (d) cases.

From the structure of the candidate codeword pairs, it follows for example that all six codeword pairs achieve D_{min}^m for $(J_1, J_2, t_1^1, t_2^1) = (1, 1, 1, 1)$, with deletions in positions $(p_1, p_2) = (2^{m-1}+1, 2^m)$ for $\mathbf{c_a} + \mathbf{c_b} = c_1^m(01)$, in positions $(p_1, p_2) = (1, 2^{m-1})$ for $\mathbf{c_a} + \mathbf{c_b} = c_1^m(10)$, and in positions $(p_1, p_2) = (2^{m-2}+1, 3 \times 2^{m-2})$ for $\mathbf{c_a} + \mathbf{c_b} = c_1^m(00)$, such that both individual values of $\mathbf{c_a}$ and $\mathbf{c_b}$ per pair are possible.

For $m = 4$ and $m = 5$ the pairs of codewords $\{\mathbf{c_a}, \mathbf{c_b}\}$ achieving the proposed D_{min}^m can be identified directly, and they have the same format as codewords achieving D_{min}^m for $m > 5$. This concludes the proof of the lemma. ■

We next determine the minimum Hamming distance between the elements of sets associated with distinct codewords of the modified code that result from the repetition of a single bit.

For $\mathbf{c} \in \hat{C}(m)$, let $S_r(\mathbf{c})$ denote the set of strings obtained by applying the repetition of a single bit to $\mathbf{c}$. Recall that $S_d(\mathbf{c})$ denotes the set of strings obtained by applying the deletion of a single bit to $\mathbf{c}$.

Lemma 10 *For $\mathbf{c_a}, \mathbf{c_b}$ distinct codewords in $\hat{C}(m)$, let $R(\mathbf{c_a}, \mathbf{c_b})$ be the smallest Hamming distance between $\mathbf{t_a}$ and $\mathbf{t_b}$ where $\mathbf{t_a}$ ranges over all elements in the set $S_r(\mathbf{c_a})$ and $\mathbf{t_b}$ ranges over all elements in the set $S_r(\mathbf{c_b})$. Let $R^m_{min} = \min_{\mathbf{c_a},\mathbf{c_b} \in \hat{C}(m), \mathbf{c_a} \neq \mathbf{c_b}} R(\mathbf{c_a}, \mathbf{c_b})$. Then for $m > 2$, $R^m_{min} = 2^{m-3} + 1$. Further, for $m \geq 3$, $R(\mathbf{c_a}, \mathbf{c_b}) = 2^{m-3} + 1$ only for $\mathbf{c_a} = c^m_j(01)$ or $c^m_j(10)$ for $j = 3 \times 2^{m-3}$, and $\mathbf{c_b}$ either $\mathbf{c_a} + c^m_1(10)$, or $\mathbf{c_a} + c^m_1(01)$, or vice versa, and in addition for $m \geq 4$, $\mathbf{c_b}$ is also $\mathbf{c_a} + c^m_1(00)$, or vice versa.*

Proof: We first observe that $0 \leq R(\mathbf{c_a}, \mathbf{c_b}) - D(\mathbf{c_a}, \mathbf{c_b}) \leq 2$, where $D(\mathbf{c_a}, \mathbf{c_b})$ is as defined in Lemma 9. To see this, consider $\mathbf{s_a}$ obtained by deleting a bit in $\mathbf{c_a}$ in position p_a, and $\mathbf{s_b}$ obtained by deleting a bit in $\mathbf{c_b}$ in position p_b. For $p_a < p_b$, $d_H(\mathbf{s_a}, \mathbf{s_b})$ is

$$\begin{aligned} d_H(\mathbf{s_a}, \mathbf{s_b}) &= d_H(\mathbf{c_a}(1, p_a - 1), \mathbf{c_b}(1, p_a - 1)) \\ &+ d_H(\mathbf{c_a}(p_a + 1, p_b), \mathbf{c_b}(p_a, p_b - 1)) \\ &+ d_H(\mathbf{c_a}(p_b + 1, n), \mathbf{c_b}(p_b + 1, n)). \end{aligned}$$

For $\mathbf{t_a} \in S_r(\mathbf{c_a})$ and $\mathbf{t_b} \in S_r(\mathbf{c_b})$ such that the bit in position p_b (p_a) is the bit that gets repeated in $\mathbf{t_a}$ ($\mathbf{t_b}$), write $d_H(\mathbf{t_a}, \mathbf{t_b})$ as

$$\begin{aligned} d_H(\mathbf{t_a}, \mathbf{t_b}) &= d_H(\mathbf{c_a}(1, p_a - 1), \mathbf{c_b}(1, p_a - 1)) + d_1 \\ &+ d_H(\mathbf{c_a}(p_a + 1, p_b), \mathbf{c_b}(p_a, p_b - 1)) + d_2 \\ &+ d_H(\mathbf{c_a}(p_b + 1, n), \mathbf{c_b}(p_b + 1, n)), \end{aligned}$$

where $d_1 = \mathbf{c_a}(p_a) + \mathbf{c_b}(p_a)$ and $d_2 = \mathbf{c_a}(p_b) + \mathbf{c_b}(p_b)$. Therefore, $0 \leq d_H(\mathbf{t_a}, \mathbf{t_b}) - d_H(\mathbf{s_a}, \mathbf{s_b}) = d_1 + d_2 \leq 2$. A similar argument gives the same inequality for $p_a > p_b$. Taking the minimum over all (p_a, p_b), the claim of this paragraph follows.

By Lemma 9, $D_{min}^m = 2^{m-3}$, so that R_{min}^m is at most $2^{m-3}+2$. We use the nomenclature introduced in Lemma 9 to determine the codewords $\mathbf{c_a}$, $\mathbf{c_b}$ for which $D(\mathbf{c_a}, \mathbf{c_b})$ yields the proposed bound on $R(\mathbf{c_a}, \mathbf{c_b})$, i.e. the codewords for which $D(\mathbf{c_a}, \mathbf{c_b})$ is at most $2^{m-3} + 1$.

1) Let us first consider the case when the constituent quadruplets in $\mathbf{c_a}$ are ‘0110’ and ‘1001’ and in $\mathbf{c_b}$ are ‘0000’ and ‘1111’, or vice versa. From the proof of Lemma 9 it follows that $D(\mathbf{c_a}, \mathbf{c_b})$ is at least 2^{m-2}, as is then $R(\mathbf{c_a}, \mathbf{c_b})$. The proposed lower bound can only be met for $m = 3$. By checking all cases, for $m = 3$, it follows that $R(\mathbf{c_a}, \mathbf{c_b})$ is at least 4, thus exceeding the proposed lower bound.

2) Suppose now that the constituent quadruplets are ‘0000’ and ‘1111’ in both $\mathbf{c_a}$ and $\mathbf{c_b}$. From the proof of Lemma 9 it follows that $D(\mathbf{c_a}, \mathbf{c_b})$ is at least $3 \times 2^{m-3}$, and thus $R(\mathbf{c_a}, \mathbf{c_b})$ is strictly greater than the proposed lower bound.

3) Finally, consider $\mathbf{c_a}$ and $\mathbf{c_b}$ with constituent quadruplets ‘0110’ and ‘1001’.

We assume that p_1', p_2', $\tilde{\mathbf{c}}_\mathbf{a}$, $\tilde{\mathbf{c}}_\mathbf{b}$, l_1, l_2, l_3 as well as t_1^0, t_1^1, t_2^0, t_2^1, s_1^1, s_2^1, s_2^0, and s_3^0 are as defined in the proof of Lemma 9.

In the notation of Lemma 9, if $p_2' - p_1' = 3$, $D(\mathbf{c_a}, \mathbf{c_b})$ is at least $2^{m-1} - 3$, thus exceeding $2^{m-3} + 1$ for $m \geq 4$. For $m = 3$, there are four codewords in $\hat{C}(m)$ having ‘0110’ and ‘1001’ as constituent quadruplets. It follows by direct checking that $R(\mathbf{c_a}, \mathbf{c_b}) = 2^{m-3} + 1 = 2$ only for $\mathbf{c_a}$ =‘01101001’ or ‘10010110’ and $\mathbf{c_b} = \mathbf{c_a} + c_1^3(01)$ or $\mathbf{c_b} = \mathbf{c_a} + c_1^3(10)$, or vice versa. In the remainder we will assume $m \geq 4$.

For $p_2' - p_1' > 3$, in the notation of Lemma 9, the overall Hamming distance between

$\tilde{\mathbf{c}}_{\mathbf{a}}$ and $\tilde{\mathbf{c}}_{\mathbf{b}}$ is

$$\begin{aligned} d_H(\tilde{\mathbf{c}}_{\mathbf{a}}, \tilde{\mathbf{c}}_{\mathbf{b}}) &= l_1 + J_1 t_1^1 + (1 - J_1) t_1^0 + s_1^1 + 2s_2^1 + \\ & \quad 2s_2^0 + 3s_3^0 + J_2 t_2^1 + (1 - J_2) t_2^0 + l_3, \end{aligned} \tag{4.1}$$

where $s_1^1 + s_2^1 + J_1 + J_2 = l_2/4$.

As established in Lemma 9, for $d_H(\tilde{\mathbf{c}}_{\mathbf{a}}, \tilde{\mathbf{c}}_{\mathbf{b}})$ to equal 2^{m-3} for $m \geq 4$ it is necessary that $\mathbf{c}_{\mathbf{a}}$ is $c_j^m(01)$ or $c_j^m(10)$ for $j = 3 \times 2^{m-3}$, and $\mathbf{c}_{\mathbf{b}}$ is either $\mathbf{c}_{\mathbf{a}} + c_1^m(10)$, $\mathbf{c}_{\mathbf{a}} + c_1^m(01)$, or $\mathbf{c}_{\mathbf{a}} + c_1^m(00)$, or vice versa. By direct checking it follows that $R(\mathbf{c}_{\mathbf{a}}, \mathbf{c}_{\mathbf{b}})$ is precisely $2^{m-3}+1$ for all six codeword pairs (since the deletions in $\mathbf{c}_{\mathbf{a}}$ and $\mathbf{c}_{\mathbf{b}}$ yielding D_{min}^m are such that one of them belongs to a run of size 2 and the other belongs to a run of size 1).

It remains to determine $\mathbf{c}_{\mathbf{a}}$, $\mathbf{c}_{\mathbf{b}}$ for which $d_H(\tilde{\mathbf{c}}_{\mathbf{a}}, \tilde{\mathbf{c}}_{\mathbf{b}})$ equals $2^{m-3}+1$, and such that both deletions occur in runs of size bigger than 1. Using the expression in (9.77) it follows that $d_H(\mathbf{c}_{\mathbf{a}}, \mathbf{c}_{\mathbf{b}}) = 2^{m-1}$, $l_1 = 0$, $l_3 = 0$, $s_2^0 = 0$, $s_3^0 = 0$, (use ($\triangle$) as a shorthand for set of conditions ($\triangle$)) and one of the following holds:

(a) $(J_1, J_2) = (0, 1)$ and $(t_1^0, t_2^1, s_2^1) = (0, 2, 0)$,

(b) $(J_1, J_2) = (0, 1)$ and $(t_1^0, t_2^1, s_2^1) = (1, 1, 0)$,

(c) $(J_1, J_2) = (1, 0)$ and $(t_1^1, t_2^0, s_2^1) = (2, 0, 0)$,

(d) $(J_1, J_2) = (1, 0)$ and $(t_1^1, t_2^0, s_2^1) = (1, 1, 0)$,

(e) $(J_1, J_2) = (0, 0)$ and $(t_1^0, t_2^0, s_2^1) = (1, 0, 0)$,

(f) $(J_1, J_2) = (0, 0)$ and $(t_1^0, t_2^0, s_2^1) = (0, 1, 0)$,

(g) $(J_1, J_2) = (1, 1)$ and $(t_1^1, t_2^1, s_2^1) = (2, 1, 0)$,

(h) $(J_1, J_2) = (1, 1)$ and $(t_1^1, t_2^1, s_2^1) = (1, 2, 0)$,

(i) $(J_1, J_2) = (0, 1)$ and $(t_1^0, t_2^1, s_2^1) = (0, 1, 1)$,

(j) $(J_1, J_2) = (1, 0)$ and $(t_1^1, t_2^0, s_2^1) = (1, 0, 1)$,

(k) $(J_1, J_2) = (0, 0)$ and $(t_1^0, t_2^0, s_2^1) = (0, 0, 1)$,

(l) $(J_1, J_2) = (1, 1)$ and $(t_1^1, t_2^1, s_2^1) = (1, 1, 1)$,

Since $l_1 = 0$ and $l_3 = 0$, all constituent quadruplets in $\mathbf{c_a}(1, p'_1 - 1)$ and $\mathbf{c_b}(1, p'_1 - 1)$ as well as in $\mathbf{c_a}(p'_2 + 1, 2^m)$ and $\mathbf{c_b}(p'_2 + 1, 2^m)$ are pairwise the same. Since s_2^0 = s_3^0 = 0, the constituent quadruplets spanning positions $p'_1 + 4$ through $p'_2 - 4$ in $\mathbf{c_a}$ and $\mathbf{c_b}$ are pairwise complements of each other. Therefore, for all cases, $\mathbf{c_a} + \mathbf{c_b}$ is of the type '0.01.10.0', with possibly one run of zeros empty (but not both as then $\mathbf{c_a}$ and $\mathbf{c_b}$ would be complements), and is such that it belongs to $\hat{C}(m)$.

First observe that for cases (a) through (h), the common constraint $s_2^1 = 0$, along with the constraint set (△) is the same as the set of constraints on the same parameters established in the proof of Lemma 9. As given in the proof of Lemma 9 under the set of constraints $(l_1 = 0, l_3 = 0, s_2^0 = 0, s_3^0 = 0, s_2^1 = 0)$, $\mathbf{c_a}$ can only be $c_j^m(01)$ or $c_j^m(10)$ for $j = 3 \times 2^{m-3}$, and $\mathbf{c_b}$ can only then be $\mathbf{c_a} + c_1^m(10)$, $\mathbf{c_a} + c_1^m(01)$, or $\mathbf{c_a} + c_1^m(00)$, or vice versa. Observe that these codeword pairs are already established in the earlier case when $d_H(\tilde{\mathbf{c}}_\mathbf{a}, \tilde{\mathbf{c}}_\mathbf{b}) = 2^{m-3}$ was analyzed (though it can also be verified that these candidate codeword pairs satisfy at least one of the (a) through (h) cases, and with deletions in appropriate runs of size 2 result in strings with Hamming distance $2^{m-3} + 1$).

The remaining cases (i) through (l) all share the same constraint that $s_2^1 = 1$, which implies that all constituent quadruplets spanning positions $p'_1 + 8$ through $p'_2 - 4$ in $\mathbf{c_a}$ and

$\mathbf{c_b}$ are the complements of their left neighboring quadruplets, except for one constituent quadruplet which is the same as its left neighboring quadruplet.

Let p_l and p_r be the positions of the leftmost and the rightmost 1 in $\mathbf{c_a} + \mathbf{c_b}$, so that (p_l, p_r) is either $(1, 2^{m-1})$ or $(2^{m-1}+1, 2^m)$ or $(2^{m-2}+1, 3 \times 2^{m-2})$. Depending on the values of J_1 and J_2 and the structure of $\mathbf{c_a} + \mathbf{c_b}$, p_1' is either $p_l - 4$ or p_l and p_2' is either $p_r + 4$ or p_r, so that all constituent quadruplets spanning positions $p_l + 8$ through $p_r - 4$ in $\mathbf{c_a}$ and $\mathbf{c_b}$ are the complements of their left neighboring quadruplets with the exception of one constituent quadruplet which is the same as its left neighboring quadruplet.

For $m > 5$, by the concatenation principle it follows that this singular constituent quadruplet must be the one starting at the position $2^{m-1}+1$ so that $\mathbf{c_a} + \mathbf{c_b}$ is $c_1^m(00)$. Moreover, the constituent quadruplets spanning positions $p_l = 2^{m-2}+1$ and 2^{m-1} are then alternating '0110' and '1001' (or vice versa), followed by alternating '1001' and '0110' (or vice versa) that span positions $2^{m-1}+1$ and $p_r = 3 \times 2^{m-2}$. As a result, it follows that $\mathbf{c_a}$ is $c_j^m(11)$ or $c_{j-1}^m(00)$ for $j = 3 \times 2^{m-3} - 1$, and $\mathbf{c_b}$ is either $\mathbf{c_a} + c_1^m(10)$, $\mathbf{c_a} + c_1^m(01)$, or $\mathbf{c_a} + c_1^m(00)$, or vice versa. However, in all cases, when $d_H(\tilde{\mathbf{c}}_\mathbf{a}, \tilde{\mathbf{c}}_\mathbf{b}) = 2^{m-3}+1$, not both deletion errors can be in runs of size bigger than 1 (which can be verified by direct checking of all possible constraint sets as given by (i) through (l)), and therefore $R(\mathbf{c_a}, \mathbf{c_b})$ is strictly greater than $2^{m-3}+1$.

For $m = 4, 5$ it can be checked directly that the only codeword pairs achieving $d_H(\tilde{\mathbf{c}}_\mathbf{a}, \tilde{\mathbf{c}}_\mathbf{b}) = 2^{m-3}+1$ under the current constraints are the same as for $m > 5$. Again, not both deletions can occur in runs of size 2, and thus $R(\mathbf{c_a}, \mathbf{c_b})$ is again strictly greater than $2^{m-3}+1$. ■

4.3.2 Decoding algorithm

In this subsection, we first propose a bounded distance decoding scheme for $\hat{C}(m)$ which corrects one deletion and up to $2^{m-4}-1$ substitution errors. We outline the algorithm and discuss its correctness and complexity.

A common technique for decoding a codeword in a Reed-Muller (1,m) code that has experienced a certain number of substitution errors involves computing a fast Hadamard transform of the received string, [41, §4, Ch.14]. Specifically, the received string s (of length n) is multiplied by a Hadamard matrix H_n to form $\mathrm{s}H_n$. The computation is done efficiently by starting with the binary string s of length $n = 2^m$ and carrying out m stages, each of which involves $n = 2^m$ additions of integers, to return the integer valued string $\mathrm{s}H_n$ of length n. Subsequently one needs to find the coordinate in this integer string of maximum absolute value. The complexity of the overall algorithm is therefore normally quoted as $O(n \log n)$.

In our situation, let $\mathbf{c} \in \hat{C}(m)$ for $m \geq 5$ be the transmitted codeword. Let s be the received string obtained from $\mathbf{c}$ by one deletion and at most $2^{m-4} - 1$ substitution errors. Thus, the received string s is of length $n - 1$. The objective is to recover $\mathbf{c}$ from s. In principle one could construct strings of length n by inserting either 0 or 1 at each position in s and compare each resulting string with candidate codewords from $\hat{C}(m)$, which would be equivalent to performing $2n$ standard decoding operations. The complexity of such an algorithm would be $O(n^2 \log n)$. However, it is possible to do much better.

For any codeword $\tilde{\mathbf{c}} \in \hat{C}(m)$, write $\tilde{\mathbf{c}} = \left[\tilde{\mathbf{c}}^L | \tilde{\mathbf{c}}^R\right]$, where $\tilde{\mathbf{c}}^L$ and $\tilde{\mathbf{c}}^R$ are each of length

2^{m-1}. In particular, the transmitted codeword $\mathbf{c}$ is written as $\mathbf{c} = \left[\mathbf{c}^L|\mathbf{c}^R\right]$. From the received string $\mathbf{s}$ we create $\mathbf{s}^L = [s(1)\dots s(2^{m-1})]$ and $\mathbf{s}^R = [s(2^{m-1})\dots s(2^m-1)]$. Each of these strings is of length 2^{m-1}.

If the location of the deletion is in the second half of the codeword, then $\mathbf{s}^L$ is obtained from $\mathbf{c}^L$ by at most $2^{m-4}-1$ substitution errors. Further, for every $\tilde{\mathbf{c}} \in \hat{C}(m)$ other than $\mathbf{c}$ and $\mathbf{c} + c_1^m(01)$ we have

$$\begin{aligned} d_H(\mathbf{s}^L, \tilde{\mathbf{c}}^L) &\geq d_H(\mathbf{c}^L, \tilde{\mathbf{c}}^L) - d_H(\mathbf{s}^L, \mathbf{c}^L) \\ &\geq 2^{m-2} - (2^{m-4}-1) \\ &> 2^{m-4}\,. \end{aligned}$$

If one uses the fast Hadamard transform to compute $\left[\mathbf{s}^L|\mathbf{0}\right] H_n$, the coordinate with maximum absolute value will then correspond to either the pair comprised of $\mathbf{c}$ and its bitwise complement or the pair comprised of $\mathbf{c}+c_1^m(01)$ and its bitwise complement. Further, there will be at most two competing locations for the maximum absolute value.

Similarly, if the location of the deletion is in the first half of the codeword, then $\mathbf{s}^R$ is obtained from $\mathbf{c}^R$ by at most $2^{m-4}-1$ substitution errors, so by using the fast Hadamard transform to compute $\left[\mathbf{0}|\mathbf{s}^R\right] H_n$, the coordinate with maximum absolute value will correspond to either the pair comprised of $\mathbf{c}$ and its bitwise complement or the pair comprised of $\mathbf{c} + c_1^m(10)$ and its bitwise complement. Again, there will be at most two competing locations for the maximum absolute value.

Thus, in $O(n \log n)$ operations we will be presented with at most 8 candidates for the transmitted codeword. We may now go the naive step of considering all the $2n$ strings of length n got by inserting either 0 or 1 at each position in s and compare each resulting string with each of these 8 candidate codewords. In $O(n)$ operations we will arrive at the true codeword.

There are some obvious inefficiencies in the algorithm just described. For instance, it is not really necessary to compare the received string with the columns of H_n that correspond to strings in $C(m)$ that are not in $\hat{C}(m)$. An analysis of this inefficiency could save a constant factor. The second stage could also undoubtedly be improved, but this is less interesting because the overall complexity is dominated by the first stage. Since using the first stage as described has the significant practical advantage that the existing hardware which is used to decode when there is no deletion can also be used when there is a deletion, we have preferred to describe the overall algorithm as above.

Finally, along similar lines, we propose a bounded distance decoding scheme for $\hat{C}(m)$ for $m \geq 4$ which corrects one repetition and up to 2^{m-4} substitution errors and discuss its correctness and complexity. Let s be the received string obtained from $\mathbf{c}$ by one repetition and at most 2^{m-4} substitution errors. Thus, the received string s is of length $n+1$. As before, for any codeword $\tilde{\mathbf{c}} \in \hat{C}(m)$, write $\tilde{\mathbf{c}} = \left[\tilde{\mathbf{c}}^L|\tilde{\mathbf{c}}^R\right]$, where $\tilde{\mathbf{c}}^L$ and $\tilde{\mathbf{c}}^R$ are each of length 2^{m-1}. The transmitted codeword $\mathbf{c}$ is written as $\mathbf{c} = \left[\mathbf{c}^L|\mathbf{c}^R\right]$. From the received string s we create $\mathrm{s}^L = [s(1)\ldots s(2^{m-1})]$ and $\mathrm{s}^R = [s(2^{m-1})\ldots s(2^m-1)]$. Each of these strings is of length 2^{m-1}.

If the location of the repetition is in the second half of the codeword, then s^L is got from $\mathbf{c}^L$ by at most 2^{m-4} substitution errors. Further, for every $\tilde{\mathbf{c}} \in \hat{C}(m)$ other than $\mathbf{c}$ and $\mathbf{c} + c_1^m(01)$ we have

$$\begin{aligned} d_H(\mathbf{s}^L, \tilde{\mathbf{c}}^L) &\geq d_H(\mathbf{c}^L, \tilde{\mathbf{c}}^L) - d_H(\mathbf{s}^L, \mathbf{c}^L) \\ &\geq 2^{m-2} - 2^{m-4} \\ &> 2^{m-4} . \end{aligned}$$

If one uses the fast Hadamard transform to compute $\left[\mathbf{s}^L|\mathbf{0}\right] H_n$ the coordinate with maximum absolute value will then correspond to either the pair comprised of $\mathbf{c}$ and its bitwise complement or the pair comprised of $\mathbf{c} + c_1^m(01)$ and its bitwise complement. Further, there will be at most two competing locations for the maximum absolute value.

Similarly, if the location of the repetition is in the first half of the codeword, then $\mathbf{s}^R$ is got from $\mathbf{c}^R$ by at most 2^{m-4} substitution errors, so by using the fast Hadamard transform to compute $\left[\mathbf{0}|\mathbf{s}^R\right] H_n$, the coordinate with maximum absolute value will correspond to either the pair comprised of $\mathbf{c}$ and its bitwise complement or the pair comprised of $\mathbf{c} + c_1^m(10)$ and its bitwise complement. Again, at most two locations will compete for the maximum absolute value.

Thus in $O(n \log n)$ operations, there will be at most 8 candidates for the transmitted codeword. We may now again follow the naive way and consider all the $2n$ strings of length n obtained by inserting either 0 or 1 at each position in s and compare each resulting string with each of these 8 candidate codewords. Using this approach, in $O(n)$ operations the true codeword will follow.

4.4 Summary and Concluding Remarks

In this chapter we studied the performance of a Reed-Muller RM(1,m) code, as an instance of a substitution-error correcting code, over channels in which, in addition to substitution errors, a sampling error can cause synchronization errors. Specifically, we studied the cases where the synchronization error results in the deletion of a single bit and where it results in the repetition of a single bit. The model we worked with is aimed at handling the kinds of errors that can occur in a variety of applications, such as magnetic recording and wireless transmission, in the absence of adequate timing recovery. Our approach to handling synchronization errors is to start with a good substitution-error correcting code, to analyze which codeword pairs cause the identification problem, and then find a linear subcode of as high a rate as possible that would both provide protection against substitution errors and be robust to the synchronization errors. The rate loss incurred from using the subcode and the increase in the complexity of the decoding algorithm should of course be reasonably small for such an approach to work.

Using the structural properties previously proved in Chapter 3, we provided an analysis that is combinatorially much tighter than might be needed for our immediate concerns. These combinatorial results may also be of independent interest. Specifically, we enumerated all pairs of codewords of the RM(1,m) codes that suffer from an identification problem over a channel allowing for the deletion of a single bit. We introduced a pruned linear subcode of the RM(1,m) code, with the loss of one information bit, which does not suffer from the identification problem under the deletion of a single bit. Given a pair of codewords in the pruned code the appropriate notion of distance between them over a channel permitting synchronization errors is the minimum Hamming distance between any pair of strings which are derived respectively from each codeword after the application of such synchronization error. We gave a combinatorially tight analysis of the the minimum distance of the pruned code for this notion of distance for both the case of the deletion of a single bit and the case of the repetition of a single bit. Specifically, we explicitly identified all pairs of codewords of the pruned code for which the post-synchronization error Hamming distance equals the corresponding post-synchronization minimum distance of the pruned code.

Finally, we provided a bounded distance decoding algorithm, suitable for the use of the pruned code over a channel where in addition to possibly one deletion error (respectively one repetition error), substitution errors can occur as well. The complexity of this algorithm is of the same order as that of the usual fast Hadamard transform based decoding for the RM(1,m) code. What is more, the proposed algorithm can in fact be essentially run on the same hardware platform as in the case without synchronization errors.

Chapter 5

Repetition Error Correcting Binary Sets

Inspired by the scenario discussed in Chapter 2, in this Chapter we study the problem of finding maximally sized subsets of binary strings (codes) that are immune to a given number r of repetitions, in the sense that no two strings in the code can give rise to the same string after r repetitions.

In Section 5.1 we mention related work on the related problem of insertion/deletion correcting codes. In Section 5.2 we introduce an auxiliary transformation that converts our problem into that of creating subsets of binary strings immune to the insertions of 0's. In Section 5.3 we focus on subsets of binary strings immune to single repetitions. We present explicit constructions of such subsets and use number theoretic techniques to give explicit formulas for their cardinalities. Our constructions here are asymptotically optimal. In Section 5.4 we discuss subsets of binary strings immune to multiple repetitions. Our constructions here are asymptotically within a constant factor of the best known upper bounds and asymptotically better, by a constant factor than the best previously known such constructions, due to Levenshtein [34].

5.1 Related Work

A closely related problem of studying codes capable of overcoming a certain number of insertions and deletions was first studied by Levenshtein [35] where it was shown that the so-called Varshamov-Tenengolts codes [57] originally proposed for the correction of asymmetric errors are capable of overcoming one deletion or one insertion. They were also shown to be asymptotically optimal. They have been further studied in [21] and [6]. In [51] further results on their cardinalities were obtained. Extensions to constructions for overcoming multiple insertions and deletions remains a difficult problem. Literature on this problem includes [25], [54].

Another interesting related problem is that of interactive communication when two users own a copy of a data stream, one corrupted and the other one uncorrupted. The owner of the uncorrupted version wants to communicate as few bits as possible to the other user so that the other user can restore the original data. A method to communicate the minimal number of bits when the data stream is corrupted by modifying the sizes of the runs of equal symbols is proposed in [44]. The difference from our model is the assumption that these communicated bits are transmitted without themselves being subjected to repetition errors, whereas in our model, any of the transmitted bits may be repeated.

5.2 Auxiliary Transformation

To construct a binary, s repetition correcting code C of length n we first construct an auxiliary code $\tilde{C}$ of length $m = n - 1$ which is s ‘0’-insertion correcting code. These two codes are related through the following transformation.

Suppose $\mathbf{c} \in C$. We let $\tilde{\mathbf{c}} = \mathbf{c} \times T_n$ mod 2, where T_n is $n \times n - 1$ matrix, satisfying

$$T_n(i,j) = \begin{cases} 1, & \text{if } i = j, j+1 \\ 0, & \text{else.} \end{cases} \tag{5.1}$$

Now, the repetition in $\mathbf{c}$ in position p corresponds to the insertion of ‘0’ in position $p-1$ in $\tilde{\mathbf{c}}$, and weight($\tilde{\mathbf{c}}$) = number of runs in $\mathbf{c}$ -1. We let $\tilde{C}$ be the collection of strings of length $n-1$ obtained by applying T_n to all strings C. Note that $\mathbf{c}$ and its complement both map into the same string in $\tilde{C}$.

It is thus sufficient to construct a code of length $n-1$ capable of overcoming s ‘0’-insertions and apply inverse T_n transformation to obtain s repetitions correcting code of length n.

5.3 Single Repetition Error Correcting Set

Following the analysis of Sloane [51] and Levenshtein [35] of the related so-called Varshamov-Tenengolts codes [57] known to be capable of overcoming one deletion or one insertion, let A_w^m be the set of all binary strings of length m and w ones, for $0 \leq w \leq m$. Partition A_w^m based on the value of the first moment of each string. More specifically, let

$S_{w,k}^{m,t}$ be the subset of A_w^m such that

$$S_{w,k}^{m,t} = \{(s_1, s_2, ..., s_m) | \sum_{i=1}^{m} i \times s_i \equiv k \bmod t\}. \tag{5.2}$$

In the subsequent analysis we say that an element of $S_{w,k}^{m,t}$ has the first moment congruent to k mod t.

Lemma 11 *Each subset* $S_{w,k}^{m,w+1}$ *is a single '0'-insertion correcting code.*

Proof: Suppose the string s′ is received. We want to uniquely determine the codeword s $= (s_1, s_2, ..., s_m) \in S_{w,k}^{m,w+1}$ such that s′ is the result of inserting at most one zero in s.

If the length of s′ is m, conclude that no insertion occurred, and that s $=$ s′.

If the length of s′ is $m+1$, a zero has been inserted. For s′ $= (s_1^{'}, s_2^{'}, ..., s_m^{'}, s_{m+1}^{'})$, compute $\sum_{i=1}^{m+1} i \times s_i^{'}$ mod $(w+1)$. Due to the insertion, $\sum_{i=1}^{m+1} i \times s_i^{'} = \sum_{i=1}^{m} i \times s_i + R_1$ where R_1 denotes the number of 1's to the right of the insertion. Note that R_1 is always between 0 and w.

Let k' be equal to $\sum_{i=1}^{m+1} i \times s_1^{'}$ mod $(w+1)$. If $k' = k$ the insertion occurred after the rightmost one, so we declare s to be the m leftmost bits in s′. If $k' > k$, R_1 is $k' - k$ and we declare s to be the string obtained by deleting the zero immediately preceding the rightmost $k' - k$ ones. Finally, if $k' < k$, R_1 is $w + 1 - k + k'$ and we declare s to be the string obtained by deleting the zero immediately preceding the rightmost $w + 1 - k + k'$ ones. ■

Before discussing the cardinality results it is worth point out that the construction presented here was used in [15] to thin the array-based LDPC code for improved performance under additive and a repetition error.

5.3.1 Cardinality Results

Since $|A_w^m| = \left(\begin{array}{c} m \\ \\ w \end{array} \right)$ there exists k such that

$$|S_{w,k}^{m,w+1}| \geq \frac{1}{w+1} \left(\begin{array}{c} m \\ \\ w \end{array} \right).$$

Since two codewords of different weights cannot result in the same string when at most one zero is inserted we may let $\tilde{C}$ be the union of largest sets $S_{w,k_w^*}^{m,w+1}$ over different weights w, i.e.

$$\tilde{C} = \bigcup_{w=1}^{m} S_{w,k_w^*}^{m,w+1},$$

where $S_{w,k_w^*}^{m,w+1}$ is the set of largest cardinality among all sets $S_{w,k}^{m,w+1}$ for $0 \leq k \leq w$. Thus, the cardinality of $\tilde{C}$ is at least

$$\sum_{w=0}^{m} \left(\begin{array}{c} m \\ \\ w \end{array} \right) \frac{1}{w+1} = \frac{1}{m+1} \left(2^{m+1} - 1\right).$$

The upper bound $U_1(m)$ on any set of strings each of length m capable of overcoming one insertion of a zero is derived in [34] to be

$$U_1(m) = \frac{2^{m+1}}{m}. \tag{5.3}$$

Hence the proposed construction is asymptotically optimal in the sense that the ratio of its cardinality to the largest possible cardinality approaches 1 as $n \to \infty$.

By applying inverse T_n transformation for $n = m+1$ to $\tilde{C}$ and noting that both pre-images under T_n can simultaneously belong to a repetition correcting set, we obtain a code of length n and of size at least $\frac{1}{n}\left(2^{n+1}-2\right)$, capable of correcting one repetition.

The cardinalities of the sets $S_{w,k}^{m,w+1}$ may be computed explicitly as we now show.

Recall that the Möbius function $\mu(x)$ of a positive integer $x = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$ for distinct primes $p_1, p_2, \ldots, p_k$ is defined as [3],

$$\mu(x) = \begin{cases} 1 & \text{for } x = 1 \\ (-1)^k & \text{if } a_1 = \cdots = a_k = 1 \\ 0 & \text{otherwise .} \end{cases} \tag{5.4}$$

and that the Euler function $\phi(x)$ denotes the number of integers y, $1 \leq y \leq x-1$ that are relatively prime with x. By convention $\phi(1) = 1$.

Lemma 12 *Let $g = gcd(m+1, w+1)$. The cardinality of $S_{w,k}^{m,w+1}$ is*

$$|S_{w,k}^{m,w+1}| = \frac{1}{m+1} \sum_{d|g} \begin{pmatrix} \frac{m+1}{d} \\ \frac{w+1}{d} \end{pmatrix} (-1)^{(w+1)(1+\frac{1}{d})} \phi(d) \frac{\mu\left(\frac{d}{gcd(d,k)}\right)}{\phi\left(\frac{d}{gcd(d,k)}\right)} \tag{5.5}$$

where $gcd(d,k)$ is the greatest common divisor of d and k, interpreted as d if $k = 0$.

Proof: Motivated by the analysis of Sloane [51] of the Varshamov-Tenengolts codes, let us introduce the function $f_{b,n}(U,V)$ in which the coefficient of $U^s V^k$, call it $g_{k,s}^b(n)$ represents the number of strings of length n, weight s and the first moment equal to $k \mod b$ (i.e.

$g^b_{k,s}(n) = |S^{n,b}_{s,k}|$,

$$f_{b,n}(U,V) = \sum_{k=0}^{b-1}\sum_{s=0}^{n} g^b_{k,s}(n)U^sV^k. \tag{5.6}$$

Observe that $f_{b,n}(U,V)$ can be written as a generating function

$$f_{b,n}(U,V) = \prod_{t=1}^{n}(1+UV^t) \mod (V^b-1)\ . \tag{5.7}$$

Let $a = e^{i\frac{2\pi}{b}}$ so that for $V = a^j$

$$f_{b,n}(U, e^{i\frac{2\pi j}{b}}) = \sum_{k=0}^{b-1}\sum_{s=0}^{n} g^b_{k,s}(n)U^s e^{i\frac{2\pi jk}{b}}\ . \tag{5.8}$$

By inverting this expression we can write

$$\begin{aligned} &\textstyle\sum_{s=0}^{n} g^b_{k,s}(n)U^s \\ = &\ \textstyle\frac{1}{b}\sum_{j=0}^{b-1} f_{b,n}(U, e^{i\frac{2\pi j}{b}})e^{-i\frac{2\pi jk}{b}} \qquad (5.9) \\ = &\ \textstyle\frac{1}{b}\sum_{j=0}^{b-1}\prod_{t=1}^{n}(1+Ue^{i\frac{2\pi jt}{b}})e^{-i\frac{2\pi jk}{b}}\ . \end{aligned}$$

Our next goal is to evaluate the coefficient U^b on the right hand side in (9.77). To do so we first evaluate the following expression

$$\prod_{t=1}^{b}(1+Ue^{i\frac{2\pi jt}{b}})\ . \tag{5.10}$$

Let $d_j = b/gcd(b,j)$ and $s_j = j/gcd(b,j)$, and write

$$\begin{aligned} & \prod_{t=1}^{b}(1+Ue^{i\frac{2\pi jt}{b}}) \\ = & \left(\prod_{t=1}^{d_j}(1+Ue^{i\frac{2\pi s_j t}{d_j}})\right)^{gcd(b,j)} \\ = & \left(1+U\sum_{t_1=1}^{d_j} e^{i\frac{2\pi s_j t_1}{d_j}}+\right. \\ & \quad U^2\sum_{t_1=1}^{d_j}\sum_{t_2=t_1+1}^{d_j} e^{i\frac{2\pi s_j (t_1+t_2)}{d_j}}+ \\ & \quad \left.+\cdots+U^{d_j}e^{i\frac{2\pi s_j (1+2+\cdots+d_j)}{d_j}}\right)^{gcd(b,j)} \quad . \end{aligned} \tag{5.11}$$

Since $gcd(d_j, s_j) = 1$, the set

$$V = \{e^{i\frac{2\pi s_j 1}{d_j}}, e^{i\frac{2\pi s_j 2}{d_j}} \ldots e^{i\frac{2\pi s_j d_j}{d_j}}\}$$

represents all distinct solutions of the equation

$$x^{d_j} - 1 = 0 \ . \tag{5.12}$$

For a polynomial equation $P(x)$ of degree d, the coefficient multiplying x^k is a scaled symmetric function of $d-k$ roots. Hence, symmetric functions involving at most $d_j - 1$ elements of V evaluate to zero. The symmetric function involving all elements of V, which is their product, evaluates to $(-1)^{d_j+1}$.

Therefore,

$$\prod_{t=1}^{b}(1+Ue^{i\frac{2\pi jt}{b}}) = \left(1+(-1)^{1+d_j}U^{d_j}\right)^{gcd(b,j)} \ . \tag{5.13}$$

Returning to the inner product in (9.77), let us first suppose that $b|n$. Then

$$\begin{aligned} & \textstyle\prod_{t=1}^{n}\left(1+Ue^{i\frac{2\pi jt}{b}}\right) \\ = & \left(\textstyle\prod_{t=1}^{b}\left(1+Ue^{i\frac{2\pi jt}{b}}\right)\right)^{n/b} \\ = & \left(1+(-1)^{1+d_j}U^{d_j}\right)^{gcd(b,j)n/b} \\ = & \textstyle\sum_{l=0}^{\frac{n}{d_j}}\begin{pmatrix}\frac{n}{d_j}\\ \\ l\end{pmatrix}(-1)^{l(1+d_j)}U^{ld_j}\ . \end{aligned} \tag{5.14}$$

Thus (9.77) becomes

$$\begin{aligned} & \sum_{s=0}^{n} g_{k,s}^{b}(n)U^{s} \\ = & \ \frac{1}{b}\sum_{j=0}^{b-1}\sum_{l=0}^{\frac{n}{d_j}}\begin{pmatrix}\frac{n}{d_j}\\ \\ l\end{pmatrix}(-1)^{l(1+d_j)}U^{d_jl}e^{-i\frac{2\pi jk}{b}}\ . \end{aligned}$$

We now regroup the terms whose j's yield the same d_j's

$$\begin{aligned} \sum_{s=0}^{n} g_{k,s}^{b}(n)U^{s} = \frac{1}{b}\sum_{d|b}\sum_{l=0}^{\frac{n}{d}}\begin{pmatrix}\frac{n}{d}\\ \\ l\end{pmatrix}(-1)^{l(1+d)}U^{dl} \\ \times \sum_{j:gcd(j,b)=b/d,0\leq j\leq b-1} e^{-i\frac{2\pi jk}{b}}. \end{aligned}$$

The rightmost sum can also be written as

$$\sum_{j:gcd(j,b)=b/d,0\leq j\leq b-1} e^{-i\frac{2\pi jk}{b}} = \sum_{s:0\leq s\leq d-1,gcd(s,d)=1} e^{-i\frac{2\pi sk}{d}}\ . \tag{5.15}$$

This last expression is known as the Ramanujan sum [3] and simplifies to

$$\sum_{s:0\leq s\leq d-1,gcd(s,d)=1} e^{-i\frac{2\pi sk}{d}} = \phi(d)\frac{\mu\left(\frac{d}{gcd(d,k)}\right)}{\phi\left(\frac{d}{gcd(d,k)}\right)}\ . \tag{5.16}$$

Now the coefficient of U^b in (9.77) is

$$\frac{1}{b}\sum_{d|b}\left(\begin{array}{c}\frac{n}{d}\\ \\ \frac{b}{d}\end{array}\right)(-1)^{\frac{b}{d}(1+d)}\phi(d)\frac{\mu\left(\frac{d}{gcd(d,k)}\right)}{\phi\left(\frac{d}{gcd(d,k)}\right)} \tag{5.17}$$

which is precisely the number of strings of length n, weight b, and the first moment congruent to $k \mod b$, i.e. $|S^{n,b}_{b,k}|$.

Consider the set of strings described by $S^{m,w+1}_{w,k}$ for $m = n - 1$ and $w = b - 1$, i.e. $S^{m,w+1}_{w,k} = S^{n-1,b}_{b-1,k}$. If we append '1' to each such string we would obtain a fraction of b/n of all strings that belong to the set $S^{n,b}_{b,k}$. To see why this is true, first note that the cardinality of the set $S^{n-1,b}_{b-1,k}$ and of the subset $T^{n}_{b,k}$ of $S^{n,b}_{b,k}$ which contains all strings ending in '1' is the same (since when a '1' is appended to each element of the set $S^{n-1,b}_{b-1,k}$, the resulting set contains strings of length n, weight b and first moment congruent to $(k + n) \mod b$, which is also congruent to $k \mod b$ since by assumption $b|n$). It is thus sufficient to show that $|T^{n}_{b,k}| = \frac{b}{n}|S^{n,b}_{b,k}|$. Let $A_k = |S^{n,b}_{b,k}|$. Write $A_k = \sum_{u,u|b} A_k(n, b, \frac{n}{u})$, where $A_k(n, b, v)$ denotes the number of strings of length n, weight b, first moment congruent to $k \mod b$, and with period v. Consider a string accounted for in $A_k(n, b, \frac{n}{u})$. Its single cyclic shift has the first moment congruent to $(k + b) \mod b$ and is thus also accounted for in $A_k(n, b, \frac{n}{u})$. Since $\frac{n}{u}$ is the period, and since $\frac{b}{u}$ is the weight per period, fraction $\frac{b/u}{n/u}$ of $A_k(n, b, \frac{n}{u})$ represents distinct strings that end in '1', have length n, weight b, first moment congruent to $k \mod b$, and period $\frac{n}{u}$. Thus, $|T^{n}_{b,k}| = \sum_{u,u|b} \frac{b/u}{n/u} A_k(n, b, \frac{n}{u}) = \frac{b}{n} A_k$, as required.

Therefore, the cardinality of $S^{m,w+1}_{w,k}$ is b/n times the expression in (9.78),

$$|S^{m,w+1}_{w,k}| =$$

$$\frac{1}{m+1}\sum_{d|w+1}\left(\begin{array}{c}\frac{m+1}{d}\\ \\ \frac{w+1}{d}\end{array}\right)(-1)^{\frac{w+1}{d}(1+d)}\phi(d)\frac{\mu\left(\frac{d}{gcd(d,k)}\right)}{\phi\left(\frac{d}{gcd(d,k)}\right)}\ . \tag{5.18}$$

Notice that the last expression is the same as the one proposed in Lemma 12 with $gcd(m+1, w+1) = w+1$.

Now suppose that b is not a factor of n. We work with $f_{g,n}(U,V)$ as in (5.7) where $g = gcd(n,b)$ and get

$$\sum_{s=0}^{n} g^{g}_{k,s}(n)U^{s} = \frac{1}{g}\sum_{d|g}\sum_{l=0}^{\frac{n}{d}}\left(\begin{array}{c}\frac{n}{d}\\ \\ l\end{array}\right)(-1)^{l(1+d)}U^{dl}$$
$$\times \sum_{j:gcd(j,g)=g/d,0\leq j\leq g-1} e^{-i\frac{2\pi jk}{g}}\ .$$

Thus the coefficient of U^{b} here is

$$\frac{1}{g}\sum_{d|g}\left(\begin{array}{c}\frac{n}{d}\\ \\ \frac{b}{d}\end{array}\right)(-1)^{\frac{b}{d}(1+d)}\phi(d)\frac{\mu\left(\frac{d}{gcd(d,k)}\right)}{\phi\left(\frac{d}{gcd(d,k)}\right)}\ . \tag{5.19}$$

This is the number of strings of length n, weight b, and the first moment congruent to $k \mod g$, namely it is the cardinality of the set $S^{n,g}_{b,k}$. Let $B_k = |S^{n,g}_{b,k}|$. Write $B_k = \sum_{u,u|g} B_k(n,b,\frac{n}{u})$ where $B_k(n,b,v)$ denotes the number of strings of length n, weight b, first moment congruent to $k \mod g$ and with period v. By cyclically shifting a string of length n, weight b, first moment congruent to $k \mod g$ and with period n/u for n/u steps, and observing that each cyclic shift also has the first moment congruent to $k \mod g$, it follows that a fraction $\frac{b/u}{n/u}$ of $B_k(n,b,\frac{n}{u})$ represents the number of strings that end in ’1’, have length n, weight b, first moment congruent to $k \mod g$, and period $\frac{n}{u}$. Thus a fraction b/n of B_k denotes the number of strings that end in '1', are of length n, weight b, and have the first moment congruent to $k \mod g$. Since each string of length $n-1$, weight $b-1$, and the first moment congruent to $k \mod g$ produces a unique string that ends in '1', is of length n, weight b, and has the first moment congruent to $k \mod g$ by appending '1', it follows that $\frac{b}{n}B_k$ is also the number of strings of length $n-1$, weight $b-1$, and the first moment congruent to $k \mod g$. Thus the number of strings given by $S^{n-1,g}_{b-1,k}$ is also $\frac{b}{n}B_k$.

Consider again cyclic shifts of a string of length n, weight b, the first moment congruent to $k \mod g$ and with period n/u. A fraction b/u of these shifts produce strings with a '1' in the last position. Let us consider one such string s_0. Its first $n-1$ bits correspond to a string of length $n-1$, weight $b-1$, and the first moment congruent to $k \mod g$. This $n-1$-bit string has the first moment congruent to $k_0 \mod b$ for some k_0. Cyclically shift s_0 for t_1 places until the first time '1' again appears in the nth position, and call the resulting string s_1 (Since $b > g$ and $u|g$, $b/u > 1$, and thus $s_1 \neq s_0$). The first $n-1$ bits of s_1 correspond to a string of length $n-1$, weight $b-1$, and the first moment congruent to $k_1 \equiv k_0 + t_1(b-1) + t_1 - n \mod g \equiv k_0 + t_1 b - n \mod b \equiv k_0 - gy \mod b$, where $y = \frac{n}{g}$. Cyclically shift s_1 for for t_2 places until the first time '1' again appears in the nth position, and call the resulting string s_2. The first $n-1$ bits of s_2 correspond to a string of length $n-1$, weight $b-1$, and the first moment congruent to $k_2 \equiv k_0 - gy + t_2(b-1) + t_2 - n \mod g \equiv k_0 - gy + t_2 b - n \mod b \equiv k_0 - 2gy \mod b$. Each subsequent cyclic shift with '1' in the last place gives a string s_i whose first $n-1$ bits have the first moment congruent to $k_i \equiv k_0 - igy \mod b$. The last such string, $s_{b/u-1}$, before the string s_0 is encountered again has the left $n-1$ bit substring whose first moment is congruent to $k_{b/u-1} \equiv k_0 - (\frac{b}{u} - 1)gy \mod b$. Note that the sequence $\{k_0, k_1, k_2, \ldots, k_{b/u-1}\}$ is periodic with period z (here gcd$(y, g) = 1$ by construction), where $z = \frac{b}{g}$. Since $z|\frac{b}{u}$, each of k_0, k_1 through $k_{\frac{b}{g}-1}$ appear equal number of times in this sequence. Consequently, the number of strings in the set $S^{n-1,b}_{b-1,k_i}$ is $\frac{g}{b}$ of the size of the set $S^{n-1,g}_{b-1,k}$ for every $k_i \equiv ig + k \mod b$.

Therefore $|S^{m,w+1}_{w,k}|$ is

$$\begin{aligned} |S^{m,w+1}_{w,k}| &= \tfrac{b}{n}\tfrac{g}{b}|S^{n,g}_{b,k}| \\ &= \tfrac{1}{m+1}\textstyle\sum_{d|g} \left(\begin{array}{c} \frac{m+1}{d} \\ \\ \frac{w+1}{d} \end{array}\right) (-1)^{(w+1+\frac{1}{d}(1+w))}\phi(d)\frac{\mu\left(\frac{d}{gcd(d,k)}\right)}{\phi\left(\frac{d}{gcd(d,k)}\right)} \end{aligned} \tag{5.20}$$

which completes the proof of the lemma. ■

5.3.2 Connection with necklaces

It is interesting to briefly visit the relationship between optimal single insertion of a zero correcting codes and combinatorial objects known as necklaces [24].

A necklace consisting of n beads can be viewed as an equivalence class of strings of length n under cyclic shift (rotation).

Let us consider two-colored necklaces of length n with b black beads and $n - b$ white beads. It is known that the total number of distinct necklaces is [24]

$$T(n) = \frac{1}{n}\sum_{d|gcd(n,b)} \left(\begin{array}{c} \frac{n}{d} \\ \\ \frac{b}{d} \end{array}\right) \phi(d)\ . \tag{5.21}$$

In general necklaces may exhibit periodicity. However, consider, for example for the case $gcd(n, b) = 1$. Then there are

$$\frac{1}{n}\left(\begin{array}{c} n \\ \\ b \end{array}\right)$$

distinct necklaces, all of which are aperiodic. Now assume that $b + 1|n$ and note that this implies $gcd(n + 1, b + 1) = 1$. Suppose we label each necklace beads in the increasing order 1 through n and we rotate each necklace by one position at the time relative to this labeling. At each step we sum mod $b + 1$ the positions of b black beads. For each necklace each of residues k, $0 \leq k \leq b$ is encountered $n/(b + 1)$ times. The total number of times each residue k is encountered is thus

$$\frac{1}{b+1}\left(\begin{array}{c} n \\ \\ b \end{array}\right) = \frac{1}{n+1}\left(\begin{array}{c} n+1 \\ \\ b+1 \end{array}\right),$$

which as expected equals the number of binary strings of weight b, length n, and the first moment congruent to k mod $b + 1$ (same for all k).

5.4 Multiple Repetition Error Correcting Set

We now present an explicit construction of a multiple repetition error correcting set and discuss its cardinality.

Let $\mathbf{a} = (a_1, a_2, ..., a_r)$ for $r \geq 1$, and consider the set $\hat{S}(m, w, \mathbf{a}, p)$ for $w \geq 1$ defined as

$$\begin{aligned} \hat{S}(m,w,\mathbf{a},p) = \{ \quad & \mathbf{s} = (s_1, s_2, ...s_m) \in \{0,1\}^m : \\ & v_0 = 0, v_{w+1} = m+1, \text{ and } v_i \text{ is the position of the } i^{\text{th}} \text{ 1 in s for } 1 \le i \le w, \\ & b_i = v_i - v_{i-1} - 1, \text{ for } 1 \le i \le w+1, \\ & \textstyle\sum_{i=1}^{m} s_i = w, \\ & \textstyle\sum_{i=1}^{w+1} i b_i \equiv a_1 \bmod p, \\ & \textstyle\sum_{i=1}^{w+1} i^2 b_i \equiv a_2 \bmod p, \\ & \qquad \vdots \\ & \textstyle\sum_{i=1}^{w+1} i^r b_i \equiv a_r \bmod p \ \}. \end{aligned} \tag{5.22}$$

The set $\hat{S}(m,0,\mathbf{0},p)$ contains just the all-zeros string. Let $\mathbf{a_0} = \mathbf{0}$ and let $\hat{S}\,(m,(\mathbf{a_1},p_1),(\mathbf{a_2},p_2),...,(\mathbf{a_m},p_m))$ be defined as

$$\hat{S}\,(m,(\mathbf{a_1},p_1),(\mathbf{a_2},p_2),...,(\mathbf{a_m},p_m)) = \bigcup_{l=0}^{m} \hat{S}(m,l,\mathbf{a_l},p_l), \tag{5.23}$$

where $b_1, \ldots, b_{w+1}$ denote the sizes of the *bins* of 0's between successive 1's.

Lemma 13 *If each p_l is prime and $p_l >$ max(r,l), the set $\hat{S}\,(m,(\mathbf{a_1},p_1),(\mathbf{a_2},p_2),...,(\mathbf{a_m},p_m))$, provided it is non empty, is r-insertions of zeros correcting.*

Proof: It suffices to show that each non-empty set $\hat{S}(m,l,\mathbf{a_l},p_l)$ is r-insertions of zeros correcting. This is obvious for $l = 0$. For $l > 0$ suppose a string $\mathbf{x} \in \hat{S}(m,l,\mathbf{a_l},p_l)$ is transmitted. After experiencing r insertions of zeros, it is received as a string $\mathbf{x}'$. We now show that $\mathbf{x}$ is always uniquely determined from $\mathbf{x}'$.

Let $i_1 \leq i_2 \leq ... \leq i_r$ be the (unknown) indices of the bins of zeros that have experienced insertions. For each j, $1 \leq j \leq r$, compute $a'_j \equiv \sum_{i=1}^{w+1} i^j b'_i \bmod p_l$, where b'_i is the size of the i^{th} bin of zeros of $\mathbf{x}'$,

$$\begin{aligned} a'_j &\equiv \textstyle\sum_{i=1}^{w+1} i^j b'_i \bmod p_l \\ &\equiv a_j + (i_1^j + i_2^j + ... + i_r^j) \bmod p_l, \end{aligned} \tag{5.24}$$

where a_j is the j^{th} entry in the residue vector $\mathbf{a_l}$ (to lighten the notation the subscript l in a_j is omitted).

By collecting the resulting expressions over all j, and setting $a''_j \equiv a'_j - a_j \bmod p_l$, we arrive at

$$E_r = \begin{cases} a''_1 \equiv i_1 + i_2 + ... + i_r \bmod p_l \\ a''_2 \equiv i_1^2 + i_2^2 + ... + i_r^2 \bmod p_l \\ \cdots\cdots\cdots \\ a''_r \equiv i_1^r + i_2^r + ... + i_r^r \bmod p_l. \end{cases} \tag{5.25}$$

The terms on the right hand side of the congruency constraints are known as power sums in r variables. Let S_k denote the k^{th} power sum mod p_l of $\{i_1, i_2, ..., i_r\}$,

$$S_k \equiv i_1^k + i_2^k + ... + i_r^k \bmod p_l, \tag{5.26}$$

and let Λ_k denote the k^{th} elementary symmetric function of $\{i_1, i_2, ..., i_r\}$ modp_l,

$$\Lambda_k \equiv \sum_{v_1 < v_2 < ... < v_k} i_{v_1} i_{v_2} \cdots i_{v_k} \bmod p_l. \tag{5.27}$$

Using Newton's identities over $GF(p_l)$ which relate power sums to symmetric functions of the same variable set, and are of the type

$$S_k - \Lambda_1 S_{k-1} + \Lambda_2 S_{k-2} - ... + (-1)^{k-1} \Lambda_{k-1} S_1 + (-1)^k k \Lambda_k = 0, \tag{5.28}$$

for $k \leq r$, we can obtain an equivalent system of r equations:

$$\widetilde{E}_r = \begin{cases} d_1 \equiv \sum_{j=1}^{r} i_j \bmod p_l \\ d_2 \equiv \sum_{j<k} i_j i_k \bmod p_l \\ \cdots\cdots\cdots \\ d_t \equiv \prod_{j=1}^{r} i_j \bmod p_l, \end{cases} \tag{5.29}$$

where each residue d_k is computed recursively from $\{d_1, ..., d_{k-1}\}$ and $\{a_1^{''}, a_2^{''}, ...a_k^{''}\}$. Specifically, since the largest coefficient in (5.28) is r, and $r < p_l$ by construction, the last term in (5.28) never vanishes due to the multiplication by the coefficient k.

Consider now the following equation:

$$\prod_{j=1}^{r}(x - i_j) \equiv 0 \bmod p_l, \tag{5.30}$$

and expand it into the standard form

$$x^r + c_{r-1}x^{r-1} + ... + c_1 x + c_0 \equiv 0 \bmod p_l. \tag{5.31}$$

By collecting the same terms in (5.30) and (5.31), it follows that $d_k \equiv (-1)^k c_{r-k} \bmod p_l$. Furthermore, by the Lagrange's Theorem, the equation (5.31) has at most r solutions. Since $i_r \leq p_l$ all incongruent solutions are distinguishable, and thus the solution set of (5.31) is precisely the set $\{i_1, i_2, ..., i_r\}$.

Therefore, since the system E_r of r equations uniquely determines the set $\{i_1, i_2, ..., i_r\}$, the locations of the inserted zeros (up to the position within the bin they were inserted in) are uniquely determined, and thus $\mathbf{x}$ is always uniquely recovered from $\mathbf{x}'$. ■

Hence, $\hat{S}\left(m, (\mathbf{a_1}, p_1), (\mathbf{a_2}, p_2), ..., (\mathbf{a_m}, p_m)\right)$ is r-insertions of zeros correcting for p_l is prime and $p_l > \max(r, l)$.

In particular, for $r = 1$, the constructions in (5.2) and (5.22) are related as follows.

Lemma 14 *For p prime and $p > w$, the set $S_{w,a}^{m,p}$ defined in (5.2) equals the set $\hat{S}(m, w, \hat{a}, p)$ defined in (5.22), where $\hat{a} = f_{m,w} - a$ mod p for $f_{m,w} = (w+2)(2m-w+1)/2-(m+1)$.*

Proof: Consider a string $\mathbf{s} = (s_1, s_2, ..., s_m) \in S_{w,a}^{m,p}$, and let p_i be the position of the i^{th} 1 in $\mathbf{s}$, so that $\sum_{i=1}^{m} is_i = \sum_{i=1}^{w} p_i$. Observe that $p_k = \sum_{i=1}^{k} b_i + k$ where b_i is the size of the i^{th} bin of zeros in $\mathbf{s}$. Write

$$\begin{aligned}
&\textstyle\sum_{i=1}^{w} p_i + (m+1) = (b_1+1) + (b_1+b_2+2) + ...+ \\
&(b_1+b_2+...+b_w+w) + (b_1+b_2+...+b_{w+1}+w+1) = \\
&\textstyle\sum_{i=1}^{w+1}(w+2-i)b_i + (w+1)(w+2)/2 = \\
&(w+2)(m-w) + (w+1)(w+2)/2 - \textstyle\sum_{i=1}^{w+1} ib_i = \\
&(w+2)(2m-w+1)/2 - \textstyle\sum_{i=1}^{w+1} ib_i.
\end{aligned} \tag{5.32}$$

Thus, for $a \equiv \sum_{i=1}^{m} is_i$ mod p, the quantity $\hat{a} \equiv \sum_{i=1}^{w+1} ib_i$ mod p is $f_{m,w} - a$ mod p. ■

Observe that the indices $i = 1, \ldots, (w+1)$ in (5.22) play the role of the “weightings” of the appropriate bins of zeros in the construction above, and that they do not necessarily have to be in the increasing order for the construction and the validity of the proof to hold. We can therefore replace each of i in (5.22) with the weighting f_i with the property that each f_i is a residue $\mod P$ and that $f_i \neq f_j$ for $i \neq j$. Let $\hat{\hat{S}}(m, w, \mathbf{a}, \mathbf{f}, p)$ for $w \geq 1$ be defined as

$$\begin{aligned}\hat{\hat{S}}(m,w,\mathbf{a},\mathbf{f},p)=\{\quad & \mathbf{s}=(s_1,s_2,...s_m)\in\{0,1\}^m:\\ & v_0=0, v_{w+1}=m+1,\text{ and } v_i \text{ is the position of the } i^{\text{th}} \text{ 1 in } \mathbf{s} \text{ for } 1\le i\le w,\\ & b_i=v_i-v_{i-1}-1 \text{ for } 1\le i\le w+1,\\ & \textstyle\sum_{i=1}^{m} s_i=w,\\ & f_i \mod P\neq f_j \mod P \text{ for } i\neq j,\\ & \textstyle\sum_{i=1}^{w+1} f_ib_i\equiv a_1 \text{ mod } p,\\ & \textstyle\sum_{i=1}^{w+1}(f_i)^2b_i\equiv a_2 \text{ mod } p,\\ & \qquad\quad\vdots\\ & \textstyle\sum_{i=1}^{w+1}(f_i)^rb_i\equiv a_t \text{ mod } p\ \}.\end{aligned} \tag{5.33}$$

The set $\hat{\hat{S}}(m,0,\mathbf{0},\mathbf{0},p)$ contains just the all-zeros string. Let $\mathbf{a_0}=\mathbf{0}$ and let $\hat{\hat{S}}\,(m,(\mathbf{a_1},\mathbf{f_1},p_1),(\mathbf{a_2},\mathbf{f_2},p_2),...,(\mathbf{a_m},\mathbf{f_m},p_m))$ be defined as

$$\hat{\hat{S}}\,(m,(\mathbf{a_1},\mathbf{f_1},p_1),(\mathbf{a_2},\mathbf{f_2},p_2),...,(\mathbf{a_m},\mathbf{f_m},p_m))=\bigcup_{l=0}^{m}\hat{\hat{S}}(m,l,\mathbf{a_l},\mathbf{f_l},p_l). \tag{5.34}$$

We note that $\hat{\hat{S}}(m,w,\mathbf{a},\mathbf{f},p)$ = $\hat{S}(m,w,\mathbf{a},p)$ when $\mathbf{f}=(1,2,\ldots,(w+1))$.

Lemma 15 *If each p_l is prime and $p_l>$ max(r,l), the set $\hat{\hat{S}}\,(m,(\mathbf{a_1},\mathbf{f_1},p_1),(\mathbf{a_2},\mathbf{f_2},p_2),...,(\mathbf{a_m},\mathbf{f_m},p_m))$ is r-insertions of zeros correcting.*

Proof: The proof follows that of Lemma 13 with appropriate substitutions of f_i for i.

■

The object $\hat{\hat{S}}(m,w,\mathbf{a},\mathbf{f},p)$ will be of further interest to us in Section 6.2 when we discuss a prefixing method for improved immunity to repetition errors.

We now present some cardinality results for the construction of present interest. For simplicity we focus on the set $\hat{S}(m, w, \mathbf{a}, p)$ as the results hold verbatim for $\hat{\hat{S}}(m, w, \mathbf{a}, \mathbf{f}, p)$ with appropriate weighting assignments.

5.4.1 Cardinality Results

Let $\hat{S}^* \left(m, (\mathbf{a_1}, p_1), (\mathbf{a_2}, p_2), ..., (\mathbf{a_m}, p_m)\right)$ be defined as

$$\hat{S}^* \left(m, (\mathbf{a_1}, p_1), (\mathbf{a_2}, p_2), ..., (\mathbf{a_m}, p_m)\right) = \bigcup_{l=0}^{m} \hat{S}(m, l, \mathbf{a_l}^*, p_l). \tag{5.35}$$

where $\hat{S}(m, l, \mathbf{a_l}^*, p_l)$ is the largest among all sets $\hat{S}(m, l, \mathbf{a_l}, p_l)$ for $\mathbf{a_l} \in \{0, 1, \ldots, p_l\}^r$. The cardinality of $\hat{S}(m, l, \mathbf{a_l}^*, p_l)$ is at least

$$\binom{m}{l} \frac{1}{p_l^r} .$$

Since for all n there exists a prime between n and $2n$ it follows that one can choose the p_l, $1 \leq l \leq m$, so that cardinality of $\hat{S}(m, l, \mathbf{a_l}^*, p_l)$ for $l \geq r$ is at least

$$\binom{m}{l} \frac{1}{(2l)^r} .$$

Thus $p_1, \ldots, p_m$ can be chosen so that the cardinality of $\hat{S}^* \left(m, (\mathbf{a_1}, p_1), (\mathbf{a_2}, p_2), ..., (\mathbf{a_m}, p_m)\right)$ is at least

$$1 + \sum_{w=1}^{r-1} \binom{m}{w} \frac{1}{(2r)^r} + \sum_{w=r}^{m} \binom{m}{w} \frac{1}{(2w)^r} , \tag{5.36}$$

which is lower bounded by

$$1 + \frac{1}{(2r)^r} \sum_{w=1}^{r-1} \left(\begin{array}{c} m \\ \\ w \end{array} \right) +$$

$$\frac{1}{(2^r)(m+1)(m+2)\dots(m+r)} \left(2^{m+r} - \sum_{k=0}^{2r-1} \left(\begin{array}{c} m+r \\ \\ k \end{array} \right) \right) . \tag{5.37}$$

The prime counting function $\pi(n)$ which counts the number of primes up to n, satisfies for $n \geq 67$ the inequalities [50]

$$\frac{n}{\ln(n) - 1/2} < \pi(n) < \frac{n}{\ln(n) - 3/2} . \tag{5.38}$$

From (5.38) it follows that

$$\frac{(1+\epsilon)n}{\ln((1+\epsilon)n) - 1/2} < \pi((1+\epsilon)n) < \frac{(1+\epsilon)n}{\ln((1+\epsilon)n) - 3/2} . \tag{5.39}$$

For a prime number to exist between n and $(1+\epsilon)n$, it is sufficient to have

$$\pi((1+\epsilon)n) > \pi(n) . \tag{5.40}$$

Using (5.38) and (5.39) it is sufficient to have

$$\pi((1+\epsilon)n) > \frac{(1+\epsilon)n}{\ln((1+\epsilon)n) - 1/2} \geq \frac{n}{\ln(n) - 3/2} > \pi(n) . \tag{5.41}$$

Comparing the innermost terms in (5.41) it follows that it is sufficient for ϵ to satisfy

$$\epsilon \ln(n) \geq \ln(1+\epsilon) + \frac{3\epsilon}{2} + 1 \tag{5.42}$$

for (5.40) to hold.

For $n \geq 67$ and $\epsilon = \frac{3}{\ln(n)}$, the left hand side of (5.42) evaluates to 3 while the right hand side of (5.42) is upper bounded by $(0.539 + 1.071 + 1) < 3$.

Since $\pi(n)$ is a non-decreasing function of n, it follows that for $n \geq 67$, there exists a prime between n and $(1+\epsilon)n$ for $\epsilon \geq \frac{3}{\ln(n)}$. Thus the lower bound on the asymptotic cardinality of the best choice over $p_1, \ldots, p_m$ of $\hat{S}^* \left(m, (\mathbf{a_1}, p_1), (\mathbf{a_2}, p_2), ..., (\mathbf{a_m}, p_m)\right)$ can be improved to

$$\frac{1}{(1+\epsilon)^r (m+1)(m+2)\ldots(m+r)} \left(2^{m+r}\right) - P(m), \tag{5.43}$$

where $\epsilon = \frac{3}{\ln m}$ and $P(m)$ is a polynomial in m. In the limit $m \to \infty$, (5.43) is approximately

$$\frac{2^{m+r}}{(m+1)^r} \ . \tag{5.44}$$

A construction proposed by Levenshtein [34] has the lower asymptotic bound on the cardinality given by

$$\frac{1}{(\log_2 2r)^r} \frac{2^m}{m^r} \ . \tag{5.45}$$

Note that both (5.36) and the improved bound (5.43) improve on (5.45) by at least a constant factor.

The upper bound $U_r(m)$ on any set of strings each of length m capable of overcoming r insertions of zero is

$$U_r(m) = c(r) \frac{2^m}{m^r},$$

as obtained in [34], where

$$c(r) = \begin{cases} 2^r r! & \text{odd } r \\ 8^{r/2}((r/2)!)^2 & \text{even } r \end{cases}$$

which makes the proposed construction be within a factor of this bound. By applying the inverse T_n transformation for $n = m + 1$ to $\hat{S}^* \left(m, (\mathbf{a_1}, p_1), (\mathbf{a_2}, p_2), ..., (\mathbf{a_m}, p_m)\right)$ and noting that both strings under the inverse T_n transformation can simultaneously belong to the repetition error correcting set, we obtain a code of length n capable of overcoming r repetitions and of asymptotic size at least

$$\frac{2^{n+r}}{n^r} \; . \tag{5.46}$$

5.5 Summary and Concluding Remarks

In this chapter we discussed the problem of constructing repetition error correcting codes (subsets of binary strings). We presented some explicit number-theoretic constructions and provided some results on the cardinalities of these constructions. Specific contributions included a generalization of a generating function calculation of Sloane [51] and a construction of multiple repetition error correcting codes that is asymptotically a constant factor better than the previously best known construction due to Levenshtein [34].

Chapter 6

Prefixing-Based Method for Multiple Repetition Error Correction

In this chapter we propose a general prefixing method which injectively transforms a given collection C of binary strings of length n into another collection of binary strings D_C of equal length, such that the collection D_C is guaranteed to be immune to the prescribed number of repetition errors. The proposed method is inspired by the number-theoretic construction developed in the previous chapter. It takes an element $\mathbf{c}$ of C and produces a string $\mathbf{t_c} = [\mathbf{p_c c}]$, $\mathbf{t_c} \in D_C$, that is, the prefix $\mathbf{p_c}$ is prepended to $\mathbf{c}$ to produce $\mathbf{t_c}$. In the proposed method, the set D_C has the property that the length of the prefix $\mathbf{p_c}$ is $O(\log(n))$. Thus, if the set C is used for transmission, the proposed method provides increased immunity to repetition errors with asymptotically vanishing loss in the rate. We also provide a message passing decoding algorithm suitable for channels with both repetitions and additive errors whose complexity is the same as of the traditional message passing decoding algorithm designed to correct additive errors only.

We start with some auxiliary results.

6.1 Auxiliary results

Consider a prime number P with the property that $lcm(2,3,..r)|(P-1)$ for a given positive integer r. Since each i, $1 \leq i \leq r$, satisfies $i|(P-1)$, it follows that in the residue set $\mod P$, there are $\frac{P-1}{i}$ elements that are ith power residues, each having i distinct roots (an ith power residue x satisfies $y^i \equiv x \mod P$ for some y), [3]. For convenience, let $G = \lfloor \log_2(P) \rfloor$.

For each i, $1 \leq i \leq r$, we will construct a specific subset V_i of the ith power residues $\mod P$ such that all other residues can be expressed as a sum of a subset of elements of V_i, and such that each V_i has size that is logarithmic in P. The set of the ith roots of the elements of the set V_i will be denoted F_i. Thus, F_i will also have size logarithmic in P. The elements of $M = \bigcup_{i=1}^{r} F_i \cup \{0\}$ (the sets F_i will be made disjoint) will be reserved for the weightings f_i of the bins of zeros of the prefix string $\mathbf{p_s}$ in the transformed domain (see the construction (5.33)). Note that M also has size that is logarithmic in P, and since each bin in the prefix will have at most one zero, the length of the prefix is also logarithmic in P. The sets V_i will serve to satisfy the ith congruency constraint of the type given in (5.33) for the string $\mathbf{t_s}$ in the transformed domain, as further explained below.

In the remainder of this section we will first show how to construct sets V_i, and then we will provide the proof that it is possible to construct sets V_i with all distinct elements as well as sets F_i (from sets V_i) that have distinct elements and are non intersecting, for the prime P large enough. We will also provide a proof that for a given integer n, for n large enough, there exists a prime P for which we can construct non intersecting sets F_i containing distinct elements, where the prime P lies in an interval that linearly depends on n.

Combined with the encoding method described in the next section we will therefore have constructed a prefix whose length is logarithmic in n such that the overall string (which is a concatenation of the prefix and original string) in the transformed domain satisfies equations of congruential type given in (5.33), for which we have already proved in the previous chapter are sufficient for the immunity to r repetition errors.

We now provide some auxiliary results. Let $[x]_P$ indicate the residue mod P congruent to x .

Lemma 16 *For an integer P, each residue v mod P can be expressed as a sum of a subset of elements of the set $T_{z,P} = \{[z]_P, [2z]_P, [2^2z]_P, ..., [2^Gz]_P\}$ where $G = \lfloor \log_2 P \rfloor$, z is an arbitrary non zero residue mod P.*

Proof: Observe that $T_{1,P} = \{1, 2, 2^2, ..., 2^G\}$. We first show that each residue v mod P can be expressed as a sum of a subset of elements of the set $T_{1,P}$. Note that each residue i, $0 \leq i \leq 2^G - 1$ (mod P) can be expressed as a sum of a subset, call this subset Q_i, of the set $\{1, 2, 2^2, ..., 2^{G-1}\}$. Here Q_0 is the empty set. Adding 2^G to the sum of each Q_i, for $0 \leq$

$i \leq 2^G - 1$, modulo P generates the remaining residues $\{2^G, 2^G+1, ..., P-1\}$. As a result every residue mod P can be expressed as a sum of a subset of $T_{1,P} = \{1, 2, 2^2, ..., 2^G\}$.

Suppose there exists an element v which cannot be expressed as a sum of a subset of elements of $T_{z,P}$, for $z > 1$, that is $v \neq \sum_{i=0}^{G} \epsilon_i z 2^i \mod P$, for all choices of $\{\epsilon_0, ..., \epsilon_G\}$, $\epsilon_i \in \{0, 1\}$. Let z^{-1} be the inverse element of z under multiplication mod P. Then the residue $v' = vz^{-1} \neq \sum_{i=0}^{G} \epsilon_i 2^i \mod P$, for all choices of $\{\epsilon_0, ..., \epsilon_G\}$, $\epsilon_i \in \{0, 1\}$, which contradicts the result from the previous paragraph. ■

For a prime number P for which $i|P-1$, and $i < P-1$, let $Q_i(P)$ be the set of distinct ith power residues mod P, let $N_i(P)$ be the set of distinct ith power non residues mod P. We also state the following convenient result.

Lemma 17 *For a prime P such that $i|(P-1)$, each residue $n \mod P$ can be expressed as a sum of two distinct elements of $Q_i(P)$ in at least $P/(2i^2) - \sqrt{P}/2 - 3$ ways.*

Proof: The result follows from Theorem II in [27] which states that over $GF(P)$ the equation

$$x^i + y^i = a \tag{6.1}$$

where $x, y, a \in GF(P)$ and nonzero and $0 < i < P-1$ has at least

$$\frac{(P-1)^2}{P} - P^{-1/2}\left(1 + (i-1)P^{1/2}\right)^2 \tag{6.2}$$

solutions. Rearrange the terms in (6.2) to conclude that (6.1) has at least

$$P - (i-1)^2\sqrt{P} - 2(i-1) - 2 + \frac{1}{P} - \frac{1}{\sqrt{P}} \tag{6.3}$$

solutions. Noting that i distinct values of x result in the same x^i, accounting for the symmetry of x and y, and omitting the case $x^i = y^i$ we obtain a lower bound on the number of ways a residue can be expressed as a sum of two distinct ith power residues to be $P/(2i^2) - \sqrt{P}/2 - 3$. ■

Equations of the type in (6.1) were also studied by Weil [59].

We now continue with the introduction of some convenient notation. For $x_{i,1}$ an ith power residue define the set $A_{i,1}(x_{i,1})$ to be

$$A_{i,1}(x_{i,1}) = \{[2^{ik}x_{i,1}]_P | 0 \leq k \leq \lfloor \frac{G}{i} \rfloor\} \; . \tag{6.4}$$

Let $x_{i,2}$ and $x_{i,3}$ be distinct ith power residues such that $x_{i,2} + x_{i,3} \equiv 2x_{i,1} \mod P$. These two power residues generate sets $A_{i,2}(x_{i,2})$ and $A_{i,3}(x_{i,3})$ where

$$A_{i,2}(x_{i,2}) = \{[2^{ik}x_{i,2}]_P | 0 \leq k \leq \lfloor \frac{G-1}{i} \rfloor\} \text{ and} \tag{6.5}$$

$$A_{i,3}(x_{i,3}) = \{[2^{ik}x_{i,3}]_P | 0 \leq k \leq \lfloor \frac{G-1}{i} \rfloor\} \; . \tag{6.6}$$

Likewise, for each $2^l x_{i,1}$ for $1 \leq l \leq i-1$ let $x_{i,2l}$ and $x_{i,2l+1}$ be distinct ith power residues such that $x_{i,2l} + x_{i,2l+1} \equiv 2^l x_{i,1} \mod P$. These residues generate sets $A_{i,2l}(x_{i,2l})$ and $A_{i,2l+1}(x_{i,2l+1})$ where

$$A_{i,2l}(x_{i,2l}) = \{[2^{ik}x_{i,2l}]_P | 0 \leq k \leq \lfloor \frac{G-l}{i} \rfloor\} \text{ and} \tag{6.7}$$

$$A_{i,2l+1}(x_{i,2l+1}) = \{[2^{ik}x_{i,2l+1}]_P | 0 \leq k \leq \lfloor \frac{G-l}{i} \rfloor\}. \tag{6.8}$$

By introducing sets $A_{i,j}(x_{i,j})$ we have effectively decomposed all residues of the type $[2^{ik+l}x_{i,1}]_P$, $0 \leq ik + l \leq G$, $1 \leq l \leq i-1$, for which i is not a divisor of l into a sum of two ith power residues, namely $[2^{ik}x_{i,2l}]_P$ and $[2^{ik}x_{i,2l+1}]_P$. For each set $A_{i,j}(x_{i,j})$, $1 \leq j \leq 2i-1$, we let $B_{i,j}(x_{i,j})$ be the set of all ith power roots of elements of $A_{i,j}(x_{i,j})$,

$$B_{i,j}(x_{i,j}) = \{[2^k y_{i,j}^{(t)}]_P | (y_{i,j}^{(t)})^i \equiv x_{i,j} \mod P, 1 \leq t \leq i, 0 \leq k \leq \lfloor \frac{G - \lfloor \frac{j}{2} \rfloor}{i} \rfloor \} \; . \quad (6.9)$$

First note that all elements in $A_{i,j}(x_{i,j})$ are ith power residues by construction. Moreover, they are all distinct since $2^{ij_1} \neq 2^{ij_2} \mod P$ for $1 \leq j_1, j_2 \leq \lfloor \frac{G-\lfloor \frac{j}{2} \rfloor}{i} \rfloor$ for $j_1 \neq j_2$ implies $x_{i,j}2^{ij_1} \neq x_{i,j}2^{ij_2} \mod P$. Thus, $|A_{ij}(x_{i,j})| = \lfloor \frac{G-\lfloor \frac{j}{2} \rfloor}{i} \rfloor + 1$ and since the ith power roots of distinct ith power residues are themselves distinct, $|B_{ij}(x_{i,j})| = i \left(\lfloor \frac{G-\lfloor \frac{j}{2} \rfloor}{i} \rfloor + 1 \right)$.

Lemma 18 *Suppose P is a prime number such that $i|(P-1)$. Let $x_{i,1}$ be an ith power residue. Suppose $x_{i,j}$ for $2 \leq j \leq 2i-1$ are ith power residues such that $2^k x_{i,1} \equiv x_{i,2k} + x_{i,2k+1} \mod P$ for $1 \leq k \leq (i-1)$. Let $A_{i,j}(x_{i,j}) = \{[2^{il}x_{i,j}]_P | 0 \leq l \leq \lfloor \frac{G-\lfloor \frac{j}{2} \rfloor}{i} \rfloor\}$ for $1 \leq j \leq 2i-1$ and $G = \lfloor \log_2 P \rfloor$. If the sets $A_{i,j}(x_{i,j})$ are disjoint for $1 \leq j \leq 2i-1$, each residue n mod P can be expressed as a sum of a subset of elements of the set $L_{z,P} = \bigcup_{j=1}^{2i-1} A_{i,j}(x_{i,j})$ where z denotes $x_{i,1}$.*

Proof: Follows immediately from Lemma 16 by observing that, with z denoting $x_{i,1}$, we have in fact decomposed elements $[2^k z]_P$ in the set $T_{z,P}$ for k not a multiple of i into a sum of two component elements such that all component elements are distinct from one another and distinct from $[2^k z]_P$ for $i|k$. ■

The following lemma proves that it is possible to construct subsets $A_{ij}(x_{i,j})$, and subsets $B_{ij}(x_{i,j})$ from them, of the set of residues $\mod P$ for P prime that satisfies $lcm(2,3,...r)|(P-1)$ for a given positive integer r, provided that P is large enough, such that for fixed

i the subsets $A_{ij}(x_{i,j})$ are disjoint, and such that *all* subsets $B_{ij}(x_{i,j})$ for $1 \leq i \leq r$, $1 \leq j \leq 2i-1$ are also disjoint. Let $W_i(n)$ denote the number of ways any residue $n \mod P$ can be expressed as a sum of two distinct non zero ith power residues $\mod P$. A universal lower bound on $W_i(n)$ that holds for all residues n was given in Lemma 17, and we will refer to it as W_i.

Lemma 19 *For a given integer r, suppose a prime number P satisfies $lcm(2,3,...r)|(P-1)$. Let $G = \lfloor \log_2 P \rfloor$. If $P-1 > (G+r)(G+r-1)(r-1)^2$ and $W_i > 2i(G+i)(G+i-1)$, for each i in the range $2 \leq i \leq r$, there exist subsets $A_{ij}(x_{i,j})$ of the type given in* (6.7) *and* (6.8) *and $B_{ij}(x_{i,j})$ of the type given in* (6.9) *such that for fixed i subsets $A_{ij}(x_{i,j})$ for $1 \leq j \leq 2i-1$ are disjoint, and for $1 \leq i \leq r$, $1 \leq j \leq 2i-1$ all subsets $B_{ij}(x_{i,j})$ are disjoint.*

Proof: We inductively build the sets $A_{ij}(x_{i,j})$ and $B_{ij}(x_{i,j})$ for $1 \leq i \leq r$ and $1 \leq j \leq 2i-1$, starting with the level $i=1$. We then increment i by one to reach the next collection of sets $A_{ij}(x_{i,j})$ and $B_{ij}(x_{i,j})$ while making sure the sets $B_{ij}(x_{i,j})$ at the current level are disjoint from one another and with all previously constructed sets at lower levels.

Consider $i=1$. Let $x_{1,1}$ be an arbitrary residue $\mod P$, and let

$$A_{1,1}(x_{1,1}) = \{[2^k x_{1,1}]_P | 0 \leq k \leq G\}.$$

Let $z_1 = x_{1,1}$ and $y_{1,1}^{(1)} = x_{1,1}$. Here $B_{1,1}(z_1)$ is simply $A_{1,1}(x_{1,1})$ for $i=1$. All elements in $B_{1,1}(z_1)$ are distinct and $|B_{1,1}(z_1)| = (G+1)$. If $r=1$, we are done, as we did not even appeal to the condition on the lower bound on $P-1$ (it is simply $P-1>0$).

If $r \geq 2$, let us consider $i = 2$. Consider quadratic residues $x_{2,1}$, $x_{2,2}$ and $x_{2,3}$. Let their respective distinct quadratic roots be $y_{2,1}^{(1)}$, $y_{2,1}^{(2)}$ (so that $(y_{2,1}^{(1)})^2 \equiv (y_{2,1}^{(2)})^2 \equiv x_{2,1} \mod P$), $y_{2,2}^{(1)}$, $y_{2,2}^{(2)}$ (so that $(y_{2,2}^{(1)})^2 \equiv (y_{2,2}^{(2)})^2 \equiv x_{2,2} \mod P$) and $y_{2,3}^{(1)}$, $y_{2,3}^{(2)}$ (so that $(y_{2,3}^{(1)})^2 \equiv (y_{2,3}^{(2)})^2 \equiv x_{2,3} \mod P$). These quadratic residues give rise to sets

$$A_{2,1}(x_{2,1}) = \{[2^{2k}x_{2,1}]_P | 0 \leq k \leq \lfloor \frac{G}{2} \rfloor\}, \tag{6.10}$$

$$A_{2,2}(x_{2,2}) = \{[2^{2k}x_{2,2}]_P | 0 \leq k \leq \lfloor \frac{G-1}{2} \rfloor\} \text{ and,} \tag{6.11}$$

$$A_{2,3}(x_{2,3}) = \{[2^{2k}x_{2,3}]_P | 0 \leq k \leq \lfloor \frac{G-1}{2} \rfloor\} . \tag{6.12}$$

Quadratic roots of elements of sets $A_{2,1}(x_{2,1})$, $A_{2,2}(x_{2,2})$ and $A_{2,3}(x_{2,3})$ give rise to sets $B_{2,1}(x_{2,1})$, $B_{2,2}(x_{2,2})$ and $B_{2,3}(x_{2,3})$,

$$B_{2,1}(x_{2,1}) = \{[2^k y_{2,1}^{(t)}]_P | 1 \leq t \leq 2, 0 \leq k \leq \lfloor \frac{G}{2} \rfloor\}, \tag{6.13}$$

$$B_{2,2}(x_{2,2}) = \{[2^k y_{2,2}^{(t)}]_P | 1 \leq t \leq 2, 0 \leq k \leq \lfloor \frac{G-1}{2} \rfloor\}, \text{ and} \tag{6.14}$$

$$B_{2,3}(x_{2,3}) = \{[2^k y_{2,3}^{(t)}]_P | 1 \leq t \leq 2, 0 \leq k \leq \lfloor \frac{G-1}{2} \rfloor\} . \tag{6.15}$$

Having fixed the set $B_{1,1}(x_{1,1})$ based on the earlier selection of the residue $x_{1,1}$, we want to show that it is possible to find quadratic residues $x_{2,1}$, $x_{2,2}$ and $x_{2,3}$ such that $x_{2,2}+x_{2,3} \equiv 2x_{2,1} \mod P$ and such that the resulting sets $B_{1,1}(x_1)$, $B_{2,1}(x_{2,1})$, $B_{2,2}(x_{2,2})$ and $B_{2,3}(x_{2,3})$ are all disjoint.

In particular we require that $x_{2,1}$ is a quadratic residue $\mod P$ (there are $(P-1)/2$ quadratic residues) with the property that the set $B_{2,1}(x_{2,1})$ is disjoint from $B_{1,1}(x_{1,1})$. That is we require

$$y_{2,1}^{(1)} 2^k \neq y_{1,1}^{(1)} 2^l \mod P$$

and

$$y_{2,1}^{(2)} 2^k \neq y_{1,1}^{(1)} 2^l \mod P$$

for $0 \leq k \leq \lfloor \frac{G}{2} \rfloor$ and $0 \leq l \leq G$. By squaring the expressions, these two conditions can be combined into

$$x_{2,1} 2^{2k} \neq (y_{1,1}^{(1)})^2 2^{2l} \mod P \tag{6.16}$$

for $0 \leq k \leq \lfloor \frac{G}{2} \rfloor$ and $0 \leq l \leq G$. For the already chosen $y_{1,1}^{(1)} (= x_{1,1})$ at most $(G + 1)(\lfloor \frac{G}{2} \rfloor + 1)$ candidate quadratic residues out of total $(P-1)/2$ quadratic residues violate (6.16). Observe that the function $(G+i)(G+i-1)(i-1)^2$ is strictly increasing for positive i, $2 \leq i \leq r$, and thus the condition $P-1 > (G+r)(G+r-1)(r-1)^2$ in the statement of the Lemma implies $P-1 > (G+2)(G+1)$. Since $\frac{P-1}{2} > \frac{(G+1)(G+2)}{2} \geq (G+1)(\lfloor \frac{G}{2} \rfloor + 1)$, such $x_{2,1}$ exists.

Fix $x_{2,1}$ such that (6.16) holds. Having chosen such $x_{2,1}$, we now look for $x_{2,2}$ and $x_{2,3}$ as distinct quadratic residues that satisfy $x_{2,2} + x_{2,3} \equiv 2x_{2,1} \mod P$. We require that $B_{2,2}(x_{2,2})$ be disjoint from both $B_{1,1}(x_{1,1})$ and $B_{2,1}(x_{2,1})$ (by construction, if $B_{2,2}(x_{2,2})$ and $B_{2,1}(x_{2,1})$ are disjoint so are $A_{2,2}(x_{2,2})$ and $A_{2,1}(x_{2,1})$) so that

$$\begin{aligned}
y_{2,2}^{(1)} 2^{k_3} &\neq y_{1,1}^{(1)} 2^{k_1} \mod P, \\
y_{2,2}^{(2)} 2^{k_3} &\neq y_{1,1}^{(1)} 2^{k_1} \mod P, \\
y_{2,2}^{(1)} 2^{k_3} &\neq y_{2,1}^{(1)} 2^{k_2} \mod P, \\
y_{2,2}^{(2)} 2^{k_3} &\neq y_{2,1}^{(1)} 2^{k_2} \mod P, \\
y_{2,2}^{(1)} 2^{k_3} &\neq y_{2,1}^{(2)} 2^{k_2} \mod P, \\
y_{2,2}^{(2)} 2^{k_3} &\neq y_{2,1}^{(2)} 2^{k_2} \mod P,
\end{aligned} \tag{6.17}$$

where $0 \leq k_1 \leq G$, $0 \leq k_2 \leq \lfloor \frac{G}{2} \rfloor$ and $0 \leq k_3 \leq \lfloor \frac{G-1}{2} \rfloor$.

Alternatively, by squaring both sides in each expression in (6.17),

$$\begin{aligned} x_{2,2} 2^{2k_3} &\neq (y_{1,1}^{(1)})^2 2^{2k_1} \qquad \bmod P, \\ x_{2,2} 2^{2k_3} &\neq x_{2,1} 2^{2k_2} \qquad \bmod P, \end{aligned} \tag{6.18}$$

where $0 \leq k_1 \leq G$, $0 \leq k_2 \leq \lfloor \frac{G}{2} \rfloor$ and $0 \leq k_3 \leq \lfloor \frac{G-1}{2} \rfloor$.

Likewise, we require that $B_{2,3}(x_{2,3})$ be disjoint from $B_{1,1}(x_{1,1})$, $B_{2,1}(x_{2,1})$ and $B_{2,2}(x_{2,2})$ (again, if $B_{2,3}(x_{2,3})$ is disjoint from $B_{2,2}(x_{2,2})$ and $B_{2,1}(x_{2,1})$, then $A_{2,3}(x_{2,3})$ is disjoint from $A_{2,2}(x_{2,2})$ and $A_{2,1}(x_{2,1})$) so that

$$\begin{aligned} x_{2,3} 2^{2k_4} &\neq (y_{1,1}^{(1)})^2 2^{2k_1} \qquad \bmod P, \\ x_{2,3} 2^{2k_4} &\neq x_{2,1} 2^{2k_2} \qquad \bmod P, \\ x_{2,3} 2^{2k_4} &\neq x_{2,2} 2^{2k_3} \qquad \bmod P, \end{aligned} \tag{6.19}$$

where $0 \leq k_1 \leq G$, $0 \leq k_2 \leq \lfloor \frac{G}{2} \rfloor$, $0 \leq k_3 \leq \lfloor \frac{G-1}{2} \rfloor$ and $0 \leq k_4 \leq \lfloor \frac{G-1}{2} \rfloor$. For the already chosen values of $x_{2,1}$ and $y_{1,1}$ at most $N_2 = 2\left[\left(\lfloor \frac{G}{2} \rfloor + 1\right)\left(\lfloor \frac{G-1}{2} \rfloor + 1\right) + (G+1)\left(\lfloor \frac{G-1}{2} \rfloor + 1\right)\right] + \left(\lfloor \frac{G-1}{2} \rfloor + 1\right)^2$ choices for $x_{2,2}$ and $x_{2,3}$ violate (6.18) and (6.19).

We thus require that W_2 be strictly larger than N_2. Dropping floor operations it is sufficient that $W_2 > \frac{(G+1)(G+2)}{2} + \frac{5(G+1)^2}{4}$. Further simplification yields that

$$W_2 > \frac{7(G+1)(G+2)}{4} \tag{6.20}$$

is sufficient to ensure that there exist $x_{2,2}$, $x_{2,3}$ that make the respective sets disjoint. Note that this last condition follows from the requirement in the statement of the Lemma for $i = 2$, namely that $W_2 > 4(G+1)(G+2)$. If $r = 2$ we are done, else we consider $i = 3$. Before considering general level i let us present the $i = 3$ case.

For $i = 3$ we seek distinct cubic residues $x_{3,1}$, $x_{3,2}$, $x_{3,3}$, $x_{3,4}$ and $x_{3,5}$ with the property that $x_{3,2} + x_{3,3} \equiv 2x_{3,1} \mod P$ and $x_{3,4} + x_{3,5} \equiv 2^2 x_{3,1} \mod P$, and such that the respective sets $B_{3,j}(x_{3,j})$ for $1 \leq j \leq 5$ generated from the cubic roots of these residues are disjoint and are disjoint from previously constructed sets $B_{1,1}(x_{1,1})$, $B_{2,1}(x_{2,1})$, $B_{2,2}(x_{2,2})$ and $B_{2,3}(x_{2,3})$.

We start with $x_{3,1}$ a cubic residue $\mod P$ (there are $(P-1)/3$ cubic residues) with the property that the set $B_{3,1}(x_{3,1})$ is disjoint from each of $B_{1,1}(x_{1,1})$, $B_{2,1}(x_{2,1})$, $B_{2,2}(x_{2,2})$ and $B_{2,3}(x_{2,3})$. That is, after raising the elements of these sets to the third power, we require

$$\begin{aligned}
x_{3,1}2^{3k_4} &\neq (y_{1,1}^{(1)})^3 2^{3k_1} \mod P,\\
x_{3,1}2^{3k_4} &\neq (y_{2,1}^{(1)})^3 2^{3k_2} \mod P,\\
x_{3,1}2^{3k_4} &\neq (y_{2,1}^{(2)})^3 2^{3k_2} \mod P,\\
x_{3,1}2^{3k_4} &\neq (y_{2,2}^{(1)})^3 2^{3k_3} \mod P,\\
x_{3,1}2^{3k_4} &\neq (y_{2,2}^{(2)})^3 2^{3k_3} \mod P,\\
x_{3,1}2^{3k_4} &\neq (y_{2,3}^{(1)})^3 2^{3k_3} \mod P,\\
x_{3,1}2^{3k_4} &\neq (y_{2,3}^{(2)})^3 2^{3k_3} \mod P,
\end{aligned} \tag{6.21}$$

where $0 \leq k_1 \leq G$, $0 \leq k_2 \leq \lfloor \frac{G}{2} \rfloor$, $0 \leq k_3 \leq \lfloor \frac{G-1}{2} \rfloor$ and $0 \leq k_4 \leq \lfloor \frac{G}{3} \rfloor$.

For the already chosen values of $x_{1,1}$ through $x_{2,3}$, which in turn determine $y_{1,1}^{(1)}$ through $y_{2,3}^{(2)}$, the condition in (6.21) prevents $N_3 = \left(\lfloor \frac{G}{3} \rfloor + 1\right) \left[(G+1) + 2\left(\lfloor \frac{G}{2} \rfloor + 1\right) + 4\left(\lfloor \frac{G-1}{2} \rfloor + 1\right)\right]$ choices for $x_{3,1}$. Since there are $\frac{P-1}{3}$ cubic residues, after simplifying and upper bounding the expression for N_3, it follows that it is sufficient that $\frac{P-1}{3}$ be strictly larger than $\frac{4(G+2)(G+3)}{3}$. Note that this condition is implied by the requirement that $P - 1 > (r -$

$1)^2(G+r)(G+r-1)$ (again, since the function $(i-1)^2(G+i)(G+i-1)$ is strictly increasing for positive i).

Fix $x_{3,1}$ such that (6.21) holds. Having chosen such $x_{3,1}$, we now look for distinct $x_{3,2}$, $x_{3,3}$, $x_{3,4}$, $x_{3,5}$ cubic residues that satisfy $x_{3,2}+x_{3,3} \equiv 2x_{3,1} \mod P$ and $x_{3,4}+x_{3,5} \equiv 2^2 x_{3,1} \mod P$ that make all sets $B_{i,j}(x_{i,j})$, $1 \leq i \leq 3$, $1 \leq j \leq 2i-1$ disjoint.

In order that residue $x_{3,2}$ generates set $B_{3,2}(x_{3,2})$ with the property that $B_{3,2}(x_{3,2})$ is disjoint from each of $B_{1,1}(x_{1,1})$, $B_{2,1}(x_{2,1})$, $B_{2,2}(x_{2,2})$, $B_{2,3}(x_{2,3})$ and $B_{3,1}(x_{3,1})$, we require that their respective elements raised to the third power be distinct,

$$\begin{aligned}
x_{3,2}2^{3k_5} &\neq (y^{(1)}_{1,1})^3 2^{3k_1} && \mod P,\\
x_{3,2}2^{3k_5} &\neq (y^{(1)}_{2,1})^3 2^{3k_2} && \mod P,\\
x_{3,2}2^{3k_5} &\neq (y^{(2)}_{2,1})^3 2^{3k_2} && \mod P,\\
x_{3,2}2^{3k_5} &\neq (y^{(1)}_{2,2})^3 2^{3k_3} && \mod P,\\
x_{3,2}2^{3k_5} &\neq (y^{(2)}_{2,2})^3 2^{3k_3} && \mod P,\\
x_{3,2}2^{3k_5} &\neq (y^{(1)}_{2,3})^3 2^{3k_3} && \mod P,\\
x_{3,2}2^{3k_5} &\neq (y^{(2)}_{2,3})^3 2^{3k_3} && \mod P,\\
x_{3,2}2^{3k_5} &\neq x_{3,1}2^{3k_4} && \mod P,
\end{aligned} \tag{6.22}$$

where $0 \leq k_1 \leq G$, $0 \leq k_2 \leq \lfloor \frac{G}{2} \rfloor$, $0 \leq k_3 \leq \lfloor \frac{G-1}{2} \rfloor$, $0 \leq k_4 \leq \lfloor \frac{G}{3} \rfloor$ and $0 \leq k_5 \leq \lfloor \frac{G-1}{3} \rfloor$.

Likewise, we require that $B_{3,3}(x_{3,3})$ be disjoint from all of $B_{1,1}(x_{1,1})$, $B_{2,1}(x_{2,1})$, $B_{2,2}(x_{2,2})$,

$B_{2,3}(x_{2,3})$, $B_{3,1}(x_{3,1})$ and $B_{3,2}(x_{3,2})$, so that

$$\begin{aligned}
x_{3,3}2^{3k_6} &\neq (y_{1,1}^{(1)})^3 2^{3k_1} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq (y_{2,1}^{(1)})^3 2^{3k_2} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq (y_{2,1}^{(2)})^3 2^{3k_2} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq (y_{2,2}^{(1)})^3 2^{3k_3} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq (y_{2,2}^{(2)})^3 2^{3k_3} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq (y_{2,3}^{(1)})^3 2^{3k_3} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq (y_{2,3}^{(2)})^3 2^{3k_3} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq x_{3,1}2^{3k_4} && \bmod P, \\
x_{3,3}2^{3k_6} &\neq x_{3,2}2^{3k_5} && \bmod P,
\end{aligned} \tag{6.23}$$

where $0 \le k_1 \le G$, $0 \le k_2 \le \lfloor \frac{G}{2} \rfloor$, $0 \le k_3 \le \lfloor \frac{G-1}{2} \rfloor$, $0 \le k_4 \le \lfloor \frac{G}{3} \rfloor$, $0 \le k_5 \le \lfloor \frac{G-1}{3} \rfloor$ and $0 \le k_6 \le \lfloor \frac{G-1}{3} \rfloor$.

From (6.22) and (6.23) it follows that at most

$$\begin{aligned}
N_3' = \; & 2\left(\lfloor \tfrac{G-1}{3} \rfloor + 1\right)\left[(G+1) + 2\left(\lfloor \tfrac{G}{2} \rfloor + 1\right) + 4\left(\lfloor \tfrac{G-1}{2} \rfloor + 1\right) + \left(\lfloor \tfrac{G}{3} \rfloor + 1\right)\right] + \\
& \left(\lfloor \tfrac{G-1}{3} \rfloor + 1\right)^2 .
\end{aligned} \tag{6.24}$$

candidate pairs $(x_{3,2}, x_{3,3})$ do not make the respective $B_{i,j}(x_{i,j})$ sets disjoint. Since

$$\begin{aligned}
N_3' &\le 2\left(\tfrac{G+2}{3}\right)\left[(G+1) + 2\left(\tfrac{G+2}{2}\right) + 4\left(\tfrac{G+1}{2}\right) + \left(\tfrac{G+3}{3}\right)\right] + \left(\tfrac{G+2}{3}\right)^2 \\
&< 2\left(\tfrac{G+2}{3}\right) \cdot 13\left(\tfrac{G+3}{3}\right) + \left(\tfrac{G+2}{3}\right)^2 \\
&< 3(G+2)(G+3),
\end{aligned} \tag{6.25}$$

it follows that it is sufficient that

$$W_3 > 3(G+2)(G+3) \;, \tag{6.26}$$

where W_3 is the number of ways a residue $\mod P$ can be expressed as a sum of two different cubic residues. Similarly, the cubic residues $x_{3,4}$ and $x_{3,5}$ for which the respective disjoint $B_{i,j}(x_{i,j})$ sets exist, provided that

$$W_3 > 2\left(\lfloor\tfrac{G-2}{3}\rfloor+1\right)\left[(G+1)+2\left(\lfloor\tfrac{G}{2}\rfloor+1\right)+4\left(\lfloor\tfrac{G-1}{2}\rfloor+1\right)+\left(\lfloor\tfrac{G}{3}\rfloor+1\right)+ \right.$$
$$\left. 2\left(\lfloor\tfrac{G-1}{3}\rfloor+1\right)\right]+\left(\lfloor\tfrac{G-2}{3}\rfloor+1\right)^2 . \qquad (6.27)$$

Some simplification of (6.27) yields

$$W_3 > \frac{31}{9}(G+2)(G+3)\ , \qquad (6.28)$$

which subsumes the lower bound on W_3 given in (6.26). Note that (6.28) is implied by the condition in the statement of the Lemma, namely $W_3 > 6(G+2)(G+3)$.

We now inductively show the existence of the appropriate ith power residues and their sets, assuming that we have successfully identified power residues at lower levels for which all the sets $B_{k,j}(x_{k,j})$ for $1 \leq k < i$, $1 \leq j \leq 2k-1$ are disjoint.

Consider $x_{i,1}$ an ith power residue $\mod P$ (there are $(P-1)/i$ such residues) with the property that the set $B_{i,1}(x_{i,1})$ is disjoint from all of $B_{k,j}(x_{k,j})$ for $1 \leq k < i$, $1 \leq j \leq 2k-1$.

These constraints on disjointness (an example of which is given in (6.16) for $i=2$ and in (6.21) for $i=3$) prevent no more than $(\frac{G+i}{i})(\frac{G+k}{k})$ choices for $x_{i,1}$ for each $y^{(t)}_{k,j}$ where $1 \leq k \leq i-1$, $1 \leq j \leq 2k-1$, and $1 \leq t \leq k$ (since $|B_{i,1}(x_{i,1})| = \lfloor \frac{G}{i} \rfloor + 1 \leq \frac{G+i}{i}$, and

$|B_{k,j}(x_{k,j})| = \lfloor\frac{G-\lfloor\frac{j}{2}\rfloor}{k}\rfloor + 1 \leq \frac{G+k}{k}$). Summing over all choices it follows that at most

$$\begin{aligned} & \left(\tfrac{G+i}{i}\right) \textstyle\sum_{k=1}^{i-1}(2k-1)k\left(\tfrac{G+k}{k}\right) \\ \leq\ & (G+i)\left(\tfrac{G+i-1}{i}\right)\textstyle\sum_{k=1}^{i-1}(2k-1) \\ =\ & (G+i)\left(\tfrac{G+i-1}{i}\right)(i-1)^2 \end{aligned} \tag{6.29}$$

ith power residues cannot be chosen for $x_{i,1}$. Since there are $\frac{P-1}{i}$ ith power residues, we thus require

$$P-1 > (G+i)(G+i-1)(i-1)^2 \tag{6.30}$$

for each level i. Note that since the expression on the right hand side of the inequality (6.30) is an increasing function of positive i, each subsequent level poses a lower bound on P that subsumes all previous ones. It is thus sufficient to have $P-1 > (G+r)(G+r-1)(r-1)^2$, as given in the statement of the Lemma.

Consider $x_{i,2}$ and $x_{i,3}$ as distinct ith power residues $\mod P$ that satisfy $x_{i,2} + x_{i,3} \equiv 2x_{i,1} \mod P$ for a previously chosen $x_{i,1}$. We require that $x_{i,2}$ and $x_{i,3}$ give rise to sets $B_{i,2}(x_{i,2})$ and $B_{i,3}(x_{i,3})$ that are disjoint and that are disjoint from each of $B_{k,j}(x_{k,j})$ for $1 \leq k < i$, $1 \leq j \leq 2k-1$ and with $B_{i,1}(x_{i,1})$. By construction, if the sets $B_{i,1}(x_{i,1})$, $B_{i,2}(x_{i,2})$, and $B_{i,3}(x_{i,3})$ are disjoint, then so are sets $A_{i,1}(x_{i,1})$, $A_{i,2}(x_{i,2})$, and $A_{i,3}(x_{i,3})$. Constraints based on the previously encountered $y_{j,k}^{(t)}$ for $1 \leq k < i$, $1 \leq j \leq 2k-1$, $1 \leq t \leq k$ prevent at most $(\frac{G+i-1}{i})(\frac{G+j}{j})$ choices for each of $x_{i,2}$ and $x_{i,3}$, for each $y_{j,k}^{(t)}$ (since $|B_{i,2}(x_{i,2})| = |B_{i,3}(x_{i,3})| = \lfloor\frac{G-1}{i}\rfloor + 1 \leq \frac{G+i-1}{i}$, and $|B_{k,j}(x_{k,j})| = \lfloor\frac{G-\lfloor\frac{j}{2}\rfloor}{k}\rfloor + 1 \leq \frac{G+k}{k}$). Combined with the restriction based on the disjointness with $B_{i,1}(x_{i,1})$ and the requirement that $B_{i,2}(x_{i,2})$ and $B_{i,3}(x_{i,3})$ be nonintersecting, it follows that

$$W_i > 2\left(\tfrac{G+i-1}{i}\right)\left[\sum_{k=1}^{i-1}(2k-1)k(\tfrac{G+k}{k}) + \left(\tfrac{G+i}{i}\right)\right] + \left(\tfrac{G+i-1}{i}\right)^2 \tag{6.31}$$

is sufficient for the pair $(x_{i,2}, x_{i,3})$ to exist.

Likewise, for $x_{i,2l}$ and $x_{i,2l+1}$ to be distinct ith power residues $\mod P$ that satisfy $x_{i,2l} + x_{i,2l+1} \equiv 2^l x_{i,1} \mod P$, that give rise to disjoint sets $B_{i,2l}(x_{i,2l})$ and $B_{i,2l+1}(x_{i,2l+1})$ and that are also disjoint from all previously constructed set $B_{k,j}(x_{k,j})$, we require

$$W_i > 2(\tfrac{G+i-1}{i})\left[\sum_{k=1}^{i-1}(2k-1)k(\tfrac{G+k}{k}) + (2l-1)\left(\tfrac{G+i}{i}\right)\right] + \left(\tfrac{G+i-1}{i}\right)^2 \tag{6.32}$$

for the pair $(x_{i,2l}, x_{i,2l+1})$ to exist. Since at each level i we construct $i-1$ pairs $x_{i,2l}$ and $x_{i,2l+1}$, and since the right hand side of (6.33) is an increasing function of l, it is sufficient to upper bound the expression in (6.33) for $l = i-1$,

$$\begin{aligned} & W_i > 2(\tfrac{G+i-1}{i})\left[\sum_{k=1}^{i-1}(2k-1)k(\tfrac{G+k}{k}) + (2i-3)\left(\tfrac{G+i}{i}\right)\right] + \left(\tfrac{G+i-1}{i}\right)^2 \\ \Leftarrow \quad & W_i > 2(\tfrac{G+i-1}{i})\left[(i-1)^2(G+i) + \tfrac{2i-3}{i}(G+i)\right] + \left(\tfrac{G+i-1}{i}\right)^2 \\ \Leftarrow \quad & W_i > (G+i)(G+i-1)\left(\tfrac{2}{i}(i-1)^2 + \tfrac{2}{i}\tfrac{2i-3}{i} + \tfrac{1}{i^2}\right) \ . \end{aligned} \tag{6.33}$$

Some simplification yields

$$W_i > (G+i)(G+i-1)\tfrac{2i^3-4i^2+6i-5}{i^2} \tag{6.34}$$

as a sufficient condition for the disjoint sets $B_{i,j}(x_{i,j})$ to exist that are also disjoint from all sets $B_{k,l}(x_{k,l})$ for $k < i$.

Further simplifying the last inequality, it is sufficient that

$$W_i > 2i(G+i)(G+i-1) \tag{6.35}$$

to make these sets disjoint. We have thus demonstrated that with the appropriate lower bounds on P and W_i's, it is possible to construct disjoint sets $B_{i,j}(x_{i,j})$. ■

Note that all residues $\mod P$ can be expressed as a sum of a subset of elements of $V_i := \bigcup_{j=1}^{2i-1} A_{i,j}(x_{i,j})$ by Lemma 18 for each i, $1 \leq i \leq r$. Also note that $|V_i|$ scales as $\log_2(P)$, since $|A_{i,j}(x_{i,j})| = \lfloor \frac{G-\lfloor \frac{j}{2} \rfloor}{i} \rfloor + 1$. For $F_i := \bigcup_{j=1}^{2i-1} B_{ij}(x_{i,j})$, $|F_i|$ also scales as $\log_2(P)$, since $|B_{i,j}(x_{i,j})| = i\left(\lfloor \frac{G-\lfloor \frac{j}{2} \rfloor}{i} \rfloor + 1\right)$.

We now discuss how large prime P needs to be so that the conditions of Lemma 19 hold. Namely we require

$$P - 1 > (r-1)^2(G+r)(G+r-1) \tag{6.36}$$

and

$$W_i > 2i(G+i)(G+i-1) \text{ for } 2 \leq i \leq r \ . \tag{6.37}$$

Using Lemma 17 it follows that it is sufficient that

$$P > 4r^3(G+r)(G+r-1) + r^2\sqrt{P} + 6r^2 \text{ , for } r \geq 2 \tag{6.38}$$

for (6.37) to hold. Moreover, if (6.38) holds , it implies (6.36).(For $r = 1$, the requirement is $P > 1$). The expression (6.38) certainly holds as $P \to \infty$, and for the finite values of P

we (loosely) have that

$$P > 200 \text{ for } r = 1,$$

$$P > 4 \times 10^3 \text{ for } r = 2,$$

$$P > 2 \times 10^4 \text{ for } r = 3,$$

$$P > 6 \times 10^4 \text{ for } r = 4,$$

$$P > 2 \times 10^5 \text{ for } r = 5.$$

For a given large enough integer n, we now show that there exists a prime number P that satisfies (6.38) (which holds for P large enough) and for which $lcm(2, 3, ..., r)|(P-1)$ such that P lies in an interval that is linear in n. Since the elements of $M := \bigcup_{i=1}^{r} F_i \cup \{0\}$ are to be reserved for the indices of bins of zeros of the prefix in the transformed domain we also require that $P - n > |M|$, since the total number of bins of zeros to be used is at most n (from the original string) + $|M|$ (from the prefix), and each bin receives a distinct index. Since $F_i = \cup_{j=1}^{2i-1} B_{i,j}(x_{i,j})$ and $|B_{i,j}(x_{i,j})|$ =$i\left(\lfloor \frac{G-\lfloor \frac{j}{2} \rfloor}{i} \rfloor + 1\right)$, whereby $i\left(\frac{G-i}{i}\right) \leq |B_{i,j}(x_{i,j})| \leq i\left(\frac{G+i}{i}\right)$, it follows that

$$|M| \leq \sum_{i=1}^{r}(2i-1)(G+i)+1 \leq (G+r)\sum_{i=1}^{r}(2i-1) = r^2(G+r)+1 \tag{6.39}$$

and

$$|M| \geq \sum_{i=1}^{r}(2i-1)(G-i)+1 \geq (G-r)\sum_{i=1}^{r}(2i-1) = r^2(G-r)+1 \tag{6.40}$$

Equation (6.39) yields a sufficient requirement on how large P needs to be

$$P > n + r^2(\log_2(P)+r)+1\ . \tag{6.41}$$

For given integers n and r (n is typically large and r is small), we essentially need to show that there exists a prime P for which $k := lcm(2, 3, ..., r)|(P-1)$ and $P \in (c_1 n, c_2 n)$ (here c_1 and c_2 are positive numbers that do not depend on n) and such that P satisfies (6.38) and (6.41).

For the asymptotic regime as $n \to \infty$ we recall the prime number theorem for arithmetic progressions [52] which states that

$$\pi(n, k, 1) \sim \frac{1}{\phi(k)} \frac{n}{\log(n)} , \tag{6.42}$$

where $\pi(n, k, 1)$ denotes the number of primes $\leq n$ that are congruent to $1 \mod k$, and $\phi(k)$ is the Euler function and represents the number of integers $\leq k$ that are relatively prime with k. As $n \to \infty$, we may let $c_1 := 2$ and $c_2 := 4$, so that

$$\frac{\pi(4n, k, 1)}{\pi(2n, k, 1)} \sim 2 , \tag{6.43}$$

and thus there exists a prime P, $k|(P-1)$ in an interval that is linear in n. Clearly, as $n \to \infty$, such P also satisfies (6.38) and (6.41).

For finite (but possibly very large) values of n and certain small r we appeal to results by Ramare and Rumely [47]. The number-theoretic function $\theta(x; k, l)$ is usually defined as

$$\theta(x; k, l) = \sum_{p \text{ prime } , p \equiv l \mod k, p \leq x} \ln p .$$

To show that there exists a prime P in the interval $(c_1 n, c_2 n)$ for which $k = lcm(2, 3, ..., r)|(P-1)$ it is sufficient to have

$$\theta(c_2 n; k, 1) > \theta(c_1 n; k, 1) , \tag{6.44}$$

where $k = lcm(2, 3, ..., r)$.

Theorem 2 in [47] states that $|\theta(x; k, 1) - \frac{x}{\phi(k)}| \leq 2.072\sqrt{x}$ for all $x \leq 10^{10}$ for k given in Table I of [47]. For larger x, Theorem 1 in [47], provides the bounds of the type

$$(1-\varepsilon)\frac{x}{\phi(k)} \leq \theta(x; k, 1) \leq (1+\varepsilon)\frac{x}{\phi(k)} \ , \tag{6.45}$$

for k given in Table I of [47], and ε also given in Table I of [47] for various x. Here $\phi(k)$ is the Euler function and denotes the number of integers $\leq k$ that are relatively prime with k.

For $c_2 n < 10^{10}$, using

$$\theta(c_1 n; k, 1) < \frac{c_1 n}{\phi(k)} + 2.072\sqrt{c_1 n}$$

and

$$\theta(c_2 n; k, 1) > \frac{c_2 n}{\phi(k)} - 2.072\sqrt{c_2 n},$$

it is thus sufficient to have

$$2.072\phi(k) < \sqrt{n}(\sqrt{c_2} - \sqrt{c_1}) \ , \tag{6.46}$$

for $\theta(c_2 n; k, 1) > \theta(c_1 n; k, 1)$ to hold.

For $c_1 n > 10^{10}$ using

$$\theta(c_1 n; k, 1) < (1+\varepsilon)\frac{c_1 n}{\phi(k)}$$

and

$$\theta(c_2 n; k, 1) > (1-\varepsilon)\frac{c_2 n}{\phi(k)},$$

after some simplification, it is sufficient to have

$$(1+\varepsilon)c_1 < (1-\varepsilon)c_2 \ , \tag{6.47}$$

for $\theta(c_2n;k,1) > \theta(c_1n;k,1)$ to hold.

Expressing $P \in (c_1n, c_2n)$ in terms of c_1n and c_2n, it is sufficient that

$$(c_1-1)n > r^2(\log_2 n + \log_2 c_2 + r) + 1 \tag{6.48}$$

for (6.41) to hold. Likewise, for $r \geq 2$, it is sufficient that

$$c_1n > 4r^3(\log_2 n + \log_2 c_2 + r)(\log_2 n + \log_2 c_2 + r - 1) + r^2(6 + \sqrt{c_2n}) \tag{6.49}$$

for (6.38) to hold.

Parameters c_1 and c_2 can be chosen as a function of r to make (6.46) (or (6.47)), (6.48) and (6.49) hold. We consider now some suitable choices for c_1 and c_2 for small values of r and some finite n.

- $r = 1$: The condition (6.48) reduces to $(c_1 - 1)n > \log_2(n) + \log_2(c_2) + 2$. For $c_2n < 10^{10}$, the condition (6.46) reduces to $\sqrt{n}(\sqrt{c_2} - \sqrt{c_1}) > 2.072$. We may let $c_2 = 4$ and $c_1 = 2$ for $12 < n < 10^{10}/4$ to ensure that there exists a prime in the interval $(2n, 4n)$ which satisfies (6.48).

  The condition (6.47) applies to $c_1n > 10^{10}$ so we may let $c_1 = 4$ for $n > 10^{10}/4$. Since all ε entries for $k = 1$ in Table I of [47] are $\ll 1/9$, we may let $c_2 = 5$ to make the condition (6.48) hold .

  Since $|M| \leq (\lfloor\log_2 P\rfloor + 2) \leq (\log_2 n + \log_2 c_2 + 2)$ (from (6.39)), and $|M| \geq \lfloor\log_2 P\rfloor \geq (\log_2 n + \log_2 c_1 - 2) + 1$ (from (6.40)) it follows that $(\log_2 n) \leq |M| \leq (\log_2 n + 4)$ for $12 < n < 10^{10}/4$ and $(\log_2 n + 1) \leq |M| \leq (\log_2 n + 5)$ for $n > 10^{10}/4$.

- $r = 2$: The conditions (6.48) and (6.49) reduce to $(c_1 - 1)n > 4(\log_2(n) + \log_2(c_2) + 2) + 1$ and $c_1 n > 32(\log_2 n + \log_2 c_2 + 2)(\log_2 n + \log_2 c + 1) + 4(6 + \sqrt{c_2 n})$.

  For $c_2 n < 10^{10}$, the condition (6.46) is again $\sqrt{n}(\sqrt{c_2} - \sqrt{c_1}) > 2.072$. We may let $c_1 = 2^{10}$ and $c_2 = 2^{11}$ to satisfy the required conditions (6.46), (6.48) and (6.49) for $10 \le n \le 10^{10}/2^{11} = 1/2 \times 5^{10}$.

  For $n \ge 1/2 \times 5^{10}$, we may let $c_1 = 2^{11}$ and $c_2 = 2^{12}$ to satisfy the required conditions (6.47) (since all ε entries in Table I of [47] are $\ll 1/3$), (6.48) and (6.49).

  Thus we have $4(\log_2 n + 7) + 1 \le |M| \le 4(\log_2 n + 14) + 1$, for $n \ge 10$.

- $r = 3$: The conditions (6.48) and (6.49) reduce to $(c_1 - 1)n > 9(\log_2(n) + \log_2(c_2) + 3) + 1$ and $c_1 n > 4 \cdot 27(\log_2 n + \log_2 c_2 + 3)(\log_2 n + \log_2 c + 2) + 9(6 + \sqrt{c_2 n})$.

  For $c_2 n < 10^{10}$, the condition (6.46) is now $\sqrt{n}(\sqrt{c_2} - \sqrt{c_1}) > 2.072 \times 2$. We may let $c_1 = 2^{12}$ and $c_2 = 2^{13}$ to satisfy the required conditions (6.46), (6.48) and (6.49) for $10 \le n \le 10^{10}/2^{13} = 1/8 \times 5^{10}$.

  For $n \ge 1/8 \times 5^{10}$ it suffices to let $c_1 = 2^{13}$ and $c_2 = 2^{14}$ to ensure (6.46), (6.48) and (6.49) are satisfied.

  Thus we have $9(\log_2 + 8) + 1 \le |M| \le 9(\log_2 n + 17) + 1$, for $n \ge 10$.

- $r = 4$: The conditions (6.48) and (6.49) reduce to $(c_1 - 1)n > 16(\log_2(n) + \log_2(c_2) + 4) + 1$ and $c_1 n > 4 \cdot 64(\log_2 n + \log_2 c_2 + 4)(\log_2 n + \log_2 c + 3) + 16(6 + \sqrt{c_2 n})$.

  For $c_2 n < 10^{10}$, the condition (6.46) is $\sqrt{n}(\sqrt{c_2} - \sqrt{c_1}) > 2.072 \times 4$. We may let

$c_1 = 2^{13}$ and $c_2 = 2^{14}$ to satisfy the required conditions (6.46), (6.48) and (6.49) for $16 \le n \le 10^{10}/2^{14} = 1/16 \times 5^{10}$.

For $n \ge 1/16 \times 5^{10}$ it suffices to let $c_1 = 2^{14}$ and $c_2 = 2^{15}$ to ensure (6.46), (6.48) and (6.49) are satisfied.

Thus we have $16(\log_2 +8) + 1 \le |M| \le 16(\log_2 n + 19) + 1$, for $n \ge 16$.

- $r = 5$: The conditions (6.48) and (6.49) reduce to $(c_1 - 1)n > 25(\log_2(n) + \log_2(c_2) + 5) + 1$ and $c_1 n > 4 \cdot 125(\log_2 n + \log_2 c_2 + 5)(\log_2 n + \log_2 c + 4) + 25(6 + \sqrt{c_2 n})$.

  For $c_2 n < 10^{10}$, the condition (6.46) is $\sqrt{n}(\sqrt{c_2} - \sqrt{c_1}) > 2.072 \times 16$. We may let $c_1 = 2^{14}$ and $c_2 = 2^{15}$ to satisfy the required conditions (6.46), (6.48) and (6.49) for $19 \le n \le 10^{10}/2^{14} = 1/32 \times 5^{10}$.

  For $n \ge 1/32 \times 5^{10}$ it suffices to let $c_1 = 2^{15}$ and $c_2 = 2^{16}$ to ensure (6.46), (6.48) and (6.49) are satisfied.

  Thus we have $25(\log_2 +8) + 1 \le |M| \le 25(\log_2 n + 21) + 1$, for $n \ge 19$.

6.2 Prefixing Algorithm

Let r denote the target synchronization error correction capability. The goal of this section it to provide an explicit prefixing scheme which, based on the string s of length n, produces a fixed length prefix $\mathbf{p}_\mathbf{s}$ of length m, where $\mathbf{p}_\mathbf{s}$ is a function of s, such that the string $\mathbf{t}_\mathbf{s} = [\mathbf{p}_\mathbf{s}\ \mathrm{s}]$ after the transformation T_{m+n} given in (5.1), call it $\tilde{\mathbf{t}_\mathbf{s}}$, satisfies first r

congruency constraints previously described in (5.33), which were shown to be sufficient to provide immunity to r repetition errors. Using judiciously chosen prefix, we will show that this will be possible for $m = |\mathbf{p_s}| = O(\log n)$.

We select as $\mathbf{p_s}$ that preimage with the property that in the concatenation $[\mathbf{p_s s}]$ the last of $\mathbf{p_s}$ is the complement of the first bit of $\mathbf{s}$. This property ensures that no bin of zeros in the transformed domain spans the boundary separating the substrings corresponding to the transformed prefix and the transformed original string.

For a given repetition error correction capability r and the original string length n let P be a prime number with the property that $lcm(2, 3, ..., r)|(P-1)$ and such that P lies in an interval that scales linearly with n, namely that $P \in (c_1 n, c_2 n)$ for $1 < c_1 < c_2$, where c_1, c_2 possibly depend on r but not on n and are chosen such that (6.46) (or (6.47), depending on how large n is), (6.48) and (6.49) hold. The existence of such P was discussed in the previous Section. Let R_P be the set of all residues $\mod P$. Recall that $M = \cup_{i=1}^{r} F_i \cup \{0\}$ denotes the set of indices of bins of zeros reserved for the prefix, where $F_i = \cup_{j=1}^{2i-1} B_{i,j}(x_{i,j})$ where $B_{i,j}(x_{i,j})$ are given in (6.9), and are constructed such that all sets $B_{i,j}(x_{i,j})$ for $1 \leq i \leq r$, $1 \leq j \leq 2i-1$ are nonintersecting. The existence of disjoint sets $B_{i,j}(x_{i,j})$ for such P was proved in Lemma 19. Let $L = |M|$. Let N denote the total number of bins of zeros of $\tilde{\mathrm{s}}$, where $\tilde{\mathrm{s}} = \mathrm{s}T_n$. By construction, $N \leq n$. Let

$$\begin{aligned} a'_1 &\equiv \textstyle\sum_{i=L+1}^{L+N} b_i f_i \text{ mod } P, \\ a'_2 &\equiv \textstyle\sum_{i=L+1}^{L+N} b_i f_i^2 \text{ mod } P \\ &\vdots \\ a'_r &\equiv \textstyle\sum_{i=L+1}^{L+N} b_i f_i^r \text{ mod } P \end{aligned} \tag{6.50}$$

where b_i is the size of the ith bin of zeros in $\tilde{\mathbf{t}_{\mathbf{s}}}$, and f_i in (6.50) are chosen in the increasing order from the set $R_P \setminus M$. Since $N \leq n$, and since by the condition (6.48), $n \leq P - L$, the set $R_P \setminus M$ is large enough to accommodate such f_i's.

We may think of a'_1 through a'_r as the contribution of the original string s (in the transformed domain) to the overall congruency value, since the ith bin of zeros for $L+1 \leq i \leq L+N$ is precisely the jth bin of zeros in $\tilde{\mathrm{s}}$ for $j = i - L$, since no run spans both $\mathbf{p_s}$ and s by the choice of $\mathbf{p_s}$.

Since not all strings in the original code may have the same number of bins of zeros in the transformed domain, we may view the unused elements of the set $R_P \setminus M$ as corresponding to ”virtual” bins of size zero. Since these bins are not altered during the transmission that causes r or less repetitions, the locations of repetitions can be uniquely determined as shown in the proof of Lemmas 13 and 15.

We now show that it is always possible to achieve

$$\begin{aligned} a_1 &\equiv \textstyle\sum_{i=1}^{L+N} b_i f_i \text{ mod } P, \\ a_2 &\equiv \textstyle\sum_{i=1}^{L+N} b_i f_i^2 \text{ mod } P, \\ &\vdots \\ a_r &\equiv \textstyle\sum_{i=1}^{L+N} b_i f_i^r \text{ mod } P, \end{aligned} \tag{6.51}$$

for arbitrary but fixed values a_1 through a_r irrespective of the values a'_1 through a'_r, where b_i is either 0 or 1 for $1 \leq i \leq L-1$, and where $f_L = 0$.

Before describing the encoding method that achieves (6.51) we state the following convenient result.

Lemma 20 *Suppose P is a prime number such that $i|(P-1)$. Suppose the equation $x^i \equiv a \mod P$ has a solution, $1 \leq a \leq P-1$. Then the equation $x^i \equiv a \mod P$ has i distinct solutions [3] and we may call them x_1 through x_i. The sum $\sum_{k=1}^{i} x_k^j \equiv 0 \mod P$ for $1 \leq j \leq i-1$.*

Proof: Let us consider the equation $x^i \equiv a \mod P$. Using Vieta's formulas and Newton's identities over $GF(P)$ it follows that $\sum_{k=1}^{i} x_k^j \equiv 0 \mod P$ for $1 \leq j \leq i-1$. ■

The encoding procedure is recursive and proceeds as follows.

Let l be the lth level of recursion for $l = 1$ to $l = r$. The lth level ensures that the lth congruency constraint in (6.51) is satisfied without altering previous $l-1$ levels. At each level l, starting with $l = 1$ and while $l \leq r$:

1. Select a subset T_l of $F_l = \cup_{j=1}^{2l-1} B_{l,j}(x_{l,j})$ such that $\sum_{k \in T_l} k^l \equiv a_l - a'_l - \sum_{i=1}^{l-1} d_{i,l} \mod P$, and such that if an element y, $y^l \equiv z \mod P$ of $B_{l,j}(x_{l,j})$ is selected, then so

are all other $l-1$ lth roots of z (which are also elements of $B_{l,j}(x_{l,j})$ by construction). For $l=1$, $\sum_{k\in T_1} k \equiv a_1 - a'_1 \mod P$.

2. Let $d_{l,j} \equiv \sum_{k\in T_l} k^j \mod P$ for $l+1 \leq j \leq r$.

3. For each i, $1 \leq i \leq |F_l|$, for which $f_i \in T_l$ we set $b_i = 1$, and for each i, for which $f_i \notin T_l$ we set $b_i = 0$.

4. Proceed to level $l+1$.

After the level r is completed, let $b_L = \sum_{i=1}^{r}(|F_i| - |T_i|)$. The purpose of this bin with weighting zero is to ensure that the overall string $\mathbf{t_s}$ has the same length irrespective of the structure of the starting string s.

The existence of $T_l, T_l \subseteq F_l$ in Step 1) follows from Lemmas in Section 6.1. In particular, recall that each residue $\mod P$ can be expressed as a sum of a subset L_l of $\cup_{j=1}^{2l-1} A_{l,j}(x_{l,j})$, by Lemma 18. We then let T_l consist of all lth power roots of elements in L_l. By construction, T_l is the union of appropriate subsets of sets $B_{l,j}(x_{l,j})$, whose lth powers are precisely the elements of L_l, and these subsets are disjoint by construction.

Recall that the sets $B_{l,j}(x_{l,j})$ are constructed such that if an lth power root of a residue y belongs to $B_{l,j}(x_{l,j})$ then all l power roots of y also belong to $B_{l,j}(x_{l,j})$. Then, by Lemma 20 the contribution to each congruency sum for levels 1 through $l-1$ of the elements of F_l is zero. Hence, once the target congruency value is reached for a particular level, it will not be altered by establishing congruencies at subsequent levels. As a result, since each string $\tilde{\mathbf{t_s}}$ satisfies congruency constraints given in (5.33), the resulting set of strings is immune to

r repetitions while incurring asymptotically negligible redundancy.

6.3 Decoding Algorithm

Suppose we wish to communicate over a channel that introduces additive and up to s repetition errors per coded sequence. We further assume that these repetitions can occur anywhere in the sequence but also that at most one repetition per original bit is allowed.

We develop an iterative decoding algorithm suitable for such a channel and for a channel code C that itself has a nice graphical representation relating bits and checks. For example the code C could be an LDPC code prepended with a repetition error correcting prefix in way that was described in the previous section.

Let nt be the length of a transmitted string from the set C. We wish to develop a variant of a message passing decoding algorithm of reasonable complexity. To facilitate message exchange, we introduce the following auxiliary variables:

- G_j for $1 \leq j \leq s$, $G_j \in \{1, ..., nt\}$. Variable G_j indicates the bit location of the jth repetition.

- $R_{i,j}$ for $1 \leq i \leq nt$ and $1 \leq j \leq s$, $R_{i,j} \in \{-1, 0, 1\}$. Variable $R_{i,j}$ indicates the relative location of the ith bit with respect to the jth repetition, so that $R_{i,j} = -1$ (0, resp. 1) if the j the repetition is after (at, resp. before) the i th bit.

- T_i for $1 \leq i \leq nt$, $T_i \in \{-s, -s+1, ..., s-1, s\}$. Variable T_i indicates the relative position of the ith bit with respect to all repetitions so that $T_i = -s$ if all repetitions

local domain	local function $\varphi(\cdot)$
$\{G_j\}$	1
$\{G_j, R_{i,j}\}$	$1(G_j > i)1(R_{i,j} = -1)+$ $1(G_j = i)1(R_{i,j} = 0) + 1(G_j < i)1(R_{i,j} = 1)$
$\{T_i, R_{i,1}, R_{i,2}, \cdots, R_{i,j}\}$	$1(T_i = -s)1(R_{i,1} = -1)1(R_{i,2} = -1)\cdots 1(R_{i,s} = -1)+$ $1(T_i = -s+1)1(R_{i,1} = 0)1(R_{i,2} = -1)\cdots 1(R_{i,s} = -1)+$ $\vdots$ $1(T_i = s)1(R_{i,1} = 1)1(R_{i,2} = 1)\cdots 1(R_{i,s} = 1)$
$\{T_i, x_i\}$	$1(T_i = -s)P(y_i \mid x_i) + 1(T_i = -s+1)P(y_i \mid x_i)P(y_{i+1} \mid x_i)+$ $1(T_i = -s+2)P(y_{i+1} \mid x_i) + \cdots + 1(T_i = s)P(y_{i+s} \mid x_i)$
$\{T_i, T_{i+1}\}$	$1(T_{i+1} \geq T_i)1(\text{parity}(T_i) = \text{parity}(s))+$ $1(T_{i+1} > T_i)1(\text{parity}(T_i) \neq \text{parity}(s))$
$\{c_k, (x_j, j \in \mathcal{N}_k)\}$	$1(c_k = \oplus_{j \in \mathcal{N}_k} x_j)$

Table 6.1: Local domains and functions.

occurred strictly after the ith bit, $T_i = -s+1$ if the first leftmost repetition occurred at the ith bit, and all remaining $s-1$ repetitions occurred strictly afterwards, $T_i = -s+2$ if the first leftmost repetition occurred strictly before the ith bit, and all remaining $s-1$ repetitions occurred strictly afterwards, and so on until $T_i = s$ if all repetitions occurred strictly before the ith bit.

We now group the variables as shown in Table 6.1 for $1 \leq i \leq nt$, $1 \leq j \leq s$ and $1 \leq k \leq M_c$, where M_c is the total number of checks of the code C. Note that $\mathbf{y}_1^{nt+s} = (y_1, y_2, \ldots, y_{nt+s})$ is viewed as evidence.

The junction graph corresponding to these local domains is shown in Figure 6.1,and has the bidirectional edges between:

- $\{G_j\}$ and $\{G_j, R_{i,j}\}$ for each $1 \leq i \leq nt$, and each $1 \leq j \leq s$.

- $\{G_j, R_{i,j}\}$ and $\{T_i, R_{i,1}, R_{i,2}, \ldots, R_{i,s}\}$ for each $1 \leq i \leq nt$, and each $1 \leq j \leq s$.

- $\{T_i, R_{i,1}, R_{i,2}, \ldots, R_{i,s}\}$ and $\{T_i, x_i\}$ for each $1 \leq i \leq nt$.
- $\{T_i, x_i\}$ and $\{T_{i-1}, T_i\}$ for each $2 \leq i \leq nt$.
- $\{T_i, x_i\}$ and $\{T_{i+1}, T_i\}$ for each $1 \leq i \leq nt - 1$.

The variables T_i and $R_{i,j}$ are introduced to ensure the consistency among the variables G_j, namely, that $G_{j_1} < G_{j_2}$ for $j_1 < j_2$, without having to directly compare G_j's. The local function at $\{T_i, T_{i+1}\}$ ensures the consistency between T_i and T_{i+1}, namely that T_{i+1} cannot be smaller than T_i, and in particular, must be strictly bigger than T_i if T_i itself corresponds to a repetition. Variables R_{ij} essentially provide local consistency between G_j's and T_i's. The benefit of introducing these variables and their local comparisons is in achieving the computational complexity of $O(nt)$, as we further discuss below.

We use a message passing algorithm along the lines of [1] to try to evaluate the posterior probability $P(x_i|\mathbf{y}_1^{nt+s})$ as an (approximate) product of all incoming messages multiplied by the local potential at $\{T_i, x_i\}$ and marginalized over T_i (see Figure 6.1).

The decoding algorithm starts with all messages being initialized to 1. Suppose $\alpha_{i,j}(R_{i,j})$ is the message sent from $\{T_i, R_{i,1}, R_{i,2}, \ldots, R_{i,s}\}$ to $\{G_j, R_{i,j}\}$ at some stage.

The message $\beta_{i,j}(G_j)$ from $\{G_j, R_{i,j}\}$ to $\{G_j\}$ is then

$$\begin{aligned}\beta_{i,j}(G_j) &= \textstyle\sum_{R_{i,j}} \alpha_{i,j}(R_{i,j})\varphi(G_j, R_{i,j}),\\ &= \begin{cases} \alpha_{i,j}(-1) & \text{if } G_j > i, \\ \alpha_{i,j}(0) & \text{if } G_j = i, \\ \alpha_{i,j}(1) & \text{if } G_j < i\ . \end{cases}\end{aligned}$$

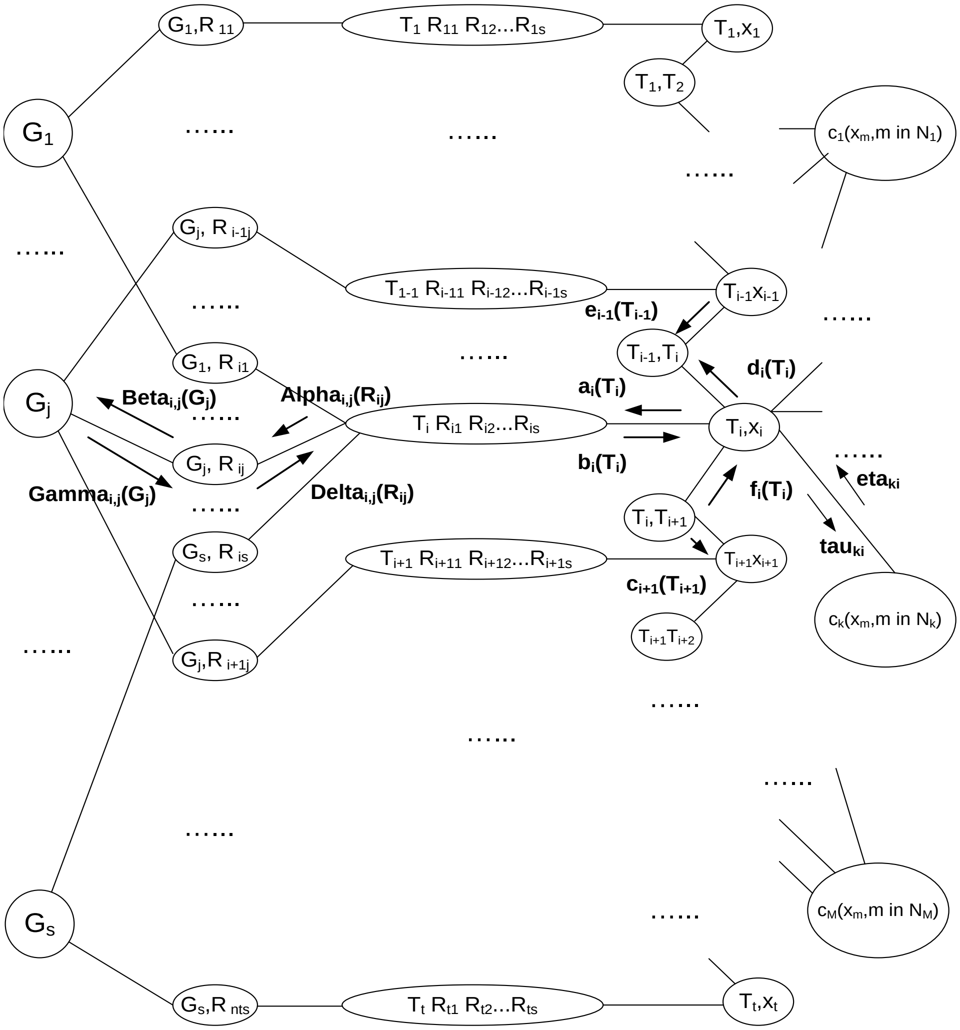
G_1,R_{11}
$T_1\ R_{11}\ R_{12}\ldots R_{1s}$
T_1,x_1
T_1,T_2
G_1
……
……
……
$c_1(x_m, m \text{ in } N_1)$
$G_j,\ R_{i-1j}$
……
$T_{1-1}\ R_{i-11}\ R_{i-12}\ldots R_{i-1s}$
$T_{i-1}x_{i-1}$
……
$e_{i-1}(T_{i-1})$
……
T_{i-1},T_i
$G_1,\ R_{i1}$
……
$d_i(T_i)$
$a_i(T_i)$
$\mathrm{Beta}_{i,j}(G_j)$
$\mathrm{Alpha}_{i,j}(R_{ij})$
G_j
……
$T_i\ R_{i1}\ R_{i2}\ldots R_{is}$
T_i,x_i
$G_j,\ R_{ij}$
$b_i(T_i)$
……
eta_{ki}
$\mathrm{Gamma}_{i,j}(G_j)$
……
$\mathrm{Delta}_{i,j}(R_{ij})$
$f_i(T_i)$
T_i,T_{i+1}
tau_{ki}
$G_s,\ R_{is}$
$T_{i+1}\ R_{i+11}\ R_{i+12}\ldots R_{i+1s}$
$T_{i+1}x_{i+1}$
$c_{i+1}(T_{i+1})$
……
$T_{i+1}T_{i+2}$
$c_k(x_m, m \text{ in } N_k)$
……
G_j,R_{i+1j}
……
……
……
……
G_s
……
$c_M(x_m, m \text{ in } N_M)$
G_s,R_{nts}
$T_t\ R_{t1}\ R_{t2}\ldots R_{ts}$
T_t,x_t

Figure 6.1: Junction graph.

The message $\gamma_{i,j}(G_j)$ sent from $\{G_j\}$ to $\{G_j, R_{i,j}\}$ is

$$\gamma_{i,j}(G_j) = \prod_{k=1,k\neq i}^{nt} \beta_{k,j}(G_j) = \frac{1}{\beta_{i,j}(G_j)} \prod_{k=1}^{nt} \beta_{k,j}(G_j)\ . \tag{6.52}$$

The message from $\{G_j, R_{i,j}\}$ to $\{T_i, R_{i,1}, R_{i,2}, \ldots, R_{i,s}\}$ is

$$\delta_{i,j}(R_{i,j}) = \sum_{G_j} \gamma_{i,j}(G_j)\varphi(G_j, R_{i,j}). \tag{6.53}$$

The message $a_i(T_i)$ from $\{T_i, x_i\}$ to $\{T_i, R_{i,1}, R_{i,2}, \ldots, R_{i,s}\}$ is expressed in terms of $c_i(T_i)$, $f_i(T_i)$ and $\mu_k(x_i)$

$$a_i(T_i) = \sum_{x_i} c_i(T_i) f_i(T_i) \prod_{k\in N(i)} \mu_k(x_i)\varphi(T_i, x_i) \tag{6.54}$$

where the message $\mu_k(x_i)$ is the message received from the kth check node in which bit x_i participates, and $N(i)$ is the set of checks in which the node i participates. The messages $c_i(T_i)$ and $f_i(T_i)$ are

$$c_i(T_i) = \sum_{T_{i-1}} e_{i-1}(T_{i-1})\varphi(T_{i-1}, T_i),\ \text{and} \tag{6.55}$$

$$f_i(T_i) = \sum_{T_{i+1}} d_{i+1}(T_{i+1})\varphi(T_i, T_{i+1}) \tag{6.56}$$

for

$$e_i(T_i) = \sum_{x_i} c_i(T_i) b_i(T_i) \prod_{k\in N(i)} \mu_k(x_i)\varphi(T_i, x_i),\ \text{and} \tag{6.57}$$

$$d_i(T_i) = \sum_{x_i} b_i(T_i) f_i(T_i) \prod_{k\in N(i)} \mu_k(x_i)\varphi(T_i, x_i)\ . \tag{6.58}$$

The message $b_i(T_i)$ from $\{T_i, R_{i1}, R_{i2}, \ldots, R_{is}\}$ to $\{T_i, x_i\}$ is

$$b_i(T_i) = \sum_{R_{i,1}} \sum_{R_{i,2}} \cdots \sum_{R_{i,s}} \delta_{i,1}(R_{i,1})\delta_{i,2}(R_{i,2}) \cdots \delta_{i,s}(R_{i,s})\varphi(T_1, R_{i,1}, R_{i,2}, \ldots, R_{i,s}) \tag{6.59}$$

and the message $\alpha_{i,j}(R_{ij})$ from $\{T_i, R_{i1}, R_{i2}, \ldots, R_{is}\}$ to $\{G_j, R_{ij}\}$ is

$$\begin{aligned}\alpha_{i,j}(R_{i,j}) \;&=\; \textstyle\sum_{T_i}\sum_{R_{i,1}}\cdots\sum_{R_{i,j-1}}\sum_{R_{i,j+1}}\cdots\sum_{R_{i,s}} a_i(T_i)\delta_{i,1}(R_{i,1})\cdots\delta_{i,j-1}(R_{i,j-1})\\ &\qquad \delta_{i,j+1}(R_{i,j+1})\cdots\delta_{i,s}(R_{i,s})\varphi(T_i, R_{i,1}, R_{i,2}, \ldots, R_{i,s}) \;.\end{aligned} \tag{6.60}$$

Recall that we want to compute the (approximate) posterior probability of x_i given the evidence $\mathbf{y}_1^{nt+s}$,

$$P(x_i|\mathbf{y}_1^{nt+s}) \approx \sum_{T_i} c_i(T_i)f_i(T_i)b_i(T_i)\prod_{k\in N(i)}\mu_k(x_i)\varphi(T_i, x_i) \;. \tag{6.61}$$

The complexity of computing all of $\alpha_{i,j}(R_{i,j})$, $\beta_{i,j}(G_j)$, $\gamma_{i,j}(G_j)$, and $\delta_{i,j}(R_{i,j})$ can be reduced by circumventing $\beta_{i,j}(G_j)$'s and $\gamma_{i,j}(G_j)$'s, and directly computing messages $\delta_{i,j}(R_{i,j})$ from $\alpha_{i,j}(R_{i,j})$ as

$$\delta_{i,j}(-1) = \sum_{G_j>i}\gamma_{i,j}(G_j) = \sum_{G_j>i}\frac{V_j(G_j)}{\beta_{i,j}(G_j)} = \frac{1}{\alpha_i(-1)}\sum_{G_j>i}V_j(G_j) \;. \tag{6.62}$$

Likewise

$$\delta_{i,j}(0) = \frac{1}{\alpha_i(0)}V_i(G_i) \;, \tag{6.63}$$

and

$$\delta_{i,j}(1) = \frac{1}{\alpha_i(1)}\sum_{G_j<i}V_j(G_j) \;, \tag{6.64}$$

where

$$V_j(G_j) = \prod_{k=1}^{nt}\beta_{k,j}(G_j) = \prod_{k=1}^{i-1}\alpha_{k,j}(-1)\cdot\alpha_{i,j}(0)\cdot\prod_{k=i+1}^{nt}\alpha_{k,j}(1) \;. \tag{6.65}$$

For fixed j ($1 \leq j \leq s$), given $\alpha_{k,j}(R_{k,j})$ for $1 \leq k \leq nt$ we can compute $V_j(G_j)$ in $O(nt)$ steps. Having computed $V_j(G_j)$'s, we can then also compute $\delta_{i,j}(R_{i,j})$ messages in

$O(nt)$ steps. Once we have all $\delta_{i,j}(R_{i,j})$ messages, each $b_i(T_i)$ message can be computed in $O(3^s)$ steps.

Given the messages $e_i(T_i)$'s, for $1 \leq i < nt$, each message $c_i(T_i)$ is computed in $O(s)$ steps. Likewise, given the messages $d_i(T_i)$'s, for $1 < i \leq nt$, each message $f_i(T_i)$ is computed in $O(s)$ steps.

The messages $\tau_{k,i}(x_i)$ and $\mu_{k,i}(x_i)$ are analogous to messages computed in a traditional message passing algorithm on a bipartite graph, so their complexity is also $O(nt)$.

Given the messages $c_i(T_i)$, $b_i(T_i)$, and the product $\prod_{k \in N(i)} \mu_{k,i}(x_i)$ (itself computed in $O(nt)$ steps) each $e_i(T_i)$, for $1 \leq i < nt$ is computed in a constant number of steps. Similarly, given the messages $f_i(T_i)$, $b_i(T_i)$, and the product $\prod_{k \in N(i)} \mu_{k,i}(x_i)$, each message $d_i(T_i)$ is computed in a constant number of steps. The same holds for messages $a_i(T_i)$ which, based on $c_i(T_i)$, $f_i(T_i)$, and the product $\prod_{k \in N(i)} \mu_{k,i}(x_i)$ are also computed in the constant number of steps. Therefore, all messages $a_i(T_i)$ through $f_i(T_i)$ are computed in $O(nt)$ steps.

Based on $a_i(T_i)$ and $\delta_{i,k}(R_{i,k})$ for $1 \leq k \leq s$ and $k \neq j$, each $\alpha_{i,j}(R_{i,j})$ can be computed in $O(s3^{s-1})$ steps. The total complexity of computing all $\alpha_{i,j}(R_{i,j})$ messages is also $O(nt)$ for the fixed parameter s.

6.4 Summary and Concluding Remarks

In this chapter we proposed a technique for prefixing a collection of binary strings of equal length to provide immunity to repetition errors. The presented prefixing scheme relies on introducing a carefully chosen prefix for each original binary string such that the resulting strings (each consisting od the prefix and one of the original strings) are immune to repetition errors. This prefix is constructed based on the number theoretic methods from the previous chapter. The prefix length is only logarithmic in the size of the original collection. We also presented a message passing decoding algorithm suitable for channels causing both repetition and additive errors. The proposed algorithm has the same complexity as the traditional message passing decoding algorithm capable of decoding under only additive errors.

Part II

Iterative Decoding of LDPC Codes

Chapter 7

Introduction

In this part of the thesis we are concerned with the analytic understanding of the LPDC code performance under iterative decoding, with the particular focus on the performance of finite-length LDPC codes in the low BER region.

Low density parity check (LDPC) codes are a class of error control codes defined on sparse graphs [26]. Their graphical representation makes them particularly amenable for low-complexity iterative decoding algorithms. LDPC codes were invented by Gallager [26] in the 1960's, but then were largely forgotten until early 1990's. Their rediscovery [39], [32] ignited intensive research in LDPC codes, as well as their wide consideration for many modern applications.

While vast empirical evidence points to the successful use of LDPC codes, most of the known theoretical results regarding the performance of LDPC codes are asymptotic in nature. A theoretical tool known as density evolution [49] operates on an infinitely long

LDPC code ensemble and it demonstrates an exponential concentration of the messages exchanged in the decoding process around their mean. The underlying assumption in density evolution is that a large enough neighborhood of each node is locally tree-like, which can be assumed as the block length tends to infinity. However, for finite-length LDPC codes (with block lengths on the order of hundreds or thousands) such assumption no longer holds, and in fact for structured finite-length LDPC codes there inevitably exist numerous relatively short cycles in the associated Tanner graph. Furthermore, in this finite blocklength regime, many LDPC codes exhibit a so-called “error floor”, corresponding to a significant flattening in the curve that relates signal to noise ratio (SNR) to the bit error rate (BER) level, typically occurring in the low BER region. Since moderate blocklengths and low BER’s are of primary interest in many communications and data storage applications, prior lack of understanding of the LDPC code performance has significantly hindered the wide-scale deployment of these very promising codes.

In this dissertation we aim to address this issue through the introduction and the subsequent study of a convenient combinatorial object, which we have termed an absorbing set.

The following chapter provides the background on the low BER performance of LDPC codes, where we discus the error floor, introduce the notion of an absorbing set and summarize some related work. Later, we will provide an in-depth study of absorbing sets for an important family of high-performance finite-length LDPC codes.

Chapter 8

Background on Iterative Decoding

In this chapter we focus on the LDPC code performance in the low BER region, and discuss the so-called "error-floor" phenomenon. We introduce the combinatorial object termed absorbing set, and relate it to some existing concepts form the literature. Having defined an absorbing set, in the next chapter we focus on the detailed theoretical analysis of absorbing sets for a family of high-rate array-based LDPC codes.

8.1 LDPC Codes, Message Passing Algorithms and Error Floors

Empirically, LDPC codes perform very well when decoded iteratively using message passing algorithms, despite the fact that such decoding algorithms are suboptimal on graphs with cycles (and graphs defining LDPC codes inevitably contain cycles). An added attractive feature of message passing algorithms is their low complexity.

However, it is also known [40], [48], that LDPC codes often exhibit an error floor phenomenon, whereby the bit error rate (BER) vs. signal to noise ratio (SNR) curve shows a significant decrease in the slope in the very low BER region. An example of the error-floor behavior is shown in Figure 8.1 (reproduced from [63]) for the Reed-Solomon based LDPC code, where different error curves correspond to specified number of iterations. The error floor implies that a significant increase in the signal power is needed for only a marginal improvement in the bit error rate. It is attributed to the suboptimal nature of the message passing algorithms on graphs with cycles.

For many applications, including data storage, gigabit ethernet, and satellite communications, it is imperative to reach this low BER region without requiring a major increase in SNR. This region, however, is out of the reach of pure software simulations, and consequently the limitations of a given LDPC code under message-passing decoding in the very low BER region are largely unknown.

In order to explain and analyze the dominant causes of decoding failures we introduce the notion of an absorbing set in the next section. The absorbing sets are related to (but not entirely equivalent to) previously introduced structures, including stopping sets [14], trapping sets [48], near codewords [40] and pseudo-codewords [20]. Fully absorbing sets are viewed as fixed points of a bit-flipping algorithm (which itself is a simplest form of message passing and can be viewed as a 1-bit approximation to the finite-precision message passing decoding algorithms that are used in practice). Our claim is that if there are fully

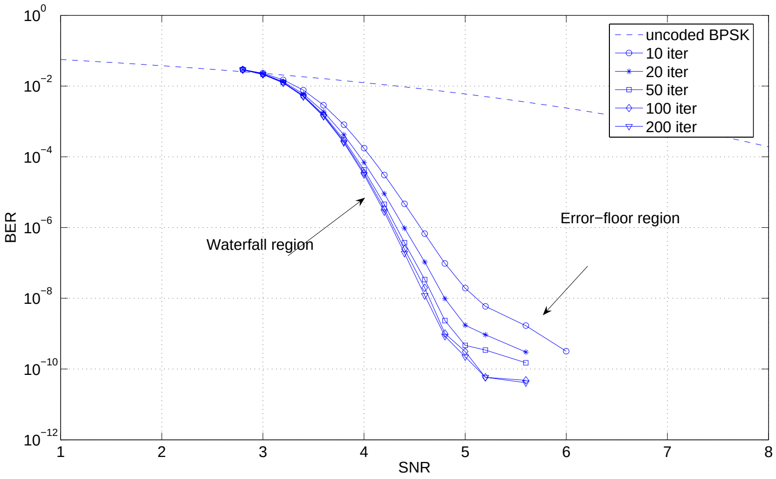

Figure 8.1: An example of error floor.

absorbing sets smaller that the minimum distance of the code, the decoder is likely to converge to these objects. As a result, under iterative decoding, the low BER performance will be dominated by the number and the size of dominant fully absorbing sets. This is in contrast to the conventional point of view which considers the number of minimum distance codewords and the minimum distance itself to be the key performance metric of a code.

8.2 Absorbing sets of LDPC codes

Several LDPC codes, while having excellent performance for moderate bit error rate (BER) levels of 10^{-6} and above, have been demonstrated using hardware based emulation [63] and software clusters [67] to suffer from the so-called “error floor”, see Figure 8.1.

This error-floor behavior can be attributed to the suboptimal nature of the message passing algorithms traditionally used in decoding of LDPC codes. Experimental results in the low FER region obtained using a hardware emulator [63] revealed that certain structures consisting of short cycles organized in particularly detrimental configurations in the Tanner graph associated with the parity check matrix of an LDPC code cause the decoder to fail by converging to a non-codeword state.

These combinatorial objects that describe the convergence to the non-codeword state are called *absorbing sets*. For many LDPC codes, designed to have sufficiently large minimum distance, the associated Tanner graphs contain absorbing sets which have strictly fewer bits than the weight of codewords at the minimum distance. As a result, the performance of the decoding algorithm in the low FER region is predominantly dictated by the number and the structure of minimal absorbing sets, rather than the number of the minimum distance codewords (which would be the key parameters in describing the performance of the code under ML decoding). Before explaining the links between absorbing sets and some related concepts, including near-codewords and trapping sets, in Subsection 8.2.2, we first provide the formal definition of these objects.

8.2.1 Formal definition

Let $G = (V, F, E)$ be a bipartite graph with the vertex set $V \cup F$, where V and F are disjoint, and with the edge set E, such that there exists an edge $e(i, j) \in E$ iff $i \in V$ and $j \in F$. One can associate a bipartite graph $G_H = (V, F, E)$ with a parity check matrix H, such that the set V corresponds to the columns of H, the set F corresponds to the rows of

H, and $E = \{e(i,j)|H(j,i) = 1\}$. Such a graph G_H is commonly referred to as the Tanner graph of the parity check matrix H of a code, [22]. Elements of V are called "bit nodes" and elements of F are called "check nodes". For the subset D of V we let N_D denote the set of check nodes neighboring the elements of D.

For a subset D of V, let $\mathcal{E}(D)$ (resp. $\mathcal{O}(D)$) be the set of neighboring vertices of D in F in the graph G with even (resp. odd) degree with respect to D. Given an integer pair (a,b), an (a,b) *absorbing set* is a subset D of V of size a, with $\mathcal{O}(D)$ of size b and with the property that each element of D has strictly fewer neighbors in $\mathcal{O}(D)$ than in $F\backslash\mathcal{O}(D)$. We say that an (a,b) absorbing set D is an (a,b) *fully absorbing set*, if in addition, all bit nodes in $V\backslash D$ have strictly more neighbors in $F\backslash\mathcal{O}(D)$ than in $\mathcal{O}(D)$.

An example of an (a,b) absorbing set with $a = 4$, $b = 4$ is given in Fig. 8.2, where full circles constitute the set D, full squares constitute the set $\mathcal{O}(D)$, empty squares constitute the set $\mathcal{E}(D)$, $E(D,\mathcal{O}(D))$ is given by solid lines, and $E(D,\mathcal{E}(D)$ is given by dashed lines. Observe that each element in D has more even-degree than odd-degree neighbors. All check nodes not in the picture are denoted by empty squares. For this set to be a fully absorbing set, every bit node not in the figure should also have strictly more empty squares than full squares as neighbors.

Note that $D \subseteq V$ is a fully absorbing set iff for all v, $wt(Hx_{D\Delta v}) > wt(Hx_D) = b$, where $D\Delta v$ denotes the symmetric difference between D and $\{v\}$, $wt(y)$ is the Hamming weight of a binary string y, and x_D is a binary string with support D.

We have introduced the notion of absorbing sets to qualitatively describe the convergent

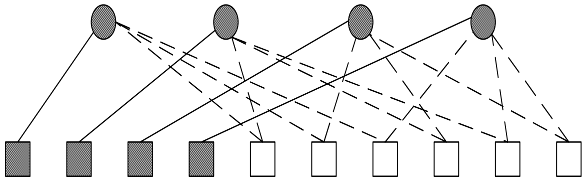

Figure 8.2: An example of a (4,4) absorbing set

non-codeword state of the message passing algorithms, when the transmission channel is additive white gaussian noise (AWGN). In the asymptotic limit given by the bit flipping algorithm, the configuration described as a fully absorbing set is stable, since each bit node receives strictly more messages from the neighboring checks that reinforce its value than messages that suggest the opposite bit value.

8.2.2 Related Work and Existing Concepts

The observation that the low BER/FER performance of finite length LDPC codes under iterative decoding is guided by non-codewords is not new. One of the first such attempts to explain these findings was described in [40] where it was recognized that non-codewords, rather than minimum distance codewords, can attribute to the error floor. There, the notion of a near-codeword was introduced. An (a, b) near-codeword refers to a binary string s of weight a whose syndrome $\mathrm{s}H^T$ has weight b. A fully absorbing set can be viewed as a near codeword as defined in [40], though the reverse is not true, since a near codeword does not necessarily describe a stable configuration.

The trapping sets were introduced in [48] as a part of the study of error floors of LDPC codes, which pioneered a simulation-emulation approach. Trapping sets as defined in [48] carry an operational, decoder dependent definition. In addition, they are defined as a union of all bits that are not eventually correct, and thus permit a situation in which the decoder oscillates among a finite number of states.

Stopping sets introduced and studied in detail in [14] refer to the subgraph of the Tanner graph with the property that no check node relative to this subgraph has degree 1. These sets describe stable combinatorial configurations in the context of a binary erasure channel (BEC), since the decoder halts once it encounters a stopping set in which all bit nodes were erased. The stopping sets have been shown to be a very useful tool in understanding the performance of LDPC codes on erasure channels, both for finite-length codes [28] as well as the asymptotic behavior of LDPC code ensambles, [45]. Nevertheless, such analysis cannot be directly applied to an AWGN channel since the nature of errors is different.

Additional related notions previously introduced in the literature include pseudo-codewords [20] and elementary trapping sets [33]. Note that the pseudo-codewords are defined in the context of the linear programming based decoding and their connection with the convergent non-codeword states of the iterative decoding algorithms though interesting is not yet fully established. Pseudo-codewords were also studied in [30] where it was observed that under so-called graph cover decoding, pseudo-codewords in the covers of the bipartite graph, along with the actual codewords, compete to be the best estimate produced by the decoder. Loop calculus method discussed in [9] provides a way to improve linear programming based decoding once the so-called critical loop is identified. While for short codes, as the (155,64) LDPC code discussed in [9], one loop may be sufficient to describe a critical state, it would be interesting to investigate how this concept extends to larger codes. While [9] uses a search algorithm to find “bad” configurations for the (155,64) LDPC code, one could, based on the structure of the parity check matrix of this code, analytically describe dominant absorbing sets, and then use these as a starting point for further analysis. It would be interesting to further pursue this connection.

Elementary trapping sets are defined as subgraphs in the Tanner graph in which each check satisfied with respect to this subgraph has degree 2 and each check unsatisfied with respect to this subgraph has degree 1, [33]. In a loose sense, one may view absorbing sets as consisting of a union of elementary trapping sets in some cases.

Chapter 9

Absorbing Sets of Array-based LDPC Codes

In this chapter we provide a detailed analysis of the minimal absorbing sets and minimal fully absorbing sets of the high rate array-based LDPC codes. In particular we will show that for $\gamma = 2$ the minimal (fully) absorbing sets are in fact minimum distance codewords, whereas, for $\gamma = 3$ and $\gamma = 4$, minimal (fully) absorbing sets are in fact strictly smaller that the minimum distance of the code.

9.1 Array-based LDPC codes

Array based LDPC codes [19] are regular LDPC codes parameterized by a pair of integers (p, γ), such that $\gamma \leq p$, and p is an odd prime, with a parity check matrix $H_{p,\gamma}$

given by

$$H_{p,\gamma} = \begin{bmatrix} I & I & I & \dots & I \\ I & \sigma & \sigma^2 & \dots & \sigma^{p-1} \\ I & \sigma^2 & \sigma^4 & \dots & \sigma^{2(p-1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ I & \sigma^{\gamma-1} & \sigma^{(\gamma-1)2} & \dots & \sigma^{(\gamma-1)(p-1)} \end{bmatrix} \tag{9.1}$$

where σ denotes a $p \times p$ permutation matrix of the form

$$\sigma = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}. \tag{9.2}$$

We use $C_{p,\gamma}$ to denote the binary linear code with parity check matrix of the form (9.1). The rate of this code is $R = 1 - \frac{\gamma p - \gamma + 1}{p^2}$, [42].

As first demonstrated by Fan [19], array-based LDPC codes have very good performance. They have been proposed for a number of applications, including digital subscriber lines [18] and magnetic recording [58].

In our earlier experimental work [63] we have observed that certain structures in the Tanner graph of the parity check matrix of the code are the limiting factor in the iterative decoding of several structured LDPC codes, including array-based codes. Motivated by the empirical findings, we introduced this object, which we call an *absorbing set*. The formal definition of absorbing sets is given in Section 8.2.1. Here we study them in detail for array-based LDPC codes $C_{p,\gamma}$ for $\gamma = 2, 3, 4$, for the standard parity check matrix $H_{p,\gamma}$.

9.2 Theoretical Results

Our goal is to describe minimal absorbing sets and minimal fully absorbing sets (a, b) of the Tanner graph of the parity check matrix $H_{p,\gamma}$, for $\gamma = 2, 3, 4$, where the minimality refers to the smallest possible a, and where b is the smallest possible for the given a.

We use the following notation throughout this chapter. Recall that $H_{p,\gamma}$ is a $\gamma p \times p^2$ matrix of 0's and 1's. It is convenient to view $H_{p,\gamma}$ as a two-dimensional array of component $p \times p$ submatrices with the rows i in the range $0 \leq i \leq \gamma - 1$ (also referred to as row groups) and the columns j in the range $0 \leq j \leq p - 1$ (also referred to as column groups). Each column of $H_{p,\gamma}$ is uniquely described by a pair (j, k) where j denotes the column index of the submatrix this column belongs to, and k, $0 \leq k \leq p - 1$ denotes the index of this column within the submatrix.

Let $G_{p,\gamma}$ be the Tanner graph associated with $H_{p,\gamma}$, so bit nodes and check nodes in $G_{p,\gamma}$ represent columns and rows in $H_{p,\gamma}$, respectively. In $G_{p,\gamma}$ bit nodes have degree γ and check nodes have degree p. There is a total of p^2 bit nodes and γp check nodes. Each bit node in $G_{p,\gamma}$ receives the unique label (j, k) that describes the corresponding column of $H_{p,\gamma}$. Each check node in $G_{p,\gamma}$ receives a label i if the corresponding row of $H_{p,\gamma}$ belongs to the row group i. Multiple bit nodes can have the same j or k label, but not both. Multiple check nodes can have the same i label.

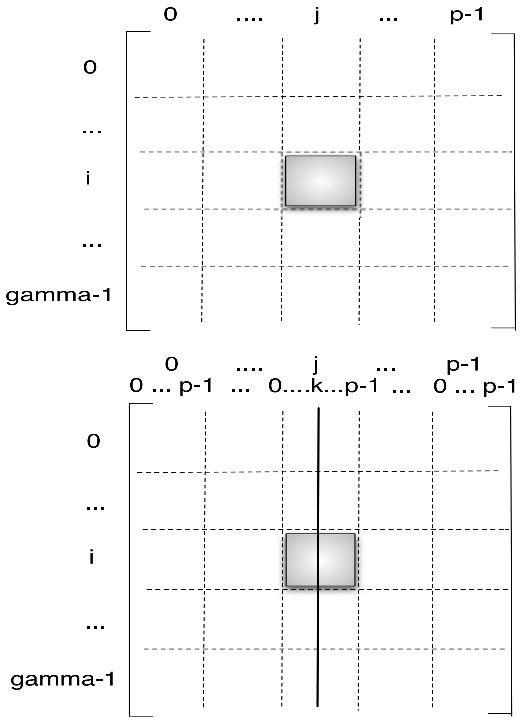

Figure 9.1: Illustration of the notation (a) Row and column groups in $H_{p,\gamma}$(b) (j, k) column label. Shaded area corresponds to the submatrix σ^{ij}.

We note that the structure of the parity check matrix imposes the following conditions on the neighboring bit nodes and check nodes:

Bit Consistency: For a bit node, all its incident check nodes, labelled i_{s_1} through i_{s_γ} must have distinct labels, i.e. these check nodes are in distinct row groups.

Check Consistency: All bit nodes, say (j_1, k_1) through (j_p, k_p), participating in the same check node must have distinct j_ℓ values, i.e. they are all in distinct column groups.

Both conditions follow from the fact that the parity check matrix $H_{p,\gamma}$ of $C_{p,\gamma}$ consists of a 2-dimensional array of permutation matrices of equal size. ■

We begin with elementary lemmas that play a central role throughout this chapter.

Lemma 21 *(Pattern Consistency): The (r, k) entry of σ^i is 1 iff $r - k \equiv i \mod p$.*

Corollary 1 *(Pattern Consistency): Let σ^{ij_1} and σ^{ij_2} be in the same row group of $H_{p,\gamma}$. If the entry (r, k_1) of σ^{ij_1} is non-zero, then so is the entry (r, k_2) of σ^{ij_2} where $k_1 + ij_1 \equiv k_2 + ij_2 \equiv r \mod p$.*

We will refer to the constraints of the type described in both Lemma 21 and Corollary 1 as *pattern consistency* constraints.

Lemma 22 *(Cycle consistency:) Consider a cycle in $G_{p,\gamma}$ of length $2t$, involving t bit nodes, with labels (j_1, k_1) through (j_t, k_t) and t check nodes, with labels i_1 through i_t, such that bit nodes (j_1, k_1) and (j_2, k_2) participate in the check labelled i_1, (j_2, k_2) and (j_3, k_3) participate in the check labelled i_2, and so on, until check labelled i_t in which (j_t, k_t) and*

(j_1, k_1) *participate. Then*

$$i_1(j_2-j_1)+i_2(j_3-j_2)+\cdots+i_{t-2}(j_{t-1}-j_{t-2})+i_{t-1}(j_t-j_{t-1})+i_t(j_1-j_t) \equiv 0 \mod p. \tag{9.3}$$

Proof: The pattern consistency constraints of Corollary 1 give:

$$\begin{aligned} k_1 + i_1 j_1 &\equiv k_2 + i_1 j_2 && \mod p, \\ k_2 + i_2 j_2 &\equiv k_3 + i_2 j_3 && \mod p, \\ &\vdots \\ k_{t-1} + i_{t-1} j_{t-1} &\equiv k_t + i_{t-1} j_t && \mod p, \\ k_t + i_t j_t &\equiv k_1 + i_t j_1 && \mod p. \end{aligned} \tag{9.4}$$

Expand $k_1 - k_2$ into $(k_1 - k_t) - (k_{t-1} - k_t) - (k_{t-2} - k_{t-1}) - \cdots - (k_2 - k_3)$. Hence,

$$i_1(j_2-j_1) \equiv i_t(j_t-j_1)-i_{t-1}(j_t-j_{t-1})-i_{t-2}(j_{t-1}-j_{t-2})-\cdots-i_2(j_3-j_2) \mod p. \tag{9.5}$$

By rearranging the terms, (9.3) follows. ■

Constraints of the type (9.3) will subsequently be referred to as cycle consistency constraints.

Our main results can be summarized as follows: Let $G_{p,\gamma}$ be the Tanner graph associated with the parity check matrix $H_{p,\gamma}$ of the array-based LDPC code $C_{p,\gamma}$.

Theorem 3 Minimality

(a) For the $G_{p,2}$ *family, all minimal absorbing sets are minimal fully absorbing sets and are of size* $(4, 0)$.

(b) For the $G_{p,3}$ family, the minimal absorbing sets are of size $(3,3)$, and the minimal fully absorbing sets are of size $(4,2)$.

(c) For the $G_{p,4}$ family, and for $p > 19$, all minimal absorbing sets are fully minimal absorbing sets, and are of size $(6,4)$. ■

Theorem 4 Scaling

(a) Suppose $\gamma = 2$ and $p > 3$. The number of minimal (fully) absorbing sets in $G_{p,\gamma}$ grows with block length n (Recall that the blocklength $n = p^2$, given by the number of columns in $H_{p,\gamma}$) as $\Theta(n^2)$.

(b) For $\gamma = 3$ and for all blocklengths $n > 3^2$ the number of minimal absorbing sets as well as the number of minimal fully absorbing sets in $H_{p,\gamma}$ is $\Theta(n^{3/2})$.

(c) For $\gamma = 4$ and for all blocklengths $n > 19^2$ the number of minimal absorbing sets as well as the number of minimal fully absorbing sets in $H_{p,\gamma}$ is $\Theta(n^{3/2})$. ■

Here we say that the number Q of particular absorbing sets grows as $\Theta(n^l)$ if $cn^l \leq Q \leq c'n^l$, for some constants c and c'.

The following three subsections provide proofs of these claims, where we separately treat each of the values of γ. While our results provide a precise count of the minimal (fully) absorbing sets, the main message is regarding the cardinality scaling is that of Theorem 4.

Note that Theorem 3(a) implies that for $\gamma = 2$ the smallest (fully) absorbing sets are in fact the minimum distance codewords. This is in contrast to the results in Theorem 3(b) for $\gamma = 3$ and 3(c) for $\gamma = 4$ for which we show the existence of (fully) absorbing sets strictly smaller than the minimum distance of the code. In particular, for $\gamma = 3$, the minimum distance is 6 [61], [42], and for $\gamma = 4$ and $p > 7$ the minimum distance is between 8 and 10, [61],[42]. The minimal absorbing sets and minimal fully absorbing sets are for both $\gamma = 3$ and $\gamma = 4$ and p large enough strictly smaller than the minimum distance of the code.

9.2.1 Absorbing sets of $H_{p,2}$

The code $C_{\gamma,2}$ has uniform bit degree 2, and is thus a cycle code. Even though such codes are known to be poor [46], we include the analysis for the sake of completeness. We start by proving the statement in Theorem 3(a).

Let $G_{p,2} = (V, F, E)$ denote the Tanner graph of $H_{p,2}$. Let D be an (a, b) absorbing set in $G_{p,2}$. Each bit node in D has degree 2 in $G_{p,2}$ and is required to have strictly more neighbors in $\mathcal{E}(D)$ than in $\mathcal{O}(D)$. This implies that $\mathcal{O}(D)$ is empty. The absorbing set is of type $(a, 0)$. It is thus a fully absorbing set, and is in fact a codeword.

Since the matrix $H_{p,2}$ has the top row consisting of identity matrices, the codewords of $C_{p,2}$ are of even weight. Moreover, since the bottom row of $H_{p,2}$ consists of distinct component submatrices, no two columns of $H_{p,2}$ sum to zero. Therefore $a > 2$ and even and there are no cycles of length 4 in this code.

We now consider $a = 4$. Let (j_1, k_1), (j_2, k_2), (j_3, k_3) and (j_4, k_4) be the bit nodes participating in a candidate $(4, 0)$ absorbing set. These nodes must necessarily be arranged as in Figure 9.2.

The following result proves Theorem 3(a).

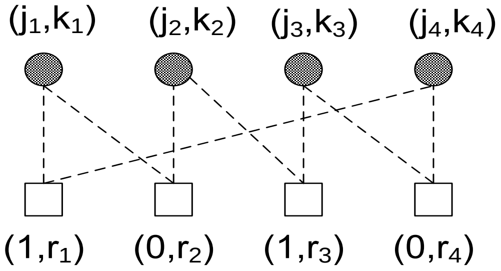

Figure 9.2: (Labelled) candidate (4,0) absorbing set

Lemma 23 *There is a total of* $p^2(p-1)^2$ $(4,0)$ *(fully) absorbing sets in the code described by* $H_{p,2}$.

Proof: The bit consistency conditions are automatically satisfied by the numbering of the row groups in Figure 9.2. The check consistency constraints give:

$$\begin{aligned} j_1 &\neq j_4 \\ j_1 &\neq j_2 \\ j_2 &\neq j_3 \\ j_3 &\neq j_4 \quad . \end{aligned} \tag{9.6}$$

The pattern consistency constraints of Corollary 1 give:

$$\begin{aligned} k_1 &= k_2 \\ k_3 &= k_4 \\ k_2 + j_2 &\equiv k_3 + j_3 \qquad \text{mod } p \\ k_4 + j_4 &\equiv k_1 + j_1 \qquad \text{mod } p. \end{aligned} \tag{9.7}$$

There are p ways of choosing k_2, which also determines k_1. Since $j_2 \neq j_3$, we must have $k_3 \neq k_2$, so we have $(p-1)$ ways of choosing k_3, which also determines k_4. We then have p ways of choosing j_2, which also determines j_3. Since $j_1 \neq j_2$, we have $(p-1)$ ways of choosing j_1, which also determines j_4. To verify that every one of these choices satisfies all the equations it only remains to verify that $j_3 \neq j_4$. This holds because

$$j_3 - j_4 \equiv (k_2 - k_3 + j_2) - (k_1 - k_4 + j_1) \equiv j_2 - j_1 \neq 0 \mod p \, . \tag{9.8}$$

Now, for any choice of row group labels for the checks, and column labels for the bits that satisfy the bit and check consistency constraints and the pattern consistency constraints of Corollary 1 there is a unique way to choose the row index in the individual row groups so that the pattern consistency constraint of Lemma 21 are satisfied. This completes the proof of Lemma 23. ■

Theorem 4(a) is now a consequence of

Corollary 2 *The number of* $(4,0)$ *(fully) absorbing sets for the code described by* $H_{p,2}$ *is* $\Theta(n^2)$*, where* n *is the codeword length.*

Proof: Follows immediately from Lemma 23 and $n = p^2$. ■

Note that $(4,0)$ absorbing sets are actually codewords, so the cycle code is dominated by low weight codewords. We now consider $\gamma > 2$, which leads to more interesting results. In particular, our results establish the existence of minimal absorbing sets and minimal fully absorbing sets, for which the number of bit nodes a is *strictly smaller* than the minimum distance d_{min} of the code.

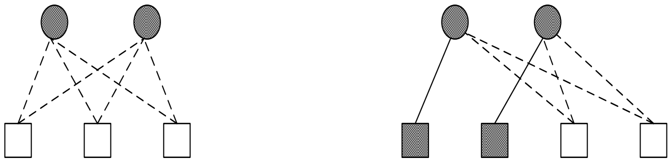

Figure 9.3: Candidate (2,b) absorbing sets

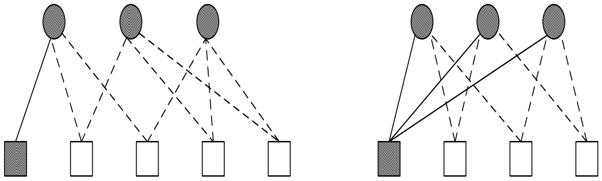

Figure 9.4: Candidate (3,1) absorbing sets

9.2.2 Absorbing sets of $H_{p,3}$

We now turn to the proof of Theorem 3(b), concerning the sizes and numbers of minimal absorbing sets in $H_{p,3}$.

Let $G_{p,3} = (V, F, E)$ denote the Tanner graph of $H_{p,3}$. Let D be an (a, b) absorbing set in $G_{p,3}$. Each bit node in D has degree 3 in $G_{p,3}$ and is required to have strictly more neighbors in $\mathcal{E}(D)$ than in $\mathcal{O}(D)$.

Suppose $a = 2$. In $G_{p,3}$ an even number of edges from D terminates in $\mathcal{E}(D)$. Thus either $b = 0$ or $b = 2$. These correspond to the situations in Figure 9.3. In either case there would be a cycle of length 4 in $G_{p,3}$, which is false [19]. Thus $a \geq 3$.

Suppose $a = 3$. In $G_{p,3}$ an even number of edges from D terminates in $\mathcal{E}(D)$. Thus either $b = 1$ or $b = 3$. Suppose $b = 1$. This must correspond to the left form in Figure 9.4, or the right form in Figure 9.4, which again involves a cycle of length 4 in $G_{p,3}$, a

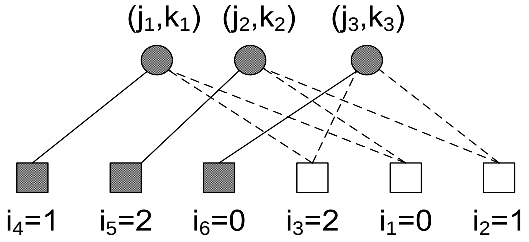

Figure 9.5: (Labelled) candidate (3,3) absorbing set

contradiction, [19].

The remaining case with $a = 3$ is $b = 3$. In this case, each bit node in D would then connect to exactly one check node in $\mathcal{O}(D)$ implying the unlabelled form of Figure 9.5. Note that there is a cycle of length 6. Suppose that these 3 bit nodes are indexed as (j_1, k_1), (j_2, k_2) and (j_3, k_3), respectively, where j_1, j_2 and j_3 are distinct (by the check consistency) and $0 \leq j_1, j_2, j_3 \leq p-1$. Without loss of generality assume that (j_1, k_1) and (j_2, k_2) share a check in the row group i_1, (j_2, k_2) and (j_3, k_3) share a check in the row group i_2, and that (j_1, k_1) and (j_3, k_3) share a check in the row group i_3, where $i_1, i_2, i_3 \in \{0, 1, 2\}$ and are distinct by the bit consistency condition. We may assume without loss of generality that $i_1 = 0$, $i_2 = 1$ and $i_3 = 2$. Note that the bit consistency constraints force the values of i_4, i_5 and i_6 to be as given in Figure 9.5.

In the remainder of the discussion we will first prove the existence of a $(3, 3)$ absorbing set. We will then show that these $(3, 3)$ absorbing sets are not fully absorbing sets. This result will in turn imply the existence of $(4, 2)$ fully absorbing sets, which are thus minimal fully absorbing sets for $\gamma = 3$.

The bit consistency constraints are automatically satisfied by our labelling of the row

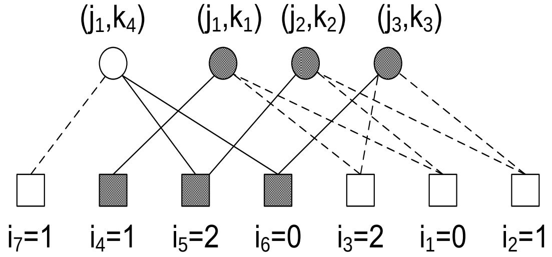

Figure 9.6: Candidate (3,3) absorbing set (solid circles), with an adjacent bit node (empty circle).

groups in Figure 9.5. The check consistency constraints reduce to the distinctness of j_1, j_2 and j_3. The pattern consistency constraints of Corollary 1 give:

$$k_1 + 2j_1 \equiv k_3 + 2j_3 \mod p, \tag{9.9}$$

$$k_1 \equiv k_2 \mod p, \tag{9.10}$$

$$k_2 + j_2 \equiv k_3 + j_3 \mod p. \tag{9.11}$$

The existence of a solution and hence of a (3,3) absorbing set is given in the proof of Lemma 24 below, which counts the number of such sets.

Even though a $(3,3)$ fully absorbing set seems plausible, care must be taken with respect to a bit node *outside* a candidate fully absorbing set when this bit node also participates in the unsatisfied checks. As we now show, a $(3,3)$ fully absorbing set cannot exist, though the existence of a $(3,3)$ absorbing set implies a $(4,2)$ fully absorbing set.

Suppose first that a $(3,3)$ fully absorbing set exists. Since $\gamma = 3$, it is then necessary that no bit node outside of the absorbing set participates in more than one unsatisfied check adjacent to a $(3,3)$ absorbing set. Since (j_1, k_1) and (j_3, k_3) share a check, $j_1 \neq j_3$. Consider the bit node labelled (j_1, k_4) that connects to i_6, as in Figure 9.6. Since $i_6 = 0$, it follows from Corollary 1 that $k_3 = k_4$. Equations (9.9)-(9.11) imply that $k_3+2j_1 \equiv k_2+2j_2 \mod p$ so that (j_1, k_3) bit node also connects to the check labelled i_5, as shown in Figure 9.6. This eliminates the possibility of a (3,3) fully absorbing set.

A (4,0) absorbing set cannot exist since the minimum distance of the code is 6 [61]. The next candidate size for the smallest fully absorbing set is (4,2). Each of the unsatisfied checks in any such configuration would necessarily connect to only one of the bit nodes, else we would have a cycle of length 4, a contradiction [19]. Given this, no satisfied check node can connect to all four bit nodes, else we would have a cycle of length 4, a contradiction [19]. Since there are 10 edges from the bit nodes that go to satisfied checks we now see that there must be 5 satisfied checks in any candidate (4,2) fully absorbing set. The two bit nodes that each have all their three edges going to satisfied check nodes must then share exactly one satisfied check (they have to share at least one, and cannot share more than one [19]). We have therefore concluded that any candidate (4,2) fully absorbing set must look like (an unlabelled version) of Figure 9.6. The existence of such (4,2) fully absorbing sets is proved in Lemma 24, which also counts the number of such sets. ■

Lemma 24 *The total number of* $(3,3)$ *absorbing sets and* $(4,2)$ *fully absorbing sets in the Tanner graph described by* $H_{p,3}$ *is* $p^2(p-1)$*, and* $3p^2(p-1)/2$*, respectively.*

Proof: Referring to Figure 9.5, the bit consistency and the check consistency constraints are satisfied for the given labels of row groups and since j_1, j_2 and j_3 are distinct. Then j_1 and k_1 can be chosen in p ways, and then j_3 can be chosen in $p-1$ ways. This fixes k_3 by equation (9.9), k_2 by equation (9.10) and then j_2 by equation (9.11). There is then a unique way to choose the row indices in the individual row groups so that the pattern consistency conditions of Lemma 21 are satisfied. Thus the total number of (3,3) absorbing sets is $p^2(p-1)$.

Turning to counting (4,2) fully absorbing set, every such set must look like an unlabelled version of Figure 9.6, and so contains exactly two distinct (3,3) absorbing sets (corresponding respectively to removing one of the bit nodes that connects to an unsatisfied check). From Figure 9.5 one can see that every (3,3) absorbing set is contained in three distinct (4,2) fully absorbing sets (for each pair of unsatisfied checks in Figure 9.5 one can find a bit node that these checks connect to, which when appended to the (3,3) absorbing set gives a (4,2) fully absorbing set). The total number of (4,2) fully absorbing sets is therefore $3p^2(p-1)/2$. ■

9.2.3 Absorbing sets of $H_{p,4}$

In order to establish that $(6,4)$ (fully) absorbing sets are minimal for $H_{p,4}$, we will first show that (a,b) absorbing sets for $a<6$ do not exist.

Let D denote an (a,b) absorbing set in $G_{p,4}=(V,F,E)$, the Tanner graph of $H_{p,4}$. If $a=2$ (respectively 3) then at least 6 (respectively 9) edges from D in $G_{p,4}$ terminate in $\mathcal{E}(D)$, which implies the existence of a cycle of length 4 in $G_{p,4}$, which is false [19]. Thus, $a \geq 4$.

Suppose $a=4$. Note that b must be even. We cannot have $b=0$, since this would imply the existence of a codeword of weight 4, which is false, [61]. If $b=2$ one can conclude

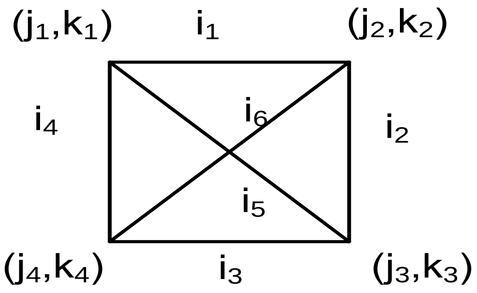

Figure 9.7: Depiction of the candidate (4,4) set

that there must be a cycle of length 4 in the code (whether the number of edges going into unsatisfied checks is 2 or 4), and this is false, [19]. Thus we must have $b = 4$ and, since each bit node must have at least three edges going to satisfied checks, the impossibility of a cycle of length 4 [19] implies that the absorbing set can be described as in Figure 9.7. In this figure each vertex represents a distinct bit node of the candidate (4,4) absorbing set and each edge represents a satisfied check node that connects to the bit nodes in the absorbing set, that correspond to its end points in the figure. The following lemma establishes that such sets do not exist if the prime p is large enough.

Lemma 25 *For $p > 7$, the Tanner graph family $G_{p,4}$ does not contain any $(4,4)$ absorbing sets.*

Proof: Without loss of generality we may let $i_1 = x$, $i_5 = y$ and $i_4 = z$, where $x, y, z \in \{0, 1, 2, 3\}$ and distinct by the bit consistency conditions. Then, by propagating the bit consistency conditions at each remaining vertex, and exploiting the symmetry, it suffices to consider $(i_1, i_2, i_3, i_4, i_5, i_6)$ either (x, y, x, y, z, z) or (x, y, x, y, z, w) where $x, y, z, w \in \{0, 1, 2, 3\}$ and are distinct.

For the case $(i_1, i_2, i_3, i_4, i_5, i_6) = (x, y, x, y, z, z)$, we establish the following cycle consistency conditions based on the cycles within the graph in Figure 9.7:

$$\begin{aligned} x(j_2 - j_1) + y(j_3 - j_2) + z(j_1 - j_3) &\equiv 0 \mod p, \\ x(j_2 - j_1) + z(j_4 - j_2) + y(j_1 - j_4) &\equiv 0 \mod p, \quad \text{and} \\ x(j_4 - j_3) + y(j_1 - j_4) + z(j_3 - j_1) &\equiv 0 \mod p. \end{aligned} \tag{9.12}$$

By adding and subtracting the conditions in (9.12), it follows that

$$\begin{aligned} (y - z)(j_3 + j_4 - j_1 - j_2) &\equiv 0 \mod p, \\ (x - z)(j_2 + j_3 - j_1 - j_4) &\equiv 0 \mod p, \quad \text{and} \\ (x - y)(j_2 + j_4 - j_1 - j_3) &\equiv 0 \mod p. \end{aligned} \tag{9.13}$$

Since x, y, z are distinct, (9.13) implies that j's would have to be all the same, which contradicts the check consistency constraint.

For the case $(i_1, i_2, i_3, i_4, i_5, i_6) = (x, y, x, y, z, w)$, again based on the cycle structure in Figure 9.7, we get the cycle consistency conditions

$$\begin{aligned} x(j_2 - j_1) + y(j_3 - j_2) + z(j_1 - j_3) &\equiv 0 \mod p, \\ x(j_2 - j_1) + w(j_4 - j_2) + y(j_1 - j_4) &\equiv 0 \mod p, \quad \text{and} \\ x(j_4 - j_3) + y(j_1 - j_4) + z(j_3 - j_1) &\equiv 0 \mod p. \end{aligned} \tag{9.14}$$

We let $u_1 := j_2 - j_1$, $u_2 := j_3 - j_1$, and $u_3 := j_4 - j_1$. By the check consistency condition, all of u_1, u_2, and u_3 are non-zero. Substituting u_1, u_2 and u_3 in (9.14) and then expressing u_2 and u_3 in terms of u_1, one arrives at the condition

$$(z - x)(w - y) + (z - y)(w - x) \equiv 0 \mod p. \tag{9.15}$$

It can be easily verified that this condition can not hold for any choice of x, y, z, w, where $x, y, z, w \in \{0, 1, 2, 3\}$ and are distinct for $p > 7$. There are $4! = 24$ ways of assigning numerical values to (x, y, z, w). The expression in (9.15) is at most 7 in absolute value for such x, y, z, and w. Substituting each numerical assignment (x, y, z, w) yields possible choices of prime p for which the expression in (9.15) becomes zero $\mod p$. The condition (9.15) holds for $p \in \{2, 5, 7\}$. Therefore, for $p > 7$, $G_{p,\gamma}$ does not contain $(4, 4)$ absorbing sets. ■

We next show that $(5, b)$ absorbing sets do not exist for the parameter p large enough. In particular we will establish a congruential constraint involving the labels of the edges emanating from the bits in the absorbing set that cannot hold for p large enough.

Lemma 26 *For $p > 19$, the Tanner graph family $G_{p,4}$ does not contain any $(5, b)$ absorbing sets.*

Proof: Since each bit node in the absorbing set has at most one neighboring unsatisfied check node, it follows that $b \leq 5$. Observe that the number of bit nodes with 3 satisfied and 1 unsatisfied check nodes is even, and thus b is even. First $b > 0$ by the minimum distance, $d_{min} \geq 8$ of the code, [61]. If $b = 2$, since we have at most five edges going to unsatisfied checks there are two cases: (a) either three of them go to one unsatisfied check and one to another, or (b) one edge goes to each unsatisfied check. In case (a), because the girth of the Tanner graph is bigger than 4, [19], none of the three bit nodes that share an unsatisfied check can share a satisfied check. Further, no two bit nodes can share a satisfied check for the same reason. By counting, this eliminates case (a). In case (b), if we drop one of the

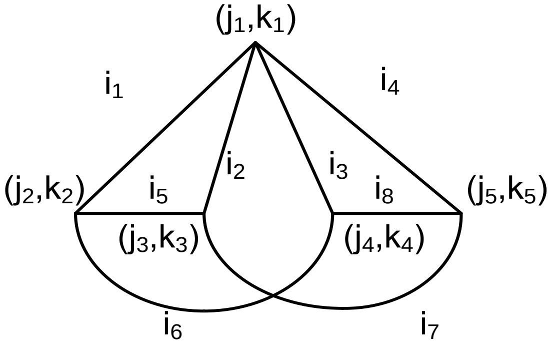

Figure 9.8: Depiction of the candidate (5,4) set

bit nodes that has an unsatisfied check we would have a (4,4) absorbing set which we have argued in Lemma 25 does not exist for $p > 7$.

Thus for $p > 7$ we are left with considering the case $b = 4$ since at most five edges go into unsatisfied checks. This means the candidate absorbing set contains 1 bit node with all checks satisfied and 4 bit nodes each with 3 satisfied and 1 unsatisfied check. The only way that such an absorbing set could exist is if one has the configuration shown in Figure 9.8, where the vertices represent bit nodes and edges represent their satisfied check nodes.

Since i_1, i_2, i_3 and i_4 are all distinct elements of the set $\{0, 1, 2, 3\}$, by the bit consistency condition, and by the symmetry of the candidate configuration in Figure 9.8, we may assume that $i_1 = 0$. We let $x := i_2$, $y := i_3$ and $z := i_4$, where $x, y, z \in \{1, 2, 3\}$ and distinct. By propagating possible values of the labels for remaining edges, while maintaining bit consistency conditions, it follows that $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8)$ is either $(0, x, y, z, y, z, 0, x)$ or $(0, x, y, z, z, x, y, 0)$.

For $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8)$ = $(0, x, y, z, y, z, 0, x)$, and for each edge and its endpoints in Figure 9.8, we write the pattern consistency constraints of Corollary 1, in terms of x, y and z,

$$k_1 \equiv k_2 \mod p, \tag{9.16a}$$

$$k_3 \equiv k_5 \mod p, \tag{9.16b}$$

$$k_1 + xj_1 \equiv k_3 + xj_3 \mod p, \tag{9.16c}$$

$$k_1 + yj_1 \equiv k_4 + yj_4 \mod p, \tag{9.16d}$$

$$k_1 + zj_1 \equiv k_5 + zj_5 \mod p, \tag{9.16e}$$

$$k_2 + yj_2 \equiv k_3 + yj_3 \mod p, \tag{9.16f}$$

$$k_2 + zj_2 \equiv k_4 + zj_4 \mod p, \text{ and} \tag{9.16g}$$

$$k_4 + xj_4 \equiv k_5 + xj_5 \mod p. \tag{9.16h}$$

This last system simplifies to

$$\begin{aligned}
k_1 + xj_1 &\equiv k_3 + xj_3 \mod p, && \text{(from (9.16c))}\\
k_1 + yj_1 &\equiv k_4 + yj_4 \mod p, && \text{(from (9.16d))}\\
k_1 + zj_1 &\equiv k_3 + zj_5 \mod p, && \text{(from (9.16b) and (9.16e))}\\
k_1 + yj_2 &\equiv k_3 + yj_3 \mod p, && \text{(from (9.16a) and (9.16f))}\\
k_1 + zj_2 &\equiv k_4 + zj_4 \mod p, \quad \text{and} && \text{(from (9.16a) and (9.16g))}\\
k_4 + xj_4 &\equiv k_3 + xj_5 \mod p. && \text{(from (9.16b) and (9.16h))}
\end{aligned} \tag{9.17}$$

Thus

$$\begin{aligned} k_1 - k_3 &\equiv x(j_3 - j_1) \equiv z(j_5 - j_1) \equiv y(j_3 - j_2) && \mod p && \text{(9.18a)} \\ k_1 - k_4 &\equiv y(j_4 - j_1) \equiv z(j_4 - j_2) && \mod p && \text{(9.18b)} \\ k_3 - k_4 &\equiv x(j_4 - j_5) && \mod p. && \text{(9.18c)} \end{aligned}$$

We let $u_1 := j_3 - j_1$, $u_2 := j_4 - j_1$, $u_3 := j_4 - j_1$, and $u_4 := j_3 - j_2$. Note that by the check consistency condition, all of u_1, u_2, u_3, and u_4 are non-zero.

We then obtain

$$\begin{aligned} xu_1 &\equiv zu_3 && \mod p && \text{(from (9.18a)),} \\ xu_1 &\equiv yu_4 && \mod p && \text{(from (9.18a)),} \\ yu_2 &\equiv z(u_2 - u_1 + u_4) && \mod p && \text{(from (9.18b)),} \\ x(u_2 - u_3) &\equiv yu_2 - xu_1 && \mod p && \text{from } k_3 - k_4 = (k_1 - k_4) - (k_1 - k_3) \text{ and} \\ &&&&& \text{substituting from (9.18c),(9.18b),(9.18a)),resp.} \end{aligned} \tag{9.19}$$

This last system can be rewritten as

$$\begin{bmatrix} x & 0 & 0 & -y \\ x & 0 & -z & 0 \\ -z & z-y & 0 & z \\ -x & y-x & x & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mod p\ . \tag{9.20}$$

Therefore, the determinant of the matrix multiplying the (non-zero) vector $[u_1 u_2 u_3 u_4]^T$ in (9.20) is itself zero, which simplifies to

$$xy(z-x)(z-y) - z^2(x-y)^2 \equiv 0 \mod p, \tag{9.21}$$

Since $x, y, z \in \{1, 2, 3\}$ and distinct we consider all $3! = 6$ assignment for (x, y, z), and for each we evaluate the left had side expression in (9.21). Note that for distinct $x, y, z \in \{1, 2, 3\}$, this expression is at most 19 in absolute value, and therefore the constraint in (9.21) does not have a solution for $p > 19$ for distinct $x, y, z \in \{1, 2, 3\}$. (Solutions exist for $p = 5, 11$ and 19, which can be verified by direct numerical substitution).

For $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8)$ = $(0, x, y, z, z, z, y, 0)$ we likewise establish the constraints as in (9.16) and (9.17). We again let $u_1 := j_3 - j_1$, $u_2 := j_4 - j_1$, $u_3 := j_4 - j_1$, and $u_4 := j_3 - j_2$, and obtain

$$\begin{bmatrix} 0 & y & -z & 0 \\ x & y-x & 0 & -x \\ x & 0 & 0 & -z \\ y-x & y & -y & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mod p \, . \tag{9.22}$$

Since the entries in $[u_1 u_2 u_3 u_4]^T$ are all non-zero, it follows that the determinant of the matrix in (9.22) is zero. Simplifying the expression for the determinant yields again the condition in (9.21). Therefore for $p > 19$, (5,4) absorbing sets do not exist. ■

We can now proceed with the analysis of $(6, b)$ absorbing sets. Since the number of bit nodes with 3 satisfied and 1 unsatisfied check node is even, b is even. First, $b = 0$ is not possible since $d_{min} \geq 8$ [61]. The following lemma considers $b = 2$.

Lemma 27 *For $p > 19$, the Tanner graph family $G_{p,4}$ does not contain any $(6, 2)$ absorbing sets.*

Proof: We first claim that there is no check node of degree at least 3 with respect to the bit

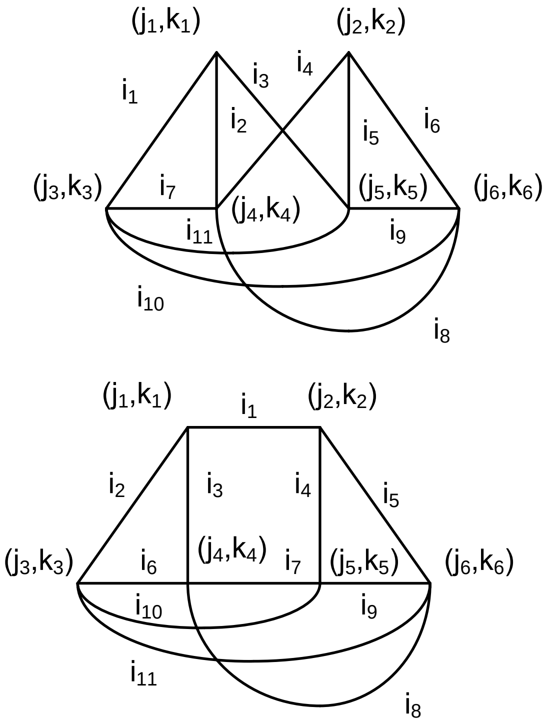

Figure 9.9: Depiction of the candidate (6,2) sets.

nodes in the absorbing set. Let us first suppose that there exists one such check node and that it has an even degree with respect to the bit nodes in the absorbing set. Since we are considering an absorbing set with 6 bit nodes, such a check node would have degree either 4 or 6 with respect to the bit nodes in the absorbing set. If this satisfied check is of degree 6, there would exist 2 bit nodes in the absorbing set which would share an additional satisfied check. This situation would imply the existence of a cycle of length 4, which is impossible by the girth condition [19].

Suppose now that this satisfied check has degree 4. Each bit node that participates in this check has at least 2 more neighboring satisfied checks, which it then necessarily must share with the remaining two bit nodes in the absorbing set that themselves do not participate in this degree-4 check. If there exists a bit node that participates in this degree-4 check and has all checks satisfied, it then shares its remaining neighboring check with one of the bit nodes with which it already shares a check. This situation violates the girth constraint [19]. If all bit nodes in the absorbing set that participate in this degree-4 check have 3 satisfied and 1 unsatisfied check, three of them would have to participate in the same unsatisfied check to make the total number of unsatisfied checks be 2. This again violates the girth condition [19].

Therefore, all satisfied checks with respect to the bit nodes in the absorbing set have degree 2. Suppose there exists a check node that is unsatisfied with respect to the bits in the absorbing set and that has degree bigger than 1. If such a check node has degree 5, there would necessarily exist 2 bit nodes in the absorbing set that share this degree-5 check and another satisfied check, which is impossible by the girth condition.

Suppose that there exists 2 degree-3 checks incident to the bit nodes in the absorbing set. First, these degree-3 checks do not have any neighboring bit nodes in common since we require that each bit node has at most 1 unsatisfied check. We can then group the bit nodes in the absorbing set into two disjoint groups, each of size 3, such that the bits in the same group share the same degree-3 check. Consider a bit node in, say, the first group. It shares its remaining 3 (satisfied) checks with one of each bit nodes in the second group. The same is true with the other two bit nodes in the first group, namely they too share their remaining 3 (satisfied) checks with the bit nodes in the second group. Therefore, there exist two bit nodes in the first group and two bit nodes in the second group such that any two share a distinct check. This configuration is not possible by Lemma 25 for $p > 7$.

Suppose now there exists one unsatisfied check of degree 3 with respect to the bit nodes in the absorbing set. The remaining unsatisfied check then has degree 1 with respect to the bit nodes in the absorbing set, and the neighboring bit nodes in the absorbing set of these two unsatisfied checks are different. There are two bit nodes in the absorbing set that have all checks satisfied. Partition the bit nodes in the absorbing set into three groups: the first group contains the three bit nodes that share a degree-3 unsatisfied check, the second group contains the one bit node that has one unsatisfied check, and the third group contains the two bit nodes that have all four checks satisfied. Each of the three bit nodes in the first group has one unsatisfied and three satisfied checks and thus it shares a satisfied check with each of the bit nodes in the second and third group since it cannot share a satisfied check with another bit node in the first group by the girth condition. The bit node in the second group also has one unsatisfied and three satisfied checks, and the latter are shared then with bit nodes in the first group. The two bit nodes in the third group have all four checks satisfied, the three of which they each share with each of the bit nodes in the first group. Since all three satisfied checks of the bit node in the second group are used up with the checks it shares with the bit nodes in the first group, the two bit nodes in the third group share a satisfied check with each other. Therefore, there exist two bit nodes in the first group and two bit nodes in the third group such that any two share a distinct check. This configuration is not possible by Lemma 25 for $p > 7$.

We conclude that no check incident to the bit nodes in the absorbing set has degree larger than 2, namely that all neighboring satisfied (resp. unsatisfied) checks have degree 2 (resp. 1). By requiring that each vertex corresponding to a bit node in the absorbing set has either 3 or 4 outgoing edges, and that there are no parallel edges, it follows that there are 2 possible configurations, as shown in Figure 9.9, that relate bit nodes in the absorbing set (vertices) and their shared satisfied checks (edges).

Observe that the bottom configuration in Figure 9.9 contains a $(4, 4)$ absorbing set which consists of (j_3, k_3), (j_4, k_4), (j_5, k_5), and (j_6, k_6). By Lemma 25 such configuration is not possible for $p > 7$. The rest of the proof focuses on the topmost configuration.

By ensuring the bit consistency, it follows that the topmost configuration in Figure 9.9 has 2 distinct edge labellings. In particular, by the bit consistency at (j_3, k_3) we may let $x := i_1$, $y := i_7$, $z := i_{11}$ and $w := i_{10}$, where $x, y, z, w \in \{0, 1, 2, 3\}$ and distinct. By propagating the labels while making sure that the bit consistency constraints are satisfied we obtain

- $x = i_1 = i_5 = i_8$, $y = i_7 = i_9$, $z = i_2 = i_6 = i_{11}$, $w = i_3 = i_4 = i_{10}$ (call this set of constraints $\star$) or
- $x = i_1 = i_4 = i_9$, $y = i_3 = i_6 = i_7$, $z = i_8 = i_{11}$, $w = i_2 = i_5 = i_{10}$ (call this set of constraints $\star\star$) where throughout x, y, z, w are distinct and belong to the set $\{0, 1, 2, 3\}$.

Case ($\star$)

Using the pattern consistency constraint (see Corollary 1) for each edge in Figure 9.9

for the current labelling we obtain

$$k_1 + xj_1 \equiv k_3 + xj_3 \mod p, \tag{9.23a}$$

$$k_1 + zj_1 \equiv k_4 + zj_4 \mod p, \tag{9.23b}$$

$$k_1 + wj_1 \equiv k_5 + wj_5 \mod p, \tag{9.23c}$$

$$k_2 + wj_2 \equiv k_4 + wj_4 \mod p, \tag{9.23d}$$

$$k_2 + xj_2 \equiv k_5 + xj_5 \mod p, \tag{9.23e}$$

$$k_2 + zj_2 \equiv k_6 + zj_6 \mod p, \tag{9.23f}$$

$$k_3 + yj_3 \equiv k_4 + yj_4 \mod p, \tag{9.23g}$$

$$k_4 + xj_4 \equiv k_6 + xj_6 \mod p, \tag{9.23h}$$

$$k_5 + yj_5 \equiv k_6 + yj_6 \mod p, \tag{9.23i}$$

$$k_3 + wj_3 \equiv k_6 + wj_6 \mod p, \text{ and} \tag{9.23j}$$

$$k_3 + zj_3 \equiv k_5 + zj_5 \mod p, \tag{9.23k}$$

We now separately consider $x = 0$, $y = 0$, $z = 0$, and $w = 0$.

1. For $x = 0$, the set of constraints (9.23a)-(9.23k) reduces to

$$\begin{aligned}
&k_1 - k_3 \equiv 0 \mod p && \text{(from (9.23a))}\\
&k_2 - k_5 \equiv 0 \mod p && \text{(from (9.23e))}\\
&k_4 - k_6 \equiv 0 \mod p && \text{(from (9.23h))}\\
&k_1 - k_4 \equiv z(j_4 - j_1) \equiv y(j_4 - j_3) \equiv w(j_6 - j_3) \mod p\\
&\qquad \text{(from (9.23b)), (9.23a) and (9.23g), and (9.23a) and (9.23j) respectively.)}\\
&k_2 - k_4 \equiv w(j_4 - j_2) \equiv z(j_6 - j_2) \equiv y(j_6 - j_5) \mod p\\
&\qquad \text{(from (9.23d)) (9.23h) and (9.23f), and (9.23e), (9.23h) and (9.23i) respectively.)}\\
&k_1 - k_2 \equiv w(j_5 - j_1) \equiv z(j_5 - j_3) \mod p\,.\\
&\qquad (\text{from (9.23c) (9.23d) and (9.23k) respectively.})
\end{aligned} \tag{9.24}$$

Since $j_1 \neq j_4$, $j_2 \neq j_4$ and $j_1 \neq j_5$ by the check consistency conditions, we have that $k_1 \neq k_4$, $k_2 \neq k_4$ and $k_1 \neq k_2$.

Since $\{y, z, w\} = \{1, 2, 3\}$ and $p > 19$ is prime, we may let

$$\begin{aligned}
k_1 - k_4 &\equiv ywzt \mod p,\\
k_2 - k_4 &\equiv ywzu \mod p, \quad \text{and}\\
k_1 - k_2 &\equiv wzs \mod p
\end{aligned} \tag{9.25}$$

for some integers t, s and u which are themselves nonzero. From $k_1 - k_2 = (k_1 - k_4) - (k_2 - k_4)$, it follows that

$$wzs \equiv yzwt - ywzu \mod p\,. \tag{9.26}$$

Write $j_5 - j_3 = -(j_6 - j_5) + (j_6 - j_3)$ to obtain

$$ws \equiv -wzu + yzt \mod p\ . \tag{9.27}$$

Likewise, from $j_5 - j_1 = -(j_6 - j_5) + (j_6 - j_2) - (j_4 - j_2) + (j_4 - j_1)$, it follows that

$$zs \equiv -wzu + ywu - yzu + ywt \mod p\ . \tag{9.28}$$

From (9.26)- (9.28), by equating the expressions for ws and zs, it follows that

$$\begin{aligned} wu(y-z) &\equiv yt(w-z) \qquad \mod p \\ wu(y-z) &\equiv yt(z-w) \qquad \mod p\ . \end{aligned} \tag{9.29}$$

The last set of constraints implies $w \equiv z \mod p$ which is a contradiction. 2. For $y = 0$ the set of constraints (9.23a)-(9.23k) reduces to

$$\begin{aligned} k_3 - k_4 &\equiv 0 && \mod p \\ k_5 - k_6 &\equiv 0 && \mod p \\ k_1 - k_3 &\equiv x(j_3 - j_1) \equiv z(j_4 - j_1) && \mod p \\ k_2 - k_5 &\equiv x(j_5 - j_2) \equiv z(j_6 - j_2) && \mod p \\ k_3 - k_5 &\equiv x(j_6 - j_4) \equiv w(j_6 - j_3) \equiv z(j_5 - j_3) && \mod p \\ k_1 - k_5 &\equiv w(j_5 - j_1) && \mod p\ . \end{aligned} \tag{9.30}$$

Note that $j_1 \neq j_3$, $j_2 \neq j_5$, $j_4 \neq j_6$ and $j_1 \neq j_5$ by the check consistency conditions. Since $\{x, z, w\} = \{1, 2, 3\}$, we may let

$$\begin{aligned} k_1 - k_3 &\equiv xzs \qquad \mod p, \\ k_1 - k_5 &\equiv wv \qquad \mod p, \\ k_2 - k_5 &\equiv xzu \qquad \mod p, \quad \text{and} \\ k_3 - k_5 &\equiv xwzt \qquad \mod p \end{aligned} \tag{9.31}$$

for some integers s, u, v and t, which are themselves nonzero. The identities $k_1 - k_3 = (k_1-k_5)-(k_3-k_5)$, $j_5-j_1 = (j_5-j_3)+(j_3-j_1)$ and $j_4-j_1 = -(j_6-j_4)+(j_6-j_3)+(j_3-j_1)$ respectively, yield the following constraints,

$$\begin{aligned} xzs &\equiv wv - xwzt && \mod p \\ v &\equiv xwt + zs && \mod p \\ xs &\equiv -wzt + xzt + zs && \mod p \ . \end{aligned} \tag{9.32}$$

Eliminating v from the top two constraints implies $zs(x-w) \equiv xwt(w-z) \mod p$, which combined with the bottom constraint yields

$$z^2(x-w)^2 \equiv xw(w-z)(x-z) \mod p \ . \tag{9.33}$$

Since $\{x, y, w\} = \{1, 2, 3\}$, this cannot hold for $p > 19$.

3. For $z = 0$ we obtain

$$\begin{aligned} k_1 - k_4 &\equiv 0 && && && \mod p \\ k_2 - k_6 &\equiv 0 && && && \mod p \\ k_3 - k_5 &\equiv 0 && && && \mod p \\ k_1 - k_3 &\equiv x(j_3 - j_1) &&\equiv w(j_5 - j_1) &&\equiv y(j_3 - j_4) && \mod p \\ k_2 - k_3 &\equiv x(j_5 - j_2) &&\equiv y(j_5 - j_6) &&\equiv w(j_3 - j_6) && \mod p \\ k_1 - k_2 &\equiv w(j_2 - j_4) &&\equiv x(j_6 - j_4) && && \mod p \ . \end{aligned} \tag{9.34}$$

As before, some algebra yields $x \equiv w \mod p$, a contradiction.

4. For $w = 0$ we obtain

$$\begin{aligned}
k_1 - k_5 &\equiv 0 && \mod p \\
k_2 - k_4 &\equiv 0 && \mod p \\
k_3 - k_6 &\equiv 0 && \mod p \\
k_1 - k_3 &\equiv x(j_3 - j_1) \equiv y(j_6 - j_5) \equiv z(j_3 - j_5) && \mod p \\
k_2 - k_3 &\equiv z(j_6 - j_2) \equiv y(j_3 - j_4) \equiv x(j_6 - j_4) && \mod p \\
k_1 - k_2 &\equiv z(j_4 - j_1) \equiv x(j_2 - j_5) && \mod p\,.
\end{aligned} \tag{9.35}$$

After some algebra, we obtain the following condition

$$xz(z-y)(x-y) \equiv -y^2(x-z)^2 \mod p, \tag{9.36}$$

which, because $\{x, y, z\} = \{1, 2, 3\}$ has no solution for $p > 19$.

Case ($\star\star$)

We separately consider $x = 0$, $y = 0$, $z = 0$, and $w = 0$, and proceed along the lines of the previous case.

For $x = 0$, resp. $y = 0$, it follows after some algebra that $y \equiv w \mod p$, resp. $x \equiv w \mod p$, a contradiction in each case.

For $z = 0$, resp. $w = 0$, it follows similarly that $xw(w-y)(x-y) \equiv y^2(x-w)^2 \mod p$, resp. $xy(y-z)(x-z) \equiv -z^2(x-y)^2 \mod p$, neiter of which can hold for $p > 19$.

This completes the proof of the Lemma. ■

Having eliminated smaller candidate absorbing sets, we now prove the following result.

Lemma 28 *For all $p > 5$, the Tanner graph family $G_{p,4}$ has $(6, 4)$ (fully) absorbing sets.*

Proof: We will first show that all satisfied checks neighboring bit nodes in one such absorbing set must have degree 2. Note that there cannot be a degree-6 check with respect to the bits in the absorbing set as then some of these bits would have to share another satisfied check which is not possible by the girth condition. Suppose that there exists a check node of degree 4 with respect to a $(6, 4)$ absorbing set. Let t_1, t_2, t_3, t_4 be the bit nodes in the absorbing set participating in degree-4 check node, and let t_5 and t_6 be the remaining two bit nodes in the absorbing set. By the girth condition there can be at most one degree-4 check incident to the bit nodes in the absorbing set. If at least one of t_1, t_2, t_3, t_4 had all check nodes satisfied, it would be necessary that such a bit node shares another distinct check node with some other bit node participating in the degree-4 check node, which is impossible by the girth constraint [19]. Thus, all of t_1, t_2, t_3, t_4 are each connected to 3 satisfied and 1 unsatisfied check node. Then t_5 and t_6 are each connected to 4 satisfied check nodes each of degree 2 with respect to the bit nodes in the absorbing set. Since t_1 through t_4 have 3 satisfied neighboring checks (one of which is a degree-4 check by assumption), they each share a check with t_5 and with t_6. Therefore, t_5 and t_6 do not share a check. Let i_j for $1 \leq j \leq 4$ be the labels of the four check nodes connecting t_j and t_5. By the bit consistency condition at t_5, they are all different. By the bit consistency condition at each of t_j for $1 \leq j \leq 4$, the label of their shared degree-4 check node must be different from all i_j for $1 \leq j \leq 4$, which is impossible as there are only 4 distinct labels available. Therefore, all satisfied check nodes neighboring bit nodes in the absorbing set have degree 2.

We first analyze the case where there exists an unsatisfied check of degree 3 with respect to the bit nodes in the absorbing set (an unsatisfied check of degree larger than 3 is not possible by the girth condition). Consider a candidate (6,4) absorbing set in which three bit nodes, call them t_1, t_2, t_3 connect to the same unsatisfied check, and the remaining three bit nodes, call them t_4, t_5, t_6, each have a distinct unsatisfied check. Since there are no cycles of length 4, each of the t_1, t_2, t_3 shares a distinct satisfied check with each of t_4, t_5, t_6. We will show that in fact for large enough prime p such a configuration is not possible.

Let the check incident to t_1, t_2, and t_3 have label x, where $x \in \{0, 1, 2, 3\}$. Using the bit consistency condition, we let y be the label of the satisfied check incident to t_1 and t_4, z be the label of the satisfied check incident to t_1 and t_5, and w be the label of the satisfied check incident to t_1 and t_6, where $y, z, w \in \{0, 1, 2, 3\}$ are distinct and are different from x.

By propagating remaining edge labels while ensuring that the bit consistency is satisfied, we obtain that the labels of the checks connecting t_2 with t_4, t_5 and t_6, respectively, are z, w and y and the labels of the checks connecting t_3 with t_4, t_5 and t_6, respectively, are w, y and z.

Let (j_l, k_l) for $1 \leq l \leq 6$ be the labels of the bit nodes t_l.

Using the pattern consistency (see Corollary 1) we write one equation for each pair of the bit nodes in the absorbing set that share a satisfied check as follows:

$$\begin{aligned}
k_1 + yj_1 &\equiv k_4 + yj_4 \mod p, \\
k_1 + zj_1 &\equiv k_5 + zj_5 \mod p, \\
k_1 + wj_1 &\equiv k_6 + wj_6 \mod p, \\
k_2 + zj_2 &\equiv k_4 + zj_4 \mod p, \\
k_2 + wj_2 &\equiv k_5 + wj_5 \mod p, \\
k_2 + yj_2 &\equiv k_6 + yj_6 \mod p, \\
k_3 + wj_3 &\equiv k_4 + wj_4 \mod p, \\
k_3 + yj_3 &\equiv k_5 + yj_5 \mod p, \\
k_3 + zj_3 &\equiv k_6 + zj_6 \mod p\ .
\end{aligned} \tag{9.37}$$

In addition we may also write

$$k_1 + xj_1 \equiv k_2 + xj_2 \equiv k_3 + xj_3 \mod p, \tag{9.38}$$

since the bit nodes (j_1, k_1), (j_2, k_2) and (j_3, k_3), all participate in the same (unsatisfied) check with label x.

Since $x, y, z, w \in \{0, 1, 2, 3\}$ and are distinct we now consider different numerical assignments of these labels. In particular, it is sufficient to consider $x = 0$ and $y = 0$, since by the symmetry of the configuration both $z = 0$ and $w = 0$ reduce to the $y = 0$ case.

1. Case $x = 0$

Equation (9.38) reduces to

$$k_1 = k_2 = k_3, \tag{9.39}$$

which combined with (9.37) gives

$$\begin{aligned} k_1-k_4 &\equiv y(j_4-j_1)\equiv z(j_4-j_2)\equiv w(j_4-j_3) \mod p \\ k_1-k_5 &\equiv z(j_5-j_1)\equiv w(j_5-j_2)\equiv y(j_5-j_3) \mod p \\ k_1-k_6 &\equiv w(j_6-j_1)\equiv y(j_6-j_2)\equiv z(j_6-j_3) \mod p \end{aligned} \tag{9.40}$$

Since y, z, w do not have any non trivial factors, we may let $yzwt \equiv k_1 - k_4 \mod p$, $yzwv \equiv k_1 - k_5 \mod p$ and $yzws \equiv k_1 - k_6 \mod p$ for some integers t, v and s.

Using the identity

$$j_5-j_4 = (j_5-j_1)-(j_4-j_1) = (j_5-j_2)-(j_4-j_2) = (j_5-j_3)-(j_4-j_3) \tag{9.41}$$

we obtain (using $(j_5 - j_1) \equiv ywv \mod p, (j_4 - j_1) \equiv zwt \mod p$, and so on),

$$ywv - zwt \equiv yzv - ywt \equiv zwv - yzt \mod p\,. \tag{9.42}$$

The last expression implies

$$y^2(w-z)^2 \equiv wz(z-y)(y-w) \mod p\,. \tag{9.43}$$

Likewise, using the identity

$$j_6-j_5 = (j_6-j_1)-(j_5-j_1) = (j_6-j_2)-(j_5-j_2) = (j_6-j_3)-(j_5-j_3) \tag{9.44}$$

we obtain

$$yzs - ywv \equiv zws - yzv \equiv yws - zwv \mod p\,, \tag{9.45}$$

and from it

$$z^2(y-w)^2 \equiv yw(z-y)(w-z) \mod p\,. \tag{9.46}$$

Using the identity

$$j_6 - j_4 = (j_6 - j_1) - (j_4 - j_1) = (j_6 - j_2) - (j_4 - j_2) = (j_6 - j_3) - (j_4 - j_3) \quad (9.47)$$

we obtain

$$yzs - zwt \equiv zws - ywt \equiv yws - yzt \mod p\ , \quad (9.48)$$

and from it

$$w^2(z-y)^2 \equiv zy(w-z)(y-w) \mod p\ . \quad (9.49)$$

Since $\{y, z, w\} = \{1, 2, 3\}$, the equations (9.43), (9.46) and (9.49) hold only for prime $p = 13$.

2. Case $y = 0$

In this case equation (9.38) implies

$$\begin{aligned} k_1 &= k_4 \\ k_3 &= k_5 \\ k_2 &= k_6 \quad . \end{aligned} \quad (9.50)$$

The relations (9.50) combined with (9.37) further yield

$$\begin{aligned} k_1 - k_3 \equiv z(j_5 - j_1) \equiv w(j_3 - j_4) \equiv x(j_3 - j_1) \mod p \\ k_1 - k_2 \equiv w(j_6 - j_1) \equiv z(j_2 - j_4) \equiv x(j_2 - j_1) \mod p \\ k_2 - k_3 \equiv w(j_5 - j_2) \equiv z(j_3 - j_6) \equiv x(j_3 - j_2) \mod p\ . \end{aligned} \quad (9.51)$$

We let $xzwt \equiv k_1 - k_3 \mod p$, $xzwv \equiv k_1 - k_2 \mod p$, and $xzws \equiv k_2 - k_3 \mod p$, for some integers t, v and s. From $k_1 - k_3 = k_1 - k_2 + k_2 - k_3$, we have

$$t \equiv v + s \mod p. \quad (9.52)$$

From $j_6 - j_1 = -(j_3 - j_6) + (j_3 - j_1)$ we get

$$zxv \equiv -wxs + zwt \mod p. \tag{9.53}$$

Likewise, from $j_5 - j_1 = (j_5 - j_2) + (j_2 - j_1)$ we get

$$xwt \equiv xzs + zwv \mod p, \tag{9.54}$$

and from $j_3 - j_4 = (j_3 - j_2) + (j_2 - j_4)$ we get

$$xzt \equiv zws + xwv \mod p\ . \tag{9.55}$$

From (9.52) and (9.53) by equating the expressions for zwt we obtain

$$zv(x - w) \equiv ws(z - x) \mod p. \tag{9.56}$$

Likewise, from (9.52) and (9.54) by equating the expressions for xwt we obtain

$$wv(z - x) \equiv xs(w - z) \mod p, \tag{9.57}$$

and from (9.52) and (9.55) by equating the expressions for xzt we obtain

$$xv(z - w) \equiv zs(w - x) \mod p, \tag{9.58}$$

From (9.56), (9.85) and (9.58), it follows that

$$\begin{aligned} w^2(z-x)^2 &\equiv xz(w-z)(x-w) \mod p, \\ -z^2(x-w)^2 &\equiv xw(z-x)(z-w) \mod p, \\ -x^2(w-z)^2 &\equiv wz(w-x)(z-x) \mod p\ , \end{aligned} \tag{9.59}$$

which are the same as conditions (9.43), (9.46) and (9.49) previously derived for the $x = 0$ case. Therefore, for prime $p, p > 13$ the candidate configuration does not exist.

We now continue with the analysis of the candidate configurations in which each satisfied check has degree 2 with respect to the bit nodes in the absorbing set, and each unsatisfied check has degree 1 with respect to the bits in the absorbing set.

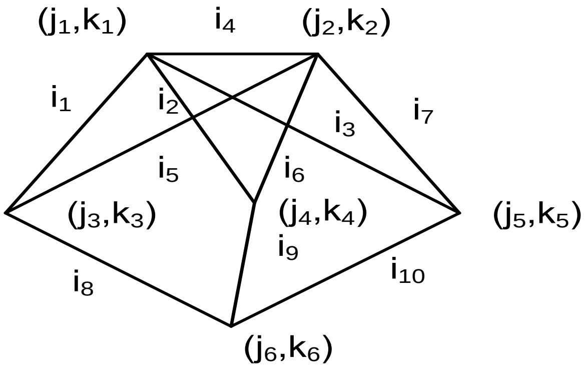

Figure 9.10: Depiction of the first candidate (6,4) set

By separately considering the cases when the two bit nodes that have all neighboring checks satisfied have a satisfied check in common, and the cases when they do not, one can show that there are 3 possible non isomorphic configurations, as shown in Figure 9.10, 9.11,

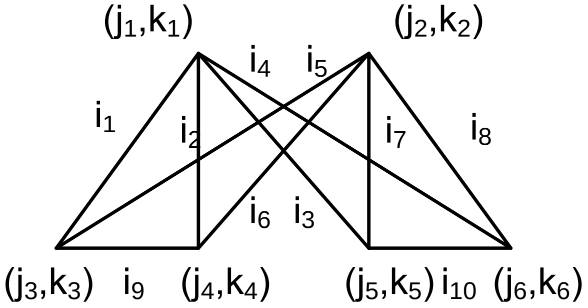

Figure 9.11: Depiction of the second candidate (6,4) set

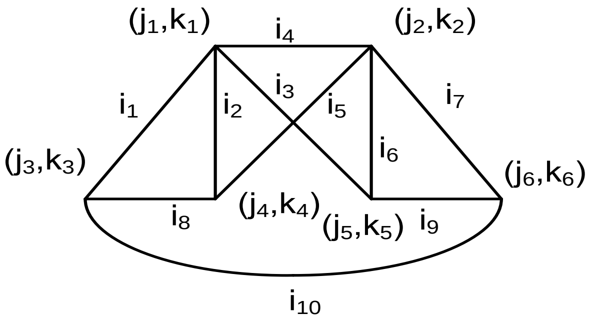

Figure 9.12: Depiction of the third candidate (6,4) set

and 9.12. By ensuring the bit consistency, it further follows that for each configuration there are 8 distinct edge labellings (as we show below).

Let us consider the topmost configuration first. The other two configurations are analyzed subsequently.

I. Candidate (6,4) configuration, given in Figure 9.10.

We first determine all possible edge labellings. For convenience, we assign (i_1, i_2, i_3, i_4) $:= (x, y, z, w)$, where $x, y, z, w \in \{0, 1, 2, 3\}$ and distinct by the bit consistency condition at (j_1, k_1). Then, by imposing the bit consistency conditions at remaining vertices, the possible assignments for the remaining edge labels are as follows

$$\begin{aligned}(i_5, i_6, i_7, i_8, i_9, i_{10}) \in \{&(y, z, x, z, x, y), (z, x, y, y, z, x),\\ &(y, z, x, z, w, y), (y, z, x, w, x, y), (y, z, x, z, x, w),\\ &(z, x, y, y, z, w)(z, x, y, y, w, x), (z, x, y, w, z, x)\}.\end{aligned} \tag{9.60}$$

We first observe that the assignments $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}) = (x, y, z, w, y, z, x, z, x, y)$ and $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$ =$(x, y, z, w, z, x, y, y, z, x)$ are in fact symmetric (exchange y and z) and is thus sufficient to analyze only one of them. Likewise, by appealing to symmetry and after appropriate renamings, the remaining six assignments also represent the same labelled configuration. In particular, third and sixth assignments in (9.60) are symmetric, as are fourth and seventh, and as are fifth and eighth assignments. Fourth assignment follows from the third by exchanging the labels x and y, and the fifth assignment follows from the third by exchanging the labels x and z. It is thus sufficient to consider only $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$= $(x, y, z, w, y, z, x, z, x, y)$ or $(x, y, z, w, y, z, x, z, w, y)$.

Consider $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$ = $(x, y, z, w, y, z, x, z, x, y)$.

By applying the pattern consistency for each edge and its end points in Figure 9.10 we obtain

$$\begin{aligned}
k_1 + xj_1 &\equiv k_3 + xj_3 \mod p,\\
k_1 + yj_1 &\equiv k_4 + yj_4 \mod p,\\
k_1 + zj_1 &\equiv k_5 + zj_5 \mod p,\\
k_1 + wj_1 &\equiv k_2 + wj_2 \mod p,\\
k_2 + yj_2 &\equiv k_3 + yj_3 \mod p,\\
k_2 + zj_2 &\equiv k_4 + zj_4 \mod p,\\
k_2 + xj_2 &\equiv k_5 + xj_5 \mod p,\\
k_3 + zj_3 &\equiv k_6 + zj_6 \mod p,\\
k_4 + xj_4 &\equiv k_6 + xj_6 \mod p, \text{and}\\
k_5 + yj_5 &\equiv k_6 + yj_6 \mod p.
\end{aligned} \tag{9.61}$$

Using the cycle consistency conditions for each of five cycles that span the cycle space of the graph in Figure 9.10 we also write

$$\begin{gathered}
w(j_2-j_1)+y(j_3-j_2)+x(j_1-j_3)\equiv 0 \mod p\\
w(j_2-j_1)+z(j_4-j_2)+y(j_1-j_4)\equiv 0 \mod p\\
w(j_2-j_1)+x(j_5-j_2)+z(j_1-j_5)\equiv 0 \mod p\\
y(j_4-j_1)+x(j_6-j_4)+z(j_3-j_6)+x(j_1-j_3)\equiv 0 \mod p\\
x(j_5-j_2)+y(j_6-j_5)+x(j_4-j_6)+z(j_2-j_4)\equiv 0 \mod p.
\end{gathered} \tag{9.62}$$

We will use the relationships in (9.62) to express j_3 through j_6 in terms of j_1 and (j_2-j_1), and then in turn use (9.61) to express k_2 through k_6 in terms of k_1, j_1 and (j_2-j_1).

By symmetry of the configuration (see Figure 9.10), for the current labelling it is sufficient to consider $x=0$ and $w=0$. Specifically, letting $y=0$ or $z=0$ reduces to the $x=0$ case.

We let $a:=j_2-j_1$, $b:=j_3-j_1$, $c:=j_4-j_1$, $d:=j_5-j_1$, and $e:=j_6-j_1$. Note that in particular by the check consistency constraint, $a\neq 0$.

1. Case $x=0$.

The system in (9.62) reduces to

$$\begin{aligned}
a(w-y)+by &\equiv 0 \mod p\\
a(z-w)+c(y-z) &\equiv 0 \mod p\\
aw-dz &\equiv 0 \mod p\\
bz+yc-ze &\equiv 0 \mod p\\
az-cz-dy+ey &\equiv 0 \mod p\,.
\end{aligned} \tag{9.63}$$

Using (9.63) we express b, c , d and e in terms of a. In particular, the last constraint in (9.63) is redundant as it follows from the previous four, as we now show.

Express b, c and d of a using top three equations in (9.63) so that

$$\begin{aligned} b &\equiv a\tfrac{y-w}{y} && \mod p \\ c &\equiv a\tfrac{w-z}{y-z} && \mod p \\ d &\equiv a\tfrac{w}{z} && \mod p\,. \end{aligned} \tag{9.64}$$

Substitute for b, c, d in terms of a in the fourth equation of the system (9.63) to obtain

$$e \equiv a(\frac{y-w}{y} - \frac{w-z}{y-z}\frac{y}{z}) \mod p\,. \tag{9.65}$$

Likewise, substitute for b, c, d in terms of a in the fifth equation of the system (9.63) to obtain

$$a\left(z - z\frac{w-z}{y-z} - \frac{wy}{z}\right) + ey \equiv 0 \mod p\,. \tag{9.66}$$

From (9.65) it follows that

$$(y-z)yze \equiv a\left(z(y-w)(y-z) + y^2(w-z)\right) \mod p, \tag{9.67}$$

and from (9.66) it follows that

$$a\left(z^2(y-z) - z^2(w-z) - wy(y-z)\right) + (y-z)yze \equiv 0 \mod p\,. \tag{9.68}$$

Rewrite (9.68) as

$$(y-z)yze \equiv a\left(-z^2(y-z) + z^2(w-z) + wy(y-z)\right) \mod p\,. \tag{9.69}$$

We expand the terms that multiply a in both (9.67) and (9.69). They both reduce to $-wyz-yz^2+wz^2+y^2w$, which makes the last equation in the system (9.63) redundant.

Therefore, for $q := j_1$ and $t := j_2 - j_1$, all of the remaining values of j_3, j_4, j_5, j_6 follow for each of the $3! = 6$ choices of (y, z, w).

Using (9.61) we further obtain

$$\begin{aligned}
k_3 &\equiv k_1 && \mod p \\
k_4 &\equiv k_1 - y(j_4 - j_1) && \mod p \\
k_5 &\equiv k_1 - z(j_5 - j_1) && \mod p \\
k_2 &\equiv k_5 && \mod p \\
k_6 &\equiv k_4 && \mod p.
\end{aligned} \tag{9.70}$$

Therefore, we can express k_2 through k_6 in terms of $s := k_1$, q and t. The results for all choices of (y, z, w) are summarized in Table 9.1.

y, z, w	j_1	j_2	j_3	j_4	j_5	j_6	k_1	k_2	k_3	k_4	k_5	k_6
$3, 2, 1$	q	$q + t$	$q + \frac{2t}{3}$	$q - t$	$q + \frac{t}{2}$	$q - \frac{5t}{6}$	s	$s - t$	s	$s + 3t$	$s - t$	$s + 3t$
$3, 1, 2$	q	$q + t$	$q + \frac{t}{3}$	$q + \frac{t}{2}$	$q + 2t$	$q + \frac{11t}{6}$	s	$s - 2t$	s	$s - \frac{3t}{2}$	$s - 2t$	$s - \frac{3t}{2}$
$2, 3, 1$	q	$q + t$	$q + \frac{t}{2}$	$q + 2t$	$q + \frac{t}{3}$	$q + \frac{11t}{6}$	s	$s - t$	s	$s - 4t$	$s - t$	$s - 4t$
$2, 1, 3$	q	$q + t$	$q - \frac{t}{2}$	$q + 2t$	$q + 3t$	$q + \frac{7t}{2}$	s	$s - 3t$	s	$s - 4t$	$s - 3t$	$s - 8t$
$1, 2, 3$	q	$q + t$	$q - 2t$	$q - t$	$q + \frac{3t}{2}$	$q - \frac{5t}{2}$	s	$s - 3t$	s	$s + t$	$s - 3t$	$s + t$
$1, 3, 2$	q	$q + t$	$q - t$	$q + \frac{t}{2}$	$q + \frac{2t}{3}$	$q - \frac{5t}{6}$	s	$s - 2t$	s	$s - \frac{t}{2}$	$s - 2t$	$s - \frac{t}{2}$

Table 9.1: Several solution sets for the (6,4) configuration.

Furthermore, under the current configuration, the bit nodes in one such $(6, 4)$ absorbing set that have 3 satisfied and 1 unsatisfied check, all have unsatisfied checks in the row group labelled w. By the bit consistency condition, no bit node can connect to more than one such check. Therefore, this configuration is in fact a $(6, 4)$ fully absorbing set.

The (absolute) indices of columns that correspond to the bit nodes in the absorbing set are $k_i + pj_i$ for $1 \leq i \leq 6$ and the indices of rows that correspond to the unsatisfied check nodes in the absorbing set are $(k_i + j_i w) \mod p + wp$, for $3 \leq i \leq 6$. In particular, the solution set in row 1 holds for all $p > 5$ and t a multiple of 6.

We complete the analysis of this label assignment by considering $w = 0$.

2. Case $w = 0$

In this case the system in (9.62) reduces to:

$$\begin{aligned} ay + b(x-y) &\equiv 0 \mod p \\ az + c(y-z) &\equiv 0 \mod p \\ ax + d(z-x) &\equiv 0 \mod p \\ b(z-x) + c(y-x) + e(x-z) &\equiv 0 \mod p \\ a(z-x) + c(x-z) + d(x-y) + e(y-x) &\equiv 0 \mod p\ . \end{aligned} \tag{9.71}$$

Note that the last relation follows from the previous four. We again express b, c d and e in terms of a, so that by setting $j_1 := q$ and $a := t$, all of j_2 through j_6 follow as a function of q and t. Then, by letting $k_1 := s$, the remaining k_2 through k_6 follow from q, t and s from (9.61). The solution set for various numerical assignments of (x, y, z) is given in Table 9.2.

As in the $x = 0$ case, the unsatisfied checks all belong in the row group labelled w. By the bit consistency condition, no bit node can connect to more than one such check. Therefore, this configuration is also in fact a $(6, 4)$ fully absorbing set.

The (absolute) indices of columns that correspond to the bit nodes in the absorbing set

x, y, z	j_1	j_2	j_3	j_4	j_5	j_6	k_1	k_2	k_3	k_4	k_5	k_6
$3, 2, 1$	q	$q+t$	$q-2t$	$q-t$	$q+\frac{3t}{2}$	$q-\frac{5t}{2}$	s	s	$s+6t$	$s+2t$	$s-\frac{3t}{2}$	$s+\frac{13t}{2}$
$3, 1, 2$	q	$q+t$	$q-\frac{t}{2}$	$q+2t$	$q+3t$	$q+\frac{7t}{2}$	s	s	$s+\frac{3t}{2}$	$s-2t$	$s-6t$	$s-\frac{13t}{2}$
$2, 3, 1$	q	$q+t$	$q+3t$	$q-\frac{t}{2}$	$q+2t$	$q+\frac{7t}{2}$	s	s	$s-6t$	$s+\frac{3t}{2}$	$s-2t$	$s-\frac{13t}{2}$
$2, 1, 3$	q	$q+t$	$q-t$	$q+\frac{3t}{2}$	$q-2t$	$q-\frac{5t}{2}$	s	s	$s+2t$	$s-\frac{3t}{2}$	$s+6t$	$s+\frac{13t}{2}$
$1, 2, 3$	q	$q+t$	$q+2t$	$q+3t$	$q-\frac{t}{2}$	$q+\frac{7t}{2}$	s	s	$s-2t$	$s-6t$	$s+\frac{3t}{2}$	$s-\frac{13t}{2}$
$1, 3, 2$	q	$q+t$	$q+\frac{3t}{2}$	$q-2t$	$q-t$	$q-\frac{5t}{2}$	s	s	$s-\frac{3t}{2}$	$s+6t$	$s+2t$	$s+\frac{13t}{2}$

Table 9.2: Several solution sets for the (6,4) configuration.

are $k_i + pj_i$ for $1 \leq i \leq 6$ and the indices of rows that correspond to the unsatisfied check nodes in the absorbing set are $(k_i + j_i w) \mod p + wp$, for $3 \leq i \leq 6$. In particular, the solution set in row 1 of the table in Figure 9.2 holds for all $p > 5$ and t even.

The remaining labelled configuration of Figure 9.10 to be considered is

$(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$ = $(x, y, z, w, y, z, x, z, w, y)$.

We let $a := j_2 - j_1$, $b := j_3 - j_1$, $c := j_4 - j_1$, $d := j_5 - j_1$, and $e := j_6 - j_1$. Note that in particular by the check consistency constraint, $a \neq 0$.

Based on the cycles consistency condition for the five cycles in Figure 9.10 we establish

$$\begin{aligned}
a(w-y) + b(y-x) &\equiv 0 \mod p \\
a(w-z) + c(z-y) &\equiv 0 \mod p \\
a(w-x) + d(x-z) &\equiv 0 \mod p \\
c(y-w) + b(z-x) + e(w-z) &\equiv 0 \mod p \\
d(x-y) + a(z-x) + c(w-z) + e(y-w) &\equiv 0 \mod p\,.
\end{aligned} \tag{9.72}$$

By expressing b, c and d in terms of a, from this system we obtain

$$a\left(\frac{(y-w)(w-z)}{y-z}+\frac{(z-x)(w-y)}{x-y}\right)+e(w-z) \equiv 0 \mod p \quad (9.73)$$
$$a\left(\frac{(x-y)(w-x)}{z-x}+(z-x)+\frac{(w-z)^2}{y-z}\right)+e(y-w) \equiv 0 \mod p, \quad (9.74)$$

where $\{x, y, z, w\} = \{0, 1, 2, 3\}$ and are distinct. For all $4! = 24$ distinct ways of assigning numerical values to x, y, z and w, the system (9.73)–(9.74) produces the unique solution $a = 0$, $e = 0$, provided that $p > 3$. Since $a \neq 0$ by the check consistency condition, we conclude that this configuration is not possible.

We now consider the second candidate (6,4) configuration.

II. Candidate (6,4) configuration, given in Figure 9.11.

We first determine all possible edge labellings. For convenience, let $(i_1, i_2, i_3, i_4) := (x, y, z, w)$, where $x, y, z, w \in \{0, 1, 2, 3\}$ and are distinct by the bit consistency condition at (j_1, k_1). Then, by imposing the bit consistency conditions at remaining vertices, the assignments for the remaining edge labels are given by the following set

$$\begin{gathered}(i_5, i_6, i_7, i_8, i_9, i_{10}) \in \{(y, x, w, z, z, x), (w, x, y, z, z, x), \\ (y, x, w, z, z, y), (y, w, x, z, z, y), (y, x, w, z, w, x), \\ (z, x, w, y, w, x), (y, z, w, x, w, y), (y, x, w, z, w, y)\}\ .\end{gathered} \quad (9.75)$$

Out of these 8 possible labelled configurations by appealing to symmetry and label renaming it is sufficient to consider only 2 of these as we now show. Note that the eighth labelling is the same as the first labelling after we exchange (j_3, k_3) and (j_4, k_4), (j_5, k_5) and (j_6, k_6), and labels y with x and w with z. Likewise, the second labelling is the same as the seventh labelling after we exchange (j_3, k_3) and (j_4, k_4), (j_5, k_5) and (j_6, k_6), and labels y with x and w with z. Sixth labelling is the same as the fourth labelling after we exchange labels z with x, y with w, and nodes (j_1, k_1) with (j_2, k_2), (j_3, k_3) with (j_4, k_4), and (j_5, k_5) with (j_6, k_6), and take the mirror image of the resulting configuration. Fifth labelling is the same as the third after we exchange labels z with x and y with w and take the mirror image of the whole configuration. Fourth (respectively first) labelling is the same as the second (respectively third) after we exchange (j_3, k_3) and (j_4, k_4) and labels x and y.

It is thus sufficient to consider only two different labellings, namely

$(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$= $(x, y, z, w, y, x, w, z, z, y)$ (third labelling) and

$(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$ = $(x, y, z, w, y, z, w, x, w, y)$ (seventh labelling).

For the first remaining case, by symmetry, it is sufficient to consider $x = 0$ and $z = 0$ as $w = 0$ and $y = 0$ reduce to the $x = 0$ and $z = 0$ case respectively. Likewise, for the second case it is sufficient to consider $x = 0$ and $y = 0$, as $z = 0$ and $w = 0$ each reduce to the $x = 0$ and $y = 0$ cases, respectively.

Consider $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$= $(x, y, z, w, y, x, w, z, z, y)$.

1. Case $z = 0$

From Figure 9.11 and under the current edge label assignment using the pattern consistency constraints of Corollary 1 we write

$$
\begin{aligned}
k_1 &\equiv k_5 && \mod p \\
k_2 &\equiv k_6 && \mod p \\
k_3 &\equiv k_4 && \mod p \\
k_1 + xj_1 &\equiv k_3 + xj_3 && \mod p \\
k_1 + yj_1 &\equiv k_4 + yj_4 && \mod p \\
k_1 + wj_1 &\equiv k_6 + wj_6 && \mod p \\
k_2 + yj_2 &\equiv k_3 + yj_3 && \mod p \\
k_2 + xj_2 &\equiv k_4 + xj_4 && \mod p \\
k_2 + wj_2 &\equiv k_5 + wj_5 && \mod p \\
k_5 + yj_5 &\equiv k_6 + yj_6 && \mod p \, .
\end{aligned}
\tag{9.76}
$$

Let $a := j_2 - j_1$, $b := j_3 - j_1$, $c := j_4 - j_1$, $d := j_5 - j_1$ and $e := j_6 - j_1$. Using the cycle constraint for four cycles spanning the cycle space of the configuration in Figure 9.11 and under the current edge labelling we have

$$
\begin{aligned}
xb + y(-c) &\equiv 0 \mod p, \\
y(b - a) + x(a - c) &\equiv 0 \mod p, \\
y(e - d) + w(-e) &\equiv 0 \mod p, \\
w(d - a) + y(e - d) &\equiv 0 \mod p.
\end{aligned}
\tag{9.77}
$$

From the system (9.76) we write

$$\begin{aligned} k_1 - k_2 \equiv k_1 - k_6 \equiv w(j_6 - j_1) \equiv we \qquad & \mod p, \\ k_1 - k_3 \equiv k_1 - k_4 \equiv y(j_4 - j_1) \equiv yc \qquad & \mod p, \\ k_2 - k_3 \equiv y(j_3 - j_2) \equiv y(b - a) \qquad & \mod p. \end{aligned} \tag{9.78}$$

Using the identity $(k_1 - k_2) = (k_1 - k_3) - (k_2 - k_3)$, and (9.78) we obtain

$$we \equiv y(c - b + a) \mod p. \tag{9.79}$$

There are six possible assignments for (x, y, w), as permutations of the set $(1, 2, 3)$. In the remainder we will show that in fact only $(x, y, w) = (1, 3, 2)$ gives rise to absorbing sets. In all other cases, using (9.77) and (9.79), we will reach a contradiction.

From (9.77) we have

$$\begin{gathered} xb \equiv yc \mod p \\ yd \equiv (y - w)e \mod p. \end{gathered} \tag{9.80}$$

We also have

$$\begin{gathered} xa - (y + x)c \equiv 0 \mod p \\ (2y - w)e \equiv ya \mod p, \end{gathered} \tag{9.81}$$

where the top expression in (9.81) follows from substituting top expression in (9.80) into the second expression of (9.77) and the bottom expression in (9.81) follows from substituting bottom expression in (9.80) into the fourth expression of (9.77).

For $(y, w, x) = (1, 2, 3)$, the bottom expression in (9.81) gives

$$a \equiv 0 \mod p,$$

which then implies

$$c \equiv 0 \mod p,$$

by the top expression in (9.81). Since $c = j_4 - j_1$, and (j_1, k_1) and (j_4, k_4) share a check, c must be non-zero, implying a contradiction.

For $(y, w, x) \in \{(1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2)\}$ we express b, c, d, e in terms of a using (9.80) and (9.81).

For $(y, w, x) = (1, 3, 2)$ we obtain:

$$b \equiv a/3 \mod p, c \equiv 2a/3 \mod p, d \equiv 2a \mod p, e \equiv -a \mod p.$$

For $(y, w, x) = (2, 1, 3)$ we obtain:

$$b \equiv 2a/5 \mod p, c \equiv 3a/5 \mod p, d \equiv a/3 \mod p, e \equiv 2a/3 \mod p.$$

For $(y, w, x) = (2, 3, 1)$ we obtain:

$$b \equiv 2a/3 \mod p, c \equiv a/3 \mod p, d \equiv -a \mod p, e \equiv 2a \mod p.$$

For $(y, w, x) = (3, 1, 2)$ we obtain:

$$b \equiv 3a/5 \mod p, c \equiv 2a/5 \mod p, d \equiv 2a/5 \mod p, e \equiv 3a/5 \mod p.$$

In all four cases, when b, c and e are substituted in (9.79) it follows that $a \equiv 0 \mod p$ (we get $-3a \equiv 4a/3 \mod p$, $2a/3 \equiv 12a/5 \mod p$, $6a \equiv 4a/3 \mod p$, and $3a/5 \equiv 12a/5 \mod p$, respectively). Since b is a multiple of a in all four cases, if $a \equiv 0 \mod p$, then $b \equiv 0 \mod p$ as well. Since $b = j_3 - j_1$ and nodes (j_1, k_1) and (j_3, k_3) share a check, b must be non-zero, thus implying a contradiction.

For $(y, w, x) = (3, 2, 1)$ we obtain:

$$b \equiv 3a/4 \mod p, c \equiv a/4 \mod p, d \equiv a/4 \mod p, e \equiv 3a/4 \mod p.$$

When b, c and e are substituted in (9.79), we obtain the identity $3a/2 \equiv 3a/2 \mod p$. Since $c \equiv d \mod p$, we have that $j_4 = j_5$ and since $b \equiv e \mod p$, we have that $j_3 = j_6$. Note that neither of these two conditions on js violates the check consistency constraint since the respective bit nodes do not share a check in Figure 12. Let $q = j_1$ and $t = j_4 - j_1$. Then $j_4 = q + t$ and $j_5 = q + t$. Since $b = 3c$, and $b = j_3 - j_1$ and $c = j_4 - j_1$, we have that $j_3 = q + 3t$. Since $j_3 = j_6$, $j_6 = q + 3t$ as well. Likewise, since $a = 4c$, and $a = j_2 - j_1$ and $c = j_4 - j_1$, we have that $j_2 = q + 4t$. We have thus expressed all of j_1 through j_6 in terms of q and t. Now the system (9.76) reduces to

$$\begin{aligned} k_1 &\equiv k_5 \mod p, \\ k_2 &\equiv k_6 \mod p, \\ k_3 &\equiv k_4 \mod p, \\ k_1 - k_3 &\equiv 3t \mod p, \\ k_1 - k_2 &\equiv 6t \mod p, \\ k_2 - k_3 &\equiv -3t \mod p. \end{aligned} \tag{9.82}$$

Thus, with $s = k_1$ and using (9.82) we can express all of k_1 through k_6 in terms of s and t. This solution set for j_1 through j_6 and k_1 through k_6 is listed in Table 9.3.

Note that the result in Table 9.3 establishes the existence of a (6,4) absorbing set. Even though $j_3 = j_6$ and $j_4 = j_5$ the check consistency constraints are not violated as (j_3, k_3)

x, y, w	j_1	j_2	j_3	j_4	j_5	j_6	k_1	k_2	k_3	k_4	k_5	k_6
$1, 3, 2$	q	$q+4t$	$q+3t$	$q+t$	$q+t$	$q+3t$	s	$s-6t$	$s-3t$	$s-3t$	s	$s-6t$

Table 9.3: A solution set for an (6,4) absorbing set.

and (j_6, k_6) do not share an edge, and neither do (j_4, k_4) and (j_5, k_5), see Figure 9.11.

We now discuss whether this set is also a (6,4) fully absorbing set. Suppose there exists a bit node (j_7, k_7) outside this absorbing set that is incident to some of the unsatisfied checks. By the bit consistency constraint, both (j_3, k_3) and (j_4, k_4) each have a neighboring unsatisfied check whose label is w. These two checks must be distinct by the girth condition. Likewise, both (j_5, k_5) and (j_6, k_6) each have a neighboring unsatisfied check whose label is x, and these are also distinct by the girth condition. By the bit consistency condition, the bit node (j_7, k_7) can then share at most 2 of these checks with the bit nodes (j_3, k_3) through (j_7, k_7).

Suppose that the bit node (j_7, k_7) shares the check with each of (j_3, k_3) and (j_5, k_5). From the cycles relating bit nodes (j_7, k_7), (j_3, k_3), (j_5, k_5), (j_1, k_1), and (j_2, k_2), we obtain

$$\begin{aligned} x(j_7 - j_5) + w(j_3 - j_7) + x(j_1 - j_3) &\equiv 0 \qquad \bmod p \\ w(j_5 - j_2) + x(j_7 - j_5) + w(j_3 - j_7) + y(j_2 - j_3) &\equiv 0 \qquad \bmod p \,. \end{aligned}$$

For $(x, y, z, w) = (1, 3, 0, 2)$ of present interest, we obtain that $j_7 \equiv q + 2t \mod p$ using the result in Table 9.3.

Since we further have

$$\begin{aligned} k_3 + 2j_3 &\equiv k_7 + 2j_7 \mod p \\ k_5 + j_5 &\equiv k_7 + j_7 \mod p, \end{aligned}$$

it follows that $k_7 \equiv s - t \mod p$. Therefore by the existence of this bit node (j_7, k_7), the current (6,4) absorbing set is not a (6,4) fully absorbing set.

We now consider $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$= $(x, y, z, w, y, x, w, z, z, y)$ with $x = 0$.

1. Case $x = 0$

As before, using the pattern consistency constraints we establish:

$$\begin{aligned} k_1 &\equiv k_3 \mod p \\ k_2 &\equiv k_4 \mod p \\ k_1 + yj_1 &\equiv k_4 + yj_4 \mod p \\ k_1 + zj_1 &\equiv k_5 + zj_5 \mod p \\ k_1 + wj_1 &\equiv k_6 + wj_6 \mod p \\ k_2 + yj_2 &\equiv k_3 + yj_3 \mod p \\ k_2 + wj_2 &\equiv k_5 + wj_5 \mod p \\ k_2 + zj_2 &\equiv k_6 + zj_6 \mod p \\ k_3 + zj_3 &\equiv k_4 + zj_4 \mod p \\ k_5 + yj_5 &\equiv k_6 + yj_6 \mod p, \ . \end{aligned} \tag{9.83}$$

Let $a := j_2 - j_1$, $b := j_3 - j_1$, $c := j_4 - j_1$, $d := j_5 - j_1$, and $e := j_6 - j_1$. Using the cycle constraints for four cycles spanning the cycle space of the configuration in

Figure 9.11 we may also write

$$\begin{aligned} z(c-b)+y(-c) &\equiv 0 \mod p, \\ y(b-a)+z(c-b) &\equiv 0 \mod p, \\ zd+y(e-d)+w(-e) &\equiv 0 \mod p, \\ w(d-a)+y(e-d)+z(a-e) &\equiv 0 \mod p\,. \end{aligned} \tag{9.84}$$

There are 6 possible assignments for (y, z, w) as permutations of the set $\{1, 2, 3\}$. We will show that in fact the only possible assignment is $(y, z, w) = (2, 1, 3)$ (a contradiction will be reached in all other cases).

Consider first the assignment $(y, z, w) = (1, 2, 3)$. Using (9.84) we express a, b, c and d in terms of e so that

$$\begin{aligned} a &\equiv 3e \mod p, \\ b &\equiv e \mod p, \\ c &\equiv 2e \mod p, \\ d &\equiv 2e \mod p\,. \end{aligned} \tag{9.85}$$

Note that since $c \equiv d \mod p$ and $b \equiv e \mod p$ the above implies that $j_4 = j_5$ and $j_3 = j_6$. Even though now some vertices have the same j components, the check consistency condition is not violated as (j_4, k_4) and (j_5, k_5) do not share an edge, and neither do (j_3, k_3) and (j_6, k_6) (see Figure 9.11).

From (9.83) and by substituting for a, c and d in terms of e using (9.85) we note that

$$\begin{aligned} k_1 - k_2 &\equiv 1(2e) \mod p, \\ k_1 - k_5 &\equiv 2(2e) \mod p, \\ k_2 - k_5 &\equiv 3(2e-3e) \mod p\,. \end{aligned} \tag{9.86}$$

The system (9.86) implies that $e \equiv 0 \mod p$ for $p > 5$. Since $e = j_6 - j_1$ and (j_1, k_1) and (j_6, k_6) do share an edge, the condition $e \equiv 0 \mod p$ violates the check consistency constraint. We thus conclude that the current numerical assignment for (y, z, w) is not possible.

By expressing a, b, c and d in terms of e as in (9.85) and then using (9.83) to express the differences $k_1 - k_2$, $k_1 - k_5$, and $k_2 - k_5$ as in (9.86) we conclude that $e \equiv 0 \mod p$ when (y, z, w) = $(1, 3, 2)$ and $p > 5$ as well as (y, z, w) = $(3, 1, 2)$ and $p > 7$ or $(3, 2, 1)$ and $p > 13$.

Consider now the assignment $(y, z, w) = (2, 3, 1)$. Using (9.84) it follows after substituting for d in terms of e in the last expression that

$$a \equiv 0 \mod p\,. \tag{9.87}$$

By substituting for b in terms of c in the second expression in (9.84) it also follows that

$$3a \equiv 4c \mod p\,. \tag{9.88}$$

Therefore $c \equiv 0 \mod p$, which violates the check consistency constraint for the edge connecting bit nodes (j_1, k_1) and (j_4, k_4). The condition $a \equiv 0 \mod p$ by itself does not yield a contradiction as the nodes (j_1, k_1) and (j_2, k_2) do not have any edges in common.

Finally, we consider the assignment $(y, z, w) = (2, 1, 3)$. First, by substituting for d in terms of e in the last expression that

$$a \equiv 0 \mod p\,. \tag{9.89}$$

Therefore $j_1 = j_2$. By substituting for b in terms of c in the second expression in (9.84) we have that

$$1a \equiv (2-2)c \mod p\,, \tag{9.90}$$

which does not tell us anything about the actual value of c. We express b, c and d in terms of e, again using (9.84), and obtain

$$\begin{aligned} b &\equiv -e \mod p, \\ c &\equiv e \mod p, \\ d &\equiv -e \mod p\,. \end{aligned} \tag{9.91}$$

Since $b \equiv d \mod p$, $j_3 = j_5$, and since $c \equiv e \mod p$, $j_4 = j_6$. Neither of these conditions on j's violates the check consistency constraints as the respective bit nodes do not share edges (see Figure 9.11). Thus, with $q := j_1$ and $t := e$ we can express all of j_1 through j_6 in terms of q and t. Having verified that all constraints given by (9.83) are in fact consistent for $s := k_1$ we obtain the solution set given in Table 9.4, in terms of q, t and s.

y, z, w	j_1	j_2	j_3	j_4	j_5	j_6	k_1	k_2	k_3	k_4	k_5	k_6
$2, 1, 3$	q	q	$q-t$	$q+t$	$q-t$	$q+t$	s	$s-2t$	s	$s-2t$	$s+t$	$s-3t$

Table 9.4: A solution set for the (6,4) configuration.

From Figure 9.11 and under current labelling, note that the bit nodes (j_3, k_3) and (j_4, k_4) both have an unsatisfied check whose label is w, and that likewise the bit nodes (j_5, k_5) and (j_6, k_6) both have an unsatisfied check whose label is x. Therefore there could be a bit node that potentially connects to 2 satisfied and 2 unsatisfied check nodes. Consider a bit node (j_7, k_7) that shares a check with each of (j_3, k_3) and (j_5, k_5). By the parity check constraint

$$k_7 + wj_7 \equiv k_3 + wj_3 \mod p,$$
$$k_7 + xj_7 \equiv k_5 + xj_5 \mod p\ .$$

for $(x, y, z, w) = (0, 2, 1, 3)$, it follows that $k_7 = k_5 \equiv s + t \mod p$ and $j_7 \equiv q - 4t/3 \mod p$. Thus, the existence of this (j_7, k_7) bit node for t a multiple of 3, makes the candidate configuration be a (6,4) absorbing set but not a (6,4) fully absorbing set.

Consider now the second remaining labelling, namely $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$ = $(x, y, z, w, y, z, w, x, w, y)$. We will show that it in fact not possible for p large enough.

Applying the cycle consistency condition to the four cycles in Figure 9.11 for $a := j_2 - j_1$, $b := j_3 - j_1$, $c := j_4 - j_1$, $d := j_5 - j_1$, and $e := j_6 - j_1$ we obtain

$$\begin{aligned}
b(x-w) + c(w-y) &\equiv 0 \mod p \\
e(w-y) + d(y-z) &\equiv 0 \mod p \\
a(x-w) + d(w-y) + e(y-x) &\equiv 0 \mod p \\
a(z-y) + b(y-w) + c(w-z) &\equiv 0 \mod p\ .
\end{aligned} \tag{9.92}$$

We can express all of b, c, d, e as certain multiples of a, depending on the actual numerical values of y, z and w.

From this figure we also obtain using the pattern consistency conditions the following:

$$\begin{aligned}
k_1 + xj_1 &\equiv k_3 + xj_3 \mod p \\
k_1 + yj_1 &\equiv k_4 + yj_4 \mod p \\
k_1 + zj_1 &\equiv k_5 + zj_5 \mod p \\
k_1 + wj_1 &\equiv k_6 + wj_6 \mod p \\
k_2 + yj_2 &\equiv k_3 + yj_3 \mod p \\
k_2 + zj_2 &\equiv k_4 + zj_4 \mod p \\
k_2 + wj_2 &\equiv k_5 + wj_5 \mod p \\
k_2 + xj_2 &\equiv k_6 + xj_6 \mod p \\
k_3 + wj_3 &\equiv k_4 + wj_4 \mod p \\
k_5 + yj_5 &\equiv k_5 + yj_6 \mod p \,.
\end{aligned} \tag{9.93}$$

1. Case $x = 0$

With $x = 0$, (9.93) yields $k_1 \equiv k_3 \mod p$ and $k_2 \equiv k_6 \mod p$ so that

$$k_1 - k_2 \equiv we \equiv y(a-b) \mod p \,. \tag{9.94}$$

From (9.92) we then have

$$\begin{aligned}
a(z-y)(y-w) + b[(-w)(w-z) + (y-w)^2] &\equiv 0 \qquad \mod p, \\
aw(y-z) + e[(w-y)^2 + y(z-y)] &\equiv 0 \qquad \mod p \,.
\end{aligned} \tag{9.95}$$

From (9.94), and (9.95) it follows that $a \equiv 0 \mod p$ for all $3! = 6$ numerical assignments of y, z and w, for $p \notin \{2, 3, 5, 7, 37\}$ and consequently $b \equiv 0 \mod p$. Since (j_1, k_1) and (j_3, k_3) share an edge in Figure 9.11, the $b \equiv 0 \mod p$ condition violates the check consistency constraint for all but a small finite number of values of p.

2. Case $y = 0$

We now have $k_1 \equiv k_4 \mod p$, $k_2 \equiv k_3 \mod p$ and $k_5 \equiv k_6 \mod p$ and

$$xb \equiv zd - w(d-a) \mod p\,, \tag{9.96}$$

which follows from $k_1 - k_2 = (k_1 - k_5) - (k_2 - k_5)$ and $k_2 - k_3$. From (9.92) we also have

$$\begin{aligned} a(-wz) + b[w^2 + (w-z)(x-w)] &\equiv 0 \qquad \mod p, \\ a(x-w)w + d(w^2 - xz) &\equiv 0 \qquad \mod p\,. \end{aligned} \tag{9.97}$$

Combining (9.93) and (9.97) it again follows that $a \equiv 0 \mod p$ for all $3! = 6$ numerical assignments of x, z and w for $p \notin \{2, 3, 5, 7, 37\}$. This in turn implies that $b \equiv 0$ $\mod p$, which violates the check consistency condition.

Lastly, we consider the third and final unlabelled candidate (6,4) absorbing set, for which we show that in fact does not yield (6,4) absorbing sets for the prime p large enough.

III. Candidate (6,4) configuration, given in Figure 9.12.

We first determine all possible edge labellings. As before we let $(i_1, i_2, i_3, i_4) := (x, y, z, w)$, where $x, y, z, w \in \{0, 1, 2, 3\}$ and distinct by the bit consistency condition at (j_1, k_1). Then, by propagating bit consistency conditions for remaining vertices, the assignments for the remaining edge labels are given by the following set

$$\begin{aligned} (i_5, i_6, i_7, i_8, i_9, i_{10}) \in \{&(x,y,z,z,x,y), (x,y,z,z,x,w), \\ &(x,y,z,z,w,y), (x,y,z,w,x,y), (x,y,z,w,w,y), \\ &(z,x,y,w,w,z), (z,y,x,w,w,y), (z,y,x,w,w,z)\}\,. \end{aligned}$$

By exploiting the symmetry, one can show that after renaming the labelling

$(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$= $(x, y, z, w, x, y, z, z, w, y)$ and $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$= $(x, y, z, w, x, y, z, w, x, y)$ reduce to the same case (by exchanging z and x).

We are thus left with analyzing the remaining seven cases.

As before, we let $a := j_2 - j_1$, $b := j_3 - j_1$, $c := j_4 - j_1$, $d := j_5 - j_1$, and $e := j_6 - j_1$. Note that in particular by the check consistency constraint, $a \neq 0$.

Consider the labelling $(i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10})$ = $(x, y, z, w, x, y, z, z, x, y)$. We apply the cycle consistency conditions to five cycles spanning the cycle space of the graph in Figure 9.12:

$$\begin{aligned}
xb + z(c-b) - yc &\equiv 0 \quad \mod p \\
yc + x(a-c) - wa &\equiv 0 \quad \mod p \\
-wa + zd + y(a-d) &\equiv 0 \quad \mod p \\
y(d-a) + x(e-d) + z(a-e) &\equiv 0 \quad \mod p \\
xb + y(e-b) + x(d-e) - zd &\equiv 0 \quad \mod p\ .
\end{aligned} \tag{9.98}$$

By expressing b, c and d in terms of a, and substituting in the bottom two constraints (9.98) we obtain

$$a\left(z - y + \frac{(y-w)(y-x)}{y-z}\right) + e(x-z) \equiv 0 \quad \mod p \tag{9.99}$$

$$a\left(\frac{(y-z)(x-w)}{x-z} + \frac{(y-w)(x-z)}{y-z}\right) + e(y-x) \equiv 0 \quad \mod p, \tag{9.100}$$

where $\{x, y, z, w\} = \{0, 1, 2, 3\}$ and are distinct. For all $4! = 24$ distinct ways of assigning numerical values to x, y, z and w, the system (9.99) – (9.100) produces the unique solution $a = 0$, $e = 0$, provided that $p > 3$. Since $a \neq 0$ by the check consistency condition, we conclude that this configuration is not possible.

One can likewise establish the constraints of the (9.98) type for the remaining six cases, from which the two equations (as in (9.99),(9.100)) relating a and e will follow. In all five cases, the unique solution for p large enough is $(a, e) = (0, 0)$. In particular, $p > 13$ is sufficient for all cases considered.

Having exhaustively considered all possible configurations of $(6, 4)$ absorbing sets, the proof of the lemma is complete. ■

Using these results the proof of Theorem 3(c) now follows. We complete our analysis of $\gamma = 4$ by proving the claim in Theorem 4: The number of $(6, 4)$ (fully) absorbing sets scales as $\Theta(n^{3/2})$, where n is the codeword length.

Proof: Recall that for the configuration in Figure 9.10 we identified two sets of labellings given in tables in Figures 9.1 and 9.2 that determine (6,4) fully absorbing sets. For each such assignment there are three parameters that determine all of j's and k's, and each parameter is chosen independently in at most p ways (to ensure the all j's and k's have integer values), yielding an upper bound which grows as $\Theta(p^3)$. A lower bound on the cardinality of the $(6, 4)$ fully absorbing sets is given by the solution set in Table 9.1, which also grows as $\Theta(p^3)$. Note that the number of solutions of absorbing sets in Table 9.3 and Table 9.4 grows as $\Theta(p^3)$ as well. Since $n = p^2$, the result follows. ■

We have thus proven Theorem 4 for $\gamma = 4$.

9.3 Experimental Results

The experiments were carried out using an LDPC code emulator described in detail in [63]. The decoder was implemented using a 4.5 (4 bits for integer and 5 bits for the fractional part) uniform quantization. The all-zero codeword was transmitted and the decoder was set to run for at most 200 iterations, halting earlier if decoding to a codeword. The frame error rate and the bit error rate for the $C_{47,4}$ code are shown in Figure 9.13, along with the uncoded BER curve. In the low BER region of 10^{-10} and below, all errors were found to be due to fully absorbing sets, and none was a codeword. A total of 25 errors were recorded at SNR = 6.4 dB, of which 18 errors were of smallest weight and all due to $(6, 4)$ fully absorbing sets.

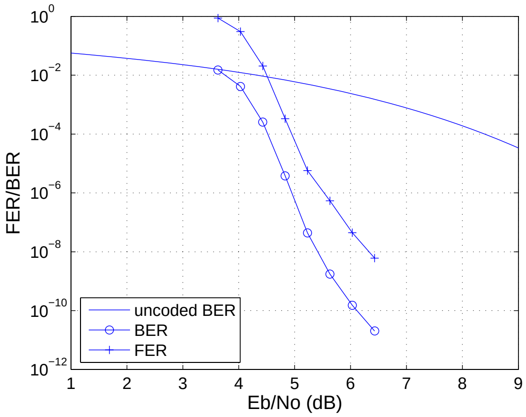

Figure 9.13: Experimental Results for $C_{47,4}$

9.4 Conclusion and Future Work

In this chapter we presented a detailed analysis of the dominant configurations in the error-floor regime of high rate array-based LDPC codes. We provided an explicit description of the minimal (fully) absorbing sets and showed the non-existence of certain candidate configurations. We also enumerated minimal (fully) absorbing sets and showed how their number scales with the codeword length. Experiments on an emulation platform were performed and were found to be in agreement with the theoretical description of the dominant errors.

Part III

Conclusion and Future Extensions

This dissertation dealt with two interesting problems in modern coding theory that arise when conventional assumptions on communication no longer hold. First, we addressed the issue of inadequate synchronization from a coding theoretic-perspective. We derived several novel structural properties of Reed-Muller codes which were subsequently used for an in-depth study of these codes in channels which in addition to substitution errors permit a bit repetition or a bit deletion. We presented explicit number-theoretic constructions for communication in the presence of repetition errors. Our constructions are asymptotically optimal for the case of one repetition, and are within a constant factor of the upper bound on the cardinality for the case of multiple repetitions. They improve on the previously best known results.

We also discussed a general prefixing method for improving the repetition error correcting capability of a given (additive) error correcting code. The proposed method leverages number-theoretic constructions and constructs a carefully chosen prefix whose length is only logarithmic in the codeword length, while providing a guarantee on the immunity to repetition errors. A possible future extension would be to implement the proposed methods as a component of a real communication system. Another interesting extension would be to combine the proposed prefixing method with a suitable decoding algorithm, of the kind presented here.

In the second part, we discussed the LDPC code performance in the low BER regime under iterative decoding. We introduced the concept of (fully) absorbing sets. Fully absorbing sets are combinatorial objects in the Tanner graph of the code, that are stable under the bit flipping algorithm. As a concrete case study, we systematically described minimal (fully) absorbing sets of high rate array-based LDPC codes. These were shown to entirely dominate the low BER performance over the AWGN channel under practical iterative decoding algorithms. This observation is in contrast to the minimum distance codewords which play the central role in describing the performance of a code under the optimal yet computationally expensive maximum likelihood decoding algorithm. Insights from the study of absorbing sets have already been used for improved decoder implementations [64], and for the error-floor prediction based on fast stochastic simulation, [17].

As a future direction, it would be interesting to establish an explicit link between absorbing sets, which play an important role in describing the performance of message passing algorithms over AWGN channels, and pseudo-codewords [20], which are useful in understanding linear programming-based decoding. It would also be useful to generalize the analysis presented here to other LDPC code families, and to derive asymptotic properties of code ensembles from the point of view of absorbing sets.

Bibliography

[1] S. Aji and R. McEliece, “The generalized distributive law”, *IEEE Transactions on Information Theory* vol. 46(2), pp. 325–43, March 2000.

[2] M. J. Ammer, *Low Power Synchronization for Wireless Communication*, PhD Thesis, EECS Dept., UC Berkeley, 2004.

[3] T. M. Apostol, “*Introduction to Analytic Number Theory*”, Springer-Verlag, NY, 1976.

[4] J. Barry, A. Kavcic, S. McLaughlin, A. Nayak, and W. Zeng, “Iterative timing recovery”, *IEEE Signal Processing Magazine*, 21:89–102, January 2004.

[5] R. C. Bose and J. G. Caldwell “Synchronizable error-correcting codes”, *Information and Control*, 10:616–630, 1967.

[6] P. A. H. Bours, “Construction of fixed-length insertion/deletion correcting runlength-limited code”, *IEEE Transactions on Information Theory*, vol. 40(6), pp. 1841–1856, Nov. 1994.

[7] L. Calabi and W. E. Hartnett, “A family of codes for the correction of substitution and synchronization errors”, *IEEE Trans. Inform. Theory*, 15(1):102–106, January 1969.

[8] G. Chen, M. Mitzenmacher, C. Ng, and N. Varnica, “Concatenated codes for deletion channels”, In *Proceedings of the IEEE International Symposium on Information Theory*, Yokohama, Japan, 2003.

[9] M. Chertkov and V.Y. Chernyak, “Loop calculus helps to improve belief propagation and linear programming decodings of low-density-parity-check codes, ” Proceedings of the *44th Annual Allerton Conf. on Communications, Control and Computing*, Monticello, IL, USA, Sep. 27-29, 2006.

[10] S.-Y. Chung, G. D. J. Forney, T. Richardson, and R. Urbanke “On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit”, *IEEE Communications Letters*, vol. 5, no. 2. pp.58–60, 2001.

[11] W. A. Clarke and H. C. Ferreira, “A new linear, quasicyclic multiple insertion/deletion correcting code”, In *2003 IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, 2003.

[12] W. A. Clarke, A. S. J. Helberg, and H. C. Ferreira, “Constrained codes for the correction of synchronization and additive errors”, In *Proceedings of the 1993 IEEE South African Symposium on Communications and Signal Processing. COMSIG*, New York, NY, USA, 1993.

[13] M. C. Davey and D. J. C. MacKay, “Reliable communication over channels with insertions, deletions and substitutions”, *IEEE Trans. Inform. Theory*, 47(2):687–698, Feb. 2001.

[14] C. Di, D. Proietti, T. Richardson, E. Telatar, and R. Urbanke, ”Finite length analysis of low-density parity-check codes on the binary erasure channel,” *IEEE Transactions on Information Theory*, vol. (48), pp. 1570–1579, 2002.

[15] L. Dolecek and V. Anantharam, “A synchonization technique for array-based LDPC codes”, *International Symposium on Information Theory*, Seattle, WA, July 9-13, 2006.

[16] L. Dolecek, “On Structural Properties of Reed-Muller RM(1,m) Codes and Their Use in Channels with Synchronization Errors”, Technical report, EECS Dept., UC Berkeley, April 2006. Available online at http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-43.pdf

[17] L. Dolecek, Z. Zhang, M. Wainwright, V. Anantharam, and B. Nikolic, “Evaluation of the low frame error rate performance of LDPC codes using importance sampling,” *Information Theory Workshop*, Lake Tahoe, CA, Sept. 2007.

[18] E. Eleftheriou and S. Ölçer, ”Low density parity check codes for digital subscriber lines”, *Proc. of the IEEE Int. Conf. on Commun.*, New York, NY, 2002, pp. 1752–1757.

[19] J. L. Fan, “Array-codes as low-density parity-check codes,” *Second Int. Symp. on Turbo Codes*, Brest, France, pp. 543–46, Sept. 2000.

[20] J. Feldman, M. J. Wainwright and D. R. Karger, “Using linear programming to decode binary linear codes”, *IEEE Transactions on Information Theory,* vol. (51), pp. 954–972.

[21] H. C. Ferreira, W. A. Clarke, A. S. J. Helberg, K. A. S. Abdel-Gaffar and A. J. Han Vinck, “Insertion/deletion correction with spectral nulls”, *IEEE Transactions on Information Theory*, vol. 43(2), pp. 722–732, March 1997.

[22] G. D. Forney, “Codes on graphs: normal realizations”, *IEEE Trans. Inform. Theory*, vol. (47), pp. 520–548, Feb. 2001.

[23] R. Gallager, *Low Density Parity Check Codes*, MIT Press, 1964.

[24] E. N. Gilbert and J. Riordan, “Symmetry types of periodic sequences”, *Illinois Journal of Mathematics,* Vol. 5, pp. 657–665, 1961.

[25] A. S. J. Helberg and H. C. Ferreira, “On multiple insertion/deletion correcting codes”, *IEEE Transactions on Information Theory* vol. 48(1), pp. 305–308, Jan. 2002.

[26] R. G. Gallager, *Low Density Parity Check Codes,* Monograph, M.I.T. Press, 1963.

[27] L. K. Hua and H. S. Vandiver, “Characters over certain types of rings with applications to the theory of equations in a finite field”, *Proc. Nat. Acad. Sci. USA*, vol. 35, pp. 481-487, 1949.

[28] N. Kashyap and A. Vardy, “Stopping sets in codes from designs,” *International Symposium on Information Theory*, pp. 122, Yokohama, Japan, July, 2003.

[29] T. Kløve, “Codes correcting a single insertion/deletion of a zero or a single peak-shift”, *IEEE Trans. Inform. Theory*, 41(1):279–283, January 1995.

[30] R. Koetter and P. Vontobel, “Graph covers and iterative decoding of finite-length codes,” Procedings of the *Third International Symposium on Turbo Codes and Related Topics*, Brest, France, pp. 75-82, Sept. 2003.

[31] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. Kurtas, “Per-survivor timing recovery for uncoded partial response channels”, In *Proceedings of the IEEE International Conference on Communications*, 2004.

[32] Y. Kou, S. Lin, and M. P. C. Fossorier, “Low-density parity-check codes based on finite geometries: arediscovery and new results” *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711 - 2736, Nov. 2001.

[33] S. Laendner and O. Milenkovic, “Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes, ” in *Wireless Comm*, Hawaii, USA, 2005.

[34] V. I. Levenshtein, ”Binary codes capable of correcting spurious insertions and deletions of ones”, *Problems of Information Transmission*, vol. 1(1),pp. 8–17, Jan. 1965.

[35] V. I. Levenshtein, ”Binary codes capable of correcting deletions, insertions and reversals”, *Sov. Phys.-Dokl.*, vol. 10(8), pp. 707–710, Feb. 1966.

[36] V. I. Levenshtein, “On perfect codes in deletion and insertion metric”, *Discrete Math. Appl.*, 2(3):241–258, 1992.

[37] S. Lin and D. Costello, *Error Control Coding*, Second Edition, Prentice Hall, 2004.

[38] J. Liu, H. Song, and B.V.K.V. Kumar, “Symbol timing recovery for low-SNR partial response recording channels”, In *GLOBECOM’02 - IEEE Global Telecommunications Conference*, 2002.

[39] D. MacKay and R. Neal, “Near Shannon limit performance of low density parity check codes”, *Electronic Letters*, vol. 32, no. 18, pp. 1645–46, August 1996.

[40] D. MacKay and M. Postol, “Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes,” *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.

[41] F. J. MacWilliams and N. J. A. Sloane, *“The Theory Of Error Correcting Codes”*, North Holland Publishing Company, Amsterdam, Holland, 1977.

[42] T. Mittelholzer, “Efficient encoding and minimum distance bounds of Reed-Solomon-type array codes,” *International Symposium on Information Theory*, Lausanne, Switzerland, July 2002, p. 282.

[43] A. R. Nayak, J. Barry, and S. W. McLaughlin, “Joint timing recovery and turbo equalization for coded partial response channels”, *IEEE Trans. On Magnetics*, 38(5):2295–97, Sept. 2002.

[44] A. Orlitsky, Interactive communication of balanced distributions and of correlated files, *SIAM Journal on Discrete Mathematics*, vol. 6(4), pp. 548–564, Nov. 1993.

[45] A. Orlitsky, K. Viswanathan, and J. Zhang, “Stopping set distribution of LDPC code ensambles,” *IEEE Transactions on Information Theory* vol. 51(3), pp. 929–53, March 2005.

[46] W. W. Peterson and E. J. Weldon, *Error Correcting Codes*, MIT Press 1972.

[47] O. Ramare and R. Rumely, “Primes in arithmetic progressions”, *Mathematics of Computation*, vol. 65, No. 213, pp. 397 –425, Jan. 1996.

[48] T. Richardson, “Error floors of LDPC codes”, Proceedings of the Allerton Conference on Communications, Control and Computing, 2003.

[49] T. Richardson and R. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding”, *IEEE Transactions on Information Theory*”, Vol. 47. no. 2, pp. 599–618, Feb. 2001.

[50] J. B. Rosser and L. Schoenfeld, “Approximate formulas for some functions of prime numbers”, *Illinois Jour. Math.*, vol. 6(1), pp. 64–94, March 1962.

[51] N. J. A. Sloane, ”On single deletion correcting codes”, 2000. Available at http://www.research.att.com/˜njas/doc/dijen.pdf

[52] I. Soprounov, “A short proof of the prime number theorem for arithmetic progressions”, available online at http://www.math.umass.edu/ isoprou/pdf/primes.pdf

[53] J. .J. Stiffler, "Comma free error correcting codes", *IEEE Trans. Inform. Theory*, 11:107–112, 1965.

[54] T. G. Swart and H.C. Ferreira, ”A note on double insertion/deletion correcting codes”, *IEEE Transactions on Information Theory* vol. 49(1), pp. 269–273, Jan. 2003.

[55] G. Tenengolts, "Nonbinary codes correcting single deletion or insertion", *IEEE Trans. Inform. Theory*, 30:766–769, Sept. 1984.

[56] J. D. Ullman. "Near optimal single synchronization error correcting code", *IEEE Trans. Inform. Theory*, 12:418–424, 1966.

[57] R. R. Varshamov and G.M. Tenengolts, "Codes which correct single asymmetric errors," *Avtomatika i Telemehkanika*, vol. 26, no. 2, pp. 288–292, 1965.

[58] B. Vasic and E. Kurtas, ”Coding and Signal Processing for Magnetic Recording Systems”, CRC press, 2005.

[59] A. Weil, ”Numbers of solutions of equations in finite fields”, in *Bull. Amer. Math. Soc*, vol. 50, pp. 497–508, 1949.

[60] S.B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, 1995.

[61] K. Yang and T. Helleseth, "On the minimum distance of array codes as LDPC codes", *IEEE Trans. Inform. Theory*, vol. 49(12), pp. 3268–3271, Dec. 2003.

[62] W. Zhang and A. Kavcic, “Optimal soft output detector for channels with intersymbol interference and timing errors”, *IEEE Trans. Magnetics*, 49(5):2555–557, Sept. 2003.

[63] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam and M. Wainwright, “Investigation of error floors of structured low-density parity-check codes via hardware simulation”, in *Proc. GLOBECOM 2006,* San Francisco, CA, Oct.-Nov. 2006.

[64] Z. Zhang, L. Dolecek, M. J. Wainwright, V. Anantharam, B. Nikolic, “Quantization effects of low-density parity-check decoders,” in Proceedings of *IEEE International Conference on Communications*, Glasgow UK, June 2007.

[65] Digital Video Broadcasting Project, available at http://www.dvb.org

[66] IEEE Standard 802.3an-2006, Sept. 2006 Available at http://ieeexplore.ieee.org/servlet/opac?punumber=4039890

[67] *Performance evaluation of low latency LDPC code*, 802.3an Standards Meeting, Sept. 2004. Available online at http://www.ieee802.org/3/an/public/sep04/seki10904.pdf