New Decoding Methods for LDPC Codes on Error and Error-Erasure Channels

MASTER OF SCIENCE THESIS

by

İlke ALTIN

August 19, 2010

Committee Members

Supervisor :	Dr. ir. J.H. Weber
Readers :	Prof. dr. ir. I. Niemegeers
	Dr. ir. H. Wang

Wireless and Mobile Communications Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands

Abstract

New Decoding Methods for LDPC Codes on Error and Error-Erasure Channels

For low-end devices with limited battery or computational power, low complexity decoders are beneficial. In this research we have searched for low complexity decoder alternatives for error and error-erasure channels. We have especially focused on low complexity error erasure decoders, which is a topic that has not been studied by many researchers.

The separation of erasures from errors idea [1] seemed profitable to design a new error erasure decoder, so we have also worked on this idea. However, the methods that are described in that paper are not realizable for practical values of code length and number of erasures. Thus, a new separation method; the rank completer is proposed, which is realizable. In the part of the research that is related to error decoding, we have proposed a modification to reliability ratio based bit flipping algorithm [2], which improves the BER performance with very small additional complexity. In the part that is related to error erasure decoding, we have given a new guessing algorithm that performs better than some known guessing algorithms, and a new error erasure decoder that uses the rank completer idea. Both simulation results and analytical models are used to adjust variables of the described methods. Also simulation results are utilized to compare the proposed methods with existing methods. The rank completer (and the separation of erasures from errors) is a promising method that can be further studied for error erasure decoding. However, we have shown that for already error erasure correcting methods, application of this method can degrade the BER performance of the original code. The modification that we propose for error decoders does not improve the BER performance much, however, it can be applied since it has nearly no additional complexity. The proposed error erasure decoders can be used for systems that do not require high reliability, but favor low complexity.

Acknowledgements

Firstly, I would like to thank my supervisor dr. J.H. Weber for his guidance during this study, and for the long discussions about my ideas and writing skills. I am thankful to him for letting me do research independently, and being there when I needed a helping hand. If he did not correct my mistakes, and provide me with useful comments, this work would not be present today.

I am thankful to my parents for their endless love and support, I would not be able to achieve anything without you. Thanks for encouraging me to try whatever I would like to do, especially this study in TU Delft.

I would like to use this opportunity to thank my dearest four friends in Delft. Delft is lovely with you, gentlemen. I also would like to thank all my friends for all the drinks, games, and joyful time that we shared.

Finally, I would like to thank my girlfriend for always being there for me. I am looking forward to joining you back in Turkey.

İlke ALTIN
Delft, August 19, 2010

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The basic communication problem, which is given in Figure 1-1, consists of three elements; the source with information to send to the sink, the sink to receive the information sent by the source, and the noisy channel which disrupts the information sent. The intended solution of this problem is transmission of data from source to the sink in an efficient and reliable way.

Figure 1-1: The communication problem [3]

The basic communication problem introduced above has been formalized by Shannon in two separate parts; the first part deals with the information theoretical aspects of the data to be sent by the source, and the second part deals with the reliable transmission of this data through the noisy channel. The structure is given in Figure 1-2. There are four new blocks that have two different functions as stated before; the source encoder/decoder and the channel coder/decoder. Source encoder removes the redundancy of the source information, while the source decoder retrieves the full source information from the encoded data. On the other hand, channel encoder introduces redundancy for reliable transmission of the data through the noisy channel (or storage medium), and the channel decoder retrieves -of course depending on the capabilities of the channel encoding/decoding blocks and the noise- the source coded data from the received data. Source coding part is not explained any further in this thesis, depending on Shannon's source-channel separation theorem, which simply states that an efficient and reliable transmission is possible by dealing with source coding and channel coding separately. Therefore, in the channel coding perspective the source and the source coding can be thought as a single large block. These blocks are shown in Figure 1-2 by the big dashed rectangles.

Using the channel coding perspective described in the previous paragraph, the transmitter part is reduced to a source that generates arbitrary number information symbols from an

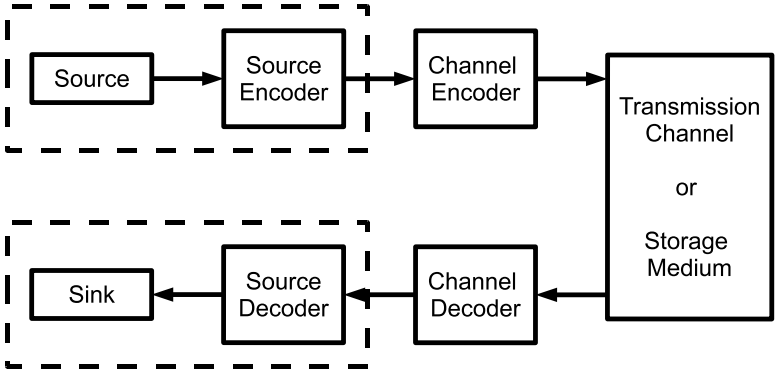

Figure 1-2: The communication problem revised by Shannon

alphabet, and a channel encoder encodes k of these symbols and produces an output of length n, where $n-k$ is the redundancy of the channel encoder. In the channel some of these n encoded symbols are corrupted due to noise which might cause errors and/or erasures. A symbol is called erroneous if its value is changed through the channel, and erased if no value or an erasure flag is received for that symbol. In the receiver, the received symbol sequence, which has length n is decoded to the k symbols that is closest to the received sequence. The term "closest" might be in the sense of maximum likelihood, Hamming distance, etc. depending on the type of the decoder.

There are two main types of channel coding techniques. The first type is called Automatic Repeat Request (ARQ), in which the receiver requests retransmission of unreliable data frames. A data frame can be declared unreliable if an erasure or an error is detected in that frame. The second type is forward error correction Forward Error Correction (FEC) where the channel decoder estimates a codeword from the received codeword. Forward error correction methods will be the focus of this thesis. An error correction/detection scheme can be evaluated by three important properties; the reliability of the scheme, the complexity of the scheme, and the efficiency of the scheme. The reliability of the scheme stands for the reliability of the decoded words in the receiver, which can be measured by Bit Error Rate (BER) or Frame Error Rate (FER). The complexity of the scheme is measured by the number of operations that is required by the system and the complexity of these operations. The efficiency of the scheme is measured by the ratio of the information sent for error correction/detection and the information sent from the source. The main trade-off in the error correction/detection technique is between these three properties.

Now, a channel coding example will be given to clarify the process for the reader. For this particular example, a (7,3,4) simplex code will be used. A simplex code is one of the basic linear block codes and can be described by its length n, dimension k and Hamming distance d and the alphabet size q. The dimension of the code defines the number of possible messages with the size of the code alphabet; there are q^k possible messages for a code that has dimension k and alphabet size q. The Hamming distance is the minimum number of different symbols between any two different codewords of the code, which can be used as a measure of error correction capability. The codewords of binary (7,3,4) simplex code are the rows of the matrix C_s in (1-1). It should be recognized that the eight possible messages that could be sent by the source are in the right-hand side of the matrix.

$$C_s = \left(\begin{array}{cccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right) \tag{1-1}$$

After introducing the possible codewords, the scenario for a source which wants to transmit message $m = 001$ is given below, where a symbol became erroneous and another one is erased through the channel;

- **Source** wants to transmit message $m_t = 001$
- **Channel encoder** encodes m_t by using C_s and outputs the codeword $c_t = 1101001$
- **Through the channel** the transmitted codeword c_t is changed into $1001x01$ where x denotes an erasure
- **Channel decoder** estimates $c_r = 1001x01$ to the closest codeword in C_s; $c_e = 1101001$, and decodes it.
- **Sink** receives the decoded message $m_r = 001$

The received message by the sink is the same with the one transmitted by the source, so we can conclude that channel coding has been successful in the presence of an error and an erasure in the codeword. It is easy to realize if channel coding has not been used, errors and erasures can not be corrected. Therefore, for a more reliable transmission through a noisy channel, channel coding techniques are indispensable. Of course, the aforementioned advantages of error correcting codes comes with a price; usage of the error correcting codes requires more symbols to be transmitted, which is costly in terms of bandwidth, power, or time depending on the communication system. However, the benefits have outweighed the aforementioned costs and the calculations for the encoding/decoding; and channel coding is widely used in communications and storage systems.

Main application areas of error detecting/correcting codes can be given as;

- **Wireless and Mobile Communications:** Error correcting techniques are widely used in mobile communication systems to cope with the perturbations caused by interference, noise, multi path fading, shadowing, propagation loss, etc. in the wireless channel. In GSM; Cyclic Redundancy Codes (CRC) are used for error detection, block and convolutional codes are applied for error correction. In CDMA2000; convolutional codes and turbo codes are used for error correction. In 3G; both convolutional and turbo codes are supported for error correction. Low-Density Parity-Check (LDPC) codes are the standard error correcting codes for many wireless communication protocols such as WiMAX (802.16e) and WLAN (802.11).

- **Deep Space Communications:** Since space and the atmosphere is the channel for these communication systems, the space radiation is the most effective disturbance to the signal which can be modeled as Additive White Gaussian Noise Channel (AWGN)(if an interleaver is used). Although the noise sources are very limited, communication from outer space would be impossible because of the high propagation loss due to the distance. Thus, error correction is the only sensible way to communicate with signal powers as low as in space communication case. First of the codes that is used for deep space communications is (32,6,16) Reed-Muller in the Mariner spacecraft. After that, mostly convolutional codes and Reed-Solomon (RS) codes are used for space communications, with improvements in rate, gain, or time performance.

- **Satellite Communications:** For digital satellite TV, RS codes had been used for a long time, however currently replaced by modern codes such as LDPC and Turbo codes.

- **Military Communications:** In military communications, besides the natural interference and noise, the communication system also needs to deal with the intentional enemy interference; therefore, the need for error correcting codes is obvious.

- **Data Storage:** RS codes is widely used for error correction in storage systems such as CDs, DVDs and hard discs. Single Error Correcting Double Error Detecting (SECDED) codes, which correct single errors and detect double errors, are widely used in many data storage units with RS codes or Hamming codes.

Before moving to the next section we will discuss some important properties of error correcting codes. Firstly, it should be noted that the error correcting codes are divided into two parts, block codes and convolutional codes.

For block codes encoding can be described as follows; the information symbols to be sent are divided into blocks of k symbols, then these k symbols are encoded to n symbols where the code rate (or the ratio of the number of information symbols to total number of symbols) is k/n. The decoding process is the reverse of this case, where the received information is chopped into blocks of n symbols and decoded to k information symbols.

For convolutional codes, the information bits to be sent are not chopped into blocks but viewed as a stream. Every symbol in this stream effects the encoder output of a number of symbols after it, depending on the memory of the convolutional code, M. The data is encoded by taking k input symbols and then computing n equations with these k symbols and the buffered $k \cdot M$ symbols. Depending on these values, it is obvious that the code rate is again k/n.

After introducing the two main types of codes, we shall explain the Hamming distance of a code in more detail. Hamming distance of a code is the minimum Hamming distance between two codewords that is the element of the code. Hamming distance between two codewords, $d(c_1, c_2)$ is the number of positions that these two codewords differ. Hamming distance had been the most important parameter of the code's that represent the error and erasure correcting ability for a long time, and a code with Hamming distance d is known to be correct $d-1$ erasures or $\lfloor \frac{d-1}{2} \rfloor$ errors. However these limits are only meaningful, while discussing hard-decision decoding, when no reliability information of received bits is available or used.

Another important property of a code is linearity; and a code is linear, if the codewords are a subspace of the length n vector space of the (finite) field. In other words if all the linear combinations of the codewords is another codeword, then we can conclude that the block code is linear. It should be obvious that a word with all zero entries is a valid codeword for all linear block codes.

Final description related to linear block codes is the parity check matrix, H which can be used to decode linear block codes. The rows of the Parity Check Matrix (PCM) form the null space of the generator matrix. Therefore, we can conclude that any valid codeword, $c \cdot H^T = 0$. If H is a $m \times n$ matrix, then there are m parity check equations for the code which is given below in (1-2).

$$PCE_i = \sum_{j=1}^{n} h_{i,j}.c_j \quad for \;\; i = 1, 2, ..., m \tag{1-2}$$

In the next section we introduce LDPC codes, since we have mainly worked on decoders for LDPC codes in this research.

1-1 LDPC codes

LDPC codes are linear block codes, that are defined by their sparse parity check matrices. By density, we mean the ratio of the number of ones in the matrix to the number of all elements in the matrix. If for each row (or column) ratio of the number of ones to the length of that row (or column) is equal, then the code is called a regular LDPC code. The low-density condition can be satisfied especially for larger block lengths.

LDPC codes have been proposed by Gallager in [4] in 1960s. However, the long block LDPC codes were too complex to be implemented at that time, and the codes were largely forgotten. In 1980s Tanner has given a bipartite graph interpretation of LDPC codes with other types of codes in [5]. After Turbo codes have emerged and iterative decoding methods for large codes are designed, two groups independently rediscovered the LDPC codes in [6] and [7].

LDPC codes that has been proposed by Gallager were regular LDPC codes, that has a constant column and row in their parity check matrices. Shortly after the rediscovery of LDPC codes, a new type of LDPC codes has been introduced in [8], which are called irregular LDPC codes. This type of LDPC codes can have different density rows and columns in its parity check matrices, and they can perform better than regular LDPC codes as shown in [9].

LDPC codes are mostly represented by their parity check matrices or the corresponding Tanner graphs. Tanner graph is a bipartite graph representation for the parity check matrix of the code. Say the size of H is a $m \times n$ matrix, the m parity check (or check) nodes and n codeword (or variable) nodes correspond to two disjoint sets in the bipartite graph. An element c_i of "check nodes" set is connected to an element v_j of "variable nodes" set if the corresponding entry in the parity check matrix is one, $h_{ij} = 1$. Previously introduced (7,3,4) simplex code is used to show the relationship between the parity check matrix and the Tanner graph. The parity check matrix is given in (1-3) and the corresponding Tanner graph representation is given in Figure 1-3. An important definition about the Tanner graph is the

definition of neighbor nodes, two nodes are neighbors if the are connected by an edge in the Tanner graph of the corresponding code.

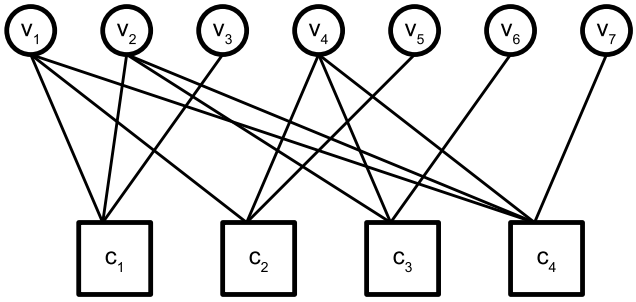

Figure 1-3: Tanner graph of (7,3,4) simplex code

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{1-3}$$

To understand why LDPC codes perform so well, Shannon's important argument about transmission on noisy channels should be taken into account. The argument states that assigning very large block codes randomly to every message, it is possible to reach the limit for nearly error-free communication which is called the Shannon limit. Therefore if very large block codes can be decoded in a feasible time, we can reach the limit set by Shannon in 1940s. The sparse parity check matrices of LDPC codes combined with iterative decoding techniques, has opened the way for decoding of large block codes and made it possible to come real close to Shannon bounds.

1-2 Motivation

In this research we have worked on new decoding alternatives for linear block codes, and especially for LDPC codes. Considering the trade-off of the error correction schemes, that is between complexity, efficiency, and reliability, we have worked on decoders that have less complexity and reasonable BER performance on channels that cause errors and channels that cause both errors and erasures. In this section, we will motivate the research on decoders for Error-Erasure Channels (EEC) by giving previous research on error correcting codes on these sort of channel types, and the research for low complexity decoders for LDPC codes by introducing two different types of decoders on different ends of the trade-off. The low complexity methods that we study in this research can be extremely useful for low-end devices that has little computational power or battery power.

1-2-1 Error Erasure Channels

There are many examples of channels in real world communication problems -either transmission or storage- that cause errors in combination with erasures, while there are not many decoders that can cope up with EEC. Some of the possible real world cases that are modeled by EEC are given below.

First case that is considered here is from mobile communications field, the Multimedia Broadcast/Multicast Service (MBMS) in GSM EDGE Radio Access Network (GERAN) [10]. The problem is decoding the Radio Link Control (RLC) layer packets which can have residual bit errors that should be corrected to prevent propagation of error to upper layers. In [11], this is handled by using RS codes with declaring erasures for unreliable bits, and then applying error-erasure decoding.

Second case is from wireless communications, systems that use frequency hopping in Medium Access Control (MAC) protocols. In [12], Binary Symmetric Error Erasure Channel (BSEEC) is introduced, which is explained in Chapter 2. Also, it is stated that frequency hopping systems might perform better by declaring erasures in case of inter-channel interference, and applying Error Erasure Decoding (EEDG). However, it is shown that for some channel conditions automatically declaring an erasure is not optimal, even with perfect side information about the interference. In [13], again the erasure declaring method is used with available side information with variable rate coding. Another method is declaring erasures by Ratio Threshold Test (RTT), which uses no side information. In [14], many channel coding techniques are given, one of them being the error-erasure decoding of RS codes, which is shown to be effective for slow frequency hopping systems. In [15], erasure declaring methods are used to solve tone jamming and Rayleigh fading problems.

Now, we move to an example in the magnetic recording channel, where the channel can be modeled as AWGN plus burst noise. The channel is assumed to be AWGN plus burst erasures with perfect side information in [16]. After the channel model is changed to AWGN plus erasures, error-erasure decoding is done by message-passing decoding of LDPC codes. In [17], AWGN with erasures is again assumed to be a good model for magnetic and optical recording systems. The authors have given analysis methods for LDPC codes over AWGN with burst and random erasures. The proposed error-erasure decoding method is again a message-passing decoder.

The combination of AWGN noise and non-stationary interference in DSL systems, is an another important real world communication problem that error-erasure channel models are appropriate. This channel is modeled as an AWGN channel with erasures in [18], by using square distance to the possible symbol points in the constellation diagram and is decoded using RS codes. Depending on this result, in [19], and [20] authors have used Additive White Gaussian Noise Channel with Erasures (AWGN+EC) and BSEEC to model the channel and used soft decision message passing (belief propagation) and hard-decision message passing with erasures (Gallager Algorithm E) for decoding, respectively.

Some other examples of EEC are; satellite communications, and digital watermarking. In [21], AWGN+EC is assumed to be a good model for satellite communications with an interleaver of proper length. In [22], LDPC codes are used to decode the word that contains errors and erasures for blind watermarking problem.

1-2-2 LDPC code decoders

Here we will explain two widely used LDPC decoders shortly; the Bit Flipping (BF) algorithms and the Message Passing (MP) algorithms to motivate our choices in this research. Both of these methods have been introduced in [4] along with the LDPC codes by Gallager, and has been studied by various researchers. Considering the three aforementioned trade-off properties of an error correction scheme, the efficiency depend on the code (the code rate to be specific) and is independent of the decoder selection. However, the complexity and the reliability depend both on the code and the decoder.

If the complexity of these methods are considered, the BF decoders are known to be less complex, since MP decoders require messages to be sent between nodes and calculation of new messages in the receiving nodes in each iteration, where the BF decoders require only the calculation of a flipping metric for each bit by using the initial reliabilities and the violated parity check equations. Comparing the reliabilities of these two methods, BF decoders perform worse than the MP decoders on error channels and erasure channels. In addition to this, no methods have been described about BF algorithms on EEC (in our best knowledge), where there have been a significant amount of research related to the decoding of LDPC codes by MP decoders on EEC. This large gap between the complexity and the reliability has motivated us to work on better performing BF algorithms on error channels. Also, the incapability of the BF decoders on EEC has motivated us to work on methods for EEDG with BF algorithms.

1-3 Problem Statement

As stated in the motivation section, the main focus of this research is the complexity-reliability trade-off of decoders for linear block codes, and especially LDPC codes. These methods can be used for low-end devices that do not require highly reliable information, but have limited resources (computational or battery). The two main aims of the research are the improvement of the reliability of existing low complexity decoders on error channels, and proposals of less complex decoders for EEC with acceptable reliability. We have selected the LDPC codes to work on, since LDPC codes are widely used for error correction in many applications, and the family of LDPC codes is a very hot research topic.

The first aim of this research is the low complexity LDPC decoders that can achieve reasonable BER performance, which can be realized by working on BF decoders, that are known to be achieving reasonable BER performance with little complexity. Especially soft decision BF algorithms are researched by various researchers on the AWGN channel, and shown to be effective with low complexity. We have worked on existing soft decision BF decoders, and tried to come up with a more reliable soft decision BF decoder with a very little increase in complexity.

The second aim of this research is to design simpler decoders for linear block codes, especially for LDPC codes on EEC. To create such decoders, we have used an idea that is introduced in [1]; the separation of erasures from errors. This idea uses puncturing and shortening operations on the code, and lets us combine an error decoder with an erasure decoder to create an Error Erasure Decoder (EED). However, this idea is not realizable for the reasons that will be explained in Chapters 2 and 3. Thus, a new method to separate the erasures

from errors is also needed to be researched before applying this idea. After such a method can be found, then this idea might be realized, and proposed as an alternative to existing EEDs. Besides this idea, simpler alternatives for EEDs, such as guessing of erasure positions are also considered in this research. Thus, we have searched for new guessing decoders that might perform better than previously existing guessing algorithms.

To conclude, our aims are;

- Studying the existing BF decoders on error channels, and searching for improvements that narrow the gap in the reliability between the MP decoders and BF decoders.
- Finding a realizable method to separate erasures from errors.
- Creating new EEDs that have low complexity and reasonable reliability (BER performance), which might be alternatives to existing high complexity EEDs (such as MP decoders).

1-4 Contributions and Thesis Outline

The contributions of this thesis are;

- A minor modification to the RRBF algorithm which results in better BER performance with very little complexity added.
- The rank completer method, which is a new and realizable way to separate errors from erasures.
- Implementation of new EEDs with less complexity and reasonable BER performance.

The outline of the thesis is;

- Chapter 2 **Background Information:** Some background information on LDPC decoding is given in detail, the channel models that are used in the thesis are introduced, and finally the idea of "Separating Errors from Erasures" is given.
- Chapter 3 **Literature Review:** Three main parts is presented in the literature review, research related to error-erasure decoding, research related to BF and RRBF.
- Chapter 4 **Modified Reliability Ratio Based Bit Flipping Algorithm:** We present our modified version of the RRBF algorithm, Modified Reliability Ratio Based Bit Flipping (MRRBF) and compare its BER performance using different parity check matrices.
- Chapter 5 **The Rank Completer:** We present the new method of separation of erasures from errors, which is realizable and does not require creation of "separating matrices".

- Chapter 6 **New Error Erasure Decoding Methods:** We present two methods for EEDG that have been originated from existing ideas, and compare BER performances of these methods on different channels.
- Chapter 7 **Conclusion and Future Work:** Summary of the results and the possible future work is given.

Chapter 2

Background

In this section we first present fundamental techniques for LDPC decoding and then we move on to the channel models that are used in this research. Finally, the "Separation of Erasures from Errors" idea is explained, since it is the basis of this work.

2-1 Decoding methods for LDPC codes

In this section, common decoding techniques are introduced for LDPC codes. Firstly, bit-flipping algorithms, message-passing algorithms, and linear programming decoding algorithms are discussed, and afterwards exhaustive and iterative erasure decoding algorithms are presented.

2-1-1 Bit-Flipping Algorithms

In [4], Gallager has introduced a simple BF algorithm to decode LDPC codes, but did not analyze the performance of that algorithm extensively. Instead, he focused on binary message passing algorithms for hard-decision decoding and the belief propagation algorithm for soft-decision decoding, since they perform better in the BER performance perspective. The BF algorithm has been proposed again in [7] with a guarantee of correct decoding result, if number of erroneous bits is bounded by a constant depending on the code.

The BF algorithm can be described as follows; the codeword bit with the highest metric is flipped in every iteration. Calculation of this metric divides the BF algorithms into two classes; hard-decision BF and soft-decision BF algorithms. For hard-decision BF, the metric depends only on the number of unsatisfied parity check equations since no reliability information is available. For soft-decision BF, the metric is calculated by the number of satisfied and unsatisfied parity check equations and their corresponding weights which is also decided by the decoding algorithms. Types of soft-decision BF algorithms will be given in Chapter 3 in a detailed fashion.

Another property that separates BF algorithms is the number of bits flipped per iteration. If a single bit is allowed to be flipped in a single iteration, then the algorithm is called single BF algorithm. If multiple bits are allowed to be flipped in a single iteration, then the algorithm is called multi BF algorithm. Single BF algorithms need much more iterations to decode a codeword than multi BF algorithms, but they are more resistant to decoding loops than multi BF algorithms also.

To finalize the introduction to BF algorithms, we give a simple hard-decision single BF algorithm to help the reader understand the structure of BF algorithms.

1. Compute the parity check equations, if all of them are satisfied, then stop decoding and output the codeword.
2. Find the number of unsatisfied parity check equations for each variable node.
3. Flip the codeword bit with largest number of unsatisfied parity check equations, select randomly in case of an equality.
4. Repeat steps 1-3 until maximum number of iterations is reached.

2-1-2 Message-Passing Algorithms

The second type of the decoding algorithm that Gallager has proposed in [4], was message-passing (MP) algorithm. He realized their superiority to BF algorithms and proposed two main types of decoders using MP in his paper; hard-decision MP decoding and soft-decision MP decoding. We will describe these algorithms after presenting some fundamental background information and a generic message-passing decoder.

To understand MP algorithms in general; one should be familiar with the term *extrinsic information*, which means that the data sent to a node depends only on other nodes. In other words, extrinsic information should not be influenced by the node that it is sent to. In a good MP algorithm the messages should contain *extrinsic information* only, therefore the messages are calculated by excluding the information received from the node that the message is intended. Messages that contain intrinsic information might result in a situation where every node is dominated by its current value (previous message is the current value).

A generic message-passing decoder is given below.

1. Initialize the decoder by sending the received values of variable nodes to check nodes.
2. In each check node, a message for each neighbor variable node is calculated and sent.
3. Each variable node calculates a new value for itself depending on the received value and the extrinsic information from the check nodes. The output is checked by parity check equations and if all are checked, then decoding is over. Otherwise, a message for each neighbor check node is calculated and sent.
4. Repeat steps 2-3 until maximum number of iterations is reached.

Now, we can describe the two main types of message-passing algorithms by defining the method of calculating the messages and the size of message alphabet. The first type is hard-decision MP algorithm (Gallager Algorithm B), which uses a binary alphabet {-1,1} for the messages and the node values. The other type is a soft-decision MP algorithm, and is called the Belief Propagation (BP) algorithm, which uses real numbers for messages and node values [4]. Min-Sum (MS) algorithm should also be introduced here, which is an approximation of the BP algorithm, with less computational complexity.

These two algorithms use the same idea; where soft-decision MP utilizes the soft information of the received values to calculate (soft) messages, and hard-decision MP uses only the sign of the received information to calculate (hard) messages. Thus, when soft information is available it is obvious that soft-decision MP algorithms outperform the hard-decision MP algorithms. However, the complexity of the calculations and real valued message transmission should be considered.

2-1-3 Linear Programming Decoding Algorithms

Linear Programming (LP) decoding corresponds to a general family of algorithms that tries to find the maximum likelihood codeword via linear programming in feasible time. Since Maximum Likelihood (ML) decoding is an Non-deterministic Polynomial-time (NP)-hard problem, LP decoders offer sub-optimal solutions to the estimation of the ML codeword. Two general methods for LP decoding are introduced in [23] and [24]. Here we shortly explain the method that has been described in [24], which has been studied by many researchers. This method is based on linear programming relaxation, which results in a more manageable representation. Although the details are omitted here, the representation is acquired by defining local codes for each check node (any set containing even number of ones in the parity check equation) and a global code that is the intersection of all local codes. This procedure defines a larger codeword set that contains the set of actual codewords and some non-integer codewords. This set is called pseudocodewords set, and from this set ML solution can always be found. However, for some cases ML solution is non-integer, which shows that LP decoding fails. One of the drawbacks of the LP decoding is the polynomial complexity of the algorithm. LP decoding is out of the scope of this research due to its complexity, and we present it here for completeness and to form a basis for one of the algorithms that is discussed in Chapter 3.

2-1-4 Erasure Decoding Algorithms

In this section we present erasure decoding algorithms that can correct erasures. First we present the exhaustive decoding technique which is not practical and not used. Then we present the iterative decoding technique which is realizable, but performs a little bit worse than the exhaustive method.

Exhaustive erasure decoding: Exhaustive decoding can correct every set of bit erasures, as long as the set of erasures do not contain the support of a non-zero codeword. The support of a non-zero codeword is the position of non-zero values in the codeword. In this method, a received word with erasures in flagged positions can be decoded by comparing every possible codeword to the received word using the bits that are not erased. This simple method is effective, however becomes infeasible when the number of codewords are large. This can be

explained by an example clearly. Using the previously introduced (7,3,4) simplex code, the channel encoder transmits the codeword $c_s = 0001111$, and some of the bits are erased in the channel. The received codeword is $c_r = 0x01x1x$, where x denotes an erased bit. After searching a candidate in all of the possible codewords from (1-1),the codeword is decoded as $c_d = 0001111$. To sum up, it should be noted that the exhaustive decoding can correct any pattern of erasures, as long as there's only one codeword that can be decoded from the received word. If the support of a codeword is erased, it is easy to see that the received word can be decoded to more than one words, and such an erasure set is called incorrigible set [25].

Iterative erasure decoding: Iterative decoding methods correct erasures one-by-one using the parity check matrix or the Tanner graph of the code. The algorithm is explained in three steps below.

1. Store the erasure list.
2. Search for a parity check equation that checks a single erasure position from the erasure list. If such an equation is not found, then stop decoding with unresolved erasures.
3. Correct the erasure position using the found parity check equation, and update the erasure list.
4. Unless the erasure list is empty, go to step 2.

The iterative decoding algorithm depends on not only the position of the erasures, but also the parity check equations to be able to check erasures in a single position. Therefore the set of erasures that causes unresolved erasures always contains the incorrigible set, but is not always equal to it. These sets are called stopping sets and are first analyzed in [26]. More information on stopping sets and incorrigible sets and their effects on iterative decoding can be found in [27].

2-2 Channel Models

In this section the channel models that are used in the research are presented. We start by describing the channel models for hard-decision decoding, and then go on with the channel models for soft-decision decoding.

2-2-1 Binary Erasure Channel

Binary Erasure Channel (BEC) cause only erasures in the transmitted codeword. The model is described by the bit erasure probability, which we call p_1 in this thesis. Each bit of the codeword is received correctly with probability of $1 - p_1$ and is erased with probability of p_1. The channel model is given in Figure 2-1.

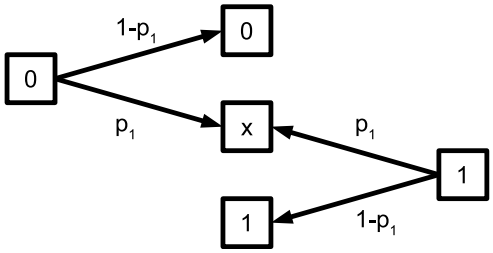

Figure 2-1: The channel model of BEC

2-2-2 Binary Symmetric Channel

Binary Symmetric Channel (BSC) cause only errors in the transmitted codeword. The model is described by the bit error probability, which we call p_2 in this thesis. Each bit of the codeword is received correctly with probability of $1 - p_2$ and is received erroneously with probability of p_2. The channel model is given in Figure 2-2.

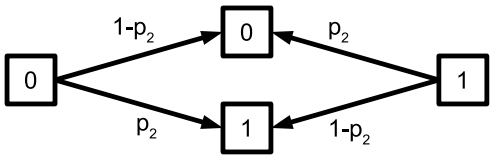

Figure 2-2: The channel model of BSC

2-2-3 Binary Symmetric Error-Erasure Channel

BSEEC cause both errors and erasures in the transmitted codeword. The model is described by the bit erasure probability p_1 and bit erasure probability p_2. Each bit of the codeword is received correctly with probability of $1 - p_1 - p_2$, is erased with probability of p_1, and is received erroneously with probability of p_2. The channel model is given in Figure 2-3.

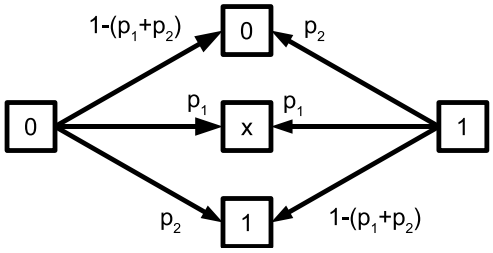

Figure 2-3: The channel model of BSEEC

2-2-4 Additive White Gaussian Noise Channel

AWGN channel is an additive noise channel, and it can not be explained in the same manner with the above channels. Instead of the logical values (0 and 1) which have been used to

describe the channels above, real values should be considered to explain the AWGN channel. Assigning {-1,+1} to the binary bit values {1,0}, the channel input is a stream of values from {-1,+1}. The channel output is modeled as adding white Gaussian noise (Gaussian distributed variable with zero mean and non-zero variance) to the input values. The effect of AWGN channel on bits is given in Figure 2-4. The effect of the channel can also be formulated as in (2-1), where $\widetilde{N}$ is a Gaussian random variable, $\widetilde{X}$ is the channel input, $\widetilde{Y}$ is the output, and σ^2 is the variance of the Gaussian random variable. The AWGN channel is described by the ratio of the signal power to the noise power, $SNR = \frac{\overline{x}_n^2}{\sigma^2}$, with $\overline{x}_n$ as symbol energy. However to compare different coding schemes with different rates, $E_b/N_0 = \frac{\overline{x}_n^2}{2 \cdot R \cdot \sigma^2}$ is used, where N_0 is the noise spectral density, E_b is energy per bit, and R is the code rate [28]. In this thesis, the SNR value is used for E_b/N_0 value, and the change in the code rate is taken into account while calculating the SNR value.

$$\widetilde{Y} = \widetilde{X} + \widetilde{N}(0, \sigma^2) \tag{2-1}$$

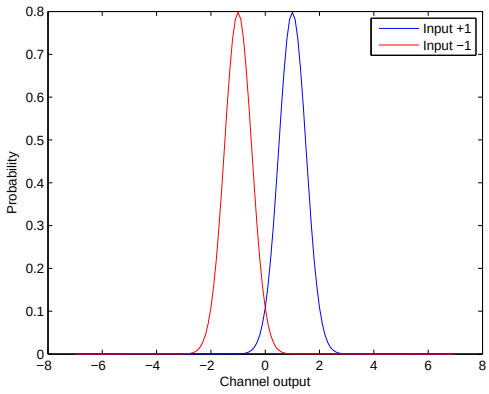

Figure 2-4: The probability distribution of AWGN channel output

2-2-5 Additive White Gaussian Noise Channel with Rayleigh Fading

This channel is simply the AWGN channel with multipath fading effects, therefore fundamental information on Rayleigh fading should be given to explain it. Rayleigh fading is a model to describe the small scale fading (multipath fading) in the transmission of electromagnetic waves in wireless communications [29]. The model has been named after the Rayleigh distribution, pdf of which is given in (2-2), since in such a channel the amplitude of the signal fades according to this distribution. We consider uncorrelated and fast fading, and model the amplitude distortions by multiplying each transmitted bit with a Rayleigh distributed random variable $\widetilde{a}$. The receiver is assumed to be capable of handling distortions due to phase differences, which is also a model used in [30]. The described channel can be formulated with a slight modification to (2-1), which is given in (2-3). AWGN with Rayleigh fading channel is also described by the SNR (which is defined the same with E_b/N_0 in this thesis) value, assuming the fading model does not disturb the mean value of the SNR, but only causes fluctuations around it. The distortions by the Rayleigh fading result in a significant amount of degradation in the channel capacity. The effect of this degradation on BER performance is

shown in Figure 2-5, where compare the uncoded AWGN channel to uncoded AWGN channel with Rayleigh fading.

$$f(x) = \frac{x}{\sigma} e^{-(x)^2/2\sigma^2} \tag{2-2}$$

$$\widetilde{Y} = \widetilde{a} \cdot \widetilde{X} + \widetilde{N}(0, \sigma^2) \tag{2-3}$$

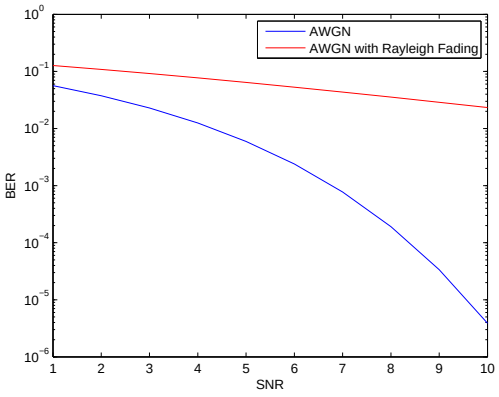

Figure 2-5: The comparison of uncoded AWGN channel to uncoded AWGN with Rayleigh fading channel

2-2-6 Additive White Gaussian Noise Channel with Erasures

AWGN channel with erasures (AWGN+EC) is described by the erasure probability p_1, and SNR. The channel can be realized as an erasure channel cascaded to an AWGN channel, which is given in Figure 2-6. The erasure channel in the block diagram differs from the BEC in the channel inputs and outputs. It accepts real numbers as inputs from the AWGN channel and outputs either erasures or real values (from AWGN channel). As it is illustrated, first additive noise is added to x[n] which forms y[n], the output of the AWGN channel. Then y[n] is fed as an input to the erasure channel, and bits are erased according to the erasure probability p_1. The output of the channel is z[n], which contains erasures and real values.

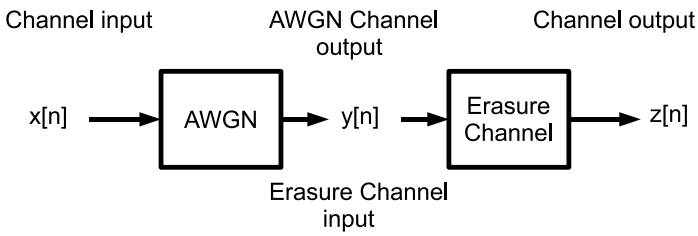

Figure 2-6: The channel model for AWGN+EC

The effect of the channel is given below with a sample word that is sent through the channel for clearer understanding of the reader. First the binary codeword is selected by the channel encoder, regarding the source message to be sent. Then, this codeword is mapped to real numbers according to a modulation scheme. The modulation and its effects are out of the scope of this thesis, therefore we assume Binary Phase Shift Keying (BPSK) modulation which maps binary one to -1 and binary zero to 1. After the BPSK modulation, AWGN with variance σ^2 is added to the channel input x[n], which gives y[n]. In the final step, bits are erased according to the erasure probability p_1, and the channel output z[n] is created.

$$
\begin{aligned}
c &= 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \\
&\downarrow \\
x[n] &= -1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad 1 \\
&\downarrow \\
y[n] &= -0.8 \quad -0.1 \quad 0.7 \quad 1.5 \quad 1.1 \quad -0.5 \quad 0.4 \quad 0.2 \quad 1.7 \quad 0.7 \\
&\downarrow \\
z[n] &= -0.8 \quad x \quad 0.7 \quad x \quad 1.1 \quad -0.5 \quad 0.4 \quad x \quad 1.7 \quad 0.7
\end{aligned}
$$

2-3 Separation of Erasures from Errors

The "Separation of Erasures from Errors" idea is introduced in [1], which works for linear block codes. If the Hamming distance d of the code satisfies $d - 1 \geq 2 \cdot t_{\#} + t_{?}$, then a received codeword with $t_{?}$ erasures and $t_{\#}$ errors can be corrected [31]. Using this idea an error-correcting decoder and an erasure correcting decoder algorithm can be used together to deal with an error-erasure channel.

An algorithm that uses this idea can be explained as follows;

1. The erasure positions are deleted from the original code, which creates a punctured code that does not check on erased bits, but checks on the rest of the bits. In other words, a new parity check matrix for the punctured code should be created from the original parity check matrix.
2. The error correcting decoder is used to correct errors in the punctured codeword which does not contain any erasures.
3. After the error correction is finished, erasure correcting decoder is used to correct the erasures and the decoding is complete.

As stated above, the separation idea uses a different punctured code for each set of erasures. To clarify the process, we give formal definitions of puncturing and shortening of codes.

Puncturing: When a code is punctured; the dimension k is kept constant and the length n is decreased for each bit that is shortened, unless the support of a non-zero codeword is punctured [32]. If the support of a non-zero codeword is punctured, then k is also decreased

for each linearly independent non-zero codeword support that is punctured. Thus, if we puncture a bits from a (n, k) code and no non-zero codeword supports are punctured, then we end up with a $(n - a, k)$ code. If we puncture a bits from a (n, k) code and b linearly independent non-zero codeword supports are separated, then we end up with a $(n - a, k - b)$ code. By puncturing, we usually have a more efficient (higher code rate) coding with less error correction capability than the previous code (d is usually decreased). Puncturing is simply erasing columns of the generator matrix of the previous code. One-bit punctured version of the (7,3,4) simplex code is given in (2-4), whose generator matrix is previously given in Chapter 1, (1-1). As it can be seen from the new generator matrix, the new code has $n = 6$, $k = 3$, and $d = 3$.

$$C_{sp} = \left(\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right) \tag{2-4}$$

Shortening: When a code is shortened; n and k are decreased for each bit that is shortened, unless the support of a non-zero codeword of the dual code is shortened [32]. If the support of a non-zero codeword of the dual code is shortened, then k is increased for every linearly independent non-zero codeword support of the dual code that is shortened (remember k is decreased by one for each shortened bit). Thus, if we shorten a bits from a (n, k) code and no non-zero codeword supports of the dual code are shortened, then we end up with a $(n-a, k-a)$ code. If we shorten a bits from a (n, k) code and b linearly independent non-zero codeword supports of the dual code are shortened, then we end up with a $(n - a, k - a + b)$ code. It is easy to observe that via shortening, we usually end up with a lower rate code than the previous code with the same error correcting capability. Shortening is done by using some of the information symbols, and forcing the other information symbols to be zero, excluding any row that has a non-zero entry in the shortened positions. Afterwards, the column with all zero entries is erased and the shortening operation is done. One bit shortened version of the (7,3,4) simplex code is given in (2-5). As it can be seen from the new generator matrix, the new code has $n = 6$, $k = 2$, and $d = 4$.

$$C_{ss} = \left(\begin{array}{cccc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right) \tag{2-5}$$

An important connection between these two operations is the fact that they are dual operations, which is given in [33]. Therefore we can claim that if we puncture the codeword, then we can work on the dual code (parity check matrix of the code) by shortening the parity check matrix. We will exploit this fact while explaining the separation idea using parity check matrices.

The separation idea punctures the received codeword in the case of erasures, and then decodes the punctured codeword using the parity check matrix of the punctured codeword. By using the duality of puncturing and shortening operations, we claim that the original parity check matrix should be shortened using the exact same bits that are punctured (erasure positions). Mathematically this is equivalent to saying if we puncture a code which is (n, k, d) in l positions, then we will have a new code that is $(n - l, k, d - l)$. Now a valid parity check matrix or the dual code should be $(n - l, n - k - l, d^{\perp})$ where $d^{\perp}$ is the distance of the dual code. In other words the new parity check matrix should have a rank of $n - k - l$, which we call the *rank property*. Such a matrix can be generated by shortening, if the original parity check matrix has enough number of rows with a zero in the shortened positions.

This can be explained clearly with examples, using the extended Hamming code (8,4,4). A parity check matrix for the code is given in (2-6). The part of the matrix that is above the line is enough for simple error decoding, however it is shown that for the separation idea having extra rows is beneficial.

$$H = \left(\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right) \tag{2-6}$$

If the received codeword is $0x011110$, then the codeword is punctured in the second position, and the parity check matrix is shortened in the exact same positions. By the properties of puncturing operation, we know that the new code should be $(n = 7, k = 4, d = 3)$ and the parity check matrix should have a rank of 3. If we use the small portion of the parity check matrix which has size 4×8, then the resulting parity check matrix has two linearly independent rows, therefore the rank is 2. However if the whole parity check matrix is used, then we have a matrix that has three linearly independent rows out of four rows, which satisfies the needed rank. The resulting parity check matrices are given in (2-7) and (2-8).

$$H_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{2-7}$$

$$H_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{2-8}$$

The new parity check matrix is created by shortening operations as above, which concludes the step one of the algorithm. Now we move to step 2, and apply error decoding with the new parity check matrix and the shortened parity check matrix. Although we do not show here, using H_1 which lacks the required rank might cause wrong decoding results. Thus, we use H_2 to decode the punctured word, 0011110. The closest codeword (of the punctured code) to this word is 0011010, which has distance one, and the error decoder outputs this codeword.

This result can be checked by multiplying all other distance one words with the parity check matrix.

After error decoding step is done, the algorithm goes to step 3, erasure correction. The erasure position can be corrected using the original code, if the iterative erasure decoding is applied, the erasure position can be identified as a one using the third row of the original parity check matrix. Hence, the final decoding result is 01011010.

Using the separation of erasures from errors, an error decoder and an erasure decoder can be used for EEDG. The novelty in this approach is the systematical method of puncturing the codes by shortening the parity check matrices. In [1], the authors have proposed the usage of separating matrices that have enough rows to satisfy the rank condition $(n - k - l)$ for all erasure sets of maximum size l. The separating matrices method will be given in Chapter 3.

Chapter 3

Literature Review

The literature review is divided into three main parts, each part related to one main contribution of this thesis. In the first part, a method to realize the idea of "Separation of Erasures from Errors" is given, which is called "Separating Matrices". In the second part, various EEDG methods are explained thoroughly. In the last section, types of Bit-Flipping algorithm are presented.

3-1 Separating Matrices

The Separating Matrices (SM) idea has been introduced in [1] with the separation idea. In the simplest manner it can be thought as creating matrices that can separate erasure sets that have size l or less. The authors define the parity check matrix as a matrix which typically has redundant rows, and contains the parity check matrices of all codes that are punctured up to a fixed number of symbols. In this research, this number of fixed symbols is called l, and it is known to be at most $d-1$, since $d-1$ is certainly the highest number of erasures that could be surely corrected by a code that has a Hamming distance d.

The authors defined l-separating matrices, if the matrix can separate any erasure set that has size l or less. They have used the notation $H(S)$ for a parity check matrix that is shortened according to set S, and they have stated that a parity check matrix separates a set S iff all $H(S)$ has the rank of $n-k-|S|$ where $|S|$ is the size of the erasure set in Lemma 2. Also they have show that if all of the erasure sets of size l can be separated by a parity check matrix, then all of the smaller erasure sets are also separable and the parity check matrix is l-separating in Lemma 3.

After presenting these results, they have shown that a parity check matrix that is l-separating has no stopping sets of size l or less in Theorem 1. Therefore if one uses the separating matrices idea, stopping sets are not an issue for the iterative decoder.

They have also found two construction methods for separating matrices which are not presented here. Using these methods separating matrices can be created for any value of l for

$l < d$, and for any linear block code. However, the number of rows in these matrices are also important for decoding purposes, since it is not efficient to store and use very large parity check matrices. Thus they have defined the term separating redundancy, which is the minimum number of rows of an l-separating parity check matrix of an arbitrary block code.

The upper bounds on separating redundancy of the two constructions are given below, where q is the alphabet size of the code.

$$s_{l1} \leq \begin{cases} \binom{n}{l} \cdot (n-k-l) & \text{if } l \leq n-k-1 \\ \binom{n}{l-1} & \text{if } l = n-k \end{cases}$$

$$s_{l2} \leq \begin{cases} \sum_{i=1}^{l+1} \binom{n-k}{i} \cdot (q-1)^{i-1} & \text{if } l \leq n-k-1 \\ \sum_{i=1}^{l} \binom{n-k}{i} \cdot (q-1)^{i-1} & \text{if } l = n-k \end{cases}$$

The authors have also presented a general, non-constructive, lower bound for separating redundancy, which is given below, where $d^{\perp}$ is the distance of the dual code (parity check matrix).

$$\frac{\binom{n}{l}(n-k-l)}{\binom{n-d^{\perp}}{l}}$$

The constructive upper bounds and the non-constructive lower bound are separated by a large gap. The constructive methods end up with matrix sizes that are not feasible when values of n or l are large. Therefore in [34], the authors have shown other constructive methods with lower separating redundancy. The amount of improvement is significant for very small values of l and n, however as these values increase, the difference is negligible.

3-2 Error-Erasure Decoders

There are many types of decoding methods for error-erasure channels; here we first present a general algorithm that is applicable to all types of codes, then we move on to code-specific algorithms mainly focusing on algorithms for LDPC codes.

3-2-1 Guessing Algorithms

The simplest guessing algorithm that can be used to deal with errors and erasures, is randomly guessing each erased bit in the codeword. By this method, a codeword that has $t_?$ erasures and $t_\#$ errors is transformed into a codeword with $t_\# + \tilde{g_e}$, where $\tilde{g_e}$ is the number of erroneously guessed bits. It is trivial to show that $E[\tilde{g_e}] = \frac{t_?}{2}$, with a binary equiprobable alphabet, and thus it can be concluded that in the mean value we have $e = \frac{t_?}{2} + t_\#$ errors in the codeword. Since only errors are left in the codeword, any simple error decoder can correct the errors as long as e is small enough (small in the sense of the distance of the code and decoder

capabilities). However, by using the methods that will be explained in this section, we can do much better.

In [31], a guessing algorithm with multiple trials that is capable of correcting errors and erasures together, which we call trial guessing, is introduced. The algorithm uses hard-decision decoding, and the number of correctable errors and erasures are bounded by $d-1 > 2 \cdot t_{\#} + t_{?}$, which has been introduced before. First we give the binary version of the algorithm, which is simpler, and then generalize it for all symbol alphabets.

The binary algorithm starts by creating two vectors; one with zeros in all $t_?$ erasure positions, and the other with ones in all $t_?$ erasure positions. Now either two vectors have $\frac{t_?}{2} + t_{\#}$ errors (this is only possible in the case there are equal number of zeros and ones erased in the transmitted message), or one of the vectors contain less errors $\left\lfloor \frac{t_?}{2} \right\rfloor + t_{\#}$. After the erasure positions are filled, error correction is applied to the two vectors. Since, at least one vector contains less than or equal to $\left\lfloor \frac{t_?}{2} \right\rfloor + t_{\#}$ errors, then error correction will certainly work for one of the vectors. If both of the vectors are corrected to the same result, then this is the final decoding result. If two results are not the same, then the one that differs at most $t_{\#}$-positions in the non-erasure positions (there is only one) is the final decoding result.

If the binary algorithm is generalized to be used with larger alphabets of size q, one should create q vectors to fill in the erasure positions with vectors 00...0, 11...1, ..., $(q-1), (q-1), ..., (q-1)$. However, such a guessing can only guarantee words that contain less than $\left\lfloor \frac{(q-1) \cdot t_?}{q} \right\rfloor + t_{\#}$ errors. It is obvious that the performance of such an algorithm converges to the performance of the simple guessing algorithm. In [31], another algorithm is defined that is superior to the simple guessing algorithm. In this algorithm, the decoder tries up to $q^{t_?}$ vectors to find the valid codeword. Due to the exponential nature of the number of trials, this algorithm is not practical, unless the alphabet size and the number of erasures are small.

3-2-2 Code-specific error-erasure decoders

In the previous sub-section, we have introduced a general method that is applicable to all of the codes. Here, we will present methods that can be applied to certain code types. First we will give examples from a large code family that has been used to deal with error and erasure channels; Bose-Chaudhuri-Hocquenghem (BCH) and RS codes. Then we will move on to the methods for error-erasure decoding in LDPC coding which is more relevant to our research, regarding the implementation of "Separating the Errors from Erasures" idea for LDPC decoding.

The first case is RS codes, which have been introduced in [35] in 1960. Soon after their discovery, it has been shown that RS codes are a sub-class of Bose-Chaudhuri-Hocquenghem (BCH) codes. Therefore we present not only the research on RS codes, but also on BCH codes here, since the work on BCH directly applies to RS codes. RS codes are generally decoded by the Berlekamp-Massey Algorithm (BMA), which has been introduced in [36]. BMA was not introduced as an error-erasure decoder, so it could handle errors or erasures separately. However, Forney has shown that the BCH codes (which also implies to RS codes) are capable of error-erasure decoding in [37]. Forney's method starts by computing an errata locator polynomial which will give the error and erasure locations in the codeword. After all the

errors are located, they are declared as erasures, and a simple algorithm is used to correct the erasures.

Many other algorithms for RS codes have been proposed after Forney's initial error-erasure decoder. One of the most important proposal is in [38], which is a method to compute the errata locator polynomial by initializing the BMA with the erasure locator polynomial. Another approach which uses the Euclidean algorithm instead of BMA for error-erasure decoding of RS codes is given in [39]. In [40], the author has shown that the BCH bound for the correctable number of errors and erasures can be improved to the Hartmann-Tzeng bound [41]. Improvements related to the complexity of the Forney approach has been studied and given in [42] and [43], where the authors propose simpler methods to calculate the errata locator vector. There are many other works that offer less complexity, easier implementation, or a new approach to the error-erasure decoding problem of RS codes, such as [44], [45], and [46].

For LDPC codes, message-passing algorithms (except for the binary message-passing decoder) can be used for error-erasure decoding. First type of the algorithms that can be used for error and erasure decoding is the belief propagation algorithm, and its approximation; the min-sum decoding. If the erasures are fed to the input as bits with zero reliability, then after a few iterations the erased bits would acquire messages from their non-zero (not erased) neighbors and the error-erasure decoding can be done without any extra calculations (if the erasure set is not a stopping set). Methods to design LDPC codes for AWGN+EC are discussed in [17] using belief propagation algorithm. It is shown that the performance of LDPC codes can be improved for AWGN channels with erasures by using LDPC codes that are designed using the methods described. In [21], several standard LDPC codes are compared to the codes designed by the authors in the AWGN channel with erasures, using the belief propagation algorithm. In [47], BP algorithm is again used for error-erasure decoding. BP algorithm and its variants seem to be the generic decoding mechanism nowadays, but they have a high complexity due to the messages and calculations that include real numbers.

Second type of the message-passing algorithms that can be used for error-erasure decoding are the quantized belief propagation algorithms. The simplest of these algorithms is called the Binary Message Passing with Erasures (BMP-E); which has been first described in [48] and then independently re-invented in [49]. In [49], the performance of BMP-E (which is also called Gallager E algorithm and other quantized versions of BP algorithm has been given through probabilistic analysis. These algorithms are much more simpler than the regular BP algorithm, since the number of possible messages are very limited and the calculations are simpler. However, these algorithms still require message transmission and are still complex for low-end devices.

Another error-erasure decoding algorithm for LDPC codes is given in [50], which uses linear programming decoding which has been introduced before. The method includes a slight modification in the algorithm with nearly no additive complexity to the plain LP decoder. The algorithm performs well, but the high complexity of the LP decoding should again be considered.

3-3 Bit-Flipping Algorithms

In this section, research related to bit-flipping algorithms is given. The main subjects are decoding methods that utilize soft information, a pre-processing method called bootstrapping, and multiple bit-flipping algorithms.

One of the first of the decoding methods which utilizes soft-information is called the Weighted Bit Flipping (WBF) algorithm [51]. The main difference of this algorithm to the simple BF algorithm that has been given before is the utilization of the received values (reliabilities) of the bits. In the previously introduced algorithm, the bit that is connected to the highest number of unsatisfied parity check equation is flipped. In this one, a more complex metric that improves the BER performance of the decoder is used. Each parity check equation is assigned a reliability value, which is the minimum of the absolute value of all of the received values of the connected variable bits. After the reliability values are assigned, the flipping metric is calculated for each bit. As it can be seen the major difference between the simple BF algorithm and the WBF algorithm is weighing the parity check equations. Also in WBF algorithm satisfied parity check equations play a role in the calculation of the metric, while in simple BF only the number of unsatisfied parity check equations is relevant.

An improvement to the weighted bit flipping algorithm is given in [52], which also uses the variable bit reliability while calculating the flipping metric. The flipping metric depends on both the minimum reliability values in the parity check equation and the reliability of the bit that the metric is calculated. The current bit reliability is weighed with a multiplicative constant α that can be determined via Monte Carlo simulations. Modified Weighted Bit Flipping (MWBF) offers a better BER performance up to 0.5 dB with a value of α that is close to optimal.

In [53], the calculation method for the flipping metric has been improved using the extrinsic information idea. The parity check equation reliability is selected from a set that excludes the variable bit that the metric is being calculated. It is shown that Improved Modified Weighted Bit Flipping (IMWBF) performs better than MWBF for some LDPC codes in [53], with a little increase in the amount of complexity.

Another bit flipping algorithm that uses soft information is reliability ratio based bit flipping (RRBF) algorithm [2]. In the RRBF algorithm, bit flipping decision is based on the ratio of the reliabilities of the variable bits that are element of a parity check equation. The calculation methods were a little complex in the original method, therefore in [54] the authors have suggested a simpler method to realize the algorithm. In this method, every variable bit in a parity check equation is assigned with a metric that is the ratio of the sum of the reliability values of the variable bits in that parity check matrix to the reliability value of the variable bit considered. The RRBF algorithm has been shown to outperform than MWBF for various codes in AWGN channels with a comparable complexity. However in uncorrelated Rayleigh fading channels, RRBF might be outperformed by MWBF depending on the code [2].

In [55], an algorithm that is close to WBF family algorithms in complexity is presented. The algorithm is called Liu-Pados Weighted Bit Flipping (LP-WBF), and is known to perform well especially for finite geometry LDPC codes. This algorithm uses the least reliable and the most reliable variable bit in a parity check equation to calculate the metric.

There are many other kinds of bit flipping algorithms that offer different amount of complexity, preprocessing, and BER for various codes, but we do not present all of these algorithms here.

The final BF algorithm that is presented is a self reliability based algorithm. The metric is computed from the number of satisfied and unsatisfied parity check equations, and the self-reliability values of the variable bits [56]. This algorithm performs as good as MWBF and indicates the importance of the self reliability in the flipping decision.

After presenting some of the important soft decision bit flipping decoders, now we shall move on some methods that can be applied to the BF decoders for different purposes, such as; lowering the number of iterations, and improving the BER performance.

The first method is cycle prevention (loop break) which lowers the number of iterations, and improves the BER performance of the BF decoder. The infinite loops occur when a bit flipping selection of size k results in the word in the beginning. Such a pattern will repeat itself, unless the selection of "the bit to be flipped" is changed to prevent the cycle. An algorithm that stores a vector of changed bits to detect and prevent loops is introduced in [55]. A simpler version of this algorithm is given in [57], which does not prevent all the loops but is shown to be effective. This simple algorithm compares the number of unsatisfied parity check equations before and after the flipping of every bit. In the case of an increase in the number of unsatisfied parity check equations; the flipped bit is inverted, and the bit with the highest flipping metric, excluding the initially flipped bit(s) is inverted.

The second method is multiple (parallel) bit flipping, which lowers the number of iterations of the BF decoder. The parallel bit flipping can be realized in different ways, the authors in [58] define a mode for the decoder, which can select multiple bit flipping or single bit flipping. In the multiple bit flipping mode bits that are smaller (in the sense of the selection metric) than a pre-defined threshold are flipped. In [59], $p = \left\lfloor w(s^{(k)})/w_c \right\rfloor$ bits are flipped in each iteration where $w(s^{(k)})$ is the weight of the syndrome of iteration k, and w_c is the column weight of the parity check matrix. In [60], a voting algorithm is used, that can be used with any metric of bit flipping algorithms. Each unsatisfied parity check equation votes for the the least reliable bit in it (depending on the metric) in each iteration. After voting the bits with more than pre-determined $F_{th}^{(k)}$ votes are flipped in parallel, where k is the iteration number again.

The last method that will be explained is a pre-processing method, that lowers the number of iterations and improves the BER performance of the decoder. The bootstrapping method, which has been introduced in [61], functions by reassigning new reliability values to unreliable bits. The algorithm uses a threshold value to select unreliable bits, and receives messages from reliable parity check equations. A parity check equation is reliable if it contains less than or equal to one unreliable variable bit. This method uses the exact same formula with the MS algorithm to calculate the messages to be passed to unreliable bits. The bootstrapping method is shown to be effective WBF in AWGN channels [61] and AWGN channels with Rayleigh fading [62]. The method has also been applied to multiple bit flipping methods [63].

Chapter 4

Modified Reliability Ratio Based Bit Flipping Algorithm

In [2], the reliability ratio based weighted bit flipping algorithm has been introduced, which is one of the best soft decision bit flipping decoders amongst MWBF, LP-WBF, and others. In (4-1), we give the calculation method of the metric, where E_i is the flipping metric (the bit with the maximum flipping metric is flipped), y_i is the received value of the bit for which the metric is calculated ($|y_i|$ is the reliability value of the bit), T_j is the summation of the reliabilities of the bits that are elements of the parity check equation, s_j is the syndrome corresponding to the parity check equation j, and S_i is the set of parity check equations that the variable bit i is an element of.

$$E_i = \frac{1}{|y_i|} \cdot \sum_{s_j \in S_i} (2 \cdot s_j - 1) \cdot T_j \tag{4-1}$$

As it can be seen from the formula, the algorithm utilizes only the reliability ratios of the bits in parity check equations to calculate the flipping metric. Although the method performs well, there's still room for improvement due to some flaws introduced by the use of the ratios only to calculate the flipping metric. Thus, we propose a modification to the RRBF algorithm, which uses the reliability of the bit itself, to calculate the flipping metric of the bit. In (4-2), the metric calculation method of the proposal is given, where β is a constant that can be determined by simulation which can be used to optimize the BER of the MRRBF decoder.

$$E_i = -\beta \cdot |y_i| + \frac{1}{|y_i|} \cdot \sum_{s_j \in S_i} (2 \cdot s_j - 1) \cdot T_j \tag{4-2}$$

4-1 Reasoning of the Modification

The reliability ratio based algorithm utilizes the ratio of the reliabilities of bits that are elements of the same parity check equation. The algorithm is shown to be better than most

of the bit flipping algorithms that are available in [2] However, cycles in the Tanner graph may cause serious degradation in the BER performance of the RRBF decoder, since each bit is flipped depending only on the reliability ratios. This can be demonstrated by an example, using the parity check matrix section given in (4-3). If the RRBF decoder is used with a code which has a parity check matrix in (4-3) (which has a length-6 cycle, consisting of columns 1, 3, and 5), errors in the cycle can not be resolved even if the reliabilities of the erroneous bits are very low. We will give an example to show that this is the case.

$$H = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & 1 & 1 & 0 & 1 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & 0 & 1 & 1 & \cdots \\ \cdots & 1 & 0 & 1 & 1 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{4-3}$$

Assume that all of the bits that are elements of the cycle (the first bit, the third bit, and the fifth bit) are erroneous with a low reliability value (assume that the received values are 0.1), and the rest of the bits are correct with high probabilities (assume that the received values are 1). First we should note that having a reliable erroneous bit is less probable due to the shape of the Gaussian distribution, and this case is very likely. When the error decoding process starts, the parity check equations will be regarded as correct, since there are two erroneous bits in each parity check equation. Thus, using (4-1), all of the bits receive negative values for their flipping metrics from the given parity check equations. All of the unreliable and the reliable bits receive the same value for their flipping metric from the three parity check equations, if the metric calculations are closely studied. The value for a reliable bit is expressed by E_r, and the value for an unreliable bit is expressed by E_u.

$$E_r = (\tfrac{1}{1}) \cdot -(1 + 0.1 + 0.1) = 1.2$$

$$E_u = (\tfrac{1}{0.1}) \cdot 2 \cdot -(1 + 0.1 + 0.1) = 24$$

As it can be seen from the calculations above, the unreliable bits (which are not correct) receive a -12 for the metric from both of the parity check equations that they are elements of. Also, the reliable bits (which are correct) receive a -1.2 for the metric from parity check equations that they are elements of. Although other parity check equations that contain these bits effect the flipping metrics, the difference between the reliable bits and the unreliable bits is high, and the unreliable bits may never be flipped. Thus, in such situations, RRBF algorithm can not decode the word correctly.

The modification that we propose introduces the self reliability of bits to the flipping metric calculation, which might solve the kind of problems described above. The self reliability value is weighed by a pre-determined constant β, to make the self reliability term comparable to the reliability ratios which can be quite high for low reliability bits. The β value should be high enough to cure the wrongly flipped bits due to problems described above, and low enough not to bias the flipping metric totally by the self reliabilities. A β value that is too high results in the flipping of the least reliable bits, shadowing any effect of the code structure and

the reliability ratio decoder. Next, we will give simulation results and discussion about the modified decoder, and all of the simulations are on AWGN channel, unless indicated to be otherwise.

4-2 The BER Minimizing Beta Value and Results

The value of the β depends on the code properties and the SNR value, as we will show in this section. For a code, there is not a beta value that minimizes the BER, for all SNR values. In Figures 4-1 and 4-2 BER performance versus the β value is given for various SNR values with a $(n = 96, k = 48, d_c = 3)$ LDPC code. The β value that minimizes the BER for each SNR value is indicated in the plots.

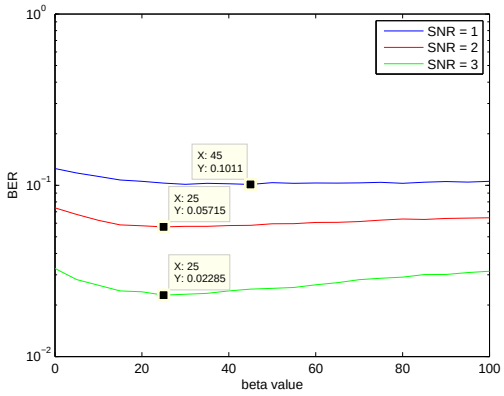

Figure 4-1: The BER performance of the modified reliability ratio algorithm for SNR values 1, 2, and 3 with $(n = 96, k = 48, d_c = 3)$ LDPC code

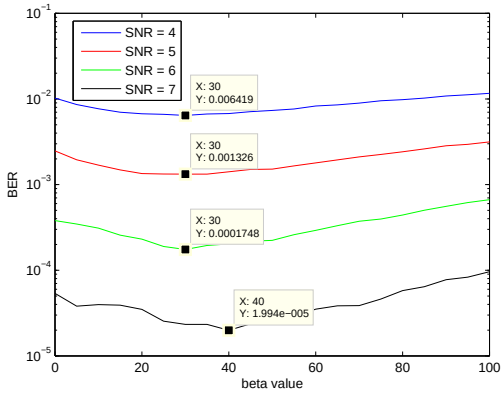

Figure 4-2: The BER performance of the modified reliability ratio algorithm for SNR values 4, 5, 6, and 7 with $(n = 96, k = 48, d_c = 3)$ LDPC code

Since the optimal β value that minimizes the BER is SNR dependent, we define and search for BER minimizing β values that minimizes the BER of the $(n = 96, k = 48, d_c = 3)$ LDPC code reasonably for all SNR values. In Figure 4-3, the performance of MRRBF with $\beta = 0$

(which is RRBF) is compared to the performance of MRRBF with three β values that offers reasonable improvement for all SNR values.

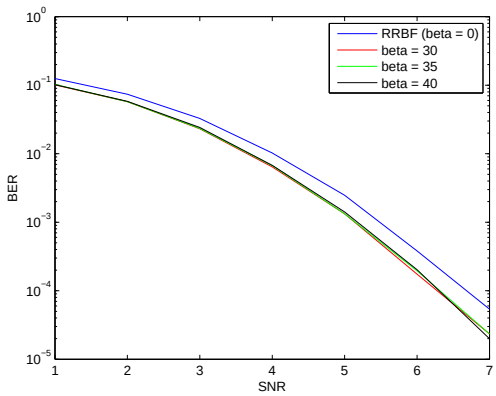

Figure 4-3: The BER performance of the modified reliability ratio algorithm with different β values using $(n = 96, k = 48, d_c = 3)$ LDPC code

The BER minimizing beta values and the amount of improvement in the BER for another $(n = 96, k = 48, d_c = 3)$ are around 30, as the first code. Thus, we take a hint that codes with similar properties have similar BER minimizing β values and BER minimization. We focus on the density of codes, while judging the amount of the improvement and the BER minimizing β values, which the ratio of number of non-zero elements (ones for binary case) in the parity check matrix to the size of the parity check matrix. The $(96, 48, 3)$ code has a density of 0.0625.

The BER minimizing β value for a $(n = 96, k = 48, d_c = 4)$ LDPC code (which has a density of 0.0833) is found to be around 50, by Monte Carlo simulations. If the optimal β values of $d_c = 3$ (which is 30) and $d_c = 4$ (which is 50) which are obtained by simulations are compared, it can be seen that increasing the column weight also increases the value of the optimal β value for most SNR values. The difference in the optimal β value is expected, considering the fact that the modification was proposed to compensate the effects of cycles on the BER performance of the RRBF algorithm. A sparser matrix generally contains less cycles than a denser matrix, unless the latter is designed to have very less cycles. Since the amount of cycles is less in a sparse matrix, the effect of the cycles to RRBF decoding is also less than a denser matrix. Thus, the value of BER minimizing β values should be lower if the parity check matrix is sparse. Also, we expect the amount of improvement to be smaller for sparser matrices by the same reasoning.

The BER performance of the $(n = 96, k = 48, d_c = 4)$ with the RRBF algorithm and the MRRBF algorithm with sub-optimal β values is given in Figure 4-4. The performance of the MRRBF and RRBF with $d_c = 4$ LDPC code is better than with $d_c = 3$ LDPC code, which arises from the fact that increasing the row weight improves the BER performance of the RRBF [2]. The amount of improvement due to the modification for this code with a denser parity check matrix, is a little bit more than the improvement for the other two (96,48) LDPC codes, as expected.

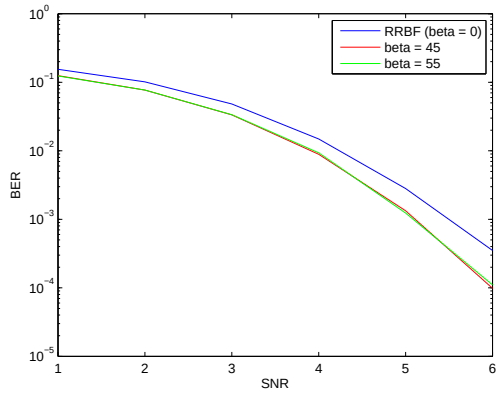

Figure 4-4: The BER performance of RRBF and MRRBF with $(n = 96, k = 48, d_c = 4)$ LDPC code

BER minimizing β values are around 20 for different SNR values for a $(n = 204, k = 102, d_c = 3)$ LDPC code (density of 0.0294), which has been found by Monte Carlo simulations. The optimal β values of $n = 96$ and $n = 204$ from simulations indicate that increasing the code length decreases the value of the BER minimizing β values. This can also be explained using the sparsity of the parity check matrix. The BER performance of the $(n = 204, k = 102, d_c = 3)$ with the RRBF and MRRBF algorithms is given in Figure 4-5. The amount of improvement is less than the previous cases, which follows from the sparser parity check matrix.

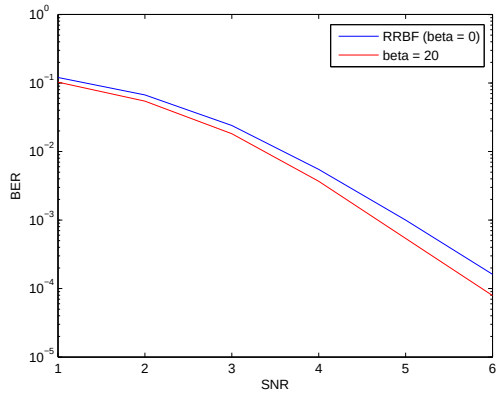

Figure 4-5: The BER performance of RRBF and MRRBF with $(n = 204, k = 102, d_c = 3)$ LDPC code

The effect of the density of the parity check matrix on BER minimizing β values is also shown by the results of two regular $(816, 408)$ LDPC codes, with column weights 3 (density of 0.0074) and 5 (density of 0.0123). The $(816, 408, 5)$ LDPC code has very good error correction capabilities for high SNR values, and the expected BER is very low. Due to very low BER values, reliable results for Monte Carlo simulations are obtained for SNR values 1-3. Thus, BER minimizing β values of these two codes are compared using the results of these SNR values only. In Figures 4-6 and 4-7, the BER performance for different beta values are

compared for SNR values 1-3. As it can be seen the denser matrix has higher BER minimizing β values.

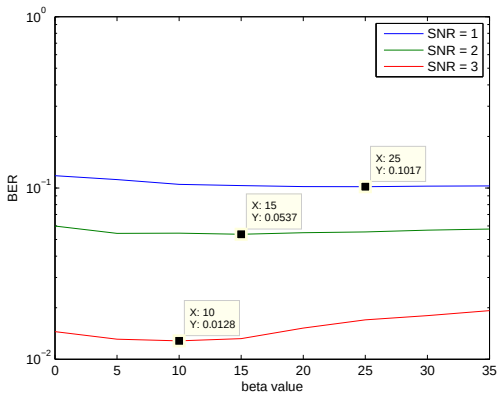

Figure 4-6: The BER performance of the modified reliability ratio algorithm for SNR values 1, 2, and 3 with $(816, 408, 3)$ code

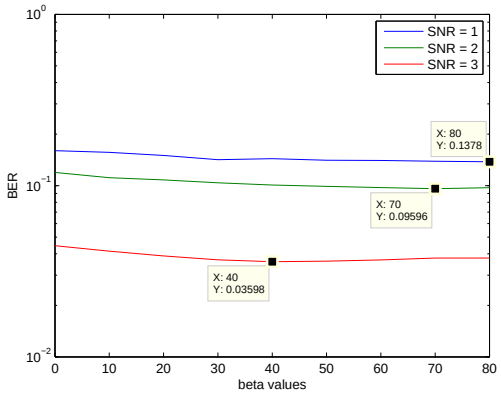

Figure 4-7: The BER performance of the modified reliability ratio algorithm for SNR values 1, 2, and 3 with $(816, 408, 5)$ LDPC code

The improvement in the BER performance of the RRBF with the sub-optimal β value is given in Figure 4-8 for $(816, 408, 3)$ LDPC code. The results for $(816, 408, 5)$ are not given here, since reliable results are available only for SNR values of 1-3 due to low BER and high number of trials in Monte Carlo simulations.

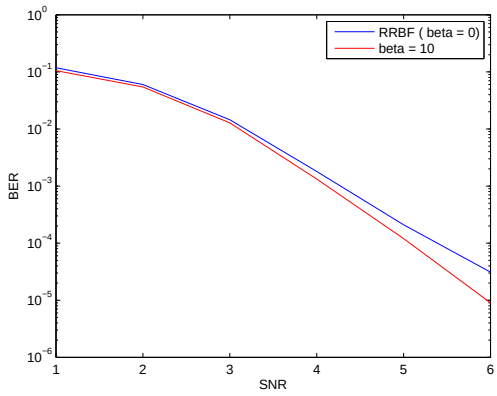

Figure 4-8: The BER performance of RRBF and MRRBF with $(816, 408, 3)$ LDPC code for SNR values 1-6

Finally, we give a result for a large LDPC code which is sparser than the previous codes; an irregular $(2048, 1030)$ LDPC code with density of 0.0036. As we have stated before, for sparse codes the BER minimizing β values are smaller and also the amount of improvement is very limited. For this code, BER minimizing β values are found to be around 8, again by Monte Carlo simulations. In Figure 4-9 the comparison of MRRBF with a BER minimizing β value to RRBF is given, which shows that the improvement is small.

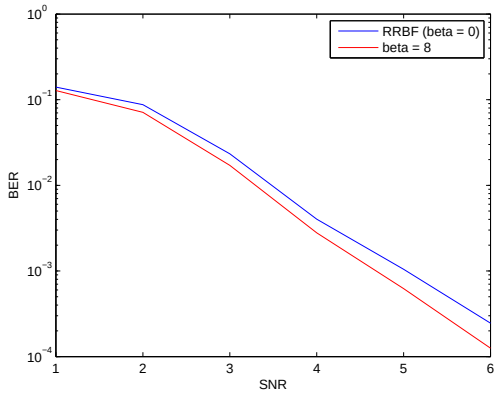

Figure 4-9: The BER performance of RRBF and MRRBF with $(2048, 1030)$ LDPC code

4-3 Conclusion

In this chapter, we have introduced the modified reliability ratio bit flipping algorithm, which outperforms the the classical reliability ratio bit flipping algorithm for small and medium length codes with BER minimizing β values. We have compared the MRRBF to the RRBF for many codes, and also discussed the BER minimizing β values for the algorithm. We have concluded that the β value is dependent on code length, density of the code, and the SNR value. The MRRBF, which is an improvement to the RRBF with very small complexity on

the calculations, can also outperform other decoders such as MWBF and LP-WBF which RRBF outperforms. For small and medium length codes, MRRBF is a good alternative to the RRBF algorithm with very small additional complexity.

Chapter 5

The Rank Completer

The Rank Completer (RC) is a new method to separate errors from erasures for linear block codes. Instead of constructing separating matrices, which are very large for practical values of n and l, the rank completer handles the separation by pre-processing the original parity check matrix to create new parity check matrices. Using the rank completer, any valid parity check matrix can be used directly for separation of erasures from errors. The method for the binary case is explained in this chapter. However, the method can be generalized to non-binary cases by modifying the method, which is shortly explained while the algorithm is described. To explain the algorithm throughly, we first give the definition of the rank property, and the rank completeness problem. Then we give the binary RC algorithm, and explain it clearly with an example. Finally, we prove that RC can guarantee separation of up to $n-k-1$ erasures, and solve the study the stopping set problem.

5-1 The Rank Property and The Rank Completer Algorithm

Lemma 1 (The rank property): Any erasure set of size l ($l \leq n-k-1$), can be separated from a (n,k) code, if there are $n-k-l$ linearly independent rows in the parity check matrix that have zeros in the erasure positions, and the erasure set does not contain the support of a non-zero codeword.

Proof. The erasure separation from the code is an operation that involves the puncturing of the codewords and shortening of the parity check matrix of the code. If the erasure set does not contain the support of a non-zero codeword, then after the puncturing we have a $(n-l,k)$ code. Thus, a valid parity check matrix must have the rank of $n-k-l$. Since, the shortening operation and the puncturing operation are dual, the original parity check matrix should be shortened in the same l positions. If there are $n-k-l$ linearly independent rows in the parity check matrix with zeros in the l erasure positions, then after the shortening operation the parity check matrix has rank of $n-k-l$, which concludes the proof. □

Using Lemma 1, we define the *rank completeness problem*; a full rank parity check matrix can not satisfy the *rank property* after shortening, if number of rows that is erased, is larger than the number of the columns to be erased. A rank deficient parity check matrix (a parity check matrix with extra rows) can be a solution to this problem, even when the number of rows that is erased is higher than the number of columns that is erased. This fact has been explained and systematized in [1], and we call it the SM method. However, the constructive methods for separating matrices end up with very large matrices, which makes the decoding process unfeasible.

Another way to solve this problem is to create extra rows that contain only zeros in the columns to be separated via pre-processing of the matrix. These new rows can be found by viewing these columns as a sub-matrix, or one-by-one for each column. We will first describe the sub-matrix method, and then move on to the one-by-one method, which we use in this research. Using the sub-matrix method, if we want to separate columns j_1, j_2, ..., j_l, then the sub-matrix has only these columns. The algorithm should create extra all-zero rows in the sub-matrix using linear matrix operations, so that in total there are $n - k - l$ linearly independent all-zero rows in this sub-matrix. However, creation of linearly independent rows can not be guaranteed without keeping track of the past calculations, which might be too complex for large codes. Hence, we have worked on the one-by-one solution; RC, which can guarantee the linear independence of the rows in the final parity check matrix.

The rank completer method can simply be explained as follows; for any column j to be separated, make sure that there are at least $a - 1$ linearly independent rows with a zero in position j, where a is the number of linear independent rows with a one in the j^{th} column of the matrix. Below, we give the algorithm for the binary case; where the shortening set is the collection of columns to be separated in descending order (the column with highest index is separated first), the modification set is the rows that contain a one in the column that the algorithm is working on, and exclusion sets are the rows or columns that have been processed by the algorithm that are stored to be erased later. The algorithm can separate one column in each loop, and a new parity check matrix is created after each of these loops. These new parity check matrices are called transitional parity check matrices, since they are used as the new basis parity check matrix when the algorithm starts separating the new element in the shortening set. By using the previous transitional matrix in each new column to be separated, the rank completer method separates an erasure set of size l in l loops. Finally, it uses one more loop to output the new parity check matrix.

1. Take the shortening set and the parity check matrix as input.
2. Choose the first element (column) in the shortening set, and store all the rows with a one in this column, in the modification set. If the shortening set is empty, erase all the columns that is in the column exclusion set, and output this new parity check matrix.
3. If the modification set is not empty, then select the first element as the basis row, and move it to the row exclusion set.
4. If the modification set is empty, then go to step 6.
5. Choose the first element (row) in the modification set, and sum this row with the basis row in mod 2. Then, store the result of this summation as an extra row in the parity check matrix, move this element to the row exclusion set, and go to step 4.

6. Erase all of the rows that is an element of the row exclusion set from the matrix (which forms the transitional parity check matrix), empty the row exclusion set, move the current element (column) from the shortening set to the column exclusion set, and go to step 2.

To clarify the algorithm, we give an example using the parity check matrix of $(8, 4, 4)$ extended Hamming code, which is given in (5-1). It is obvious that the parity check matrix has three extra rows, since its rank is 4. If we assume bits 2 and 7 are received as erasures using this code, then our shortening set is {2,7}. We have selected a matrix with extra rows in this example to demonstrate that having a few extra rows can not guarantee separation of erasures from errors. If we directly shorten this matrix, then the resulting matrix has rank zero. However, from the *rank property*; the rank of the shortened matrix should be 2. Next we show how the rank completer algorithm proceeds, and separates the erasure set. It is worth to note here that the erasures can also be separated using the full rank parity check matrix without the extra rows.

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{5-1}$$

The shortening set {7,2} and the parity check matrix is given to the rank completer as input in step 1. After step 1, the algorithm enters its first loop. In step 2, the algorithm selects column 7 to work on, and stores the rows {1,2,6,7}, in the modification set. In step 3, row 1 is selected as the basis row, and it has been moved to the row exclusion set. Since the modification set is not empty, the algorithm goes to step 5, and sums the first element in the modification set; row 2 with the basis row. The resulting row is stored in the parity check matrix, and the algorithm goes to step 4, again. With the loop structure in steps 4 and 5, all of the rows in the modification set is added to the basis row one-by-one, and then moved to the row exclusion set. When the modification set is empty, the algorithm moves to step 6. In step 6, all of the elements in the row exclusion set {1,2,6,7} are removed from the matrix, the row exclusion set is emptied, and the column that is worked on is moved to the column exclusion set. After step 6; the column exclusion set is {7}, the shortening set is {2}, and the transitional parity check matrix of loop 1 is given in (5-2).

$$H = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{5-2}$$

Since the first loop of the algorithm is over, and the transitional matrix is created, the algorithm can start its second loop. In step 2, the algorithm selects column 2 and creates the modification set; {2,3,4}. In step 3, row 2 is selected as the basis row, and moved to the row exclusion set. Since the modification set is not empty, we move to step 5 again. In step 5, row 3 is added to the basis row (row 2) in mod 2, and the resulting row is added to the parity check matrix. Also, row 3 is moved to the row exclusion set, and the algorithm goes back to step 4. Since the modification set is again not empty, the algorithm goes to step 5, and adds row 4 to the basis row in mod 2. The resulting row is again added to the parity check matrix, and row 4 is moved to the row exclusion set. The algorithm goes to step 4 a third time, and since the modification set is empty, it directly goes to step 6. In step 6, rows {2,3,4} are removed from the matrix, the row exclusion set is emptied, and column 2 is moved to the column exclusion set. Now the column exclusion set is {7,2}, the shortening set is empty, and the transitional parity check matrix of loop 2 is given in (5-3).

$$H = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{5-3}$$

Since the shortening set is empty, it is obvious that this is the final loop of the algorithm. The algorithm goes to step 2, and since the shortening set is empty, it erases columns 2 and 7 from the previous transitional parity check matrix, and outputs the parity check matrix in (5-4).

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{5-4}$$

The rank completer can easily be generalized for non-binary codes; the algorithm again selects a basis row, and create enough number of linearly independent extra rows that have zeros in the selected column position. For binary codes the creation process is fairly simple; the basis row is added to the other rows one-by-one in mod 2, which guarantee the creation of a zero in the desired position. For non-binary codes, the basis row should be divided (in mod q) by the value in the position to be separated to have a one in this position. Afterwards, the divided version of the basis row is subtracted from the other rows to get zeros in the column position to be separated. The non-binary RC algorithm is more complex than the binary RC algorithm, that needs operations including division, multiplication, and subtraction.

5-2 Separation Guarantee

In this section we study the separation guarantee of the rank completer, and prove upper bounds for the maximum number of erasures that can be separated.

Lemma 2 (Linear independence in RC): The rank completer method guarantees the creation of $a-1$ linearly independent extra rows from a linearly independent rows (in any of the loops).

Proof. For the first part of the proof, assume that we have a full rank parity check matrix. Then any sub-matrix which contains some of the rows of this matrix has linearly independent rows as elements. Hence, if one selects a sub-matrix with rows which have a one in a certain position (say with size a), then these a rows are also linearly independent. If one of the rows of this sub-matrix is selected as the basis row, and added to all of the other rows in mod 2, it is obvious that the resulting $a-1$ rows are also linearly independent of each other. This arguments are valid for any l^{th} separation, since for each additional column to be separated, the algorithm uses the parity check matrix of the previous separation, which is again a full rank parity check matrix.

Now, let's assume the parity check matrix which is not full rank. If we consider the scenario above with a rank deficient parity check matrix, then the corresponding sub-matrix has $a+b$ rows, where a of them is linearly independent. Now, we will show that if one of these $a+b$ rows is selected as the basis vector, we again end up with $a-1$ linearly independent rows plus some linearly dependent rows for one column separation. There are two possible scenarios that should be considered to prove the lemma. In the first scenario, the basis row is linearly independent from any other rows. It is obvious that the sub-matrix with $a+b-1$ rows has rank $a-1$, since an independent row is removed. We can consider this one as a special case of the full rank parity check matrix condition above, where b linearly dependent vectors are added to the sub-matrix. Hence, if the basis row is added to all of the rows of the sub-matrix, then $a-1$ of the resulting rows are linearly independent. In the second scenario, the basis row is linearly dependent to some of the other rows of the sub-matrix. If we rearrange the order of the rows of the sub-matrix with the basis row as the first row and the first a rows are linearly independent of each other (the rank of the sub-matrix is a), then again this can be seen as a special case of the full rank parity check matrix condition. Thus, it is obvious that if we add the basis row to the rows to the sub-matrix, the first $a-1$ rows are linearly independent. □

The following lemma shows that any column that is selected to be separated contains at least a single one, unless the support of a non-zero codeword is separated.

Lemma 3: While using the RC algorithm, all columns that are an element of the shortening set have at least a single one in the parity check matrix that is to be shortened (either the original parity check matrix and the whole shortening set, or the transitional parity check matrices and the corresponding reduced shortening set), unless the support of a non-zero codeword is separated.

Proof. Before starting the proof, we shall give an observation about parity check matrices from [32]. If the column j of a parity check matrix is an all-zero column, then the unit vector

e_j is a valid codeword of this code. Depending on this observation, if we have a distance one code with a e_j as a codeword, then the original parity check matrix can have an all-zero column. However, if we separate that column, then the support of a non-zero codeword is separated. For transitional parity check matrices; again if there's an all-zero column with index j, then e_j is a codeword of the punctured code. The unit vector e_j can only be a codeword of the punctured code, if all bits of a codeword c of the original code is separated except for the one in position j. If this position j is also in the shortening set, then we are separating a codeword. This proves that an all-zero column can not be an element of the shortening set, unless the support of a non-zero codeword is separated. □

Theorem 1: The rank completer method can separate any erasure set of size l ($l \leq n-k-1$), that does not contain the support of a non-zero codeword.

Proof. To prove this theorem, we first make use of Lemma 1, which states that any erasure set of size l ($l \leq n-k-1$) and that does not contain the support of a non-zero codeword can be separated, if there are $n-k-l$ linearly independent rows in the parity check matrix with zeros in the erasure positions. Thus, if we can show that the rank completer method can create $n-k-l$ linearly independent rows with zeros in erasure positions, the theorem is proven. To show that, we should inspect the behavior of the rank completer. Lemma 3 states, a column in the shortening set has at least a single one, unless the support of a non-zero codeword is separated. Thus, a column in any loop of the rank completer, has either multiple ones or a single one. If it contains a single one, then the column is easily separated by erasing the row that contains this single one. It is trivial to show that after removing the single row with a one, the rank of the new transitional matrix is $A-1$, if the previous transitional matrix of rank A. If there are multiple ones in the column, then we use Lemma 2, which states that the rank completer creates $a-1$ linearly independent rows with zeros in the selected column from a linearly independent rows in any loop. If the final transitional matrix that is formed by the rank completer using these two operations is considered, it is obvious that it has rank $n-k-l$ with zeros in the erasure positions. Thus, the rank completer can separate any erasure set, unless the erasure set contains the support of a non-zero codeword. □

Theorem 1 states that the rank completer method can separate any erasure set size l ($l \leq n-k-1$), as long as it does not contain the support of a non-zero codeword. For the cases $l > n-k-1$, the number of linearly independent rows in the parity check matrix (which is $n-k-l$) becomes zero, which means a parity check matrix with zero rank. No error detection/correction is possible with a zero rank parity check matrix, so regarding Theorem 1, $n-k-1$ is the upper bound for the number of erasure to be separated. We will extend Theorem 1 for other cases using Lemma 4 we give next.

Lemma 4: If the erasure set that is separated from a (n,k,d) linear block code contains the supports of s linearly independent codewords, then the rank of the parity check matrix should be $n-k-l+s$, where l is the number of erasures that is separated.

Proof. To prove this lemma, one should keep in mind that separating the erasures from errors is done by puncturing the code in the erasure positions. If the puncturing operation effects only the support of a single codeword, we can claim that the length of the code is reduced by l, and the rank is reduced by one as we have stated in Chapter 2. The new code parameters are

$(n-l, k-1, d^*)$, and the rank of the parity check matrix should be $n-l-(k-1) = n-k-l+1$. If supports of s linearly independent non-zero codewords are separated, then the rank of the parity check matrix should be $n-k-l+s$, since the number of codewords is divided by the alphabet size (which is 2 for the binary case that we consider) for each linearly independent codeword support that is separated. □

Theorem 2: The rank completer method can separate any erasure set of size l ($l \leq n-k-1+s$) that contains the support of a number of non-zero codewords, where s is the number of non-zero linearly independent codewords, whose supports are elements of the erasure set.

Proof. First, we should utilize Lemma 3 and its proof; an all-zero column in the parity check matrix, can only occur when the support of a codeword is separated, at the last bit of the support of the codeword. If we try to separate the support of a codeword of weight w_c, w_c-1 of the bits can be separated using Theorem 1. However, for the last bit of the codeword support to be separated, the parity check matrix has an all-zero column in the last bit position to be separated. Thus, the parity check matrix can be shortened directly at this position, resulting in a parity check matrix of rank $n-k-l+1$. For multiple codeword supports that are linearly independent, the rank of the parity check matrix becomes $n-k-l+s$, since s all-zero columns are shortened in the parity check matrix. Also, regarding Lemma 4, we know that a valid parity check matrix for such a code has the rank $n-k-l+s$. Since the *rank property* is satisfied, we can conclude that the erasure set can be separated by the rank completer. □

Theorem 2 is an extension of Theorem 1, and shows that erasure sets containing codeword supports can also be separated by the rank completer. Based on Theorem 2, we can claim that any erasure set that has less than $n-k+s$ elements can be separated by the rank completer. Since $s \geq 0$, $n-k$ can be selected as a general bound for the rank completer. All of the Lemmas and the Theorems if we consider non-zero values instead of ones, and the aforementioned non-binary RC instead of the binary RC, all of these results are applicable for the non-binary case. However, we only give the binary versions, since binary results are directly applied to our implementations and results.

5-3 Stopping Sets

The iterative erasure decoder might get stuck because of the stopping sets, as explained in the background section. Before analyzing stopping sets, we shall give the definition; a stopping set is a collection of the columns of the parity check matrix, such that the sub-matrix consisting of these rows do not contain a weight one row. We can divide the stopping sets into two groups; stopping sets that can be removed by the parity check matrix construction (type I), and the stopping sets that is the support of a non-zero codeword and can not be removed by any means (type II). Stopping sets are a cause of uncorrected erasures, and they play a role in the BER performance of the decoder, especially in the low error probability region. Thus, prevention of stopping sets is favorable for the performance of the decoder.

In Theorem 1 of [1], the authors prove that an l-separating parity check matrix, where $0 \leq l \leq \min\{d, n-k\} - 1$, does not contain any stopping set size l or less. Here we present a similar theorem for a modified version of the rank completer method with any erasure set that

can be separated. The proposed modification is adding the extra rows computed by the RC, to the original parity check matrix; and then using this new matrix in the iterative erasure decoder.

Theorem 3: There are no stopping sets in the erasure set that is separated by the modified rank completer, unless the erasure set contains the support of a non-zero codeword. In other words, the only stopping sets in the modified version of the rank completer is the stopping sets of type II.

Proof. We have shown in Lemma 3 that the any column that is an element of the shortening set, contains at least a single one, unless the support of a codeword is separated. Assuming that no support of a codeword is separated, l bits can be separated from the code, where $l \leq n-k-1$ by Theorem 1. Then the transitional matrix of $(l-1)^{th}$ separation contains a row that has a one in the l^{th} bit position, and zeros in the previously separated bit positions. If all such extra rows are added to the original parity check matrix, then for any erasure set l, there is always a weight-one row in the sub-matrix of the erasure set (thus, no stopping sets exist). If the support of a codeword is separated, there's an all-zero column in one of the transitional parity check matrices, which prevents us from guaranteeing that there are no stopping sets. □

The shortcoming of Theorem 3 for stopping sets of type II is very well expected, since if the support of a codeword is erased, it is not possible to retrieve the bit values in these positions. Another result that can be deduced from Theorem 3 is the following, if an erasure set contains the support of a codeword and some other bits, these bits can be corrected by the iterative erasure decoder, unless they are the support of another non-zero codeword.

5-4 Conclusion

To sum up, the rank completer is a new method for separation of erasures from errors, which does not require special matrix constructions that might end up with very large number of rows. Using the rank completer method, any set of erasures (up to size $n-k-1$) might be separated from errors which lets us to decode the errors and erasures separately. We have also shown that there are no non-empty stopping sets in these separated positions (and the iterative erasure decoder can decode all the separated positions), unless the support of a non-zero codeword is not separated.

Chapter 6

New Error Erasure Decoding Methods

In this chapter, we propose new error erasure decoding methods, which have originated from the ideas that are given in the background and literature review. First, we give a new guessing method; reliability guessing (RG) for binary codes, which is superior to simple guessing algorithm and trial guessing in the BER perspective. Afterwards, we move on to the error erasure decoder using the previously introduced RC method. The error erasure decoder that we describe here, uses the "separation idea" from [1] and uses the same principle with the EED described in that article. The notable difference between these two decoders is the method that is used to separate the erasures; which is RC in the decoder described here, and SM for the decoder that is described in [1]. Finally, in the results section, we will give BER performances of the new methods introduced here, and compare them to the previously known methods.

6-1 Reliability Guessing Decoder

The reliability guessing method is a type of guessing algorithm, in which we randomly assign 0s and 1s to the erasure positions as in simple guessing algorithm. The method differs from the simple guessing method due to the fact that we assign very low reliability values to those assigned bit values. After the erasure positions are filled and reliability values are assigned, the word can be decoded using any soft decision algorithm. We have selected soft decision BF algorithms due to their low complexity. The careful reader may remember that the BF algorithms are the less complex alternatives to MP algorithms, however, they can not used for EEDG. Here we present a simple method for EEDG with BF decoders.

Example guessings are given below for hard and soft channels using reliability guessing method, where the erasures are shown as x and the assigned low reliability values are shown as L. For the hard channel (BSEEC), the channel output $\{0, 1, x\}$ is mapped to $\{1, -1, L\}$ to be fed to the soft decision BF decoder.

$$\begin{array}{cccccccccc} 1 & 1 & 0 & 0 & x & x & 0 & 1 & 0 & x \\ & & & & \downarrow & & & & & \\ -1.0 & -1.0 & 1.0 & 1.0 & L & L & 1.0 & -1.0 & 1.0 & L \end{array}$$

For the soft channel (AWGN+EC), the channel output can be directly fed to the soft decision BF decoder, after the erasures are mapped to the low reliability values.

$$\begin{array}{cccccccccc} 0.8 & -0.5 & -0.7 & 1.5 & -1.1 & x & -0.5 & x & 1.7 & 0.7 \\ & & & & \downarrow & & & & & \\ 0.8 & -0.5 & -0.7 & 1.5 & -1.1 & L & -0.5 & L & 1.7 & 0.7 \end{array}$$

As it can be realized from the decoder inputs, the method always uses soft decision algorithms, even if the communication channel is a hard decision one. Therefore, we can conclude that the decoding complexity is increased for the hard channels, since simple BF could have been used for decoding instead. Next we shall explain the RG algorithm in more detail, to give a better idea about the implementation, complexity and the main principle of the algorithm. To do so, first we give the definition of the low reliability flag L with some simulations for the optimum value, and then argue the selection of the soft decision decoder.

The low reliability flag L, a random variable with two equiprobable values, where a is a pre-determined constant of the RG algorithm.

$$L = \begin{cases} a & p = 1/2 \\ -a & p = 1/2 \end{cases}$$

Before RG method for BSEEC and AWGN+EC is explained in detail, we give a block diagram of a simple decoder using reliability guessing method in Figure 6-1 to make the process clearer to the reader. As it can be seen from the block diagram, first the erasure positions are filled with low reliability values (that are real values) and then fed to a soft decision BF algorithm. Also, the binary mapping for BSEEC should be kept in mind, which is denoted by a star in the block diagram, to adapt the received word as an input to soft decision BF algorithms.

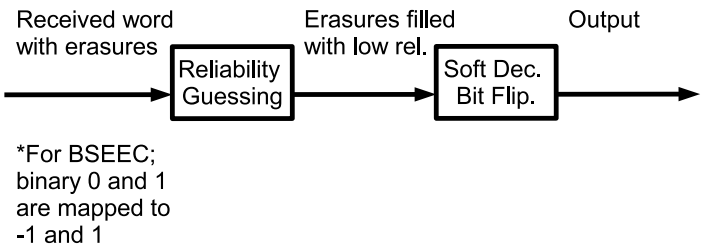

Figure 6-1: Block diagram of a simple decoder using the reliability guessing method

6-1-1 RG in BSEEC

The BSEEC is described by two probabilities; erasure probability p_1, and error probability p_2 as explained in Chapter 2. In the simulations for RG in BSEEC, error probability is given in the x axis, and $p_1 = 0.1$. The maximum number of iterations is 100 for all of the decoders. This value might seem too big at first glance, but we selected this high value to ensure that the only limit to error decoding is decoder capabilities (and not number of iterations). This maximum number of iterations might be lowered by using multiple bit flipping algorithms.

We start explaining RG in BSEEC with the value of variable a. The effect of the value of a on the decoder performance with $(n = 96, k = 48, d_c = 3)$ LDPC code are given in Figure 6-2 where d_c is the column weight of the LDPC code. Our first observation is that decreasing the value of a improves the BER performance up to a point. The second observation is that for very low values of a, the performance is identical (or very close to each other). Therefore, we define a boundary value for a, from where decreasing the a value further does not improve the BER performance. Having such a boundary value for a makes sense, since a defines how unreliable filled erasure positions are compared to rest of the received word. After this boundary value, the decoder sees the filled positions are extremely unreliable compared to the rest of the word, and lowering a does not improve the BER performance. The boundary is not strictly defined, and therefore we only can talk about a region around some value of a. For $(96, 48, 3)$ code, this boundary value for a is around 0.1. Any a value that is lower than the boundary value (except for zero) minimizes the BER value, and we call these values BER minimizing a values. The value of a can not be equal to zero, since zero reliability does not stand for any binary bit value and thus BF decoders can not work with zero reliabilities.

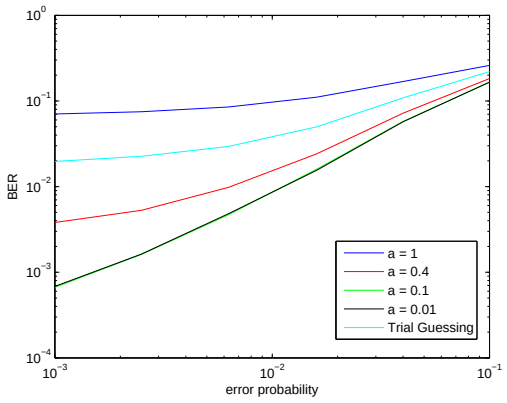

Figure 6-2: The BER performance of RG with different a values for (96,48) and Trial Guessing code on BSEEC with $p_1 = 0.1$

Using Figure 6-2, we can compare the RG method to trial guessing and simple guessing methods; the BER performance of the trial guessing method is given in the plot, where the simple guessing is a special case of the reliability guessing with $a = 1$. The results show that reliability guessing performs better than the trial guessing algorithm. This is expected, since the trial guessing algorithm promises only a small improvement to the simple guessing algorithm (the guarantee of guessing at least half of the erasures correctly). On the other hand, the reliability guessing method improves the simple guessing method, by feeding the decoder with the information that the filled erasure positions are unreliable. This is an important

innovation, since the filled erasure positions can be either zero or one equiprobably, which mean they are completely unreliable. Other guessing algorithms do not make use of this fact, and this is the main reason why RG outperforms both trial guessing and simple guessing algorithms.

The results for $(n = 204, k = 102, d_c = 3)$ is given in Figure 6-3, which agree with the previous results; a value improves the performance up to a boundary, and RG performs better than trial guessing and simple guessing $(a = 1)$. The BER values for the decoder with $a = 0.1$ and $a = 0.01$ are nearly the same, and only a single line can be observed for the BER performance of the decoder with these values. This is expected, since both values are BER minimizing.

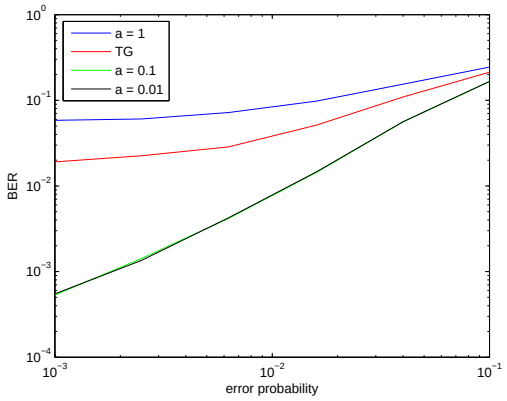

Figure 6-3: The BER performance comparison of RG, simple guessing, and trial guessing for (204,102) code on BSEEC with $p_1 = 0.1$

The final comparison related to the RG method in BSEEC is the use of different soft decision decoders. We have compared WBF, MWBF, LP-WBF, RRBF, and a modified version of RRBF with RG method in Figure 6-4, using (96,48,3) LDPC code. As it can be seen, MRRBF, MWBF, and LP-WBF are the best three decoders. One should realize that the algorithms which use the self reliability (MRRBF,MWBF,LP-WBF) in the calculation of the flipping metric, are a big step ahead than the ones that do not use(RRBF,WBF).

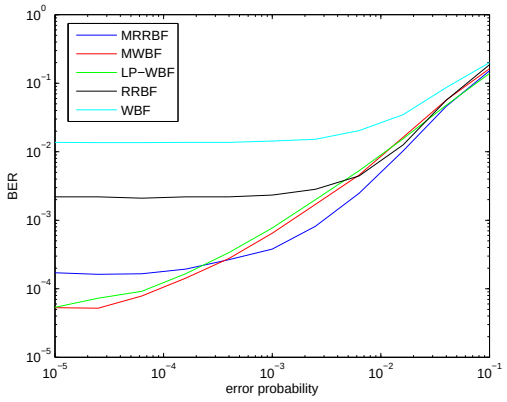

Figure 6-4: The BER performance comparison of RG with different decoders on BSEEC with erasure probability 0.1

To compare the three best performing decoders, we will divide the figure into two at the error probability of 0.001. We see that MRRBF outperforms MWBF and LP-WBF in the right hand side, and is outperformed by these two algorithms in the left hand side. To analyze these BER curves, one should realize that there are two types of errors in the guessed codewords in a BSEEC; the errors that are caused by the channel (and has reliability value of 1), and the errors that are caused by the guessing (and has reliability of a). Mostly, channel errors cause the effect in the right hand side since the error probability is higher than the erasure probability, and guessing errors cause the effect in the left hand side due to lower error probability (compared to the erasure probability). Thus, we can deduce that MRRBF performs better when the error probability is relatively higher, and the other two algorithms are better when the erasure probability is relatively higher. The performance difference originates from the metric calculation of these decoders, however we will not give a complete mathematical analysis of these algorithms with the reliability guessing algorithm on BSEEC. The point of this sub-section is to introduce a promising and a simple method for EEDG.

6-1-2 RG in AWGN+EC

As we have introduced in Chapter 2, AWGN+EC is described by the SNR value and the erasure probability p_1. In the simulations SNR is given in the x axis and p_1 is 0.1. The maximum number of iterations is 100 for all of the decoders.

In this sub-section we will not compare RG with other algorithms, since the guessing algorithms in the literature have only been defined for hard channels. We will just give the performance of RG method with different a values and with different decoders.

We use the (96,48,3) code again to show the effects of different a values, which is shown in Figure 6-5. It can be observed from the plot that the boundary value for a is around 0.05 for AWGN+EC, which is lower than the boundary value for BSEEC. This is very well expected, since the bits that are not erased can have smaller reliabilities now, unlike the BSEEC case. Hence, a smaller value of a is needed to show the decoder that the filled erasure positions are very unreliable.

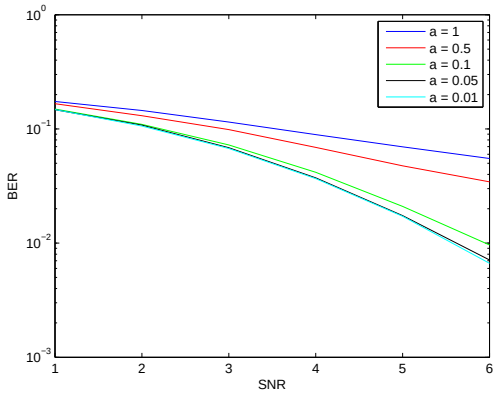

Figure 6-5: The BER performance comparison of RG with different a values on AWGN+EC with $p_1 = 0.1$

The effect of different decoder selections are given in Figure 6-6 for (96,48,3) LDPC code. The results show that MRRBF algorithm is the best decoder for RG in AWGN+EC in the feasible SNR region, however MWBF outperforms all the other algorithms when the SNR value is very high. Again, we see the two algorithms behaving better than each other in error and erasure influenced regions, as in BSEEC.

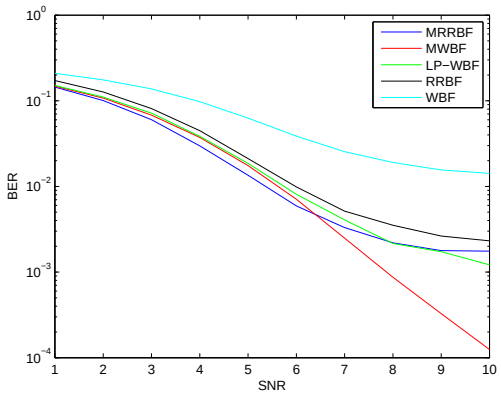

Figure 6-6: The BER performance comparison of RG with different decoders using (96,48,3) on AWGN+EC with $p_1 = 0.1$

6-1-3 RG and bootstrapping

The bootstrapping method which has been introduced in Chapter 3, corrects error positions by updating the low reliability bits using the method given in (6-1), where y_i^b is the bootstrapped value of bit i, y_i is the received reliability of bit i, $S_r^{(i)}v$ is the set of the reliable parity check equations that variable bit i is an element of, and $S_c^{(j)}$ is the set of variable nodes that are element of parity check equation (or neighbors of check bit) j.

$$y_i^b = y_i + \sum_{j \in S_{rv}^{(i)}} \min_{i' \in S_c^{(j)} \setminus i} |y_{i'}| \cdot \prod_{i' \in S_c^{(j)} \setminus i} \text{sign}(y_{i'}) \tag{6-1}$$

Regarding the fact that the reliability guessing method fills the erasure positions with low reliability values, it is obvious that the bootstrapping method corrects both errors and erasures, when used with RG. The block diagram of a decoder using reliability guessing method with bootstrapping and MWBF is given in Figure 6-7.

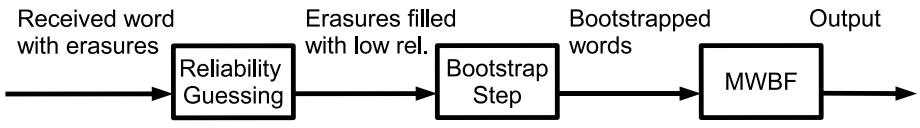

Figure 6-7: Block diagram of a bootstrapped decoder using the reliability guessing method

The improvement in the BER performance of the decoder can be seen from Figure 6-8 for (504,252) LDPC code on AWGN+EC ($p_1 = 0.1$).

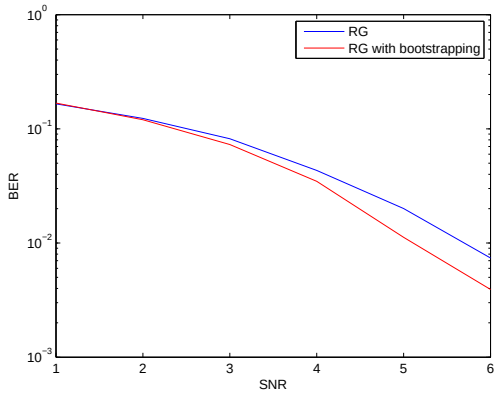

Figure 6-8: The BER performance of RG with and without bootstrapping using (504,252,3) on AWGN+EC with $p_1 = 0.1$

Bootstrapping also improves the BER performance of the decoder on BSEEC, by correcting wrongly guessed, low reliability filled erasure positions (any channel error has the same reliability with a correct bit). In Figure 6-9, the comparison of the reliability guessing method with and without bootstrapping on BSEEC ($p_1 = 0.1$) is given.

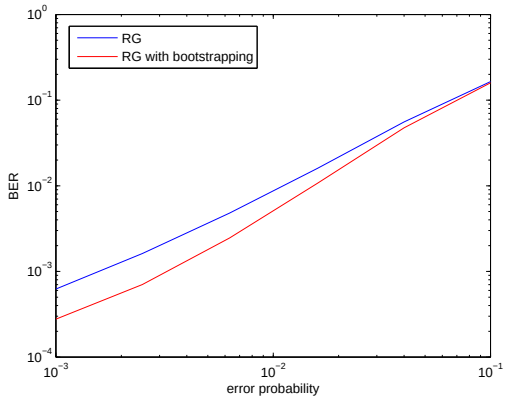

Figure 6-9: The BER performance of RG with and without bootstrapping using (96,48,3) on BSEEC with $p_1 = 0.1$

To sum up, RG is a simple method that can be used both for hard and soft error-erasure channels. For hard channels, it increases the complexity, since without RG hard channels are decoded by hard decision algorithms. However, for soft channels, no additional calculations are required. This method is inspired by the error-erasure decoding ability of belief propagation decoders. Belief propagation algorithm can simply handle erasures in the presence of errors by inserting zero reliability as the reliability value of the erased bit. Since the receiver does not know anything about the erased bit, it can either be zero or one (in the binary sense) and the reliability value should be zero. BF algorithms do not possess this ability since every bit should have a value (zero or one in binary case) in each iteration, and zero reliability

value does not state such a value. RG method is an approximation to zero reliability insertion, where we assign a random value to every bit with very low reliability value. Thus, RG method is only meaningful for decoders that can not cope up with erasures in the presence of errors directly.

The two main properties of the reliability guessing algorithms are the value of a, and the decoder selection. We have shown that the value of a should be low for good performance, however lowering this value does not improve the performance after a certain point; which we call the boundary value. For the decoder selection, we have indicated that self reliability should be taken into account while the flipping metric is calculated (the erased bit have low reliabilities). We have observed that MRRBF performs better in the error influenced region, which makes sense since it's the best performing error decoder according to our simulations, which has been introduced in Chapter 4. In the erasure influenced region we have seen LP-WBF and MWBF are good alternatives, where MWBF outperformed all other alternatives on AWGN+EC. Their better performance in this region depends on how the utilize the reliability values, and this can be researched in future work. Also, a hybrid decoder (MWBF+MRRBF) can as well be designed for RG method, using both the reliability ratios and the minimum reliabilities in the parity check equation.

6-2 Error Erasure Decoder with RC

The complete error-erasure decoder which uses the rank completer is mainly the same decoder that is introduced in [1]. The error erasure decoding is done in three main steps with three main modules. In the first step, erasures are separated from errors, and a new codeword which does not contain erasures are created. Then, this new codeword is decoded for errors using the parity check matrix that is created by the rank completer method. Finally, the erasures are decoded using the complete parity check matrix, and error erasure decoding operation ends. The block diagram for the complete EED, is given in Figure 6-10.

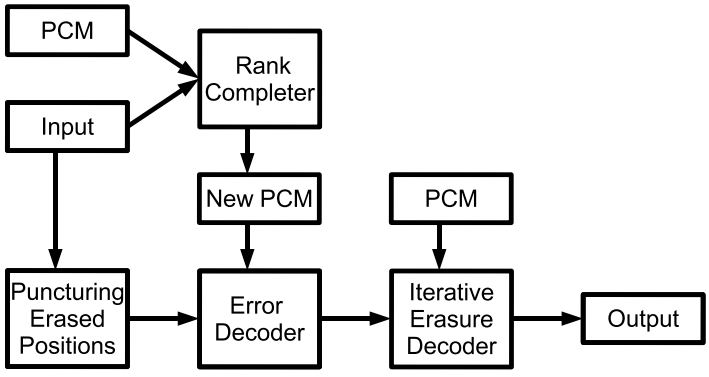

Figure 6-10: Block diagram for the complete EED with RC

The block diagram can be explained as follows. The rank completer module tailors parity check matrices for error decoding, using the separation of erasures from errors idea, as introduced in Chapter 5. The puncturing module is a simple module that punctures the received

word, and outputs a new word that does not contain the erased positions. The error decoder, applies error decoding using the specially tailored parity check matrix by the rank completer. In our implementation, we have used the bit flipping algorithms as the error decoder. The final module is the iterative erasure decoder, which is used to decode the erasures that have been separated from the codeword.

The decoding process can be explained as follows; the received word, which contains errors and erasures are fed into the RC module, with the parity check matrix of the code. Then, the RC module outputs the new parity check matrix, and error decoding is done using this matrix and the word that is punctured in the erased positions. After the error decoding is done, the word is fed to the iterative erasure decoder with the original parity check matrix and the EEDG is finalized.

Next, we discuss the separation bound, which is defined as the maximum number of erasures that can be separated by the rank completer. Then we will analyze the stopping set problem of the iterative erasure decoder, and propose a solution.

6-2-1 Separation bound and its analysis

The separation bound is a pre-determined parameter of the decoder; an erasure set is separated by the rank completer and decoded by the EED, if the erasure set is smaller than the separation bound b. A separation bound larger than $n-k$ is not feasible, since the resulting parity check matrix would be zero (unless a codeword support is separated). The calculations here are for BSEEC, which includes both erasure probability and error probability. However, the reasoning can be generalized, since the main focus of this section is erasures, and both BSEEC and AWGN+EC use the binomial distribution for erasures. Thus, we will extend any conclusions that are drawn here are also valid for AWGN+EC.

There are three possible outcomes of the error erasure decoder depending on the separation bound and the received word; the received word can be discarded, the iterative erasure decoder may fail (due to a stopping set), or the word can be decoded. The probability of a word to be discarded p_d, which directly depends on the separation bound b and the erasure probability p_1, is given in (6-2). This is simply the probability of having b erasures in the received word, since the rank completer do not separate erasure sets of size b or higher.

$$p_d = \sum_{i=b}^{n} \binom{n}{i} \cdot p_1^i \cdot (1-p_1)^{n-i} \tag{6-2}$$

The probability of the iterative erasure decoder to fail p_f, is simply the probability of a codeword having a stopping set in the erasure positions. This is given in (6-3), where D_i is the number of sets that contain a stopping set of size i [27]. This formula gives the probability of the encounter of the decoder with a stopping set (in the words that are not discarded). This can be observed from the lower limit, which is the smallest non-empty stopping set size s, and the upper limit, which is $b-1$, the highest number that is below the separation bound.

$$p_f = \sum_{i=s}^{b-1} D_i \cdot p_1^i \cdot (1-p_1)^{n-i} \tag{6-3}$$

Finally, the probability that a word is decoded (which means that the word is not discarded or there are no codeword supports in the erasure set) p_s is $1-p_f-p_d$. Increasing the separation bound by one, removes $i = b$ term from the summation of p_d, and adds the same term to the summation of p_f. Since the p_d is larger than p_f when i values are equal, increasing the separation bound increases p_s. Thus, if b is increased, the received word is decoded by the EED more probably. However, decoding in every possible case (which is increasing the separation bound) might not be beneficial considering the BER performance of the EED. Thus, before inspecting the BER performance, we should not conclude that increasing p_s is desired, so b should be as high as possible.

Now, let's study the BER performance of the decoder regarding the decoder output which might contain errors and erasures (due to word discarding or stopping sets). The BER formula is given in (6-4), where n_1 is the number of erasures in a codeword, n_2 is the number of errors in the codeword, T is the number of codewords that has been received, and L is the length of a single codeword. It can be observed that only half of the erasures are taken into account, since with a random guessing half of the erasures can be guessed correctly in the mean value.

$$BER = \frac{\sum_{i=1}^{T} (n_1(i)/2) + n_2(i)}{T \cdot L} \tag{6-4}$$

When a codeword is discarded ($p = p_d$), the number of errors and erasures can be approximated using the channel properties. If the channel is BSEEC with $p_1 = 0.1$ and $p_2 = 0.1$, then the mean value of the erasures caused by a discarded word is;

$$d_? = \frac{\sum_{i=b}^{n} \binom{n}{i} \cdot i \cdot p_1^i \cdot (1-p_1)^{n-i}}{\sum_{i=b}^{n} \binom{n}{i} \cdot p_1^i \cdot (1-p_1)^{n-i}},$$

which is the direct calculation of the mean value of a stochastic variable. The mean value of the number of errors caused by a discarded word is;

$$d_\# = (n - d_?) \cdot p_2,$$

which is calculated by using the a priori error probability and the calculated mean value of number of erasures. If $d_?$ is the mean value of number of erasures, then the mean value of number of errors can directly be found by multiplying the error probability to the bits that are not erased.

The total effect of discarding a word on the BER calculation (in the mean value) is given in (6-5), using the mean value of erasures and errors in the discarded word, $d_\#$ and $d_?$. The formula is the same with (6-4), except for the fact that the calculation of the bit errors and erasures are not done for each word, but the mean values for these figures are directly calculated and inserted to the formula.

$$BER_{dc} = \frac{p_d \cdot (d_\# + d_?/2)}{n} \tag{6-5}$$

In Table 6-1, we give the probability of a word to be discarded p_d with (15,6,5) LDPC code on BSEEC ($p_1 = 0.1$ & $p_2 = 0.1$), BER_{dc} which is the effect of the discarded words on BER calculated approximately (the approximation is in the calculation of $d_?$) by (6-5) and Monte Carlo simulations in Matlab. As it can be seen from the table, the approximated results agree with the simulation results.

Table 6-1: The probability of a word to be discarded and the effects of discarding a word with (15,6,5) LDPC code on BSEEC

Separation bound	5	6	7	8	9
p_d	$1.27 \cdot 10^{-2}$	$2.2 \cdot 10^{-3}$	$3.1 \cdot 10^{-4}$	$3.4 \cdot 10^{-5}$	$2.8 \cdot 10^{-6}$
BER_{dc} calculation	$3.1 \cdot 10^{-3}$	$6.2 \cdot 10^{-4}$	$9.7 \cdot 10^{-5}$	$1.3 \cdot 10^{-5}$	$1.0 \cdot 10^{-7}$
BER_{dc} simulation	$3.2 \cdot 10^{-3}$	$6.4 \cdot 10^{-4}$	$9.2 \cdot 10^{-5}$	$1.3 \cdot 10^{-5}$	$1.1 \cdot 10^{-6}$

As it can be seen the analytical formulas that are given and the simulation results agree, although the mean value of number of erasures is approximately calculated. However, such analytical results can not be found for the case, when a word is not discarded, since the BF decoders are not easy to describe analytically in a probabilistic fashion. Thus, the error rate in the probability of no decoder fail, which is shown by p_s can not be expressed analytically. Also the error rate in the probability of iterative decoder fail due to stopping sets can only be analyzed considering the effects of erasures. If the size of the stopping set that is an element of D_i is shown by j, then the mean value of the erasures due to decoder failure is;

$$f_? = \frac{\sum_{i=s}^{b-1} D_i \cdot j \cdot p_1^i \cdot (1-p_1)^{n-i}}{\sum_{i=s}^{b-1} D_i \cdot p_1^i \cdot (1-p_1)^{n-i}},$$

which is again the direct calculation of the mean value of a stochastic variable.

The effect of erasures on BER is given in (6-6), and the effect of errors in the codeword is directly added to the decoder errors, since the failure only effects the iterative erasure decoder. As stated before, we can not do a direct comparison using these values, since the effects of errors are not calculated here.

$$BER_{fe} = \frac{p_f \cdot (f_?/2)}{n} \tag{6-6}$$

In Table 6-2, we give the probability of the iterative erasure decoder to fail p_f, the effect of erasures due to discarding words BER_{de} (which can also be calculated by (6-5) using only the erasure term), and the effect of the erasures due to decoder failures on BER BER_{fe} for (15,6,5) LDPC code, with $p_1 = 0.1$ & $p_2 = 0.1$ on BSEEC by Matlab simulations. We know that the code has twelve weight-five codewords and no stopping sets of weight-five except for the codewords (in our best knowledge), so we can use this value to check the validity of the formulas given in (6-3) and (6-6). If we select $b = 6$, then (6-3) gives $4.18 \cdot 10^{-5}$ for p_f, and (6-6) gives $6.96 \cdot 10^{-6}$ for BER_{fe}.

Table 6-2: The probability of a word to be discarded and the effects of erasures to BER for different values of the separation bound with (15,6,5) LDPC code on BSEEC

Separation bound	5	6	7	8	9
p_f	0	$4.27 \cdot 10^{-5}$	$9.41 \cdot 10^{-5}$	$1.15 \cdot 10^{-4}$	$1.22 \cdot 10^{-4}$
BER_{fe}	0	$7.12 \cdot 10^{-6}$	$1.57 \cdot 10^{-5}$	$1.94 \cdot 10^{-5}$	$2.05 \cdot 10^{-5}$
BER_{de}	$2.1 \cdot 10^{-3}$	$4.5 \cdot 10^{-4}$	$7.23 \cdot 10^{-5}$	$8.96 \cdot 10^{-6}$	$8.52 \cdot 10^{-7}$

In Table 6-1, effects of discarding words have been inspected in detail, both for errors and erasures in the codeword. However, we do not have an analytical solution for the error decoder, as we have for the erasure decoder. Thus, we compare only the effects of erasures on the BER calculation in Table 6-2, and observe that the total effect of the erasure, which is the sum of BER_{fe} and BER_{de} is smaller for higher separation bounds. This is very well expected from the formulas of p_f and p_d, from p_f formula we see that only valid codewords cause uncorrected erasures, and from p_d formula we see that all possible bit combinations cause uncorrected erasures. The error decoding perspective is different, since all of the punctured codes with different distances and decoding performances get into the picture. A simple opinion that might be stated is that coding should always be preferred, since our aim error correction by coding. However, it might be logical to focus on the simulation results to judge the performance of the complete error erasure decoder regarding the separation bound. In Figure 6-11 the BER performance of the complete EED on BSEEC ($p_1 = 0.1$) using the ($n = 15, k = 6, d = 5$) LDPC code with different separation bounds. The BER values of separation bound 8 and 9 are nearly the same, thus only a single line can be observed for these two values.

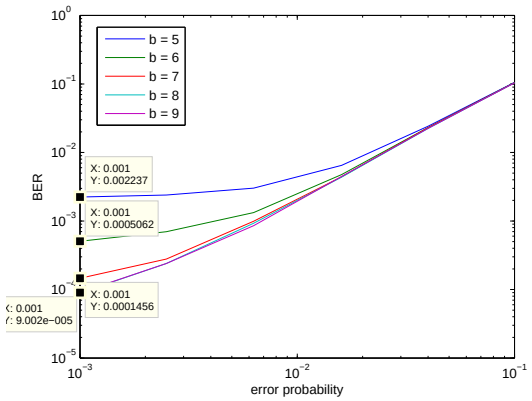

Figure 6-11: BER performance of the complete EED with different separation bounds on BSEEC with $p_1 = 0.1$ using (15,6,5) LDPC code

The BER values in the low error probability region are highly influenced by the erasures that can not be corrected, which can be seen when the figure is observed in the light of Table 6-2. Especially for separation bound values 5 and 6, the data cursors show that the BER value is nearly equal to the effect of the uncorrected erasures given in Table 6-2. However, the improvement is not limited to the low error probability region, only the effect of corrected erasures is much more significant in that region.

In Figure 6-12, the BER performance of the complete EED on BSEEC with $p_1 = 0.1$, using the $(n = 96, k = 48, d = 6)$ LDPC code is given for separation bound of $d = 6$ and $n - k = 48$. It is obvious that the decoder with the higher separation bound significantly outperforms the one with the lower separation bound. We can explain this difference by the amount of words that is discarded. When the separation bound is 6, most of the words are discarded (92% of the words), which can also be calculated using (6-2). On the other hand, all of the codewords are decoded when the separation bound is 48 which result in better BER performance, except for high error probability region. For very high probability region, the decoder may decode the codewords as wrong (but valid) codewords due to the high amount of wrong bits, which increases the BER further. We can conclude that especially for large codes with small distances, selecting the separation bound as the code distance damages the decoding process severely. Also we have observed that selecting the separation bound as large as possible (upper bounded by $n - k$ of course) is beneficial and results in improvement.

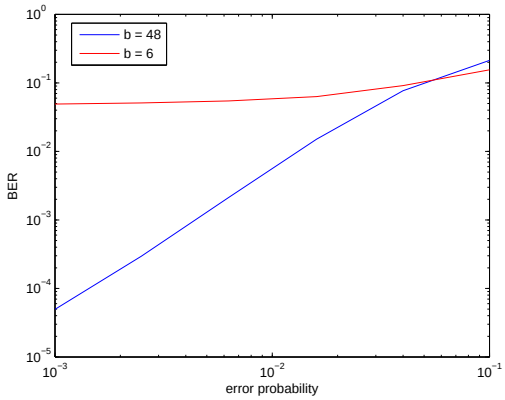

Figure 6-12: BER performance of the complete EED with separation bounds 6 and 48, with $(n = 96, k = 48, d = 6)$ LDPC code on BSEEC with $p_1 = 0.1$

Finally in Figure 6-13, the same comparison above is given for AWGN+EC. Again, the erasure probability is 0.1, and two different separation bounds; $d = 6$ and $n - k = 48$ are compared.

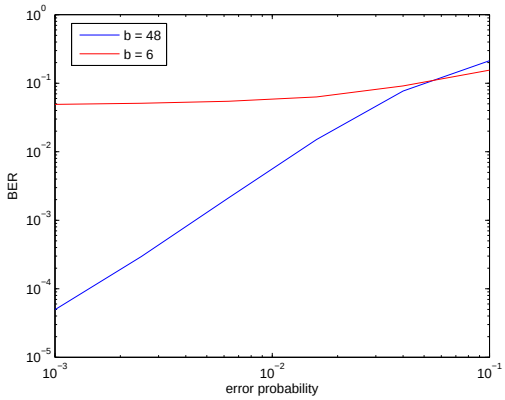

Figure 6-13: BER performance of the complete EED with separation bounds 6 and 48, with $(n = 96, k = 48, d = 6)$ LDPC code on AWGN+EC with $p_1 = 0.1$

As we have stated, most of the formulas that we have given are for BSEEC, however, the main idea is applicable to AWGN+EC. The large gap is again due to the words that are discarded as the case above. The number of words that is discarded is the same, since erasure probabilities are same in these two channels.

6-2-2 The modification for stopping set prevention

In Chapter 5, we have given Theorem 3, which introduces a modification the rank completer which prevents all of the stopping sets that are not the support of a codeword. The theorem simply states that if the extra rows that are calculated in the rank completer algorithm are added to the initial matrix, then this new matrix (big PCM) does not have any stopping sets (except for codeword supports). The new block diagram for the system is given in Figure 6-14.

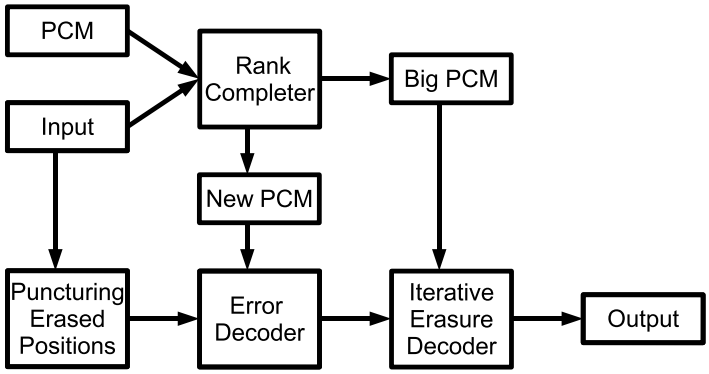

Figure 6-14: Block diagram for the complete EED with stopping set prevention

6-2-3 The BF decoder selection

Finally, we give the performance of the rank completer method on AWGN+EC ($p_1 = 0.1$), with MWBF, LP-WBF, and MRRBF in Figure 6-15 using $(96, 48, 3)$ LDPC code. The best performing decoder with the rank completer method is MWBF. This is surprising at first glance, because MRRBF and LP-WBF outperform MWBF with no erasures. The differences in the performance can be tied to the loss of sparsity and the regularity of the matrices due to separation of the erased bits. The rank completer method creates irregular matrices from regular LDPC parity check matrices due to linear matrix operations. Thus, the number of ones in every row/column is not the same, which disturbs the MRRBF and LP-WBF decoders. MRRBF decoder uses reliability ratios of the bits; and if the number of ones in a row (parity check equation) differ row by row, then the ratios might alter between bits that are an element of different parity check equations. LP-WBF sums the self reliability and the minimum reliability of the bits for all parity check equations, and if the number of ones in a column may differ from column to column, then some of the bits might be considered as reliable, although they are not, and vice versa. On the other hand, MWBF is much more stable against irregularity, since it does not work with ratios or it does not calculate the metric

in a way that is dependent to the number of parity check equations the bit is an element of. The decoders behave the same for larger LDPC codes such as (504,252) and (816,408) LDPC codes.

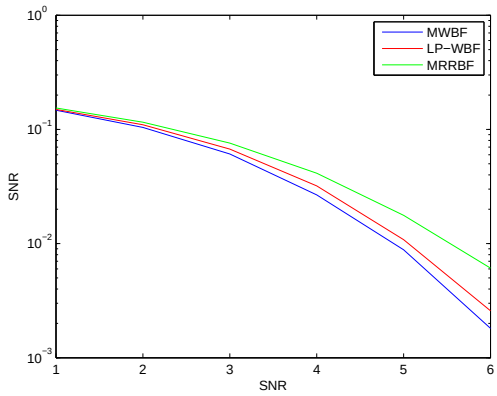

Figure 6-15: The performance of RC with different BF decoders on AWGN+EC with 0.1 erasure probability, using (96,48,3) LDPC code

To sum up, in this section, we have shown that the rank completer method can be used to work with the EED that is described in [1]. We have also investigated the separation bound; which is the minimum number of erasures that the decoder does not separate from the word. We have shown that selecting the separation bound as d can cause serious degradation in the decoder performance especially for LDPC codes, and proposed the selection of the highest separation bound, $n-k$. By selecting a higher separation bound, we have shown that the decoder performance can be improved, since more received words are decoded by the EED.

6-3 Results and Comparisons

In this section, the BER results of the decoding methods that are introduced here are compared to previously known decoders on different channels. For each channel the complexity of methods will be given; however, only rough comparisons will be made, since the scope of this research is not the implementation of these methods.

6-3-1 The performance on BSEEC

Both of the methods that are introduced in this chapter can be used for error erasure decoding on the hard error erasure channel BSEEC. The BER performance comparison of these methods with previously known methods such as min-sum decoding and the binary message passing with erasures are given in this sub-section. MS decoding is a widely used approximation of the belief propagation algorithm which has been introduced in Chapter 2. The BMP-E is a quantized version of the belief propagation, with an alphabet of $\{1,-1,0\}$ for the messages. The BMP-E is also known as Gallager algorithm E, which we have introduced in Chapter 3.

In Figure 6-16; we compare the rank completer method using the standard single bit flipping algorithm, the bootstrapped reliability guessing method using the modified weighted bit flipping method, the min-sum decoding, and the binary message passing with erasures with (96,48) LDPC code.

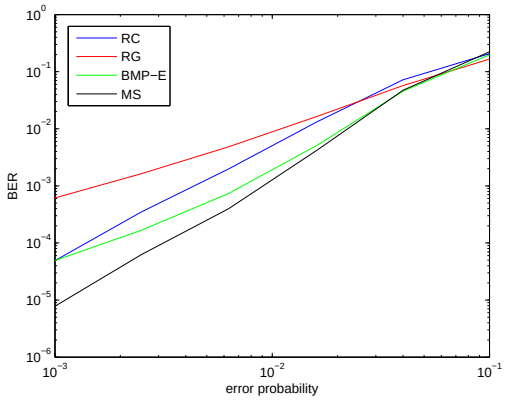

Figure 6-16: The performance of RC, RG, MS, and BMP-E on BSEEC with $p_1 = 0.1$ with (96,48) LDPC code

The min-sum algorithm which is known to be one of the best decoding options for the LDPC codes performs better than the rest, which is very well expected. The min-sum algorithm deals with erasures by inserting zero reliabilities for the erasure positions, which has no additional complexity than error decoding. The rank completer performs worse than the binary message passing decoder, however its performance gets closer to the binary message passing decoder as the error probability gets lower. This is very well expected, since the rank completer method uses the iterative erasure decoder for erasure decoding, which can be seen a form of message passing decoder for erasures only. If the erasure probability is relatively higher than the error probability, then we can claim that the rank completer behaves like the binary message passing decoder. The reliability guessing algorithm, which we have shown previously to be the best of the guessing algorithms on BSEEC performs worse than all other alternatives for low and mid error probability region. However, it still has a reasonable BER performance regarding its simplicity.

In Figure 6-17, the same comparison is given for the (204,102) LDPC code. As it can be seen the performance of the decoders are close to the previous comparison. The gap between the message passing decoders and our methods has increased since the message passing decoders are better than bit flipping algorithms with larger codes.

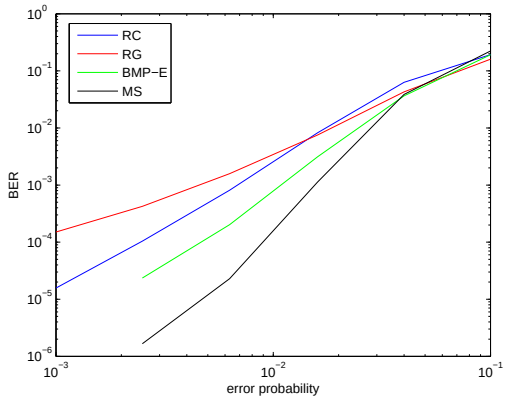

Figure 6-17: The performance of RC, RG, MS, and BMP-E on BSEEC with $p_1 = 0.1$ with (204,102) LDPC code

In Figure 6-18, we compare the rank completer with the bootstrapped reliability guessing algorithm for larger codes; (816,408) and (504,252). The behavior of the two algorithms resemble the previous comparisons, which shows us that the results can be generalized. The results for BMP-E and MS are not given due to large complexity of the decoders with increasing code length. However, both variants of the belief propagation algorithm perform better than the two methods we have introduced in this chapter, which is expected.

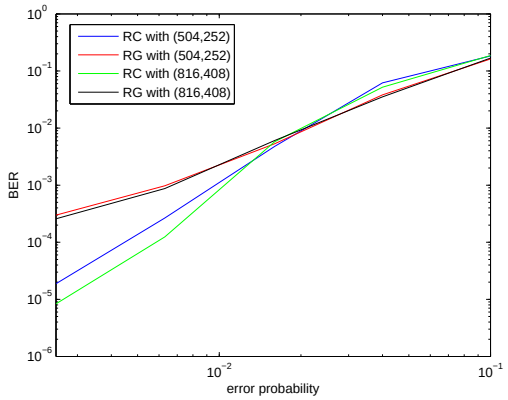

Figure 6-18: The performance of RC and RG on BSEEC with $p_1 = 0.1$ with (504,252) and (816,408) LDPC codes

Finally, the performance of the rank completer method with the min-sum algorithm is compared to the classical min-sum algorithm on BSEEC. In Figure 6-19, the performance of the RC+MS, RC+BF, and MS is given on BSEEC with (96,48) LDPC code. As it can be observed, rank completer with min-sum (RC+MS) algorithm is inferior to min-sum (MS) algorithm and performs close to the RC+BF. This inferiority is due to the puncturing operation, which reduces the error correction capabilities of the code. Assuming erasures and errors in the initial word, the RC+MS separates all erasure positions by puncturing the code. Now the error correction is done using the punctured code, which has a smaller Tanner graph. Since some variable nodes are removed, some routes in the original code are also removed and

the error decoding capability is reduced. Then, the erasure decoding is done using the large graph, and some errors might propagate from the error decoding step. On the other hand, MS algorithm directly uses the original graph to correct errors and erasures by message passing, and for low error probability most of the erasure positions are correctly resolved after a few message passing rounds. Then after these few rounds, the erroneous bits and the wrongly decoded erasure positions, are decoded using the original code which is more competent than the punctured code. This result can be generalized for all channels, although we only present the results for BSEEC in this thesis. Thus, using the rank completer with message passing algorithms (or any type of algorithm that can cope up with both errors and erasures) is not favorable, considering the BER performance.

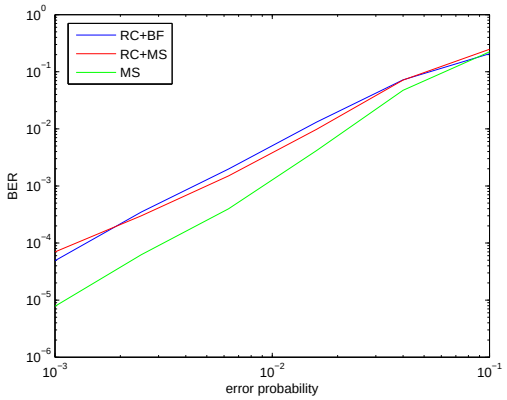

Figure 6-19: The performance of RC+BF, RC+MS, MS on BSEEC with $p_1 = 0.1$ with (96,48) LDPC code

To sum up, the rank completer method performs better than the reliability guessing method on BSEEC, and both of them are outperformed by message passing decoders. However, our methods are less complex than message passing algorithms, since the reliability guessing method does not need any messages to be passed, and the rank completer only uses binary messages for erasure decoding (iterative erasure decoder is a message passing algorithm for the erased bits). The reliability guessing method is the simplest of these methods, although it uses a kind of weighted bit flipping algorithm for error erasure decoder, which is more complex than the plain bit flipping algorithm used by the rank completer. However, the reliability guessing algorithm can be simplified further by designing a weighted bit flipping algorithm that works with values; $\{1, -1, L, -L\}$.

6-3-2 The performance on AWGN+EC

In Figure 6-20; we compare RC and Rank Completer with Bootstrapping (RCB) using the modified weighted bit flipping algorithm, the Reliability Guessing with Bootstrapping (RGB) using the modified weighted bit flipping method, the MS decoding, and the BMP-E with (96,48) LDPC code. The binary message passing with erasures performs the worst since it does not use the soft information in the decoding process. It can be observed that the min-sum decoding outperforms the methods that we have introduced also on AWGN+EC. Bootstrapped rank completer performs better, since bootstrapping improves the performance of the modified weighted bit flipping decoder.

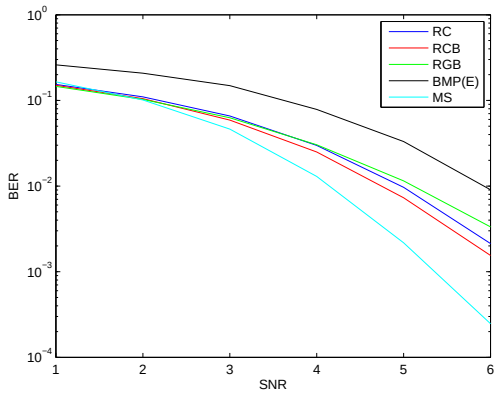

Figure 6-20: The performance of RC, RCB, RGB, MS, and BMP-E on AWGN+EC with $p_1 = 0.1$ with (96,48) LDPC code

In Figure 6-21, the same comparison is given for the (204,102) LDPC code. As it can be seen the performance of the decoders are close to the previous comparison. The gap between the MS and the RC & RG methods is larger in this graph, since the message passing algorithms perform better with increasing code length. Also the gap between BMP-E and the RC & RG methods are smaller, which can be justified by the same reasoning. Regarding this behavior, there might be a code length, where the BMP-E (a hard decision decoder) can beat RG and even RC (soft decision decoders). However, due to time consuming simulations we have not searched for such a code length.

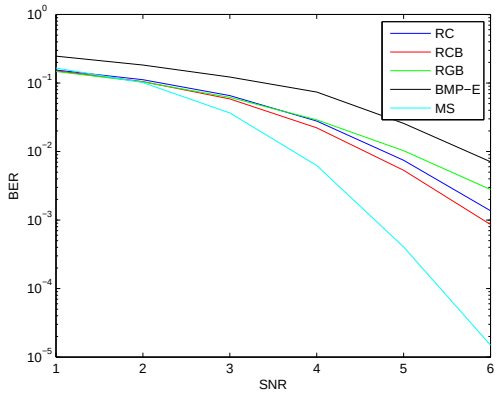

Figure 6-21: The performance of RC, RCB, RGB, MS, and BMP-E on AWGN+EC with $p_1 = 0.1$ with (204,102) LDPC code

In Figure 6-22, the comparison of the bootstrapped rank completer with the bootstrapped reliability guessing method is given for the (816,408) LDPC code. We label them directly as RC and RG, because we only compare the bootstrapped versions. The message passing algorithms are not presented here, since reliable Monte Carlo simulations take too long for large code lengths.

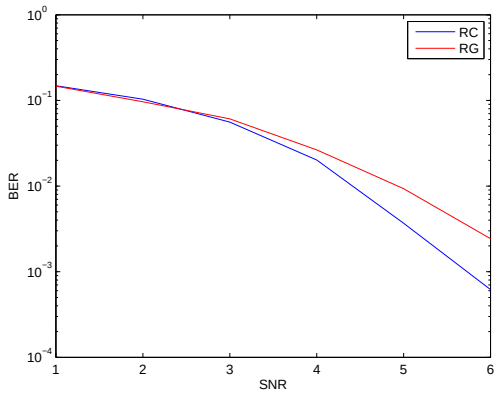

Figure 6-22: The performance of bootstrapped RC and bootstrapped RG on AWGN+EC with $p_1 = 0.1$ with (816,408) LDPC code

The performance of the AWGN+EC can be summarized as follows; the BMP-E performs the worst (for the small and medium length codes), and the RC & RG are outperformed by the MS algorithm. The complexity of the RG method is lower than the RC method, since both methods use the modified weighted bit flipping decoder, where RC method also uses the separation of erasure positions and the iterative erasure decoder. The complexity of the RC and BMP-E is hard to compare, since RC uses the soft information (which requires floating point calculations) and BMP-E uses message passing (which requires routing of messages and connections between nodes). The selection between these two methods depend on the constraints of the implementation and the end-user needs.

6-3-3 The performance on AWGN channel with Rayleigh fading

Finally, we discuss the benefits of erasure declaration while using bit-flipping algorithms in the AWGN channel with Rayleigh fading. In Chapter 1, we have given many examples of communication systems which uses Erasure Declaration (ED) schemes are used to cope up with channel effects such as jamming, fading, and interference. ED has been used for RS codes and convolutional codes, but it has not been proposed for LDPC codes (in our best knowledge). The bit-flipping algorithms can not decode errors and erasures together, thus, using ED with BF algorithms is not possible directly. Since the belief propagation algorithm uses soft information to create messages, ED throws away some of the available information. However, with the rank completer method that we propose, it is possible to use the ED methods with BF algorithms, and then decode errors and erasures altogether.

There are two main types of ED schemes, schemes of the first type use side information about the channel (i.e. information about the fading process for each symbol or bit), and schemes of the second type only uses the reliability value of the received bit. We use a simple ED scheme that does not use any side information about the channel, which is very close to the method that has been described in detail [64]. In this method, the bits are flagged as reliable and unreliable depending on the magnitude of the received value for that bit (the reliability value of the bit) and a pre-determined threshold T. The bits with reliability values less than the threshold are flagged as unreliable and erased from the word, and error

erasure decoding is performed with the word that contains bits that are flagged as reliable and erasures. However, this method has a flaw that has also been expressed in [64], it is not always guaranteed that flagging will erase the bits that are effected by the fading process, since we use no side information. However, as the results will show, this simple ED method is also quite useful especially for high SNR values.

The ED scheme that we describe above can be applied to the LDPC codes using soft decision bit flipping algorithms using the rank completer algorithm. We have selected the MWBF algorithm as the bit flipping algorithm, since in the previous section we have shown that the RC method performs best with MWBF. The ED scheme is modified to make it compatible with the rank completer method; if the number of bits that are flagged as unreliable is larger than the separation bound of the rank completer $(n-k-1)$, erasures are inserted in the least reliable $n-k-1$ bits.

In Figure 6-23, the BER performance of MRRBF, LP-WBF, MWBF, and MWBF+RC (ED) with threshold reliability $T = 0.5$ is given for $(n = 96, k = 48, d_c = 3)$ LDPC code. As it can be seen ED improves the BER performance significantly for high SNR values. The low SNR performance is affected badly, because some bits that have low reliability values due to AWGN channel effects are also erased. This problem can be solved (at least partially) with the use of side information about the channel (and the fading). The high SNR performance is not affected, since the noise very small compared to the signal in the mean, and an erased symbol is most probably has low reliability due to fading.

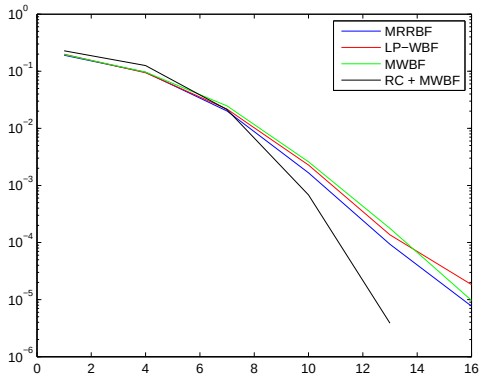

Figure 6-23: The performance of MRRBF, LP-WBF, MWBF, and MWBF+RC (ED) on AWGN channel with Rayleigh fading using $(n = 96, k = 48, d_c = 3)$ LDPC code

In Figure 6-24, the BER performance of MRRBF, LP-WBF, MWBF, and MWBF+RC (ED) is given for $(n = 204, k = 102, d_c = 3)$ LDPC code with $T = 0.5$. The behavior is close to the previous case, which shows that the improvement by ED is not code dependent.

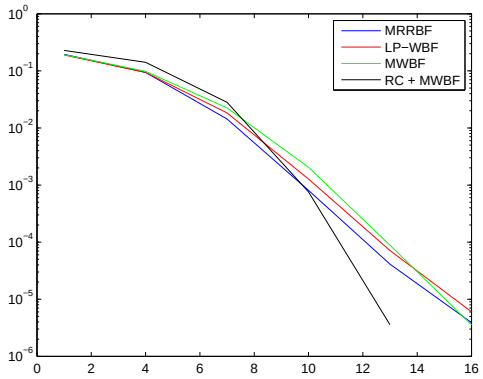

Figure 6-24: The performance of MRRBF, LP-WBF, MWBF, and MWBF+RC (ED) on AWGN channel with Rayleigh fading using $(n = 204, k = 102, d_c = 3)$ LDPC code

If the threshold value is selected to be too low, then the number of bits that are erased and the error erasure decoding provides less improvement. Also if the threshold value is selected to be too high, then useful information might be thrown away due to ED and the error erasure decoding results in worse BER performance than classical error decoding in AWGN channel with Rayleigh fading. This situation is given in Figure 6-25, with threshold values 0.25, 0.5, and 0.75.

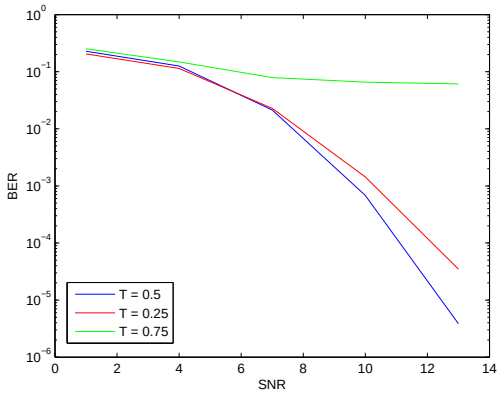

Figure 6-25: The performance of MWBF+RC (ED) on AWGN channel with Rayleigh fading using $(n = 96, k = 48, d_c = 3)$ LDPC code with different T values

The ED method improves the performance of the soft decision bit flipping algorithms, even without side information. However, one should keep in mind that the complexity is increased, due to the declaration of erasures, separation of variables, and iterative erasure decoding (the most complex one). Thus, the RC method can be used to improve the performance of soft decision bit flipping algorithms with an increase in the complexity.

6-4 Complexity Issues and Conclusion

The reliability guessing method is a very simple method, and has nearly the same complexity with any soft decision bit flipping decoder, if the erasure filling part is omitted. On the other hand for the EED with RC; erasures need to be separated from errors, an error decoder is used for error decoding, and an erasure decoder is used for erasure decoding. The most complex of the operations for EED with RC is the iterative erasure decoder, since it is based on message passing. Hence, for an EED with RC with a comparable complexity to the RG method, a simpler erasure decoder should be used. Next, we will compare the complexity of the rank completer method to the separating matrices method.

The only difference in the complexity of the separating matrices and the rank completer is in the initiation period, and the pre-processing part of the rank completer. The complexity of the pre-processing part is dependent on the number of ones in the parity check matrix, and the mean value of the number of additions for a word to be decoded is $(w_c - 1) \cdot p_1 \cdot n$, where w_c is the (mean) column weight, p_1 is the erasure probability, and n is the word length. If one considers the bit flipping algorithms and the LDPC codes, the amount of operations is very low compared to some known soft decision bit flipping decoders such as MWBF and MRRBF. For example MRRBF requires $(w_c - 1) \cdot n$ additions, and other kinds of operations (division, multiplication, modifying the most significant bit) in each iteration. Thus, at least for LDPC codes, we can claim that the pre-processing does not increase the complexity of the system significantly. When we compare the two decoders in the erasure decoding part, the rank completer is way simpler since the known separating matrices are too large.

To conclude, in this chapter we have introduced two new methods for error erasure decoding for linear binary block codes. Although the methods can be generalized for non-binary cases, we have only given the implementations for the linear binary block codes. The reliability guessing method is a new guessing algorithm that assigns reliabilities to guessed bits and can be used for simple systems that has does not need very good BER performance. We have shown it to be more effective than other guessing algorithms that are known. We have studied the EED with RC and taken the "separation idea" further by introducing the separation bound. Also limits of the separation idea are obtained by the simulation results, considering the BER performance. For example, the separation of erasures from errors should not be used with message-passing algorithms (and most probably with any soft decision decoder that is directly capable of error erasure decoding) since the puncturing operation reduces the error correction ability of the code. However, the idea can be used to combine error decoders that are not capable of error erasure decoding (such as BF algorithms) with erasure decoders. We have presented the results of the reliability guessing method and the rank completer method with various decoders on different channels. These results show that our methods are alternatives to message passing decoders for error erasure decoding with worse BER performance, but also with less complexity.

Chapter 7

Conclusions

This research has the objective of creating reliable decoders with low complexity for error channels and error-erasure channels. We have presented the modified reliability ratio bit flipping algorithm, which is a simple modification to the reliability ratio bit flipping algorithm, that is effective for small and medium length LDPC codes. We have also worked on existing ideas for simple EEDG, and chosen to work on guessing algorithms and the separation of erasures from errors idea. Since the separation idea was not realizable, we have searched for a simple and effective method to separate erasures from errors. We have found and presented such a method for separation of erasures from errors in Chapter 5, and shown that it can separate up to $n - k - 1$ erasures from a (n, k, d) linear block code. We have worked on LDPC codes and implemented an EED that utilizes the rank completer to separate erasures from errors. We have also discovered and presented a simple guessing algorithm that assigns low reliability values to guessed bits positions, which we have shown to be better than other guessing algorithms for LDPC codes.

The soft decision bit flipping decoders are quite useful for low end devices that can not handle the complexity of message passing decoders, and has been studied deeply by many researchers. The reliability ratio bit flipping algorithm is known to be one of the best soft decision bit flipping algorithms. We have proposed a modified version of the reliability ratio bit flipping algorithm, which improves the BER performance of the reliability ratio bit flipping algorithm, at the expense of very little additional complexity. The modified algorithm uses the self reliability of each bit weighed by a pre-determined constant β to calculate the flipping metric. The β value that minimizes the BER can be found via Monte Carlo simulations. We have shown that for small and medium length LDPC codes, the proposed algorithm with BER minimizing β values can outperform the classical RRBF algorithm. We have studied the relation between BER minimizing β values and the density of the codes. For very sparse codes, the modification causes a minor improvement in the BER performance, which makes the modification irrelevant. The MRRBF can outperform many good soft decision decoders in small and medium length codes, such as RRBF, MWBF, and LP-WBF.

Separation of erasures from errors is an idea to realize error erasure decoding, with the aid of separating matrices. However, the given construction methods for separating matrices can

guarantee separation for practical values of n and l only for very large matrices. This fact makes the separation idea hard to realize, and to implement. We have proposed the rank completer method to implement the separation idea, which can be applied to any parity check matrix to separate the given erasure set. The rank completer is a pre-processing method that is applied to the parity check matrix of the code, to create the parity check matrix of the punctured code. The pre-processing is done by addition operations for binary codes, and is pretty straightforward to implement. For LDPC codes, the number of operations is very low compared to the number of operations in BF error decoders. For other types of codes, the pre-processing might become more complex, as the number of ones in a column increases. Also for non-binary codes, multiplication and division operations might be used to create the new parity check matrix. We have also introduced the separation bound, which limits the maximum number of erasures that can be separated from a word. An erasure set can only be separated if the size of it is smaller than the separation bound b. We have found it to be $n - k$, unless the support of a codeword is separated (which increases this value). Also we have shown that we can ensure the separated words contain no stopping sets, unless the support of a codeword is separated.

As stated before, decoders are mostly designed for error correction or erasure correction. We have presented two error erasure decoders for linear block codes, which can be considered as simple alternatives to some known decoders. The error erasure decoders that we have presented, originate from well known decoders in the literature. The reliability guessing method is the improved version of the simple guessing algorithm, which guesses the erasure positions and inserts very low reliability values to those positions. After the guessing operation, error decoding is applied, which concludes the error erasure decoding operation. The error erasure decoder that uses the rank completer is the same decoder that is proposed in [1], except for the separation method that is applied. This error erasure decoder separates the erasure positions, and error decoding is applied to the punctured word with the matrix that is found by the rank completer method. After error decoding, the decoding process is ended by applying iterative erasure decoding to recover the erasure positions in the word. These new methods are implemented for LDPC codes and BER performances of these methods are compared to the message passing algorithms in the BSEEC and AWGN+EC. The results show that these new methods are alternatives to message passing decoders with worse performance but less complexity for error erasure decoding. Also the rank completer method can be applied to the bit flipping algorithms on AWGN channels with Rayleigh fading, to implement ED techniques which improves the BER performance of the code. Our results also indicate that this is the case and ED can be used with bit flipping algorithms to improve the BER performance significantly at high SNR values.

Possible future work is summarized below.

- The complexity of RC can be studied for other codes which have larger column weights or which are non-binary.
- Searching for a new erasure decoder for EED with RC, since iterative erasure decoder is too complex (it is based on MP).
- The modified reliability ratio algorithm can be studied further with different codes, more extensive study about BER minimizing β values can be conducted, and other modification methods can be considered.

Appendix A

Matlab Codes

A-1 Appendix Section

A-1-1 MATLAB source code for the Modified Reliability Ratio Based Decoder

```
function [word]=modified_reliability_ratio_hard_decoder(H,b,beta)
    word=b<0;%map the real valued word to binary values
    cons=abs(b);%store the bit reliabilities
    summer=abs(repmat(abs(b),size(H,1),1).*H);
    div=sum(summer,2);
    clear summer;
    miner=abs(repmat(1./abs(b),size(H,1),1).*H);
    HH=repmat((div),1,size(H,2)).*miner;%create a matrix with reliability
        ratios in each position that is a one in H
    clear miner;
for k=1:1:150
    if mod(word*H',2)==0%check if decoding is done
        break;
    else
    check_errors_1=mod(word*H',2);%find the violated equations
    check_errors_2=(2*check_errors_1-1)*HH-beta*cons;%the MRRBF equation
    sortce=sort(check_errors_2,'descend');%order the bits according to
       the metric
    a=find(check_errors_2==sortce(1));%select the bit with highest metric
    a1=a(ceil(size(a)*rand));%if more than one bit has the highest,
       select randomly
    word(a1)=mod(word(a1)+1,2);%flip that bit
    end
end
```

A-1-2 MATLAB source code for the Rank Completer method

```
function [Hp,H2]=parity_check_creator_alt(H,erased_bits)
sortie=erased_bits; %get list of erased bits
Hp=H;%store the original parity check matrix to modify
H2=H;%store the original matrix to add extra rows for stopping set
    prevention
for i=1:1:length(erased_bits)%for all erased bits
        a=find(Hp(:,sortie(i)));%select the rows with a one in the column
            to be separated
        for j=1:1:sum(Hp(:,sortie(i)))-1%for all such rows do the
            addition
            ca=size(Hp,1);%and create extra rows
            Hp(ca+1,:)=mod(Hp(a(1),:)+Hp(a(j+1),:),2);%add extra rows to
                matrix
            cb=size(H2,1);
            H2(cb+1,:)=Hp(ca+1,:);%add extra rows to matrix for stopping
                set prevention
        end
        Hp(a,:)=[];%remove rows with a one in the separated column
end
Hp(:,sortie)=[];%remove separated columns
```

A-1-3 MATLAB source code for the Reliability Guessing method

```
function word_guess_out=soft_erasure_guess(word_channel_out,erased_bits,L
    )
word_guess_out=word_channel_out;
for i=1:1:length(erased_bits)%fill erasure positions with reliability L
if rand<0.5
    word_guess_out=[word_guess_out(1:erased_bits(i)-1) L word_guess_out(
        erased_bits(i):end)];
else
    word_guess_out=[word_guess_out(1:erased_bits(i)-1) -L word_guess_out(
        erased_bits(i):end)];
end
end
```

Bibliography

[1] K. Abdel-Ghaffar and J. Weber, "Separating erasures from errors for decoding," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, 6-11 2008, pp. 215 –219.

[2] F. Guo and L. Hanzo, "Reliability ratio based weighted bit-flipping decoding for ldpc codes," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 1, 30 2005, pp. 709 – 713 Vol. 1.

[3] T. Richardson and R. Urbanke, *Modern Coding Theory*, 1st ed. Cambridge, UK: Cambridge University Press, 2008.

[4] R. G. Gallager, *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.

[5] R. Tanner, "A recursive approach to low complexity codes," *Information Theory, IEEE Transactions on*, vol. 27, no. 5, pp. 533 – 547, sep 1981.

[6] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, p. 1645, 29 1996.

[7] M. Sipser and D. Spielman, "Expander codes," *Information Theory, IEEE Transactions on*, vol. 42, no. 6, pp. 1710 –1722, nov 1996.

[8] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1998, pp. 249–258.

[9] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 585 –598, feb 2001.

[10] L. Provvedi, C. Rattray, J. Hofmann, and S. Parolari, "Provision of mbms over the geran: technical solutions and performance," in *3G Mobile Communication Technologies, 2004. 3G 2004. Fifth IEE International Conference on*, 2004, pp. 494 – 498.

[11] W. Xu, T. Gasiba, T. Stockhammer, H. Jenkac, and G. Liebl, "Iterative decoding for geran mbms," in *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, vol. 3, 11-14 2005, pp. 1724 –1728 Vol. 3.

[12] K. Cheun and W. Stark, "Probability of error in frequency-hop spread-spectrum multiple-access communication systems with noncoherent reception," *Communications, IEEE Transactions on*, vol. 39, no. 9, pp. 1400 –1410, sep 1991.

[13] A. Park, R. Buehrer, and B. Woerner, "Throughput performance of an fhma system with variable rate coding," *Communications, IEEE Transactions on*, vol. 46, no. 4, pp. 521 –532, apr 1998.

[14] V. S. Lin and G. J. Pottie, "Channel coding for a frequency-hopped wireless transceiver," in *Information Theory and Applications*, ser. Lecture Notes in Computer Science, vol. 793. Springer Berlin / Heidelberg, 1994, pp. 109 –124.

[15] S. Ahmed, L.-L. Yang, and L. Hanzo, "Erasure insertion in rs-coded sfh mfsk subjected to tone jamming and rayleigh fading," in *Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd*, vol. 2, 25-28 2005, pp. 917 – 921.

[16] M. Yang and W. Ryan, "Performance of efficiently encodable low-density parity-check codes in noise bursts on the epr4 channel," *Magnetics, IEEE Transactions on*, vol. 40, no. 2, pp. 507 – 512, march 2004.

[17] J. Ha and S. McLaughlin, "Low-density parity-check codes over gaussian channels with erasures," *Information Theory, IEEE Transactions on*, vol. 49, no. 7, pp. 1801 – 1809, july 2003.

[18] D. Toumpakaris, J. Cioffi, D. Gardan, and M. Ouzzif, "A square distance-based byte-erasure method for reduced-delay protection of dsl systems from non-stationary interference," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, vol. 4, 1-5 2003, pp. 2114 – 2119 vol.4.

[19] M. Ardakani, F. Kschischang, and W. Yu, "Low-density parity-check coding for impulse noise correction on power-line channels," in *Power Line Communications and Its Applications, 2005 International Symposium on*, 6-8 2005, pp. 90 – 94.

[20] M. Ivkovic and S. Cui, "Hard decision error correcting scheme based on ldpc codes over impulse noise channels," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, oct. 2006, pp. 2204 –2208.

[21] A. Doenmez, T. Hehn, S. Laendner, and J. B. Huber, "Comparison of high-performance codes on awgn channel with erasures," in *4th International Symposium on Turbo Codes in Connection with the 6th International ITGConference on Source and Channel Coding*, apr. 2006.

[22] C. Dikici, K. Idrissi, and A. Baskurt, "Dirty-paper writing based on LDPC codes for data hiding," in *Multimedia Content Representation, Classification and Security*, ser. Lecture Notes in Computer Science, vol. 4105. Springer Berlin / Heidelberg, 2006, pp. 114–120.

[23] E. Candes and T. Tao, "Decoding by linear programming," *Information Theory, IEEE Transactions on*, vol. 51, no. 12, pp. 4203 – 4215, dec. 2005.

[24] J. Feldman, M. Wainwright, and D. Karger, "Using linear programming to decode binary linear codes," *Information Theory, IEEE Transactions on*, vol. 51, no. 3, pp. 954 – 972, march 2005.

[25] J. Weber and K. Abdel-Ghaffar, "Results on parity-check matrices with optimal stopping and/or dead-end set enumerators," *Information Theory, IEEE Transactions on*, vol. 54, no. 3, pp. 1368 –1374, march 2008.

[26] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *Information Theory, IEEE Transactions on*, vol. 48, no. 6, pp. 1570 –1579, jun 2002.

[27] J. Weber and K. Abdel-Ghaffar, "On decoding failure probabilities for linear block codes on the binary erasure channel," in *Information Theory Workshop, 2006. ITW '06 Chengdu. IEEE*, 22-26 2006, pp. 24 –28.

[28] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[29] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[30] Y. Zhang, D. Yuan, and H. Zhang, "Adaptive ldpc for rayleigh fading channel," in *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, vol. 2, 28 2004, pp. 651 – 656 Vol.2.

[31] S. A. Vanstone and P. C. van Oorschot, *An Introduction to Error Correcting Codes with Applications*. Kluwer Academic Publishers, 1989.

[32] J. L. Goldwasser, "Shortened and punctured codes and the macwilliams identities," *Linear Algebra and its Applications*, vol. 253, no. 1-3, pp. 1 –13, 1997.

[33] J. Justesen and T. Høholdt, *A course in error-correcting codes*. EMS Textbooks in Math, 2004.

[34] K. Ngo Minh Tri, Abdel-Ghaffar and J. Weber, "New upper bounds on the separating redundancy of linear block codes," in *Thirtieth Symposium on Information Theory in the Benelux, Eindhoven, The Netherlands*, may 2009, pp. 209 – 216.

[35] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM Journal of Applied Math.*, vol. 8, pp. 300 –304, 1960.

[36] J. Massey, "Shift-register synthesis and bch decoding," *Information Theory, IEEE Transactions on*, vol. 15, no. 1, pp. 122 – 127, jan 1969.

[37] J. Forney, G., "On decoding bch codes," *Information Theory, IEEE Transactions on*, vol. 11, no. 4, pp. 549 – 557, oct 1965.

[38] R. E. Blahut, "A universal reed-solomon decoder," *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 150 –158, march 1984.

[39] T. Truong, I. Hsu, W. Eastman, and I. Reed, "Simplified procedure for correcting both errors and erasures of reed-solomon code using euclidean algorithm," *Computers and Digital Techniques, IEE Proceedings E*, vol. 135, no. 6, pp. 318 – 324, nov 1988.

[40] P. Stevens, "Error-erasure decoding of binary cyclic codes, up to a particular instance of the hartmann-tzeng bound," *Information Theory, IEEE Transactions on*, vol. 36, no. 5, pp. 1144 –1149, sep 1990.

[41] C. Hartmann and K. Tzeng, "Generalizations of the bch bound," *Information and Control*, vol. 20, no. 5, pp. 489 –498, Jun. 1972.

[42] J.-H. Jeng and T.-K. Truong, "On decoding of both errors and erasures of a reed-solomon code using an inverse-free berlekamp-massey algorithm," *Communications, IEEE Transactions on*, vol. 47, no. 10, pp. 1488 –1494, oct 1999.

[43] T.-K. Truong, J. Jeng, and K.-C. Hung, "Inversionless decoding of both errors and erasures of reed-solomon code," *Communications, IEEE Transactions on*, vol. 46, no. 8, pp. 973 –976, aug 1998.

[44] T.-C. Chen, C.-H. Wei, and S.-W. Wei, "Step-by-step error/erasure decoding reed-solomon codes," in *Communication Systems, 2002. ICCS 2002. The 8th International Conference on*, vol. 1, 25-28 2002, pp. 322 – 326 vol.1.

[45] L. Atieno, J. Allen, D. Goeckel, and R. Tessier, "An adaptive reed-solomon errors-and-erasures decoder," in *ACM/SIGDA 14th international Symposium on Field Programmable Gate Arrays*, 2006, pp. 150 – 158.

[46] T.-K. Truong, J.-H. Jeng, and T. Cheng, "A new decoding algorithm for correcting both erasures and errors of reed-solomon codes," *Communications, IEEE Transactions on*, vol. 51, no. 3, pp. 381 – 388, march 2003.

[47] S. Karande and H. Radha, "The utility of hybrid error-erasure ldpc (heel) codes for wireless multimedia," in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 2, 16-20 2005, pp. 1209 – 1213 Vol. 2.

[48] M. Mitzenmacher, "A note on low density parity check codes for erasures and errors," in *SRC Tech. Note 1998-017, COMPAQ*, 1998.

[49] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599 –618, feb 2001.

[50] A. Krishnan, R. Radhakrishnan, and B. Vasic, "Ldpc decoding strategies for two-dimensional magnetic recording," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, nov. 2009, pp. 1 –5.

[51] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *Information Theory, IEEE Transactions on*, vol. 47, no. 7, pp. 2711 –2736, nov 2001.

[52] J. Zhang and M. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *Communications Letters, IEEE*, vol. 8, no. 3, pp. 165 – 167, march 2004.

[53] M. Jiang, C. Zhao, Z. Shi, and Y. Chen, "An improvement on the modified weighted bit flipping decoding algorithm for ldpc codes," *Communications Letters, IEEE*, vol. 9, no. 9, pp. 814 – 816, sep 2005.

[54] C.-H. Lee and W. Wolf, "Implementation-efficient reliability ratio based weighted bit-flipping decoding for ldpc codes," *Electronics Letters*, vol. 41, no. 13, pp. 755 – 757, 23 2005.

[55] Z. Liu and D. Pados, "Low complexity decoding of finite geometry ldpc codes," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 4, 11-15 2003, pp. 2713 – 2717 vol.4.

[56] J. Sha, M. Gao, Z. Zhang, L. Li, and Z. Wang, "Self reliability based weighted bit-flipping decoding for low-density parity-check codes," in *5th WSEAS Int. Conf. on Instrumentation, Measurement, Circuits and Systems, Hangzhou, China*, 2006.

[57] M. Shan, C. Zhao, and M. Jiang, "Improved weighted bit-flipping algorithm for decoding ldpc codes," *Communications, IEE Proceedings-*, vol. 152, no. 6, pp. 919 – 922, 9 2005.

[58] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding ldpc codes," *Communications, IEEE Transactions on*, vol. 58, no. 6, pp. 1610 –1614, june 2010.

[59] T. Ngatched, F. Takawira, and M. Bossert, "A modified bit-flipping decoding algorithm for low-density parity-check codes," in *Communications, 2007. ICC '07. IEEE International Conference on*, 24-28 2007, pp. 653 –658.

[60] X. Wu, C. Zhao, and X. You, "Parallel weighted bit-flipping decoding," *Communications Letters, IEEE*, vol. 11, no. 8, pp. 671 –673, august 2007.

[61] A. Nouh and A. Banihashemi, "Bootstrap decoding of low-density parity-check codes," *Communications Letters, IEEE*, vol. 6, no. 9, pp. 391 – 393, sep 2002.

[62] Y. Inaba and T. Ohtsuki, "Performance of low density parity check (ldpc) codes with bootstrap decoding algorithm on a fast fading channel," in *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, vol. 1, 17-19 2004, pp. 333 – 337 Vol.1.

[63] G. Li and G. Feng, "Improved parallel weighted bit-flipping decoding algorithm for ldpc codes," *Communications, IET*, vol. 3, no. 1, pp. 91 –99, january 2009.

[64] S. Wicker, "Reed-solomon error control coding for rayleigh fading channels with feedback," *Vehicular Technology, IEEE Transactions on*, vol. 41, no. 2, pp. 124 –133, may 1992.

Glossary

List of Acronyms

ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian Noise Channel
AWGN+EC	Additive White Gaussian Noise Channel with Erasures
BCH	Bose-Chaudhuri-Hocquenghem
BEC	Binary Erasure Channel
BER	Bit Error Rate
BF	Bit Flipping
BMA	Berlekamp-Massey Algorithm
BMP-E	Binary Message Passing with Erasures
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
BSEEC	Binary Symmetric Error Erasure Channel
CRC	Cyclic Redundancy Codes
ED	Erasure Declaration
EEC	Error-Erasure Channels
EED	Error Erasure Decoder
EEDG	Error Erasure Decoding
FEC	Forward Error Correction

FER	Frame Error Rate
GERAN	GSM EDGE Radio Access Network
IMWBF	Improved Modified Weighted Bit Flipping
LDPC	Low-Density Parity-Check
LP	Linear Programming
LP-WBF	Liu-Pados Weighted Bit Flipping
MAC	Medium Access Control
MBMS	Multimedia Broadcast/Multicast Service
ML	Maximum Likelihood
MP	Message Passing
MRRBF	Modified Reliability Ratio Based Bit Flipping
MS	Min-Sum
MWBF	Modified Weighted Bit Flipping
NP	Non-deterministic Polynomial-time
PCM	Parity Check Matrix
RC	Rank Completer
RCB	Rank Completer with Bootstrapping
RGB	Reliability Guessing with Bootstrapping
RLC	Radio Link Control
RS	Reed-Solomon
RTT	Ratio Threshold Test
SECDED	Single Error Correcting Double Error Detecting
SM	Separating Matrices
WBF	Weighted Bit Flipping