LDPC Codes from Partially Balanced Incomplete Block Designs

by

Elif Üsküplü

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Mathematics

KOÇ
UNIVERSITY

Date

LDPC Codes from Partially Balanced Incomplete Block Designs

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Elif Üsküplü

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assoc. Prof. Emine Şule Yazıcı Yuret

_____

Prof. Selda Küçükçifçi

_____

Asst. Prof. Emre Kolotoğlu

Date: _____

ABSTRACT

This thesis is a survey on a combinatorial construction of low-density parity-check (LDPC) codes using difference covering arrays. Since Gallager built LDPC codes for the first time, these codes have attracted a lot of attention since they provide ease in error correction algorithms. Therefore, researchers have developed various methods of designing LDPC codes. Initially, such codes were obtained through random processes. Then, LDPC codes were started to be constructed more structurely by combinatorial methods. In this study, one of these methods was discussed. After providing the necessary information to build and analyze the code, the results obtained on error correction performances were presented. An improvement on the construction technique has been suggested.

ÖZETÇE

Bu tez, fark kümelerinden elde edilen düşük yoğunluklu parite kontrol (DYPK) kodları üzerine bir araştırma niteliğindedir. Gallager'in DYPK kodlarını ilk kez inşa etmesinden bu yana, bu kodlar, hata düzeltme algoritmalarında kolaylık sağladığı için oldukça dikkat çekmiştir. Araştırmacılar, bu nedenle DYPK kodların çeşitli tasarım metotlarını geliştirmiştir. Başlangıçta, bu tip kodlar rassal süreçler sonucu elde ediliyordu, fakat daha sonraları kombinatoryel metotlar kullanılarak da üretilmeye başlandı. Bu çalışmamızda, bu metotlardan biri ele alınmıştır. Kodu inşa etmek ve analiz etmek için gerekli bilgiler verildikten sonra, hata düzeltme performansları hakkında elde edilen sonuçlar sunulmuştur. Ayrıca daha iyi bir kod elde etmek için bahsi geçen inşada bir iyileştirme metodu da önerilmiştir.

ACKNOWLEDGMENTS

I would like to thank to my supervisor Associate Professor Emine Şule Yazıcı Yuret not only for her enlightening comments, but also for her patience, refinement and temperate approaches during our studies. I thank to Professor Selda Küçükçifçi for her support. Without their endless support, guidance and patience, I would have not finished this thesis.

I also thank Associate Professor Emre Kolotoğlu for his inputs as committee member.

I thank Professor Ali Nesin, Associate Professors Sinan Unver, Kemal Ilgar Eroğlu, and Sonat Süer for the inspiration they have given to me in studying and teaching mathematics.

I thank Ayçin İplikçi and Oğuz Doğan for their friendship and wise advises.

I finally thank my husband Yasin Emre Üsküplü and my beloved family for their existence in my life and their endless support.

TABLE OF CONTENTS

NOMENCLATURE

LIST OF SYMBOLS/ABBREVIATIONS.

AWGN	Additive white Gaussian noise channel.
BSC	Binary symmetric channel.
BER	Bir error rate.
FER	Frame/block error rate.
E_b/N_0	Signal-to-noise ratio.
$\max f$	The points of the domain of a fuction f at which the function values are maximized.
LDPC	Low-density parity-check code.
QC-LDPC	Quasi-cyclic low-density parity-check code.
DCA	Difference covering array.
SDCA	Standard difference covering array.
PBIBD	Partially balanced incomplete block design.

LIST OF FIGURES

Chapter 1

INTRODUCTION

This thesis presents an analysis of a type of linear error-correcting codes, *Low Density Parity Check*(LDPC) codes. Any code can be uniquely determined by its parity-check matrix. Simply, an LDPC code is a linear code with few number of 1's in its parity-check matrix. The parity-check matrix of an LDPC code can be produced either in a structured way [13] or in an unstructured way [9]. The latter ones are generally obtained pseudorandomly through computer searches. However, in this case, it is difficult to determine their code rates or minimum distance which are main properties of a code. As for the structured LDPC codes, there are a variety of approaches. Some of those uses geometric approaches [12], and some of those uses combinatorial designs [3].

In this thesis, we study LDPC codes which are constructed by difference covering arrays (DCA) [7]. We illustrate some advantages of these codes over other structured LDPC codes. For example, the code rate and the minimum distance of the proposed code are given explicitly. In the next chapter, we present the basics of coding theory so that we comprehend the nature of LDPC codes. We also provide some criteria for achieving a *good* LDPC code. For example, in general, the higher the minimum distance of a linear code is, the better the code performs. We will give some techniques to determine the minimum distance of an LDPC code, and provide a lower bound. Moreover, a decoding algorithm will be provided for these codes to simulate their performances. In the third chapter, we give preliminary definitions in the combinatorial design theory and provide some basics results which are related to our construction. The construction of LDPC codes from PBIBDs that are constructed by difference covering arrays are then given in the last chapter. We also provide a performance analysis of the proposed LDPC codes in this chapter.

Chapter 2

CODING THEORY

This chapter will introduce the mathematical aspects of *error-correcting code*. To begin with, let us describe a typical communication system. There is information coming from some source, and it is transmitted over a noisy channel to a receiver. Our task is just communicating with as few error as possible.

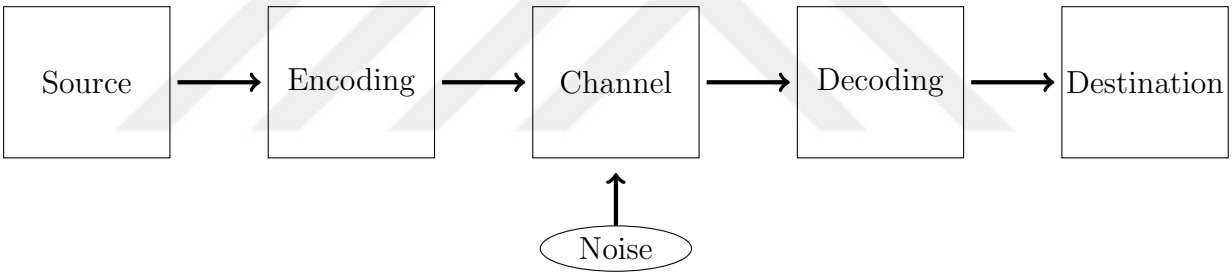


Figure 2.1: A Communication System

The idea behind error-correction can be understood in the following example. Suppose Alice wants to transmit meaningful words in English to Bob over a noisy channel. We assume that the error probability is less than $\frac{1}{2}$. Since the channel is not perfect, she sends each letter three times. For example, she wants to transmit the word "MERIT", but Bob receives the following message

MMA EEE RUR III QTT.

After dividing these into components of three letters, he can predict the message that she sent as "MERIT" because in the first, third, and last components, there are two repeated Ms, Rs, and Ts, respectively, and in the second and fourth components, there are three repeated Es and Is. Consequently, an error in a long word is recognized and corrected because the word is changed into something that is close to

the transmitted word with respect to alphabetical order. This is the principle of redundancy. According to this principle, we transmit a message longer than necessary for the information transmitted. An error-correcting code is based on this principle. The error-correction encoder maps each message of K lengths to a longer message of N lengths. The transmitted message may be corrupted by the channel, but the error-correction decoder uses the added redundancy to determine the real message. The *code rate* $\frac{K}{N}$ defines the amount of redundancy.

In 1948 [14], Shannon showed that it is possible to transmit data with arbitrarily high reliability, over a noisy channel. It is important to comprehend Shannon's result in terms of the purpose of this article. We begin with a formal definition of a channel. To see the significance of Shannon's work, see Information Theory in Encyclopedia Britannica.

Definition 1. A *channel* is a triple (A, B, P), where A is the input alphabet, B is the output alphabet, and P is the set of channel probabilities. If $A = \{a_i : i = 1, \ldots, r\}$ and $B = \{b_j : j = 1, \ldots, s\}$, then the transmission behaviour of the channel is described by the probabilities in $P = \{P(b_j|a_i) : i \in \{1, \ldots, r\}, j \in \{1, \ldots, s\}\}$, where $P(b_j|a_i)$ is the probability that the output symbol b_j will be received if the input symbol a_i is transmitted.

A channel is called a *binary-input channel* if $|A| = 2$. As convention, we take $A = \{0, 1\}$ in this case. Also, a channel is *memoryless* if the output of the channel at any time instant does not depend on previously transmitted symbols. Formally, if a sequence of N symbols $a = [a_1, \ldots, a_N]$ is transmitted and a sequence of N symbols $b = [b_1, \ldots, b_N]$ is received, then

$$p(b|a) = \prod_{i=1}^{N} p(b_i|a_i).$$

Throughout the article, all channels that we consider will be binary input memoryless channels.

Example 1. *The Binary Symmetric Channel* (BSC) is a channel with $A = B = \{0, 1\}$

and the channel transition probabilities are

$$\begin{aligned} p(0|1) &= \epsilon \\ p(0|0) &= 1-\epsilon \\ p(1|1) &= 1-\epsilon \\ p(1|0) &= \epsilon \end{aligned}$$

The parameter $\epsilon \in [0,1]$ is called *crossover probability* of the channel.

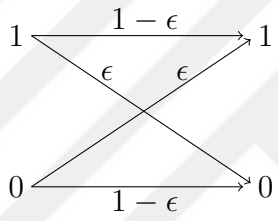


Figure 2.2: Binary Symmetric Channel

The uncertainty of the channel changes with the value of ϵ. For instance, if $\epsilon = 0$, then the channel is noiseless. Interestingly enough, the BSC with $\epsilon = 1$ is also a perfect (noiseless) channel. In this case, we may regard all 1s as 0s and all 0s as 1s. On the other hand, the uncertainty is maximized when $\epsilon = 1/2$ because all bits are corrupted with the same probability, and it is impossible to decide which bit was transmitted. The BSC is used mostly in information theory because this is the simplest channel with errors, and analyzing over a BSC gives a good estimate of the general problem [4].

Example 2. *The Binary-Input Additive White Gaussian Noise* (AWGN) channel can be described by the equation $y_i = x_i + z_i$ where $x_i \in \{-1, +1\}$ is the ith transmitted symbol, y_i is the ith received symbol, and z_i is the additive noise sampled from a Gaussian random variable with mean 0 and variance σ^2. The probability density function for z is

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-z^2/2\sigma^2}.$$

Conventionally [11], when we transmit a binary codeword on the AWGN channel, the codeword bits 0, 1 are mapped to 1, -1 respectively.

The AWGN channel is a good model for real-world channels such as wired and wireless telephone channels and satellite links [4].

In his paper "A Mathematical Theory of Communication", Claude Shannon introduced the revolutionary notion of *information entropy* [14]. Roughly speaking, it is the amount of uncertainty in given information. Also, the capacity of a channel defines the maximum amount of information that can be transmitted through the channel. Now, Shannon proved that, provided that the code rate of transmission R is less than the channel's capacity, there exists an error correcting code that will achieve an arbitrary low probability of error despite the noisy channel. However, the proof is not constructive. Therefore, coding theorists have tried to construct such codes since 1948. In the remaining part of this chapter, we will introduce one of the most studied types of such codes.

2.1 Linear Codes

A *binary code* $\mathcal{C}$ is a set of finite $\{0,1\}-$sequences. We say $\mathcal{C}$ is a *block code* if each sequence in $\mathcal{C}$ has the same length n. In this case, the elements of $\mathcal{C}$ are called the *codewords*. Thus, we can say that a block code $\mathcal{C}$ of length n is a subset of $\{0,1\}^n$.

Definition 2. If $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, then the Hamming distance $d(\mathbf{x}, \mathbf{y})$ of $\mathbf{x}$ and $\mathbf{y}$ is defined by $d(\mathbf{x}, \mathbf{y}) := |\{i : 1 \leq i \leq n, x_i \neq y_i\}|$.

The Hamming weight $w(\mathbf{x})$ of the codeword $\mathbf{x}$ defined by $w(\mathbf{x}) = d(\mathbf{x}, \mathbf{0})$.

The *minimum distance* d_{min} of a block code $\mathcal{C}$ is $\min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}\}$.

The *code rate* R of $\mathcal{C}$ is $\frac{log_2|\mathcal{C}|}{n}$.

Let us consider $\{0,1\}^n$ as a vector space over the field $GF(2)$. Then a block code is said to be linear if its codewords form a vector space over $GF(2)$.

Definition 3. A binary linear block code $\mathcal{C}$ is a linear subspace of $\{0,1\}^n$. If $\mathcal{C}$ has dimension k, then it is called (n,k) code. In this case, we have $|\mathcal{C}| = 2^k$.

For an (n, k) code, the code rate is $\frac{log_2|\mathcal{C}|}{n} = \frac{k}{n}$ as we expected.

Note that in a binary linear block code, it is easy to see that $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} + \mathbf{y})$ for any codewords $\mathbf{x}$ and $\mathbf{y}$. Therefore, the minimum distance d_{min} is equal to the minimum weight of nonzero codewords. The code's minimum distance is important in determining its error correction performance as we will discuss later. Therefore, a binary linear block code $\mathcal{C}$ is determined by three parameters n, k, d_{min} where n is the codelength, k is the code dimension, and d_{min} is the minimum distance.

From now on, we will assume that all codes that we consider will be binary linear block codes. To reduce the notation, we will say "linear code" instead of binary linear block code.

Definition 4. A *generator matrix* G for a (n, k) linear code $\mathcal{C}$ is a $k \times n$ matrix for which the rows are basis of $\mathcal{C}$.

It is not difficult to see that if G is a generator matrix for a (n, k) linear code $\mathcal{C}$, then we obtain $\mathcal{C} = \{aG : a \in \{0, 1\}^k\}$.

Example 3. Let $\mathcal{C} = \{0000, 0001, 1000, 0110, 1001, 0111, 1110, 1111\}$. It is a $(4, 3)$ linear code and $\{0110, 1000, 0001\}$ is a basis for this code. Thus, the matrix

$$G = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is a generator matrix for $\mathcal{C}$. We can obtain the code again by using the matrix.

$$[000]G = [0000] \qquad [001]G = [0001] \qquad [010]G = [1000] \qquad [100]G = [0110]$$
$$[011]G = [1001] \qquad [101]G = [0111] \qquad [110]G = [1110] \qquad [111]G = [1111]$$

Definition 5. A *parity check matrix* H for a (n, k) linear code $\mathcal{C}$ is a $(n - k) \times n$ matrix such that

$$\mathbf{x} \in \mathcal{C} \iff \mathbf{x}H^T = 0.$$

The parity check matrix of a linear code is useful for detecting errors. Suppose the codeword $\mathbf{x}$ goes to the binary vector $\mathbf{y}$ over a noisy channel. If $\mathbf{y}H^T \neq 0$, then we conclude that $\mathbf{y}$ is not a codeword, and so there must be at least one flipped bit. Furthermore, a parity check matrix gives information about the minimum distance of a code. When we try to calculate precisely the minimum distance of a code whose code length is short, the following theorem is quite useful.

Theorem 1. *Let $\mathcal{C}$ be a linear code with a parity check matrix H. The minimum distance d_{min} of $\mathcal{C}$ is equal to the smallest positive number of columns of H which are linearly dependent. Namely, all combinations of $d_{min} - 1$ columns are linearly independent, and there is some set of d_{min} columns which are linearly dependent. [11]*

Proof. Let $h_1, h_2, \ldots, h_n$ be the columns of H. Since $\mathbf{c}H^T = 0$ for any codeword $\mathbf{c} \in \mathcal{C}$, we have

$$0 = c_1h_1 + c_2h_2 + \ldots + c_nh_n.$$

Let c be the codeword of smallest weight. Recall that, for a linear code, the minimum distance d_{min} is equal to the minimum weight of nonzero codewords. Thus, c has nonzero positions only at indices $i_1, i_2, \ldots, i_{d_{min}}$. Then

$$c_{i_1}h_{i_1} + c_{i_2}h_{i_2} + \ldots + c_{i_{d_{min}}}h_{i_{d_{min}}} = 0.$$

Clearly, these columns are linearly dependent.

On the other hand, if there were a linearly dependent set of $u < d_{min}$ columns of H, there would be a codeword in $\mathcal{C}$ of weight u.

QED

The minimum distance of the code gives a bound for error-detecting capability of the code. Consider a linear code $\mathcal{C}$ with the minimum distance d_{min}. If the codeword $\mathbf{x}$ is received as $\mathbf{y}$ over a noisy channel and there are t many flipped bits, we obtain $d(\mathbf{x}, \mathbf{y}) = t$. Therefore, we detect an error if we obtain $t < d_{min}$. In this case, $\mathbf{y}$ cannot be a valid codeword. In general, a code with d_{min} can detect t bit-flipping

errors whenever $t < d_{min}$. The minimum distance of the code also gives a bound for error-correcting capability of the code.

One of the well-known error correction methods is the maximum likelihood (MLD) decoder. Roughly speaking, the MLD decoder always chooses the codeword that is most likely to have produced the received code. Let us define formally.

Definition 6. Let $\mathcal{C}$ be a linear code and (X, Y, P) be a binary input/output channel. If $\mathbf{y}$ is the received vector after the transmission over the channel, the MLD decoder returns the decoded codeword $\hat{\mathbf{x}}$ according to the rule

$$\hat{\mathbf{x}} \in \max_{\mathbf{c}\in\mathcal{C}} p(\mathbf{y}|\mathbf{c}) \tag{2.1}$$

That is, $\hat{\mathbf{x}}$ should be a code word satisfying the maximum value of $p(\mathbf{y}|\mathbf{c})$ on $\mathcal{C}$. Using a specific channel, we will derive a criterion about the error-correcting capability of a linear code. Let (A, B, P) be the binary symmetric channel. If the crossover probability ϵ is $> 1/2$, we can easily convert the channel into another BSC with $\epsilon < 1/2$. Without loss of generality, we may assume $\epsilon < 1/2$. Now, by Equation 2.1, we obtain

$$\max_{\mathbf{c}\in\mathcal{C}} p(\mathbf{y}|\mathbf{c}) = \max_{\mathbf{c}\in\mathcal{C}} \epsilon^{d(\mathbf{y},\mathbf{c})}(1-\epsilon)^{n-d(\mathbf{y},\mathbf{c})}.$$

Since $1 - \epsilon > 1/2$, the maximum is achieved for the smallest value of $d(\mathbf{y}, \mathbf{c})$. Therefore, the MLD in this case, is equivalent to choosing the codeword closest in Hamming distance to $\mathbf{y}$. It is easy to see that the MLD will correct t many bit-flipping error for any $t \leq \lfloor \frac{d_{min}-1}{2} \rfloor$. Indeed, if the codeword $\mathbf{x}$ was sent and the received code vector is $\mathbf{y}$ with t bit flips, then ML decoder choose $\mathbf{x}$ correctly. If there is another codeword $\mathbf{x}' \in \mathcal{C}$ such that $d(\mathbf{x}', \mathbf{y}) \leq t$, then by the triangle inequality, we obtain

$$d(\mathbf{x}, \mathbf{x}') \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{x}') \leq 2t < d_{min}.$$

This contradiction implies that there is only one codeword closest to $\mathbf{y}$.

As a result, any linear code $\mathcal{C}$ with minimum distance d_{min} can detect t bit flips whenever $t < d_{min}$, and it can correct t bit flips whenever $t \leq \lfloor \frac{d_{min}-1}{2} \rfloor$.

Note that for a channel with non-binary output (like AWGN) symbols, the Hamming distance is replaced by the Euclidean distance given below [11]

$$d(\mathbf{x}, \mathbf{y}) := \mathbf{x} - \mathbf{y} = \sum_i |x_i - y_i|.$$

As for performance measurement, one can easily observe that unless the channel is completely noise-free, there may be wrongly decoded codewords. We can measure the rate of such errors.

Definition 7. The *frame error rate* (FER) for a given decoder on a given channel is the number of wrongly decoded codewords as a fraction of the total number of decoded codewords. The *ber error rate* (BER) is the number of wrongly decoded codeword bits as a fraction of the total number of decoded codeword bits.

The FER and BER values of a linear code $\mathcal{C}$ indicate the error-correction capability of the code. For example, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two codes such that in a certain noise level, their FER results are 10^{-3} and 10^{-6}, respectively. Then it is obviously true that the latter code shows better performance because the probability of wrong decoding is smaller for the latter code $\mathcal{C}_2$.

2.2 Low Density Parity Check Codes

Recall that any linear code $\mathcal{C}$ has a parity check matrix H such that $c \in \mathcal{C}$ if and only if $cH^T = 0$. If H contains only a very small number of nonzero entries, then the corresponding code $\mathcal{C}$ is called *low density parity check* (LDPC) code.

LDPC codes were developed by Robert G. Gallager in his doctoral dissertation in 1962. Thus, LDPC codes are also known as Gallager codes. After their invention, they were largely forgotten because they were thought to be impractical. They were rediscovered by MacKay and Neal. They indicated that the sparsity of the parity check matrix is key property that allows for the decoding algorithm efficiency of LDPC codes. For details about these codes, see [1].

Before giving some important results about LDPC codes, let us provide more intuition about parity check matrices.

Example 4. $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$

Let $\mathcal{C}$ be linear code with parity check matrix H. Let $\mathbf{c} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix}$ be code vector in $\{0,1\}^5$. We know $c \in \mathcal{C}$ if and only if $cH^T = 0$. This gives three different equations over $GF(2)$:

$$\begin{aligned} c_1 + c_5 &= 0 \\ c_2 + c_4 &= 0 \\ c_3 + c_4 + c_5 &= 0 \end{aligned}$$

These equations are called parity-check equations. From this perspective, each row of H corresponds to a parity check equation and each column of H corresponds to a codeword bit. We say $\mathbf{c}$ is a valid codeword if it satisfies all parity-check equations.

An LDPC code parity check matrix H is called (w_c, w_r) regular if each column of H has Hamming weight w_c, and each row of H has Hamming weight w_r. If H is regular with size $m \times n$, then $mw_r = nw_c$.

Note that although we took H as $(n-k)\times n$ matrix, it is not an essential condition. Indeed, if H is $m \times n$ parity-check matrix for a (n, k) linear code, then only $n-k$ of the rows will be linearly independent. Therefore, we have $n-k \leq m$ and $n-k = rank_2 H$ where $rank_2$ stands for taking rank over $GF(2)$. Also, the code rate is given by $\frac{n-rank_2 H}{n}$. For the regular case, if H has full rank, then we obtain

$$\frac{k}{n} = \frac{n-m}{n} = \frac{n - n\frac{w_c}{w_r}}{n} = 1 - \frac{w_c}{w_r}$$

In 1981, Tanner proposed the graphical representation of LDPC codes using bipartite graphs [16]. Such a bipartite graph is called Tanner graph. The graph consists of two sets of nodes: N nodes for the codeword bits (called bit nodes) and m nodes for the parity-check equations (called check nodes) where H is $m \times N$ matrix. A bit node and a check node are adjacent to each other if that bit is included in the corresponding equation. Clearly, the number of edges in the Tanner graph is the number of 1s in the parity-check matrix.

Example 5. The Tanner graph of the parity check matrix in the previous example is given in Figure 2.3.

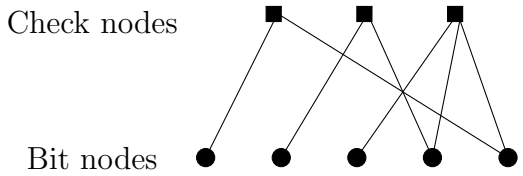


Figure 2.3: Tanner Graph

Now, we will construct a special type of LDPC codes. This type is called quasi-cyclic.

Definition 8. An LDPC code is quasi-cyclic (QC-LDPC) if the parity-check matrix H can be written as a $1 \times L$ array of $z \times z$ circulants given by $H = \begin{bmatrix} H_1 & H_2 & H_3 & \dots & H_L \end{bmatrix}$ where a circulant H_i is defined as a square matrix such that every row is a cyclic shift of the row above it.

Example 6. $H = \left[\begin{array}{cccc|cccc|cccc} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$ produces a quasi-cyclic LDPC code.

In [7], the quasi-cyclic codes are defined in a more general manner. We will use this definition.

Definition 9. An LDPC code is quasi-cyclic if the parity-check matrix H can be written as a $1 \times L$ array of $tz \times z$ circulants given by $H = \begin{bmatrix} H_1 & H_2 & H_3 & \dots & H_L \end{bmatrix}$ where a circulant H_i partition the columns into sub-blocks of length z such that each row is a cyclic shift of the previous row. It is easy to observe that each circulant H_i is described by its first column. In the new definition, the only change is that H_i's consist of t pieces $z \times z$ circulant matrix. In Example 7, we take $t = 2$ and $z = 4$.

Example 7. $H =$

$$\left[\begin{array}{cccc|cccc|cccc} 0&0&0&1&0&0&1&0&1&0&0&0 \\ 1&0&0&0&0&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&0&0&0&0&1&0 \\ 0&0&1&0&0&1&0&0&0&0&0&1 \\ \hline 1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1&0&0&0&0 \\ 0&0&0&1&1&0&0&0&0&0&0&0 \end{array}\right]$$

2.3 Minimum Distance

Let $\mathcal{C}$ be an LDPC code with parity check matrix H. Recall that the minimum distance d_{min} is the smallest Hamming distance between any pair of distinct codewords. By linearity, this is equivalent to the smallest Hamming weight of nonzero codeword in $\mathcal{C}$. In general, finding minimum distance of the code can require an exhaustive search. The exact calculation of the minimum distance is infeasible for long unstructured codes. Therefore, the general approach is to find a lower bound for d_{min}.

For a codeword $c \in \mathcal{C}$, define S_c to be set of location of nonzero codeword bits of c. For example, if $c = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$, then $S_c = \{1, 4, 5\}$. Also define A_k as the set of indices of codeword bits occuring in the parity check equation e_k. For example, if the equation e_k is $c_1 + c_4 + c_6 = 0$, then $A_k = \{1, 4, 6\}$.

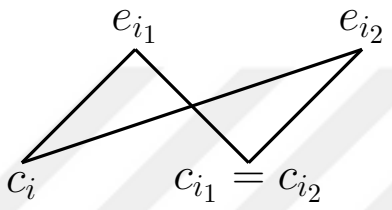
Now, since $c \in \mathcal{C}$ if and only if c satisfies all parity check equations, we can make the following observation

$$|S_c \cap A_k| \text{ is even.} \tag{2.2}$$

Indeed, if $|S_c \cap A_k|$ is odd, then adding these bits modulo 2 results in 1, which means c does not satisfy the equation e_k. This is a contradiction to $c \in \mathcal{C}$. Note also that $|S_c| = w(c)$. Thus, $d_{min} = \min\{|S_c| : c \in \mathcal{C} \setminus \{0\}\}$.

Let us consider (w_c, w_r) regular LDPC code $\mathcal{C}$ with parity check matrix H and the Tanner graph T, and let $c \in \mathcal{C}$ be nonzero. For $i \in S_c$, the codeword bit c_i occurs

in w_c many parity check equations, say $e_{i_1}, e_{i_2}, \ldots, e_{i_{w_c}}$. By the observation 2.2, we obtain $|S_c \cap A_{i_j}|$ is even for $j \in \{1, \ldots, w_c\}$. This yields c has at least one other nonzero codeword bit from each of w_c many equations, say $c_{i_1}, c_{i_2}, \ldots, c_{i_{w_c}}$. Without loss of generality, suppose $c_{i_1} = c_{i_2}$. Then we obtain a subgraph in the Tanner graph like below



There can be such a subgraph only if the girth of T is four. However, if we assume that the girth of T is at least six (a bipartite graph does not have an odd cycle), then we conclude that there cannot be such a subgraph. This implies that S_c contains at least $1 + w_c$ elements. Therefore, the girth of the Tanner graph is at least six implies the result $d_{min} \geq 1 + w_c$.

Now, let us consider the codeword bit c_{i_1}. Except for the equation e_{i_1}, the codeword bit c_{i_1} occurs in $w_c - 1$ equations, say $e_{i_{1_1}}, e_{i_{1_2}}, \ldots, e_{i_{1_{w_c-1}}}$. By the argument 2.2, c has nonzero codeword bits other than c_{i_1}, say $c_{i_{1_1}}, c_{i_{1_2}}, \ldots, c_{i_{1_{w_c-1}}}$.

As long as we assume the girth of T is at least six, we know that $\{i, i_1, \ldots, i_{w_c}\}$ are all different. Similarly, the elements in $\{i_j, i_{j_1}, \ldots, i_{j_{w_c-1}}\}$ are all different. If we want to increase the minimum distance, we need to be sure that these new codeword bits are differet from old ones. If $i_{t_k} = i_s$ for $t, s \in \{1, \ldots, w_c\}$ and $k \in \{1, \ldots, w_c - 1\}$, this yields a six cycle in the Tanner graph. If $i_{t_k} = i_{s_l}$ for $t, s \in \{1, \ldots, w_c\}$ and $k, l \in \{1, \ldots, w_c - 1\}$, then this yields an eight cycle in T. If we omit these cases and assume the girth is at least 10, then all codeword bits are different from each other. Therefore, we obtain at least $1 + w_c + w_c(w_c - 1)$ elements in S_c. Thus, in this case, we get $d_{min} \geq 1 + w_c + w_c(w_c - 1)$.

Actually, this lower bound can be generalized as follows

$$d_{min} \geq \begin{cases} 1 + w_c + w_c(w_c - 1) + \ldots + w_c(w_c - 1)^{\frac{g-6}{4}} & , g = 4k + 2 \\ 1 + w_c + w_c(w_c - 1) + \ldots + w_c(w_c - 1)^{\frac{g-8}{4}} & , g = 4k \end{cases} \tag{2.3}$$

where g is the girth of the Tanner graph. This lower bound is proposed and proved using a recursive approach by Tanner [16]. Therefore, for regular LDPC codes, the greater the column weight or the code girth is, the greater the minimum distance is.

2.4 Decoding Algorithm

When we transmit a binary data over a noisy channel, some of the codeword bits may be flipped. In this case, the task of the decoder is to detect flipped bits and, if possible, to correct them. Thanks to the parity check matrix, it is easy to detect errors. However, as for error correction, direct comparison of the received vector with every other codeword in the code is feasible only when the codelength is small. Therefore, the maximum likelihood decoder is not always useful.

There are many way of decoding, other than maximum likelihood decoding, which can perform very well for LDPC codes. The algorithms used to decode LDPC codes are generally called message-passing algorithms, since they operate by passing messages along the edges of a Tanner graph. For example, in bit-flipping decoding, the messages are binary codeword bits, and in belief-propagation decoding, the messages are probabilities that indicate what the codeword bit could be. For the details, one can read [11]. We will only deal with the belief propagation decoding because we will simulate the code samples using this decoder. The decoding is also called *Sum-Product Decoding* (SPD). From now on, we will follow notations and definitions in [11] generally.

For a codeword bit c_i, *a posteriori probability* (APP) is

$$P_i = \{c_i = 1 | \underbrace{\text{all parity-check equations are satisfied}}_{\text{the event S}}\}.$$

The intrinsic probability P_i^{int} is the original bit probability independent of the knowledge of the code constraints, and the extrinsic probability P_i^{ext} is what has been learnt from the event S.

The sum-product algorithm iteratively computes an approximation of the APP value for each codeword bit. Initially, we know only P_i^{int} values, and extrinsic infor-

mation is gained from the parity-check equations. The value P_i^{ext} obtained in one iteration is used as a priori information for the subsequent iterations.

As we said already, the SPD is a message-passing algorithm. There are messages passing between check nodes and bit nodes. The extrinsic message $E_{j,i}$ from check node j to bit node i, or equivalently, the extrinsic probability of a codeword bit i from the jth parity-check equation needs to be calculated. It is determined by the probability that the parity-check equation is satisfied if the bit is a 1, namely, the probability that an odd number of the other codeword bits are a 1. It is given by[1]

$$P_{j,i}^{ext} = \frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} \left(1 - 2P_{j,i'}^{int}\right)$$

where B_j is the set of column locations of the bits in the jth parity-check equation of the code, and $P_{j,i}^{int}$ is the intrinsic information sending from bit node i to check node j, that is, the current estimate of the probability $c_i' = 1$. Therefore, the probability that the parity-check equation is satisfied if $c_i = 0$ is $1 - P_{j,i}^{ext}$. Now, we set the extrinsic information $E_{j,i}$ expressed as a log likelihood ratio:

$$E_{j,i} = LLR(P_{j,i}^{ext}) = \ln \frac{1 - P_{j,i}^{ext}}{P_{j,i}^{ext}}.$$

Note that log likelihood ratios are used the represent the metrics for a binary variable by a single value

$$LLR(x) = \ln \frac{p(x=0)}{p(x=1)}.$$

The sign of $LLR(x)$ provides a hard decision on x. Namely, if $LLR(x) > 0$, then we conclude $p(x=0) > p(x=1)$, so we take the value x as 0. Similarly, if $LLR(x) < 0$, we take the value of x as 1. Using the identity[2]

$$\tanh\left(\frac{1}{2}\ln\left(\frac{1-p}{p}\right)\right) = 1 - 2p \ ,$$

we get

$$LLR(P_{j,i}^{ext}) = \ln\left(\frac{1 + \prod_{i' \in B_j, i' \neq i} \tanh\left(M_{j,i'}/2\right)}{1 - \prod_{i' \in B_j, i' \neq i} \tanh\left(M_{j,i'}/2\right)}\right) \tag{2.4}$$

[1]The proof is given in the Appendix A.

[2]The proof is given in the Appendix B.

where $M_{j,i'} = LLR(P_{j,i'}^{int})$. Then, the LLR of the estimated APP of the ith bit at each iteration

$$LLR(P_i) = LLR(P_i^{int}) + \sum_{j \in A_i} E_{j,i}$$

where A_i is the set of row locations of the parity check equations which check on the ith bit of the code.

Now, let us analyze the algorithm.

Step 1. The initial message sent from bit node i to check node j is the LLR of the received signal y_i obtained by the channel properties. For an AWGN channel with signal-to-noise ratio E_b/N_0, this is

$$M_{i,j} = LLR(y_i) = 4y_i \frac{E_b}{N_0}.$$

Step 2. The extrinsic message $E_{j,i}$ from check node j to bit node i is given in the Equation 2.4.

Step 3. After the second step, the LLR values of the bit nodes changed. For hard decision, we should keep these values.

$$L_i = LLR(y_i) + \sum_{j \in A_i} E_{j,i}.$$

On the other hand, if the decoded vector is not a valid codeword after the first iteration, we should send a new message from the bit nodes to the check nodes:

$$M_{j,i} = LLR(y_i) + \sum_{j' \in A_i, j' \neq j} E_{j',i}.$$

Note that we precluded the term $E_{j,i}$ because we do not want to send the same message twice.

Step 4. We update the received vector $\hat{\mathbf{y}} = (\hat{y}_i)$ according to the following rule

$$\hat{y}_i = \begin{cases} 0 & \text{if } L_i > 0 \\ 1 & \text{if } L_i \leq 0 \end{cases}$$

If the new code vector satisfies the relation $\hat{\mathbf{y}}H^T = 0$, then it is a valid codeword, and the algorithm terminates. Otherwise, we operate a new iteration starting from Step 2 with new values of $M_{j,i}$ found in Step 3.

In some cases, the algorithm may create a loop and it does not terminate. This is because of the possible small cycles in the Tanner graph. To prevent loops, we determine the maximum number of iterations in the simulations.

Example 8. We will simulate the sum-product algorithm using an AWGN channel with $E_b/N_0 = 1.75$. Let us use the following parity check matrix

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

and suppose that we send the codeword $\hat{\mathbf{c}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$. We implemented algorithm via MATLAB[3]. After the channel transmission, the received vector is

$$\mathbf{y} = \begin{bmatrix} -1.0624 & -0.4612 & -1.3318 & -0.3566 & -0.3647 & 0.3287 \end{bmatrix}$$

The hard decision implies $\hat{\mathbf{y}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$ which is not a valid codeword because $\hat{\mathbf{y}}H^T \neq 0$.

Now, the initial message from bit nodes to check nodes is LLR values of $\mathbf{y}$ which is $LLR(y) = \begin{bmatrix} -3.7183 & -1.6143 & -4.6614 & -1.2483 & -1.2766 & 1.1505 \end{bmatrix}$. As in first step of the algorithm, we take $M_{j,i} = LLR(y_i)$. For example, the message sent from the first bit node to the first and the fourth check nodes is

$$M_{1,1} = M_{4,1} = LLR(y_1) = -3.7183.$$

Note that if ith bit does not appear in the jth parity check equation, then there is no message between them. Thus, we may assume $M_{j,i} = 0$ in this case. In the second

[3]A pseudo code of the algorithm is given in the Appendix C.

step, the extrinsic information that pass from check nodes to bit nodes are calculated. For example, the message sent from the first check node to the fifth bit node is

$$\begin{aligned} E_{1,5} &= \ln\left(\frac{1+\tanh\left(M_{1,1}/2\right)\tanh\left(M_{1,6}/2\right)}{1-\tanh\left(M_{1,1}/2\right)\tanh\left(M_{1,6}/2\right)}\right) \\ &= \ln\left(\frac{1+\tanh\left(-3.7183/2\right)\tanh\left(1.1505/2\right)}{1-\tanh\left(-3.7183/2\right)\tanh\left(1.1505/2\right)}\right) \\ &= \ln\left(\frac{1+(-0.9526)(0.5192)}{1-(-0.9526)(0.5192)}\right) \\ &= -1.0842. \end{aligned}$$

Similarly, we may assume $E_{j,i} = 0$ if ith bit does not appear in the jth parity check equation. By continuing the algorithm, either we converge to a valid codeword or the algorithm terminates after the maximum number of iterations. In the simulation, first iteration does not produce a valid codeword, so another iteration is needed. The details of the simulation is given below.

–Iteration 1–

$$LLR(y) = \begin{bmatrix} -3.7183 & -1.6143 & -4.6614 & -1.2483 & -1.2766 & 1.1505 \end{bmatrix}$$

$$M = \begin{bmatrix} -3.7183 & 0 & 0 & 0 & -1.2766 & 1.1505 \\ 0 & -1.6143 & 0 & -1.2483 & 0 & 1.1505 \\ 0 & 0 & -4.6614 & -1.2483 & -1.2766 & 0 \\ -3.7183 & -1.6143 & -4.6614 & 0 & 0 & 0 \end{bmatrix}$$

$$E = \begin{bmatrix} -0.6030 & 0 & 0 & 0 & -1.0842 & 1.1999 \\ 0 & -0.5920 & 0 & -0.7237 & 0 & 0.7770 \\ 0 & 0 & 0.6462 & 1.2459 & 1.2186 & 0 \\ 1.5698 & 3.3896 & 1.5040 & 0 & 0 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} -2.7515 & 1.1833 & -2.5112 & -0.7261 & -1.1422 & 3.1274 \end{bmatrix}$$

$$\hat{\mathbf{y}} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\hat{\mathbf{y}}H^T = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T \Longrightarrow \text{Continue}$$

–Iteration 2–

$$M = \begin{bmatrix} -2.1485 & 0 & 0 & 0 & -0.0580 & 1.9275 \\ 0 & 1.7753 & 0 & -0.0024 & 0 & 2.3504 \\ 0 & 0 & -3.1574 & -1.9720 & -2.3608 & 0 \\ -4.3213 & -2.2063 & -4.0152 & 0 & 0 & 0 \end{bmatrix}$$

$$E = \begin{bmatrix} -0.0433 & 0 & 0 & 0 & -1.3556 & 0.0459 \\ 0 & -0.0020 & 0 & 1.3449 & 0 & -0.0017 \\ 0 & 0 & 1.4675 & 1.9926 & 1.7112 & 0 \\ 2.0565 & 3.4637 & 2.0938 & 0 & 0 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} -1.7050 & 1.8474 & -1.1001 & 2.0893 & -0.9210 & 1.1947 \end{bmatrix}$$

$$\hat{\mathbf{y}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\hat{\mathbf{y}}H^T = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \Longrightarrow \text{Terminate}$$

In the next chapter, we will give the necessary information to construct a special type of LDPC codes. After giving the construction, we will simulate the proposed code over AWGN channel using Sum-Product Decoding. We record the FER and BER results for different noise levels. The results obtained by the simulation will indicate the performance of the code that we construct.

Chapter 3

DESIGN THEORY

In this chapter, we will briefly introduce the concepts of combinatorial design theory.

3.1 Balanced Incomplete Block Designs

We start with some basic definitions.

Definition 10. Let $v, k,$ and λ be positive integers satisfying $v > k \geq 2$. A $(v, k, \lambda)-$*balanced incomplete block design*, denoted by $(v, k, \lambda)-$BIBD, is a pair $(X, \mathcal{A})$ where

i. X is a set with v elements which are called points,

ii. $\mathcal{A}$ is a set of subsets of X and its elements are called blocks,

iii. Each block contains exactly k points, and

iiii. Every pair of distinct points from X is contained in exactly λ blocks.

Definition 11. Given a pair $(X, \mathcal{A})$ satisfying i, ii, and iii. in Definition 10, if the value of λ varies across the pairs of points in X, then we call it *partially balanced incomplete block design* (PBIBD).

When $v = k$, we can say that the design is *complete*. In some textbooks, a BIBD is defined with two additional parameters b and r, such that $|\mathcal{A}| = b$ and every point in X occurs in exactly r blocks. However, these two parameters are completely determined by the triple (v, k, λ). Hence, we can characterize a BIBD with only v, k and λ.

Lemma 1. *Let $(X, \mathcal{A})$ be a $(v, k, \lambda)-$BIBD. Then every point occurs in exactly r blocks where*

$$r = \frac{\lambda(v-1)}{k-1}.$$

Proof. Let $x \in X$ and r_x be the number of blocks that contain the point x. Let $I = \{(y, A) : x \neq y \in X \text{ and there exists } A \in \mathcal{A} \text{ such that } \{x, y\} \subset A\}$. We know that for any $y \in X$ different from x, there are λ blocks that contain x and y. Thus we have $|I| = \lambda(v-1)$. On the other hand, since x is contained in r_x blocks and each such block contains $k-1$ many elements different from x, we also obtain $|I| = r_x(k-1)$. This yields $r_x(k-1) = \lambda(v-1)$.

QED

Lemma 2. *Let $(X, \mathcal{A})$ be a $(v, k, \lambda)-$BIBD and r be as in the previous lemma. Then there are exactly b blocks in $\mathcal{A}$ where*

$$b = \frac{vr}{k}.$$

Proof. Assume $|\mathcal{A}| = b$. We want to achieve the equality in the statement. Let us define

$$\mathcal{C} = \{(x, A) : x \in X,\ A \in \mathcal{A},\ x \in A\}.$$

We will count the elements in $\mathcal{C}$ in two ways. Since each point is contained in r blocks, we obtain there are vr pairs (x, A). On the other hand, there are b blocks and each contain k points. Therefore, there are bk pairs (x, A). These yield $bk = vr$ as desired.

QED

These two lemmas give the necessary conditions for the existence of a BIBD.

Theorem 2. *Let $(X, \mathcal{A})$ be a $(v, k, \lambda)-$BIBD. Then $\lambda(v-1) = 0 \bmod (k-1)$ and $\lambda v(v-1) = 0 \bmod k(k-1)$.*

The necessary condition in Theorem 2 is not sufficient. For example, there is no (15,5,2)-BIBD even if the parameters satisfy the conditions in Theorem 2 (See the chapter *BIBDs with Small Block Size* written by Abel and Greig in [3]).

Definition 12. Let $(X, \mathcal{A})$ be a $(v, b, r, k, \lambda)-$BIBD where $X = \{x_1, \ldots, x_v\}$ and $\mathcal{A} = \{A_1, \ldots, A_b\}$. The *incidence matrix* of $(X, \mathcal{A})$ is the $v \times b$ matrix $H = [h_{ij}]$ defined by the rule

$$h_{ij} = \begin{cases} 1 & \text{if } x_i \in A_j \\ 0 & \text{if } x_i \notin A_j \end{cases}$$

The incidence matrix H of a $(v, b, r, k, \lambda)-$BIBD is a regular matrix. Namely, it has a constant column/row weight. Every column of H has exactly k many 1s, and every row of H has exactly r many 1s. Moreover, the dot product of the rows of H is λ. An incidence matrix of a PBIBD is defined in the same way.

Remark. As we will indicate in the next chapter, every incidence matrix can be thought as a parity-check matrix for a code. The proposed code will be constructed by the incidence matrix of a PBIBD.

3.2 Latin Squares

Let us begin with a real-life problem. Suppose that there is a family, and their grandmother is confined to bed. However, other family members must leave the house at certain times due to their duties. In weekdays, they should look after her according to a schedule. For the sake of fairness, they determine some rules about the schedule. There are five weekdays and each day is divided into 5 time slots.

i) Everybody should be responsible for the grandmother for a time slot every weekday.

ii) Everybody should be scheduled to a different slot every weekday.

This schedule requires a table as in Figure 3.1. If we want to achieve a suitable plan, we need to put each person, say A, B, C, D, and E, to the squares in such a way that each row and each column must contain all of these symbols. It is easy to observe that the plan below is admissible for the all family obeying the given rules.

Although our discussion is quite simple, the solution is related to a well-known object in combinatorics, which is a *Latin square*. It has a long history, but Euler was

	Mon	Tue	Wed	Thu	Fri
P1	A	B	C	D	E
P2	B	C	D	E	A
P3	C	D	E	A	B
P4	D	E	A	B	C
P5	E	A	B	C	D

Figure 3.1: Schedule

probably the first to define it using mathematical terminology and to investigate its properties [8]. After some motivation, we are ready to give a formal definition.

Definition 13. A *Latin square of order* n with entries from a set X of size n, is an $n \times n$ array L such that every element in X appears in each row and in each column exactly once.

1	3	0	2
2	1	3	0
3	0	2	1
0	2	1	3

Figure 3.2: A Latin square of order 4

In many cases, we will take $X = \{1, 2, \ldots, n\}$. When we define a combinatorial object, the first question that emerges is whether such an object exists or not. Unsurprisingly, we have a Latin square of order n for all $n \in \mathbb{N}$. One of the most trivial constructions is given by the addition table of $\mathbb{Z}_n$. For example, Figure 3.1 was created in this way, we just replaced the elements of $\mathbb{Z}_n$ with $\{A, B, C, D, E\}$

respectively. For understanding the properties of Latin squares, we first observe that an algebraic object is closely related to Latin squares.

Definition 14. Let X be a set of n elements, and $\circ : X \times X \to X$ be a binary operation on X. We say that $(X, \circ)$ is a *quasigroup* if for any $x, y \in X$, the equations $x \circ z = y$ and $z \circ x = y$ have unique solutions.

The following theorem gives the relation between a Latin square and a quasigroup.

Theorem 3. *Let X be a set with n elements, and $\circ$ be a binary operation on X. Then $(X, \circ)$ is a quasigroup if and only if its operation table is a Latin square.*

One of the major properties of pairs of Latin squares arose from the following problem.

> *The Euler Officer Problem.* Six officers from each of six different regiments are selected so that the six officers from each regiments are of six different ranks, the same six ranks being represented by each regiment. Is it possible to arrange these 36 officers in a 6×6 array so that each regiment and each rank is represented exactly once in each row and column of this array? [8]

If we label the regiments and the ranks with 1,2,3,4,5, and 6, then each officer in the array is represented by a pair (x, y) such that x denotes his regiment and y denotes his rank. Therefore, if there exists such an array, then its rows and columns contain all pairs in $X \times X$ where $X = \{1, 2, 3, 4, 5, 6\}$. In other words, the first coordinate of the array creates a Latin square, and the second coordinate does the same. If there are two such Latin squares, then we say that these are orthogonal.

Definition 15. Let L_1 and L_2 be two Latin squares of order n on the set of symbols X. We say these are *orthogonal* to each other if for every $x, y \in X$, there is a unique cell (i, j) such that $L_1(i, j) = x$ and $L_2(i, j) = y$. In other words, the superposition of L_1 and L_2 contains every pair in $X \times X$.

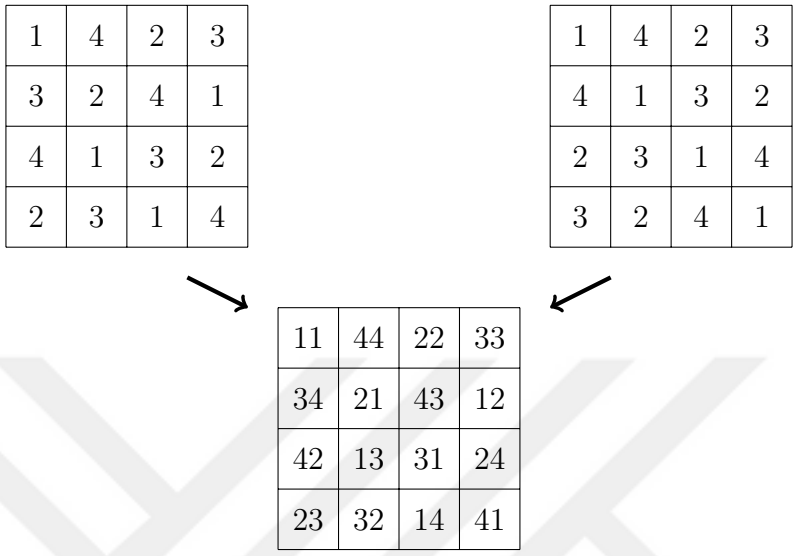


Figure 3.3: Two orthogonal Latin squares of order 4

Now, it is clear that in order to solve the Euler Officer Problem, we need to find a pair of orthogonal Latin squares of order 6. In 1782 when Euler studied the problem, he could not find such a pair. Since a trivial observation shows that there is no pair of orthogonal Latin squares of order 2, he made the following conjecture.

"*A pair of orthogonal Latin squares of order* n *exists if and only if* $n \neq 2 (\bmod 4)$."

The reason why he used the term 'if and only if', is that he was able to construct a pair for all other orders. In 1900, G. Tarry proved that Euler's problem has no solution, namely, there is no pair of orthogonal Latin squares of order 6. However, the proof contained brute force. In 1984, Doug Stinson gave a nonbrute force proof [15]. On the other hand, Euler's conjecture was wrong. There exists a pair of orthogonal Latin squares of all orders n except for 2 and 6. The disproof was given by Bose, Shrikhande, and Parker in 1960 [2]. We define a set of s-many Latin squares of order n, say $L_1, L_2, \ldots, L_s$, as *mutually orthogonal* Latin squares if each pair of latin squares are orthogonal in this set. We denote this set by MOLS(n), and let $N(n)$ be the maximum number of elements in MOLS(n). We know $N(2) = 0$ and $N(6) = 0$. Moreover, we have an upper bound on $N(n)$ for $n > 1$. It is easy to show that $N(n) \leq n - 1$ for $n > 1$. In some cases, this bound can be achieved, but we do not

know the exact value of $N(n)$ for general n.

A broader class of Latin squares can be obtained by changing the orthogonality condition. For two Latin squares L_1 and L_2 of order n on the set of symbols X, we say these are *pseudo-orthogonal* if given $\mathcal{O} = \{(L_1(i,j), L_2(i,j)) | 0 \geq i, j \geq n-1\}$, for all $x \in X$

$$|\{(x, L_2(i,j)) | (x, L_2(i,j)) \in \mathcal{O}\}| = n - 1.$$

This means that, each symbol in L_1 is paired with every symbol in L_2 exactly once, except for one symbol with which it is paired twice and one symbol with which it is not paired. This kind of Latin squares can be used to construct linear codes as in [7].

4	1	2	3
1	2	3	4
2	3	4	1
3	4	1	2

1	2	3	4
3	4	1	2
4	1	2	3
2	3	4	1

Figure 3.4: Two pseudo-orthogonal Latin squares of order 4

In the next chapter, we will construct some PBIBDs using difference covering arrays(DCA), whose incidence matrices will give us LDPC codes. Notice that same difference covering arrays could be used to construct pseudo-orthogonal Latin squares [5], and these then could be used to construct PBIBDs. However, we will construct PBIBDs using DCAs directly, instead of obtaining them by Latin squares.

Chapter 4

LDPC CODES BASED ON COMBINATORIAL DESIGNS

4.1 LDPC Codes from PBIBDs constructed by difference covering arrays

In this section, we will look at a combinatorial construction of QC-LDPC codes from partially balanced incomplete block designs (PBIBD) constructed using difference covering arrays [7]. After we give the construction and some properties of the proposed code, we will present its error correction performance.

Definition 16. Let $(G, e, \star)$ be an abelian group of order m. A *difference covering array* $\mathrm{DCA}(k, n; m)$ is an $n \times k$ matrix $Q = [q(i, j)]$ such that $q(i, j) \in G$ and for any distinct pair of columns $0 \leq j, j' \leq k-1$, the difference set

$$\Delta_{j,j'} = \{q(i, j) \star q(i, j')^{-1} : 0 \leq i \leq n-1\} = G.$$

Note that adding a constant vector to any row or any column does not change the set $\Delta_{j,j'}$. Similarly, permuting the rows does not alter the difference set, and hence we will assume that the first row is all 0's, and delete this row. Therefore, we will define a *standard* difference covering array of the group $\mathbb{Z}_{2n}$ with addition modulo $2n$ as a matrix Q that satisfies the following properties:

D1. The first column contains only 0s. Also, the remaining columns contain each entry of $\mathbb{Z}_{2n}$ precisely once. Assume the second column is equal to

$$[0\ 1\ 2\ 3\ \ldots\ 2n-1]^{\top}.$$

D2. For all pairs of distinct columns $j \neq 0$ and $j' \neq 0$, we have

$$\Delta_{j,j'} = \{q(i, j) - q(i, j') \bmod 2n : 0 \leq i \leq 2n-1\} = \mathbb{Z}_{2n} \setminus \{0\}.$$

Let us denote it by sDCA$(k, 2n; 2n)$. For any standard difference covering array, for $j \neq 0$ and $j' \neq 0$, we have $\Delta_{j,j'} = \{1, 2, \ldots, n, n, \ldots 2n-1\}$ (see [5] for the proof). Namely, the repeated difference is always n.

Example 9. The following matrices are examples of SDCA$(3, 6; 6)$ and SDCA$(3, 10; 10)$ respectively

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 5 & 0 & 2 & 4 \end{bmatrix}^\top \qquad Q' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 7 & 0 & 8 & 6 & 9 & 5 & 4 & 3 & 2 \end{bmatrix}^\top$$

For details and related results on difference covering arrays, see [17]. Existence of difference covering arrays for general k is not resolved completely. But it is known that when $k = 3$, the matrix $Q = [q(i, j)]$ where

$$q(i,j) = \begin{cases} 0, & j = 0 \\ i, & j = 1 \\ 2i+1, & j = 2 \text{ and } 0 \leq i \leq n-1 \\ 2(i-n), & j = 2 \text{ and } n \leq i \leq 2n-1 \end{cases} \tag{4.1}$$

forms a SDCA$(3, 2n; 2n)$. The matrix Q in the example above was constructed using this structure. Now, we will construct a LDPC code using SDCA$(3, 2n; 2n)$s in the following algorithm. In the paper [7], the code was constructed from SDCA$(3, 2n; 2n)$ obtained by Equation 4.1. However, any SDCA$(3, 2n; 2n)$ can be used in the construction.

Step 1. Let Q be a SDCA$(3, 2n; 2n)$, and define $SB_i = (0, i, q(i, 2))$ for $i \in \mathbb{Z}_{2n}$ and call them as starter blocks. If Q has the row $(0, n, 0)$, then delete the corresponding starter block. Otherwise, delete the starter block, say SB_d, where the difference value n occurs first. Thus, we obtain $2n - 1$ starter blocks.

Step 2. Construct the set of blocks B_{ia} for $0 \leq a \leq 2n - 1$ using the starter blocks

$$B_{ia} := \{a,\ [(i + a) \bmod 2n] + 2n,\ [(q(i, 2) + a) \bmod 2n] + 4n\}$$

where $(0, i, q(i, 2)) = SB_i$.

Step 3. Take the orbits O_i for $i \in \mathbb{Z}_{2n} \setminus \{n\}$ as the set of blocks B_{ia} that is

$$O_i = \{B_{ia} : 0 \leq a \leq 2n-1\}.$$

Define $\mathcal{B} = \bigcup O_i$. Then $(\mathbb{Z}_{6n}, \mathcal{B})$ gives a $(6n, 3)-$PBIBD with $\lambda \leq 1$.

Step 4. And finally, take the incidence matrix $H = [h(j,i)]$ where the columns are indexed by the blocks $B_{ia} \in \mathcal{B}$ and set

$$h(j, B_{ia}) = \begin{cases} 1, & \text{if } j \in B_{ia} \\ 0, & \text{otherwise} \end{cases} .$$

Example 10. Let $n = 3$ and consider the following SDCA$(3, 6; 6)$:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 4 & 0 & 3 & 2 \end{bmatrix}^{\top}$$

The starter blocks and their orbits are listed below

$SB_0 = (0,0,1)$	$SB_1 = (0,1,5)$	$SB_2 = (0,2,4)$	$SB_4 = (0,4,3)$	$SB_5 = (0,5,2)$

$O_0 =$	$O_1 =$	$O_2 =$
$\{B_{00} = \{0,6,13\},$	$\{B_{10} = \{0,7,17\},$	$\{B_{20} = \{0,8,16\},$
$B_{01} = \{1,7,14\},$	$B_{11} = \{1,8,12\},$	$B_{21} = \{1,9,17\},$
$B_{02} = \{2,8,15\},$	$B_{12} = \{2,9,13\},$	$B_{22} = \{2,10,12\},$
$B_{03} = \{3,9,16\},$	$B_{13} = \{3,10,14\},$	$B_{23} = \{3,11,13\},$
$B_{04} = \{4,10,17\},$	$B_{14} = \{4,11,15\},$	$B_{24} = \{4,6,14\},$
$B_{05} = \{5,11,12\}\}$	$B_{15} = \{5,6,16\}\}$	$B_{25} = \{5,7,15\}\}$

$O_4 =$	$O_5 =$
$\{B_{40} = \{0,10,15\},$	$\{B_{50} = \{0,11,14\}\},$
$B_{41} = \{1,11,16\},$	$B_{51} = \{1,6,15\},$
$B_{42} = \{2,6,17\},$	$B_{52} = \{2,7,16\},$
$B_{43} = \{3,7,12\},$	$B_{53} = \{3,8,17\},$
$B_{44} = \{4,8,13\},$	$B_{54} = \{4,9,12\},$
$B_{45} = \{5,9,14\}\}$	$B_{55} = \{5,10,13\}\}$

The incidence matrix of PBIBD on the points set $V = \mathbb{Z}_{18}$ with the set of blocks $\mathcal{B} = \bigcup O_i$ is

$$H = \left[\begin{array}{cccccc||cccccc||cccccc||cccccc||cccccc}
1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1\\
\hline\hline
1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&0\\
\hline\hline
0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0
\end{array}\right]$$

Now, the matrix H forms a parity-check matrix of a QC-LDPC code, say $\mathcal{C}$. When we take Q as defined in Equation 4.1, the following theorem was proved in [7].

Theorem 4. *Let H be the incidence matrix given by the algorithm. Then H is the parity-check matrix of a QC-LDPC code of length $4n^2 - 2n$, girth at least 6, rank $6n - 2$, the code rate $(4n^2 - 8n + 2)/(4n^2 - 2n)$ and minimum distance $d = 6$ when n is odd and $d = 4$ when n is even.*

We will give some major detailes of the proof in the following lemmas.

Lemma 3. *The LDPC code with the parity-check matrix H has girth at least 6.*

Proof. A four cycle in the Tanner graph is obtained by two columns in H such that they both have 1s at the same two rows like in Figure 4.1. If there is such a cycle in the graph of the code, then there exist two blocks of $\mathcal{B}$ intersecting in two elements, but this contradicts the fact that $\lambda_{y,z} \leq 1$ for all $y, z \in V = \mathbb{Z}_{6n}$. Since the Tanner graph is bipartite, it has no odd cycles. Thus, it has no cycles of length less than 6.

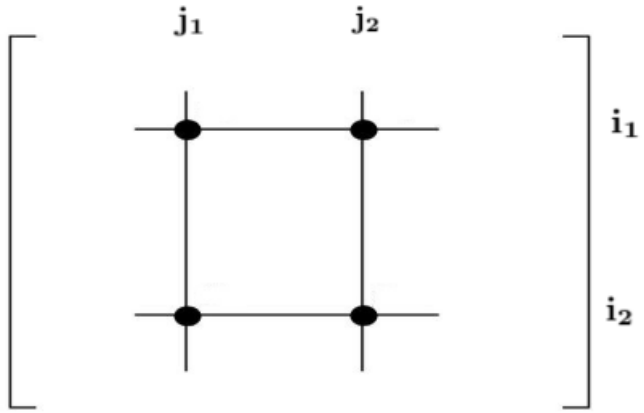


Figure 4.1: A four cycle in a parity-check matrix

QED

Lemma 4. *The rank of H is $6n - 2$.*

Proof. Since the size of H is $(6n) \times (4n^2 - 2n)$, the rank is at most $6n$. We will show firstly that the rank of H is at most $6n - 2$.

Recall that H comes from the blocks of $\mathcal{B}$ and each block contains exactly one element from each of the sets of points $V_1 = \{0, \ldots, 2n-1\}$, $V_2 = \{2n, \ldots, 4n-1\}$, and $V_3 = \{4n, \ldots, 6n-1\}$. Let $\{r_0, \ldots, r_{6n-1}\}$ be the set of rows of H.

Let R_i be the set of rows corresponding to the points from V_i. Since the column sum of H restricted to R_1 and R_2, for example, is 2, these rows are linearly dependent over $GF(2)$. Thus, any row in $R_1 \cup R_2$ can be written as the sum of other $4n - 1$ rows. The same deduction can be made for both $R_1 \cup R_3$ and $R_2 \cup R_3$. Since any set

of $6n-1$ rows must contain one of these unions, we conclude that any set of $6n-1$ rows are linearly dependent, namely, the rank is at most $6n-2$.

In order to show that the rank is exactly $6n-2$, it is enough to observe that the columns of H that correspond to following sets of blocks are linearly independent (See [7] for the details).

$$\begin{aligned}
C_1 &= \{\{a, a+2n, (a+1) \bmod 2n+4n\},\ a=0,\ldots,2n-1\},\\
C_2 &= \{\{b, b+1+2n, (b+3) \bmod 2n+4n\},\ b=0,\ldots,2n-2\},\\
C_3 &= \{\{c, c+2+2n, (c+5) \bmod 2n+4n\},\ c=0,\ldots,2n-4\},\\
C_4 &= \{\{0, 3n+1, 4n+2\}, \{1, 3n+2, 4n+3\}\}.
\end{aligned}$$

QED

By Lemma 4, we conclude that the rate of this code is

$$\frac{\text{code length} - rank_2 H}{\text{code length}} = \frac{(4n^2-2n)-(6n-2)}{4n^2-2n} = \frac{4n^2-8n+2}{4n^2-2n}.$$

In the last lemma, we will calculate the minimum distance of the proposed code.

Lemma 5. *The LDPC code with the parity-check matrix H obtained by the SDCA$(3, 2n; 2n)$ in Equation 4.1 has minimum distance 6 when n is odd and 4 when n is even.*

Proof. First, recall Equation 2.3 that if the code is 4-cycle free and its parity-check matrix has the constant column weight w_c, then the minimum distance is at least w_c+1. In our case, therefore, the minimum distance is at least 4.

By the observation above, if n is even number, it is enough to find a codeword of weight 4. In this case, the columns of H corresponding to the following blocks are linearly dependent

$\{0, 2n, 4n\}$, $\{0, 2n+n/2, 5n+1\}$,
$\{n-1, 2n, 5n+1\}$, $\{n-1, 2n+n/2, 4n+1\}$.

If n is an odd number, then the columns corresponding to the blocks

$\{0, 2n+1, 4n+3\}$, $\{0, 2n+2, 4n+5\}$,

$\{1, 2n+2, 4n+4\}$, $\{1, 3n+2, 4n+3\}$,

$\{2n-1, 2n+1, 4n+4\}$, $\{2n-1, 3n+2, 4n+5\}$,

are linearly dependent. Therefore, there exists a codeword of weight 6. Now, it remains to show that, in the case of odd n, there are no r linearly dependent columns for $r \leq 5$, we then prove the lemma thanks to Theorem 1. See [7] for the details.

QED

The main advantage of the proposed code in [7] is that we are able to give and verify explicit algebraic expressions for the rate of the code as well as its minimum distance. Moreover, these codes have high information(code) rate at lower code lengths. The codes we obtain achive rate ≥ 0.8 for code with length of 240 bits. As it can be observed from the Figure 4.2, for quite small values of n, we obtain a code with high rate.

n	Code length m	Code dim	Rate of code
5	90	62	0.688
6	132	98	0.742
7	182	142	0.780
8	240	194	0.808
9	306	254	0.830
10	380	322	0.847
11	462	398	0.861
12	552	482	0.873
13	650	574	0.883
14	756	674	0.891
15	870	782	0.899
16	992	898	0.905

Table 4.1: The rates of the codes given by the Theorem 4 for some $n \geq 5$.

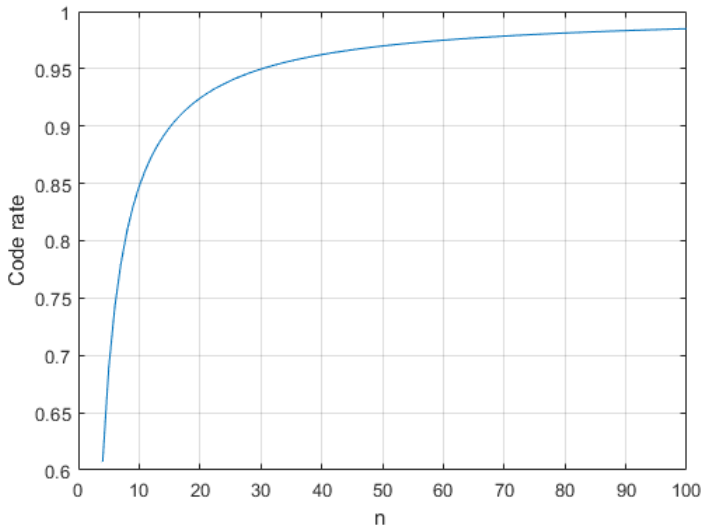


Figure 4.2: The code rate graph of the code with length $N = 4n^2 - 2n$ for $n \geq 5$.

It is not difficult to see that different SDCA$(3, 2n; 2n)$s produce different LDPC codes. For example, for $n = 6$, the code proposed in [7] has the minimum distance 4. However, if we use the SDCA$(3, 12; 12)$ below, the resulting code has the minimum distance 6.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 6 & 0 & 9 & 3 & 11 & 8 & 10 & 5 & 4 & 2 & 7 \end{bmatrix}^{\top}$$

In the Figure 4.3 and 4.4, we proposed the BER and FER results for some QC-LDPC codes obtained by a SDCA$(3, 2n; 2n)$, $n = 5, 6, 7, 8, 9$, other than one obtained by Equation 4.1 [1]. As it can be observed, their performance are different. We calculated their minimum distance. It is six for $n = 5, 6, 7$ while it is four for $n = 8, 9$. Moreover, they all have girth six.

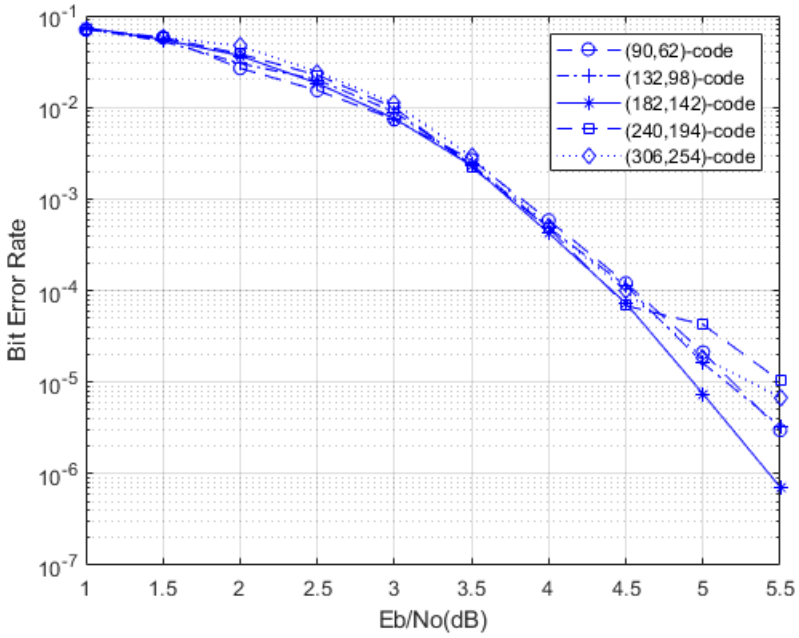


Figure 4.3: BER results of codes obtained from SDCA$(3, 2n; 2n)$ for $n = 5, 6, 7, 8, 9$

[1] These DCAs are given in the Appendix D.

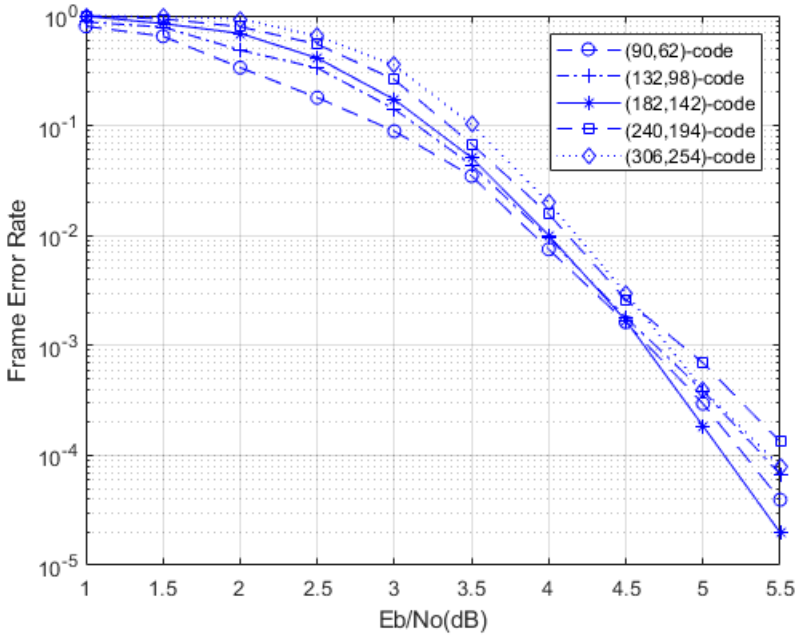


Figure 4.4: FER results of codes obtained from SDCA$(3, 2n; 2n)$ for $n = 5, 6, 7, 8, 9$

Note that if the PBIBD that is obtained by a SDCA$(3, 2n; 2n)$ has no Pasch configuration (a set of 4 blocks of the form $\{a, b, c\}$, $\{a, d, e\}$, $\{f, b, d\}$, $\{f, c, e\}$), then its incidence matrix has no set of four linearly dependent columns. In this case, there cannot be a set of five linearly dependent columns also. Otherwise, since the column weight of H is 3, one of the vectors must vanish, namely, we obtain four linearly dependent columns which yield a contradiction. Therefore, if there is a Pasch-free PBIBD that is obtained by a SDCA$(3, 2n; 2n)$, then its corresponding code has the minimum distance 6. Achieving a higher minimum distance is important for the error correcting performance. We observed that there exists a SDCA$(3, 2n; 2n)$ which creates a Pasch-free PBIBD in each case $2n = 8, 10, 12, 14, 16$ using a computer search. However, we could not gurantee that for any n there is a standard DCA that creates a Pasch-free PBIBD. We have the result for odd n, and future work is needed to explore the result for even n.

Although the minimum distance can vary depending on the SDCA$(3, 2n; 2n)$ that we used, we observed in all examples we analysed that no matter what standard DCA

we use, the rank of H does not change. Therefore, we may expect that the code rate may remain the same for any such construction. This means that changing SDCA will only alter the minimum distance.

In order to compare different SDCA$(3, 2n; 2n)$s for constant n, we will refer to the paper [6]. According to the paper, there are 68 different SDCA$(3, 10, 10)$s and 1140 different SDCA$(3, 12; 12)$s. We completed the analysis of each of the codes constructed using these difference covering arrays, using MATLAB simulations, and the results can be summarized as follows:

All LDPC codes that are obtained from a SDCA$(3, 10; 10)$ has a (30×90)-parity-check matrix of rank 28 and has the code rate 0.688. There are 40 of them that have the minimum distance 4, and the others have the minimum distance 6. In the Figure 4.5 and 4.6, the error correction performance is presented. The best performing codes are ones with minimum distance 6 as we expected, and the worst are ones with minimum distance 4.

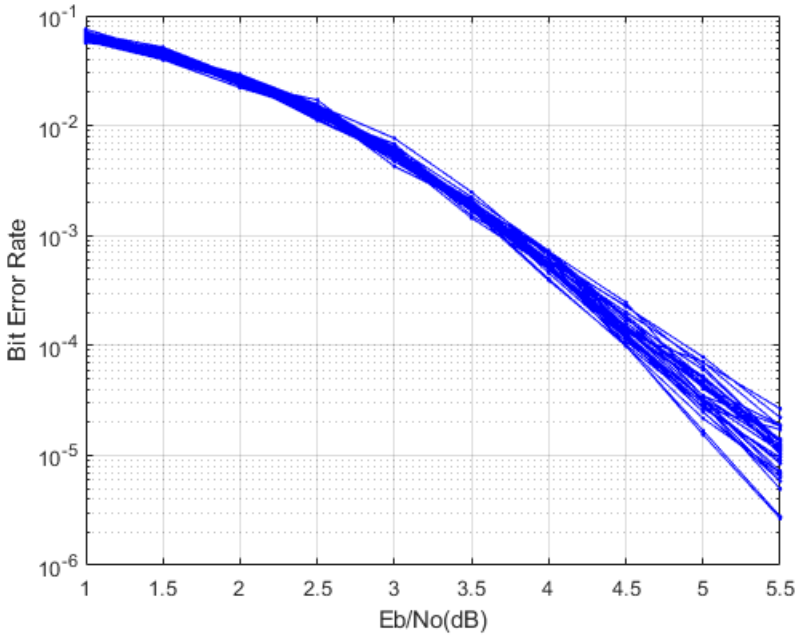


Figure 4.5: BER comparison of QC-LDPC codes for $n = 5$

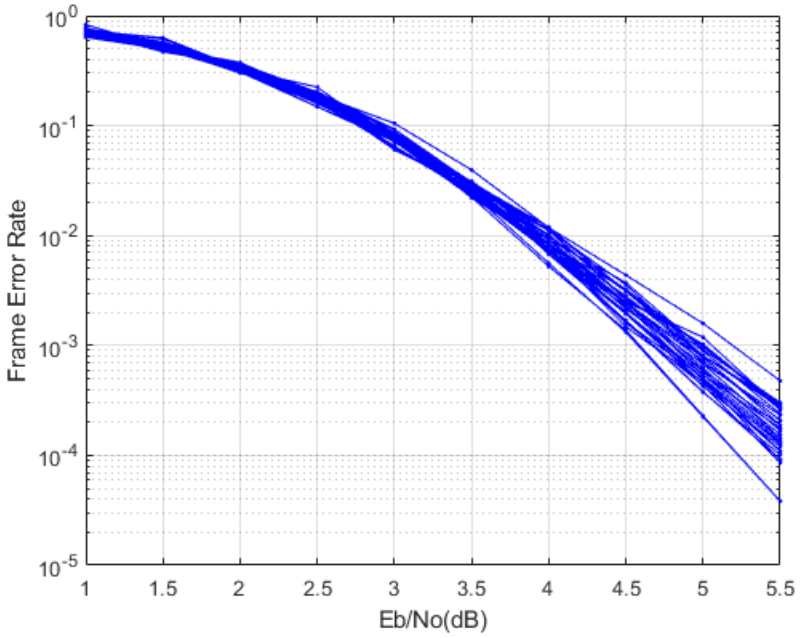


Figure 4.6: FER comparison of QC-LDPC codes for $n = 5$

All LDPC codes that are obtained from a SDCA$(3, 12; 12)$ has a (36×132)-parity-check matrix of rank 34 and has the code rate 0.742. There are 1140 different SDCA$(3, 12; 12)$s, and 884 of them that have the minimum distance 4, and the others have the minimum distance 6. In the Figure 4.7 and 4.8, the error correction performance of these codes is presented. The best performing codes are ones with minimum distance 6 as before, and the worst are ones with minimum distance 4.

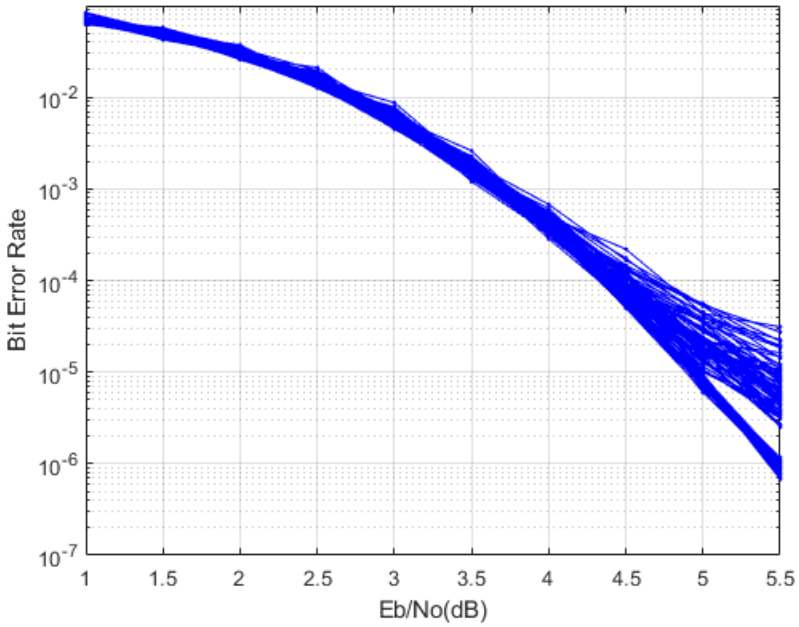


Figure 4.7: BER comparison of QC-LDPC codes for $n = 6$

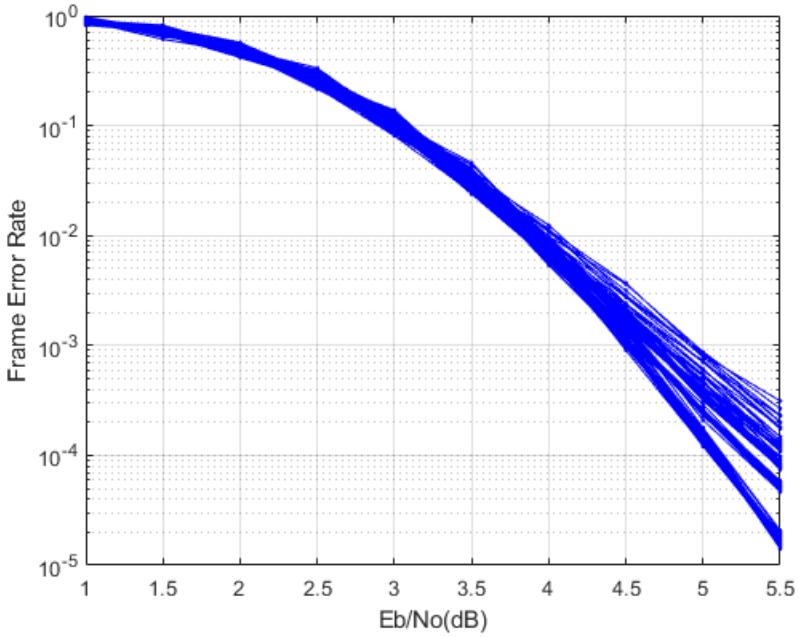


Figure 4.8: FER comparison of QC-LDPC codes for $n = 6$

4.2 Increasing Column Weight of The Parity Check Matrix

As it was also indicated in the second chapter, the most common way to improve a linear code is to increase the minimum distance of the code without changing the rate significantly. In this section, a different version of the construction given in [7] will be presented.

Let $Q = [q(i,j)]$ be the SDCA$(3, 2n; 2n)$ given in Equation 4.1. As in the previous construction, we define $SB_i = (q(i,0), q(i,1), q(i,2)) = (0, i, q(i,2))$ for $i \in \mathbb{Z}_{2n} \setminus \{n\}$ as starter blocks. The set of blocks B_{ia} for $0 \leq a \leq 2n-1$ are then constructed in the following way:

$$B_{ia} = \{i,\ a + 2n,\ [(i + a) \bmod 2n] + 4n,\ [(q(i,2) + a) \bmod 2n] + 6n\}$$

where $(0, i, q(i,2)) = SB_i$. Taking $O_i = \{B_{ia} | 0 \leq a \leq 2n-1\}$ as the orbit set for $i \in \mathbb{Z}_{2n} \setminus \{n\}$, we define the union of orbits

$$\mathcal{B} = \bigcup_{i \in \mathbb{Z}_{2n} \setminus \{n\}} O_i.$$

Then $\mathcal{B}$ consists of $b = 4n^2 - 2n$, 4-subsets (blocks) of $V = \mathbb{Z}_{8n} \setminus \{n\}$ so that each pair of points $y, z \in V$ occurs together in $\lambda_{y,z}$ blocks where

$$\lambda_{y,z} = \begin{cases} 0, & \text{if } y, z \in \{0, \ldots, 2n-1\} \setminus \{n\}, \\ 0, & \text{if } y, z \in \{2n, \ldots, 4n-1\}, \\ 0, & \text{if } y, z \in \{4n, \ldots, 6n-1\}, \\ 0, & \text{if } y, z \in \{6n, \ldots, 8n-1\}, \\ 0, & \text{if } z = y + 4n \text{ and } y \in \{2n, \ldots, 4n-1\}, \\ 0, & \text{if } z = (y+n) \bmod 2n + 4n \text{ and } y \in \{2n, \ldots, 4n-1\}, \\ 0, & \text{if } z = y + 2n \text{ and } y \in \{4n, \ldots, 6n-1\}, \\ 1, & \text{otherwise.} \end{cases}$$

Note that there is no block containing n since we precluded the starter block $(0, n, 0)$. We then construct an incidence matrix $H = [h(j,i)]$ where the columns are

indexed by the blocks $B_{ia} \in \mathcal{B}$ and set

$$h(j, B_{ia}) = \begin{cases} 1, & \text{if } 0 \leq j \leq n-1 \text{ and } j \in B_{ia} \\ 1, & \text{if } n \leq j \leq 8n-2 \text{ and } j+1 \in B_{ia} \\ 0, & \text{otherwise} \end{cases} . \tag{4.2}$$

This construction produces a $(8n-1) \times (4n^2 - 2n)$, 0-1 matrix.

Example 11. Let $n = 2$. Then starter blocks are $SB_0 = (0, 0, 1)$, $SB_1 = (0, 1, 3)$, and $SB_2 = (0, 3, 2)$. Their orbits and the incidence matrix H are given below

$O_0 =$	$O_1 =$	$O_3 =$
$\{B_{00} = \{0, 4, 8, 13\},$	$\{B_{10} = \{1, 4, 9, 15\},$	$\{B_{20} = \{3, 4, 11, 14\},$
$B_{01} = \{0, 5, 9, 14\},$	$B_{11} = \{1, 5, 10, 12\},$	$B_{21} = \{3, 5, 8, 15\},$
$B_{02} = \{0, 6, 10, 15\},$	$B_{12} = \{1, 6, 11, 13\},$	$B_{22} = \{3, 6, 9, 12\},$
$B_{03} = \{0, 7, 11, 12\}\}$	$B_{13} = \{1, 7, 8, 14\}\}$	$B_{23} = \{3, 7, 10, 13\}\}$

$$H = \left[\begin{array}{cccc||cccc||cccc} 1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1 \\ \hline\hline 1&0&0&0&1&0&0&0&1&0&0&0 \\ 0&1&0&0&0&1&0&0&0&1&0&0 \\ 0&0&1&0&0&0&1&0&0&0&1&0 \\ 0&0&0&1&0&0&0&1&0&0&0&1 \\ \hline\hline 1&0&0&0&0&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&0&0&0&0&1&0 \\ 0&0&1&0&0&1&0&0&0&0&0&1 \\ 0&0&0&1&0&0&1&0&1&0&0&0 \\ \hline\hline 0&0&0&1&0&1&0&0&0&0&1&0 \\ 1&0&0&0&0&0&1&0&0&0&0&1 \\ 0&1&0&0&0&0&0&1&1&0&0&0 \\ 0&0&1&0&1&0&0&0&0&1&0&0 \end{array}\right]$$

We now prove some basic properties of the LDPC code with the parity-check matrix H given by Equation 4.2, and then provide BER/FER results in order to compare with the previous construction. From now on, we will assume $n \geq 6$. The proof for the smaller sizes can be obtained by direct calculation.

Lemma 6. *The LDPC code with the parity-check matrix H given by Equation 4.2 has girth at least 6.*

Proof. It follows similarly as in Lemma 3. QED

We observed in many examples that the rank of the matrix obtained by this construction is $8n-6$. However, we did not prove this for general n. We now calculate the minimum distance of the proposed code. We will apply Theorem 1 for the proof.

Lemma 7. *The LDPC code with the parity-check matrix H obtained from Equation 4.2 has the minimum distance 8.*

Proof. It is easy to observe that the columns corresponding to the blocks

$$\{0, 2n+1, 4n+1, 6n+2\},$$

$$\{0, 3n-2, 5n-2, 7n-1\},$$

$$\{1, 3n-2, 5n-1, 7n+1\},$$

$$\{1, 4n-1, 4n, 6n+2\},$$

$$\{n-2, 2n+1, 5n-1, 8n-2\},$$

$$\{n-2, 3n+2, 4n, 7n-1\},$$

$$\{n-1, 3n+2, 4n+1, 7n+1\},$$

$$\{n-1, 4n-1, 5n-2, 8n-2\},$$

are linearly dependent. If we show that there is no r linearly dependent column set for $r < 8$, we then prove the lemma thanks to Theorem 1. However, this is a direct consequence of the paper [10]. It was shown that there is no stopping set[2] of size ≤ 7

[2]A stopping set S is a set of code bits with the property that every parity-check equation that checks on a bit in S in fact checks on at least two bits in S. Observe that any linearly dependent set of c columns in a parity check matrix corresponds to a stopping set of size c. [11].

in a pair of partially orthogonal Latin squares. QED

As for performance results, we present FER graphs for different values of n in both constructions as in Figure 4.9. Although the minimum distance was 4 for even n and was 6 for odd n, in the first version of the construction, the new version produces a code with minimum distance 8. Thus, the new codes perform better, as we expect. Especially for even values of n, we get definitely better codes. Like in this improvement, further research may be carried out to increase the minimum distance of the code by using difference covering arrays with more columns.

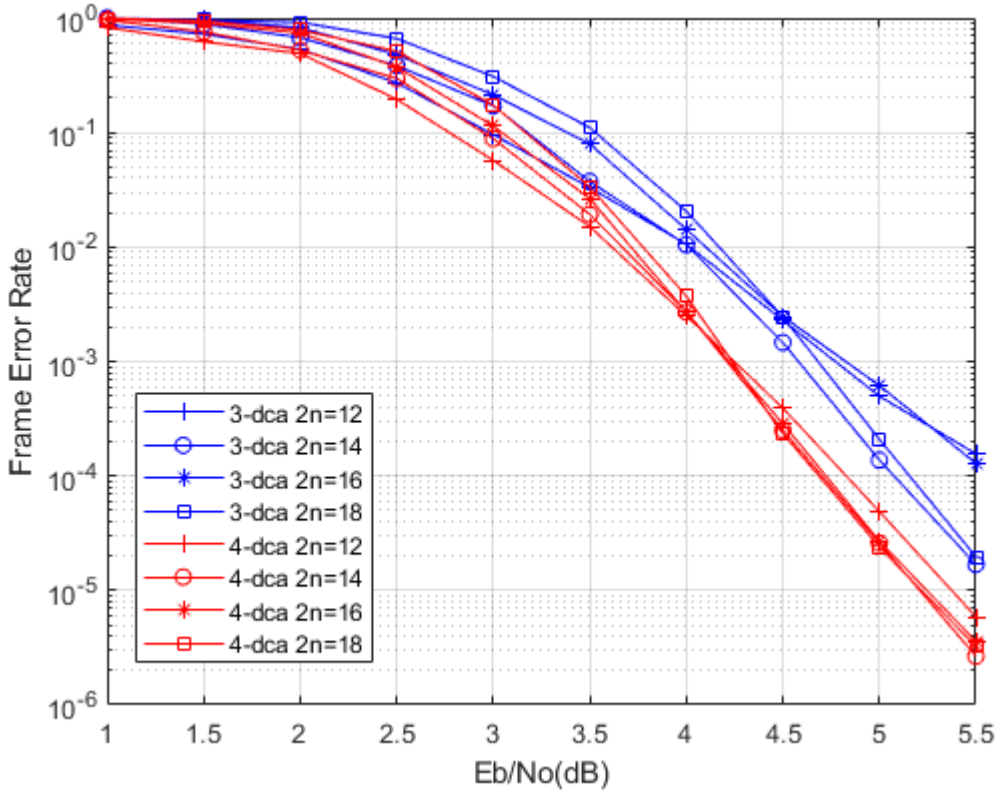


Figure 4.9: FER graphs for both versions of the construction

Chapter 5

CONCLUSION

In this thesis, we introduced a construction of one of the well-known error-correcting codes, namely, LDPC codes, using difference covering arrays. Although pseudorandomly generated LDPC codes are used in practice more than any other kinds [11], we especially studied a form of structured ones due to their certain advantages. Our main purpose was to give a broad analysis of LDPC codes obtained from PBIBDs constructed by difference covering arrays.

First of all, we provided the basic definitions from coding theory. An LDPC code is a kind of linear block code as we defined in the second chapter. Thus, we mentioned properties of linear codes in order to understand LDPC codes. We focused on an important property of a linear code, namely, its minimum distance. We emphasized the importance of having high minimum distance, and provided a lower bound for the minimum distance of an LDPC code, which is given by Tanner [16]. Additionally, the sum-product decoding (SPD) algorithm was presented in this chapter to be used for the codes which we constructed. An example of SPD algoritm, launched via MATLAB, was given.

In the third chapter, we gave some basic definitions and results from combinatorial design theory. Partially-balanced incomplete block-designs and pseudo-orthogonal Latin squares were presented to be used in the construction of proposed LDPC codes [7]. However, we mostly used PBIBDs directly instead of obtaining them by Latin squares.

In the fourth chapter, we constructed LDPC codes from PBIBDs using difference covering arrays. As we mentioned before, the PBIBDs that we constructed could be obtained by pseudo-orthogonal Latin squares. But we made this construction directly

via DCAs. We analysed some LDPC codes with small code length, and provided their BER/FER results so that we made some observation about the relation between performance of these codes and their minimum distances.

Furthermore, we presented another construction of LDPC codes. The minimum distance of the new code was greater than the old one. We compared the frame-error rate results of both constructions, and we observed that new ones perform better.

Chapter 6

APPENDICES

6.1 Appendix A

Proposition. *If ith bit node appears in the jth parity check equation, then the probability that the parity-check equation is satisfied if the bit is a 1, is equal to the probability that an odd number of the other codeword bits are a 1. It is given by*

$$\frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} \left(1 - 2P_{j,i'}^{int}\right)$$

where B_j is the set of column locations of the bits in the jth parity-check equation of the code, and $P_{j,i'}^{int}$ is the intrinsic information sending from bit node i' to check node j, that is, the current estimate of the probability $c_{i'} = 1$.

Proof. The first assertion is clear because the scalar field is $GF(2)$. We will prove the second one by using induction on the number of elements in $B_j \setminus \{i\}$. If the set is empty ($|B_j \setminus \{i\}| = 0$), then the parity check equation consists of only ith bit. In this case, the probability that an odd number of the other codeword bits are a 1, is zero as we expected. Suppose the assertion is true when the set $B_j \setminus \{i\}$ has t elements.

Now assume the set contains $t+1$ elements. Let P be the probability that an odd number of the $t+1$ codeword bits are a 1. We divide it into two independent cases, c_{t+1} is 0 or 1. By induction hypothesis, the probability of the first event is

$$\left(\frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j}^{i' \neq i, t+1} \left(1 - 2P_{j,i'}^{int}\right)\right) \left(1 - P_{j,t+1}^{int}\right). \tag{6.1}$$

In the second case, we must have the total number of nonzero bits in the other t many bits is even. Thus, this situation can happen with the probability

$$\left[1 - \left(\frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j}^{i' \neq i, t+1} \left(1 - 2P_{j,i'}^{int}\right)\right)\right] \left(P_{j,t+1}^{int}\right). \tag{6.2}$$

Therefore, the equations 6.1 and 6.2 imply that P is equal to their sum. After the following calculations, we obtain the result for general case.

$$\begin{aligned}
P &= \left(\frac{1}{2} - \frac{1}{2}\prod_{i'\in B_j}^{i'\neq i,t+1}\left(1-2P_{j,i'}^{int}\right)\right)\left(1-P_{j,t+1}^{int}\right) \\
&\quad + \left(\frac{1}{2} + \frac{1}{2}\prod_{i'\in B_j}^{i'\neq i,t+1}\left(1-2P_{j,i'}^{int}\right)\right)\left(P_{j,t+1}^{int}\right) \\
&= \frac{1}{2} - \frac{1}{2}\prod_{i'\in B_j}^{i'\neq i,t+1}\left(1-2P_{j,i'}^{int}\right) - \cancel{\frac{P_{j,t+1}^{int}}{2}} + \frac{P_{j,t+1}^{int}}{2}\prod_{i'\in B_j}^{i'\neq i,t+1}\left(1-2P_{j,i'}^{int}\right) \\
&\quad + \cancel{\frac{P_{j,t+1}^{int}}{2}} + \frac{P_{j,t+1}^{int}}{2}\prod_{i'\in B_j}^{i'\neq i,t+1}\left(1-2P_{j,i'}^{int}\right) \\
&= \frac{1}{2} - \frac{1}{2}\left(\prod_{i'\in B_j}^{i'\neq i,t+1}\left(1-2P_{j,i'}^{int}\right) - 2P_{j,t+1}^{int}\prod_{i'\in B_j}^{i'\neq i,t+1}\left(1-2P_{j,i'}^{int}\right)\right) \\
&= \frac{1}{2} - \frac{1}{2}\left(\prod_{i'\in B_j}^{i'\neq i}\left(1-2P_{j,i'}^{int}\right)\right).
\end{aligned}$$

QED

6.2 Appendix B

Proposition. *For $p \geq 0$, we have*

$$\tanh\left(\frac{1}{2}\ln\left(\frac{1-p}{p}\right)\right) = 1-2p.$$

Proof. By definition of hyperbolic tangent, we obtain

$$\tanh\left(\frac{1}{2}\ln\left(\frac{1-p}{p}\right)\right) = \frac{e^{2\left(\frac{1}{2}\ln\left(\frac{1-p}{p}\right)\right)}-1}{e^{2\left(\frac{1}{2}\ln\left(\frac{1-p}{p}\right)\right)}+1} = \frac{\left(\frac{1-p}{p}\right)-1}{\left(\frac{1-p}{p}\right)+1} = 1-2p.$$

QED

6.3 Appendix C

Pseudo Code for Sum-product Decoding Algorithm over AWGN Channel [11]

function(H, $\mathbf{c}$, E_b/N_0, I_{max})

% H is the parity check matrix

% $\mathbf{c}$ is the codeword that is sent

% E_b/N_0 is the noise level

% I_{max} is the maximum number of iterations

m = the number of rows of H

N = the number of columns of H

$\mathbf{mc} = 1 - 2\mathbf{c}$ % BPSK modulation for $\mathbf{c}$ ($0 \mapsto 1$, $1 \mapsto -1$)

$\mathbf{y} = \mathbf{mc}$+noise % The noise is obtained by AWGN with E_b/N_0

$I = 0$ % Count for iteration

for $i = 1 : N$ %Initial Message

  for $j = 1 : m$

    $M_{j,i} = LLR(y_i)$

  end for

end for

---

repeat

for $j = 1 : m$ %Check Message

  for $i \in B_j$

    $E_{j,i} = \ln\left(\frac{1+\prod_{i' \in B_j, i' \neq i} \tanh\left(M_{j,i'}/2\right)}{1-\prod_{i' \in B_j, i' \neq i} \tanh\left(M_{j,i'}/2\right)}\right)$

  end for

end for

for $i = 1 : N$

$L_i = \sum_{j \in A_i} E_{j,i} + LLR(y_i)$

$\hat{\mathbf{y}}_i = \begin{cases} 0 & \text{if } L_i > 0 \\ 1 & \text{if } L_i \leq 0 \end{cases}$ **end for**

---

if $I = I_{max}$ or $\hat{\mathbf{y}}^T H \neq 0$

Terminate

else

for $i = 1 : N$ %Bit Message

for $j \in A_i$

$M_{j,i} = \sum_{j' \in A_i, j' \neq j} E_{j',i} + LLR(y_i)$

end for

end for

$I = I + 1$

end if

until Terminate

end function

6.4 Appendix D

The SDCA$(3, 2n; 2n)$s used in the Figures 4.3 and 4.4 are listed below

$$2n = 10 \Longrightarrow Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 3 & 8 & 6 & 9 & 0 & 5 & 4 & 2 & 7 \end{bmatrix}^\top$$

$$2n = 12 \Longrightarrow Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 3 & 5 & 10 & 8 & 11 & 0 & 6 & 4 & 2 & 7 & 9 \end{bmatrix}^\top$$

$$2n = 14 \Longrightarrow Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 1 & 3 & 5 & 13 & 8 & 10 & 12 & 0 & 7 & 2 & 4 & 6 & 9 & 11 \end{bmatrix}^\top$$

$$2n = 16 \Longrightarrow Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 1 & 3 & 5 & 7 & 15 & 10 & 12 & 14 & 0 & 8 & 2 & 4 & 6 & 9 & 11 & 13 \end{bmatrix}^{\top}$$

$$2n = 18 \Longrightarrow Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ 1 & 3 & 5 & 7 & 17 & 10 & 12 & 14 & 16 & 0 & 9 & 2 & 4 & 6 & 8 & 11 & 13 & 15 \end{bmatrix}^{\top}$$

BIBLIOGRAPHY

[1] N. Bonello, S. Chen, and L. Hanzo. Low-density parity-check codes and their rateless relatives. *IEEE Communications Surveys Tutorials*, 13, 2011.

[2] R. C. Bose and S. S. Shrikhande. On the falsity of euler's conjecture about the non-existence of two orthogonal latin squares of order 4t + 2. *Proceedings of the National Academy of Sciences*, 45(5):734–737, 1959.

[3] C. J. Colbourn and J. H. Dinitz. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2006.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006.

[5] F. Demirkale, D. Donovan, J. Hall, A. Khodkar, and A. Rao. Difference covering arrays and pseudo-orthogonal latin squares. *Graphs and Combinatorics*, 32(4):1353–1374, Jul 2016.

[6] F. Demirkale, D. M. Donovan, J. I. Kokkala, and T. G. Marbach. The enumeration of cyclic mutually nearly orthogonal latin squares. *Journal of Combinatorial Designs*, 27(5):265–276.

[7] D. Donovan, A. Rao, and E. S. Yazici. *High Rate LDPC Codes from Difference Covering Arrays*, volume abs/1701.05686. 2017.

[8] L. Euler. Recherche sur une nouvelle espèce de quarrès magiques. *Verhand. Zeeuwsch Gen. Wet. Vlissingen*, 9, 1782.

[9] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962.

[10] A. Gruner and M. Huber. Low-density parity-check codes from transversal designs with improved stopping set distributions. *IEEE Transactions on Communications*, 61(6):2190–2200, June 2013.

[11] S. J. Johnson. *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*. Cambridge University Press, 2009.

[12] Y. Kou, S. Lin, and M. P. C. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Transactions on Information Theory*, 47(7):2711–2736, Nov 2001.

[13] J. M. F. Moura, Jin Lu, and Haotian Zhang. Structured low-density parity-check codes. *IEEE Signal Processing Magazine*, 21(1):42–55, Jan 2004.

[14] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[15] D. Stinson. A short proof of the nonexistence of a pair of orthogonal latin squares of order six. *Journal of Combinatorial Theory, Series A*, 36(3):373 – 376, 1984.

[16] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, Sep. 1981.

[17] J. Yin. Cyclic difference packing and covering arrays. *Designs, Codes and Cryptography*, 37(2):281–292, Nov 2005.