INVESTIGATION OF ERROR CORRECTING CODES OVER BINARY ERASURE CHANNEL AND IMPLEMENTATION OF DIFFERENT TECHNIQUES FOR DECODING

by

Zeliha Asena Kırık

A final report submitted for EE492 senior design project class
in fulfillment of the requirements for the degree of
Bachelor of Science
(Department of Electrical and Electronics Engineering)
in Boğaziçi University

July 10th, 2020

Principal Investigator:
Prof. Dr. Ali Emre Pusane

ACKNOWLEDGMENTS

This project is supported by Boğaziçi University, Department of Electrical and Electronics Engineering. I would like to express my sincere gratitude to my supervisor, Prof. Ali Emre Pusane, for his advice, guidance, and encouragement throughout the course of this work.

I would like to extend my thanks to doctoral student Mustafa Cemil Coşkun from the Technical University of Munich for his contributions, valuable advice, and discussion.

ABSTRACT

We investigate the inactivation decoding algorithm for low-density parity-check (LDPC) and Reed-Muller (RM) codes over the binary erasure channel(BEC). This decoder combines the optimality of the Gaussian elimination(GE), i.e., maximum-likelihood(ML) decoding, with the efficiency of the belief-propagation algorithm(peeling decoder). For the case of the number of inactivations is unbounded, inactivation decoder is equivalent to an ML decoder. The main motivation behind inactivation decoding is to provide a much more powerful decoding algorithm, i.e., decode whenever ML decoder can succeed, but at the same time apply the efficient belief propagation decoding to the utmost.

The proposed decoder employs peeling decoder(PD) and inactivation procedure successively. It employs PD until the decoder gets stuck, then, assigns a dummy variable to an erased information bit. PD continues by letting the upcoming messages be affine functions of these inactivated variables. In the end, inactivated bits are resolved using information gathered from code constraints.

We present the block erasure rate (BLERs) vs. bit erasure rate for various code-length LDPC and RM codes via Monte-Carlo simulations. Simulations show the selection of which variable to be inactivated matters. Among various selection procedures, inactivating the most connected variable shows superior performance. Accordingly, we inactivated the variables which are connected mostly in the following simulations. We also provide the average number of inactivation vs. the bit erasure rate by simulations.

Another interesting issue is to study the structure and minimum size of the stopping sets as one of the most common causes as the failure of PD. We point out the relation between the rank of stopping sets vs. the average number of inactivation by simulations. These results give insight into the dynamics of the inactivation decoding, showing the effect of adjusting the probable causes of peeling decoder failure.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii

ABSTRACT iii

LIST OF FIGURES v

CHAPTER

1. INTRODUCTION 6

2. PRELIMINARIES 8

2.1 Channel Models 8
2.2 Linear Block Codes 9
2.2.1 Reed-Muller Codes 10
2.2.2 LDPC Codes 11
2.3 Maximum-likelihood Decoding 11

3. PEELING DECODER 13
3.1 Introduction 13
3.2 Algorithm 13

4. GUESSING DECODER 16
4.1 Algorithm 16

5. INACTIVATION DECODER 17

5.1 Introduction 17
5.2 Algorithm 17

6. RESULTS 29

7.CONCLUSION 29

7.1 Discussion 29
7.2 Further Work 29
7.3 Social, Environmental and Economical Impact 30
7.4 Cost Analysis 30

BIBLIOGRAPHY 32

LIST OF FIGURES

Figure 1.1: Basic digital communication system block diagram 6
Figure 2.1: Binary Symmetric Channel (BSC) and Binary Erasure Channel (BEC) ... 8
Figure 2.2: Tanner Graph Representation 10
Figure 3.1: Peeling Decoder 14
Figure 5.1:Inactivation Decoder Steps 19
Figure 5.2: Inactivation Decoding-Vector Implementation 20
Figure 6.1:Peeling/ML/Inactivation Performance on RM(3,7) over BEC 22
Figure 6.2: Peeling/ML/Inactivation Performance on RM(3,7) ,RM(2,5)over BEC.. 23
Figure 6.3: Average Number of Inactivations for Max-Min for RM(3,7) 24
Figure 6.4: Average Number of Inactivations vs Erasure Probability for RM(2,5) 24
Figure 6.5: Average Number of Inactivations vs Erasure Probability for RM(3,7) 24
Figure 6.6: Average Number of Inactivations vs Erasure Probability for RM(4,9) 24
Figure 6.7: Number of Inactivation vs Size of Full-Rank Stopping Sets at e=0.35 25
Figure 6.8: Size of Full-Rank Stopping Sets vs Erasure Probability for RM(2,5) 26
Figure 6.9: Size of Full-Rank Stopping Sets vs Erasure Probability for RM(3,7) 26
Figure 6.10: Size of Full-Rank Stopping Sets vs Erasure Probability for RM(4,9) 26
Figure 6.11: Peeling/ML/Inactivation Performance on LDPC(64,128) over BEC 27
Figure 6.12: Average Number of Inactivations vs Erasure Probability for LDPC(64,128) 27
Figure 6.13: Average Number of Guesses in Guessing Decoder for RM(3,7) 27

CHAPTER 1
INTRODUCTION

Communication has been one of the most profound needs of humanity throughout history. It is essential to forming societies, educating young generations, and expressing emotions, needs, and in general any information. Our desire to have more reliable communication goes back to the earliest times, such that smoke signals of primitive societies can be considered as one of the earliest steps in communication technology. Ever since then, the various communication disciplines in engineering have the objective of providing technical assistance to communication.

Initially, these communication technologies were developed as separate networks. As these networks grew, however, the constraint that all parts of a given network had to work together, caused an increased focus on the underlying principles and architectural understanding required for continued system evolution among many research centers[1].

One of the greatest contributions of these institutes was the creation of Information Theory by Claude Shannon in 1948. In his seminal paper, A Mathematical Theory of Communication, Shannon established a digital communication framework[2]. He proved the existence of channel codes which ensure reliable communication provided that the information rate for a given code did not exceed the so-called capacity of the channel [3]. The pioneering block diagram is depicted in Figure 1.1, whose various components are described as follows:

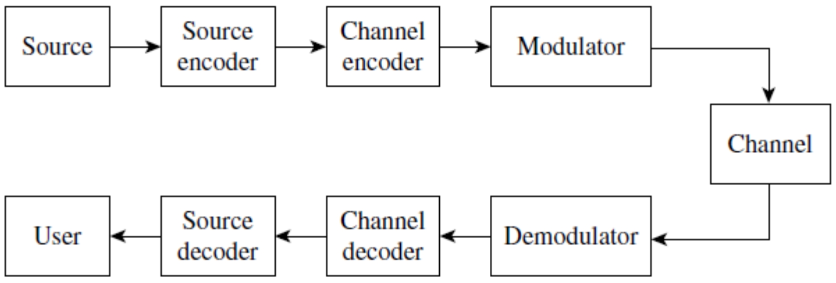

Fig. 1.1 Basic digital communication system block diagram.

Source coding is a mapping from source symbols, to a sequence of alphabet symbols such that the source symbols can be exactly recovered from the binary bits. This concept became the backbone of data compression. Channel coding is performed both at the transmitter and the receiver. At the transmitter side, channel coding is referred to as encoder, where extra bits (parity bits) are added with the raw data before modulation. At the receiver side, channel coding is referred to as the decoder. Channel coding enables the receiver to detect and correct errors due to noise, interference and fading.

In the first 45 years following Shannon's publication, various effective coding systems had been designed. However, none of them had achieved Shannon's theoretical limit. The first breakthrough came in 1993 with the discovery of turbo codes, which are the first class of codes shown to operate near Shannon's capacity limit[4]. A second breakthrough came around in 1996 with the rediscovery of LDPC codes[5]. Among this large variety of code developments, one of the first, simplest, and most ubiquitous code is the Reed-Muller(RM) code. Following their introduction [6], various decoding algorithms for RM codes have been proposed to achieve performance close to maximum a-posteriori (MAP). Recently, it has been shown that RM codes can achieve capacity on the BEC under MAP decoding [7].

The recent surge of interest in the BEC is mostly because it is the prime example of a channel over which the performance of iterative decoding algorithms can be analyzed precisely. It was shown by Di, Proietti, Telatar, Richardson, and Urbanke [8] that the performance of a LDPC code under iterative decoding on the BEC is completely determined by certain combinatorial structures.

Our proposed decoding scheme in this study, inactivation decoding, belongs to a large class of GE algorithms tailored to the solution of large sparse linear systems [9]–[11]. The algorithm can be seen as an extension of PD, where whenever the PD stops, a variable node is declared as inactive (i.e., removed from the equation system and check nodes are updated) and PD resumes. At the end of the process, the inactive variables have to be solved using GE. If a unique solution is found, it is possible to recover all the remaining input symbols by back-substitution.

CHAPTER 2
PRELIMINARIES

2.1 Channel Models

The Binary Erasure Channel (BEC) is a discrete memoryless channel (DMC) with binary input alphabet X = {0, 1}, ternary output alphabet Y = {0, 1, ε} and probability of erasure δ. According to this model, a transmitter sends a bit and the receiver either receives the bit or it receives a message that the bit was not received (erased). This channel is used frequently in information theory because it is one of the simplest channels to analyze.

$P_{Y|X}(0|0) = P_{Y|X}(1|1) = 1 - \delta \quad P_{Y|X}(\varepsilon|0) = P_{Y|X}(\varepsilon|1) = \delta \quad P_{Y|X}(1|0) = P_{Y|X}(0|1) = 0$

The symbol ε denotes the erasure event.

The Binary Symmetric Channel (BSC) is a DMC with binary input alphabet X = {0,1}, binary output alphabet Y = {0, 1} and crossover probability p. According to this model, a transmitter sends a bit and the receiver receives a bit. It is assumed that the bit is usually transmitted correctly, but that it will be "flipped" with a small probability.

$P_{Y|X}(0|0) = P_{Y|X}(1|1) = 1 - p \quad P_{Y|X}(1|0) = P_{Y|X}(0|1) = p$

where δ is the channel error probability.

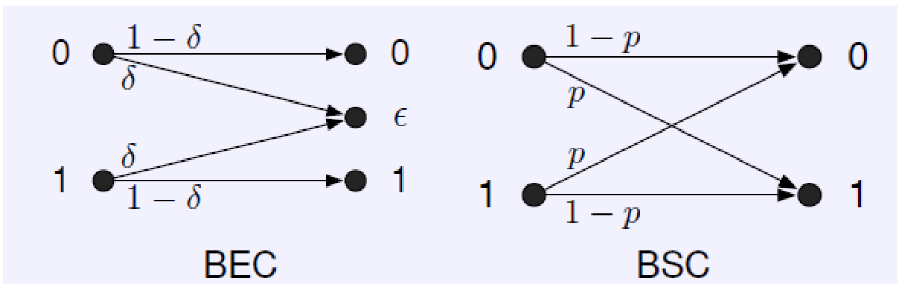

Fig. 2.1 Binary Symmetric Channel (BSC) and Binary Erasure Channel (BEC)

2.2 Linear Block Codes

An *(n,k)* linear code is a k-dimensional subspace of the vector space of all the binary n-tuples, so it is possible to find k linearly independent code words $g_0, g_1, \cdots, g_{k-1}$ to span this space.

An *(n, k)* binary linear block code *C* is usually defined via its binary generator matrix *G*, i.e. a *(k×n)* matrix. The k rows of the generator matrix span the k-dimensional code space. The generator matrix allows encoding by mapping the k-bits information word u onto a n-bits codeword **c** through the linear equation.

$$\mathbf{c} = \mathrm{u}G \tag{1}$$

If the generator matrix possesses a *(k×k)* sub-matrix in identity form, the matrix is referred to as systematic-form generator matrix, and the resulting encoder is said to be systematic.

The code rate (R) is defined as

$$\mathrm{R} = \frac{\mathrm{k}}{\mathrm{n}} \tag{2}$$

Alternatively, a binary linear block code may be defined through its (n−k) × n parity-check matrix *H* via the equation

$$\mathbf{c}H_T = 0 \tag{3}$$

where 0 is the n-elements all-zero vector. The parity-check matrix rows span the subspace orthogonal to *C*. We denote such subspace as $C_\perp$. Thus, we may write $\mathbf{c} \perp \mathbf{v}$ $\forall \mathrm{c} \in C, \quad \mathbf{v} \in C_\perp$ .The vectors in $C_\perp$ form a linear block code of dimension $n-k$, referred to as dual code of *C* [12].

In coding theory, a Tanner graph, proposed by Michael Tanner, is a bipartite graph used to state constraint equations which specify error correcting codes[13]. Tanner graphs are partitioned into subcode nodes and digit nodes. For linear block codes, the check nodes denote rows of the parity-check matrix *H*. The variable nodes represent the columns of the matrix *H*. An edge connects a check node to a variable node if a nonzero entry exists in the intersection of the corresponding row and column. In Fig 2.2, check and variable nodes of corresponding *H* matrix can be seen on the Tanner Graph.

$$\mathbf{H} = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & c_1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & c_2 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & c_3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & c_4 \end{array}$$

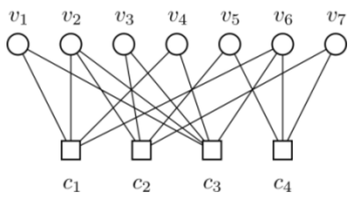

Fig. 2.2. Tanner Graph Representation

2.2.1 Reed-Muller Codes

When r and m are integers with $0 \leq r \leq m$, the Reed–Muller code with parameters r and m is denoted as RM(r, m). When it is asked to encode a message consisting of k bits, where

$$k = \sum_{i=0}^{r} \binom{m}{i}$$ holds, the RM(*r*, *m*) code produces a codeword consisting of 2^m bits.

There is a general recipe for Reed Muller generator matrices. Generator matrix of an (r, m) Reed - Muller Code can be constructed by defining a matrix G_2m where

$$[G_2]_m = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}_{m,}$$

where ‘$G^{*}{}_m$’ denotes an m-field Kronecker product of matrix G. Then, the generator matrix can be obtained by selecting $k = \sum_{i=0}^{r} \binom{m}{i}$ rows of the matrix whose weights are not smaller than 2^{m-r}.

2.2.2 LDPC Codes

LDPC codes, first described in 1960 by Gallager, are efficient channel coding codes that allow transmission errors to be corrected. These codes did not gain general acceptance due to high the computational effort required and they got forgotten. By the development of Turbo codes, LDPC codes have been rediscovered. Turbo codes were successful due to the use of iterative decoding. This type of decoding provides powerful protection against errors for the LDPC codes as well[14].

As the name suggests, matrix H of LDPC codes contains very sparse ones compared to size. Hence, the parity-check equations generally contain few elements and this provides an advantage for iterative decoding. LDPC codes are generally described by Tanner Graph, where check nodes must be zero in modulo 2.

LDPC codes continue to be characterized by a girth. This states how many edges are required as a minimum to leave a node in the bipartite graph and to return to it. Because of the structure of the graph, such a loop length is always an even number and the smallest possible loop length is equal to four. It seemed that codes with a girth of only four are generally bad. Such short loops have therefore been removed from developed graphs. However, in [15], it was demonstrated that there are also powerful codes with a girth of just four.

2.3 Maximum-Likelihood Decoding (ML)

Assuming that the channel has introduced $e \in \{0,1,\ldots,n\}$ erasures, we define

$$\mathbf{x}=[\mathbf{x}_k \mid \mathbf{x}_u],$$

where $\mathbf{x}_u$, is a vector of length e associated with the erased bits and $\mathbf{x}_k$ is a vector of length $(n-e)$ associated with the correctly received bits. We also denote by H a permuted version of a $(n-k) \times n$ parity check matrix of C. Correspondingly, H may be split up as $H= [H_k \mid H_u]$, where H_u is an $(n-k) \times e$ and H_k is an $(n-k) \times (n-e)$ matrix. Upon receiving a word $\mathbf{y}$ from the erasure channel, ML decoding consists of taking the decision x~ about the transmitted codeword according to the decision rule

$$\tilde{x} = \text{argmax} \Pr(y \mid x).$$

To this aim, the subset $C' = \{ x \in C \mid x_k = y_k \}$ is considered. If $C' = \{x'\}$, C' possesses one element only, then $\tilde{x} = x'$ and decoding is successful[16].

$$H_u \mathbf{x}_u^T = H_k \mathbf{x}_k^T \quad (4)$$

CHAPTER 3
PEELING DECODER

In this chapter, we will discuss the peeling decoder. PD, introduced for the BEC by Luby[17], is one of the simplest examples of iterative decoding. This decoder is based on the parity-check matrix of the code and can be applied to arbitrary linear code. Let H be the parity-check matrix of an (n, k) linear code.

3.1 Introduction

The parity-check matrix, H, may be represented by a bipartite graph with variable node set, $\mathbf{V}$, and check node set, $\mathbf{C}$. This bipartite graph is denoted $G(H) = (\mathbf{V} \cup \mathbf{C}, \mathbf{E})$, where the columns of H indicate the variable nodes in $\mathbf{V}$ and the rows of H indicate the check nodes in $\mathbf{C}$. For $i \in \mathbf{V}$ and $j \in \mathbf{C}$, $(i , j) \in \mathbf{E}$ if and only if $H_{ij} =1$. This bipartite graph is known as a Tanner graph with $r=n-k$ check nodes and n variable nodes.

Let $\mathbf{c} \in \mathbf{C} \subseteq \{0,1\}_n$ be a binary code word transmitted over the BEC and $\mathbf{v} \in \{0, 1, e\}_n$ be the received vector.

3.2 Algorithm

The Peeling Decoder algorithm proceeds as follows:

1. Initialize the variables $x1,...,xn$ to ? and the variables $y1 ,...,ym$ to zero.
2. For each non-erased code symbol, let j $\in$ $\mathbf{V}$ be its index and set variable x_j to the known value.
3. If there is a degree-1 check node, let j $\in$ $\mathbf{C}$ be its index, i $\in$ $\mathbf{V}$ be the index of the adjacent variable node, and set $x_j = H_{ij-1}y_i$.
4. If the graph contains a variable node whose value is known (i.e., x_j not =?) , let j $\in$ $\mathbf{V}$ be its index and
   (a) for all i such that (i , j) $\in$ $\mathbf{E}$, update $y_i = y_i - H_{ij}x_j$
   (b) remove bit j and all adjacent edges from the graph (i.e., V $\leftarrow$ V \ j , $\mathbf{E} \leftarrow \mathbf{E} \setminus \{(i, j') \in \mathbf{E} \mid j = j'\})$
   (c) Go to Step 3

5. When the algorithm reaches this point, either decoding is successful and x_j is not equal to ? for all j $\in$ $\mathbf{V}$ or the decoder is stuck in a configuration where there are no

degree-1 check nodes and the graph contains only variable nodes whose values are unknown.

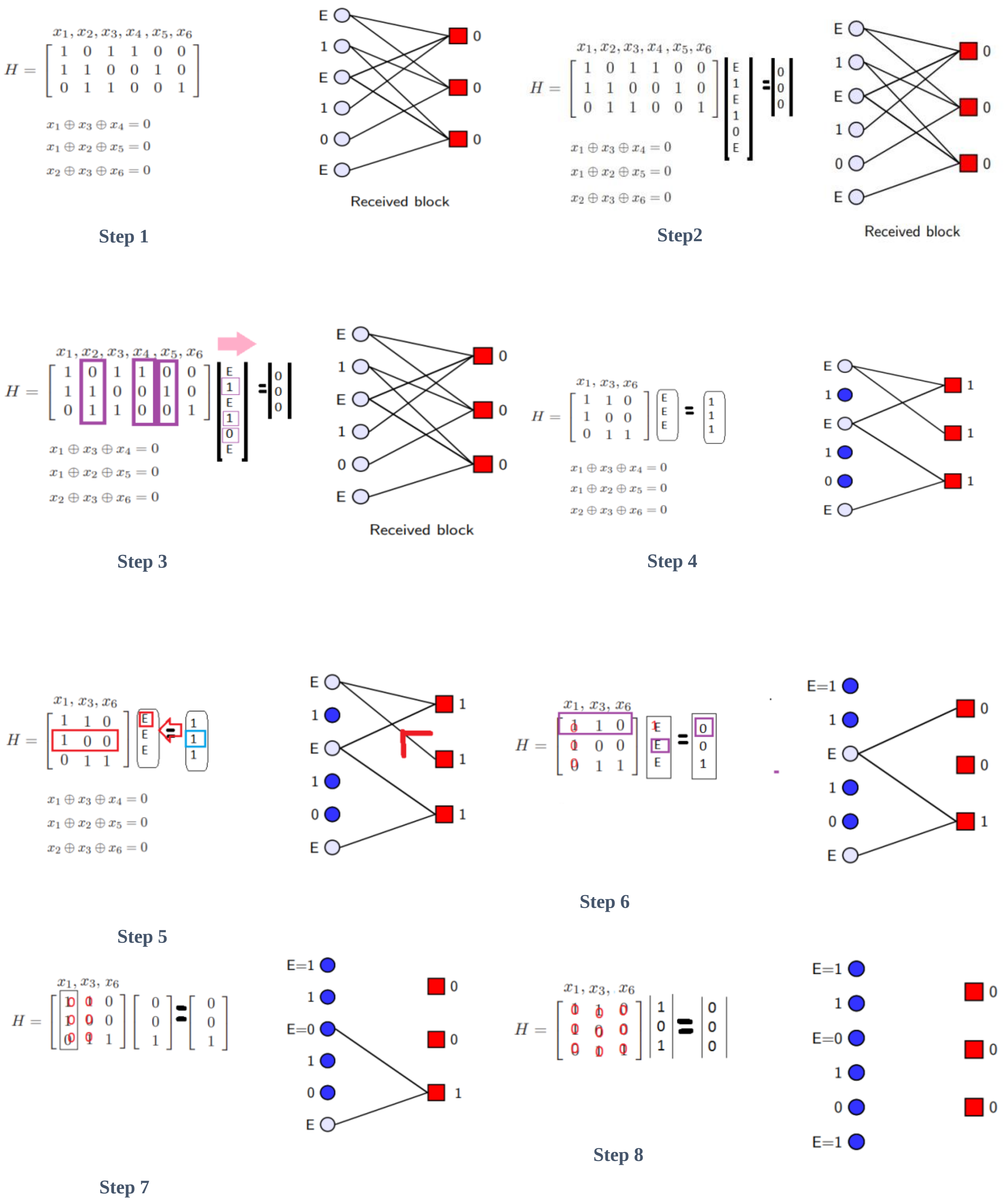

Fig. 3.1 : Peeling Decoder Example

For step (5) above, if decoder gets stuck, then the decoder has failed due to the presence of a stopping set. Stopping sets are collections of variable and check nodes in the Tanner graph of an LDPC code which greatly reduce its error correcting ability. These sets cause decoding to fail when certain variable nodes are affected by errors after transmission. Stopping sets were first described in 2002 by Di et al [8], who were researching the average erasure probabilities of bits and blocks over the BEC. In the Figure 4.1 above, this algorithm can be seen as depicted.

CHAPTER 4
GUESSING DECODER

In this chapter, we will discuss another decoding technique,namely guessing decoder. When the PD is applied to a linear code over the BEC, it results in a very fast decoding algorithm. However, the performance of this decoder is inferior to that of the ML decoder. Asymptotic analysis of the performance of codes has been worked on by various authors [8], [11], [17] and for some codes done successfully. Although using the asymptotic analysis, good degree distributions can be found, generating good finite-length codes has always been an issue too. Some authors instead of working on design on codes, proposed new decoding techniques such as guessing decoder [19],[20].

4.1 Algorithm

Guessing decoder algorithm proceeds as follows:

1. Register the initial values of check nodes to another memory unit.
2. Initial Step: The value of each received vector bit, v_i is assigned to each variable node $i \in \boldsymbol{V}$ of the Tanner Graph.
3. The check nodes $\boldsymbol{c}_i \in \mathbf{C}$ count the number of erased bits which are neighbours in the Tanner graph, g.
4. If check node c_i neighbours only one e symbol in $\mathbf{v}$, the even parity constraint uniquely determines the original value of e for that variable node.
5. Repeat steps (2) and (3) until either all erasures have been recovered or until every check node that is a neighbour of an erased bit is a neighbour of at least two erased bits. If the former is the case, decoding is succesful, otherwise continue to (6).
6. Guess one of the unknown variable node bits either ‘0’ or ‘1’. Update the values of check nodes which are connected to these variable values and remove the edges corresponding guessed variable. Check if every erasures have been either guessed or recovered.
7. If this is the case, and there is no contradiction in check equations, decoding is succesful. Otherwise, Go back to the step (6) and change the guess.

CHAPTER 5
INACTIVATION DECODER

5.1 Introduction

When the PD is applied to a linear code over the BEC, it can operate very fast. However, the performance of this decoder is inferior to that of the ML decoder. As we already mentioned in the beginning of the Chapter 4, we will propose another technique to improve decoding of the existing codes. This proposed decoding scheme is applicable to any kind of linear code. Although the method we propose can be applied to any code length, we will consider moderately short lengths.

ML decoding has the best possible BLER. Since we are concerned with improving the iterative decoding of linear codes, it is useful to study the ML decoder and its properties. The iterative decoding of codes over the BEC is much faster than the ML decoding. However, PD stucks in most cases. Our aim is to achieve ML decoder optimality while keeping the decoding fast.

As an example to this problem, let's investigate LDPC problem. Any LDPC code has a threshold ε_{th} such that if $\varepsilon > \varepsilon_{th}$, then the error probability of the standard iterative decoding is bounded away from zero by a strictly positive constant. On the other hand, if $\varepsilon < \varepsilon_{th}$ an arbitrarily small error probability is attainable if n , the length of the code, is large enough [11]. However, for finite-length codes, the situation is different. First, we may get an error floor and cannot decrease the error probability as much as we want. Moreover, to decrease the error probability, for example from 10^{-3} to 10^{-6}, we need to decrease ε by a considerable amount. This example illustrate the impact of proposed decoding algorithm that has the same complexity as the message-passing decoder.

5.2 Algorithm

For the erasure patterns that PD finishes the decoding successfully, both PD's and inactivation decoder's working principles are the same. The difference between the two algorithms is when PD fails to complete the decoding of a received codeword. In this case, inactivation decoder resumes decoding as follows:

A stopping set **S** is defined in [8] as a subset of **V** such that all neighbors of **S** are connected to **S** at least twice.

1. Initial Step: The value of each received vector bit, v_i is assigned to each variable node $i \in V$ of the Tanner Graph.

2. The check nodes $c_i \in \mathbf{C}$ count the number of erased bits which are neighbours in the Tanner graph, *g*.

3. If check node c_i neighbours only one *e* symbol in **v**, the even parity constraint uniquely determines the original value of *e* for that variable node.

4. Repeat steps (2) and (3) until either all erasures have been recovered or until every check node that is a neighbour of an erased bit is a neighbour of at least two erased bits. If the former is the case, decoding is succesful, otherwise continue to (5).

5. Label one of the unknown variable node bits as ‘inactivated’, and assign a variable to this inactivated bit. Update the values of check nodes which are connected to these variable values and remove the edges. Check if every erasures have been either inactivated or recovered. If this is the case, continue to (6). If latter is the case, Go back to the step (2).

6. By using the equation $cH_T=0$ (Formula 3), all the inactivated bits can be recovered from updated final check node values. Since the variable nodes are linear combination of these inactivated bits, variable nodes can be recovered.

This proposed algorithm depicted below is shown in the Figure 5.1. There are different ways of implementing this procedure in MATLAB. In our implementation, multiples of 10 are assigned to every inactivated variables. Initially, we assigned 10 to the first inactivated variable. If PD can continue, it resumes. Whenever PD is stuck and new inactivation is needed, the coefficient is multiplied with 10, kept in memory and assigned to a new inactivated variable. Even though this implementation works fast, we need to apply mode operation to check nodes at every step of decoding, to prevent the variable values to accumulate and be entangled to each other. This modulo operation is reducing check nodes to odd coefficient part of variables. For instance, for the case coefficient value is assigned as 10 and check node variable is ‘347’ reduces to ‘101’ by [347=(200+100)+(40)+(6+1)].

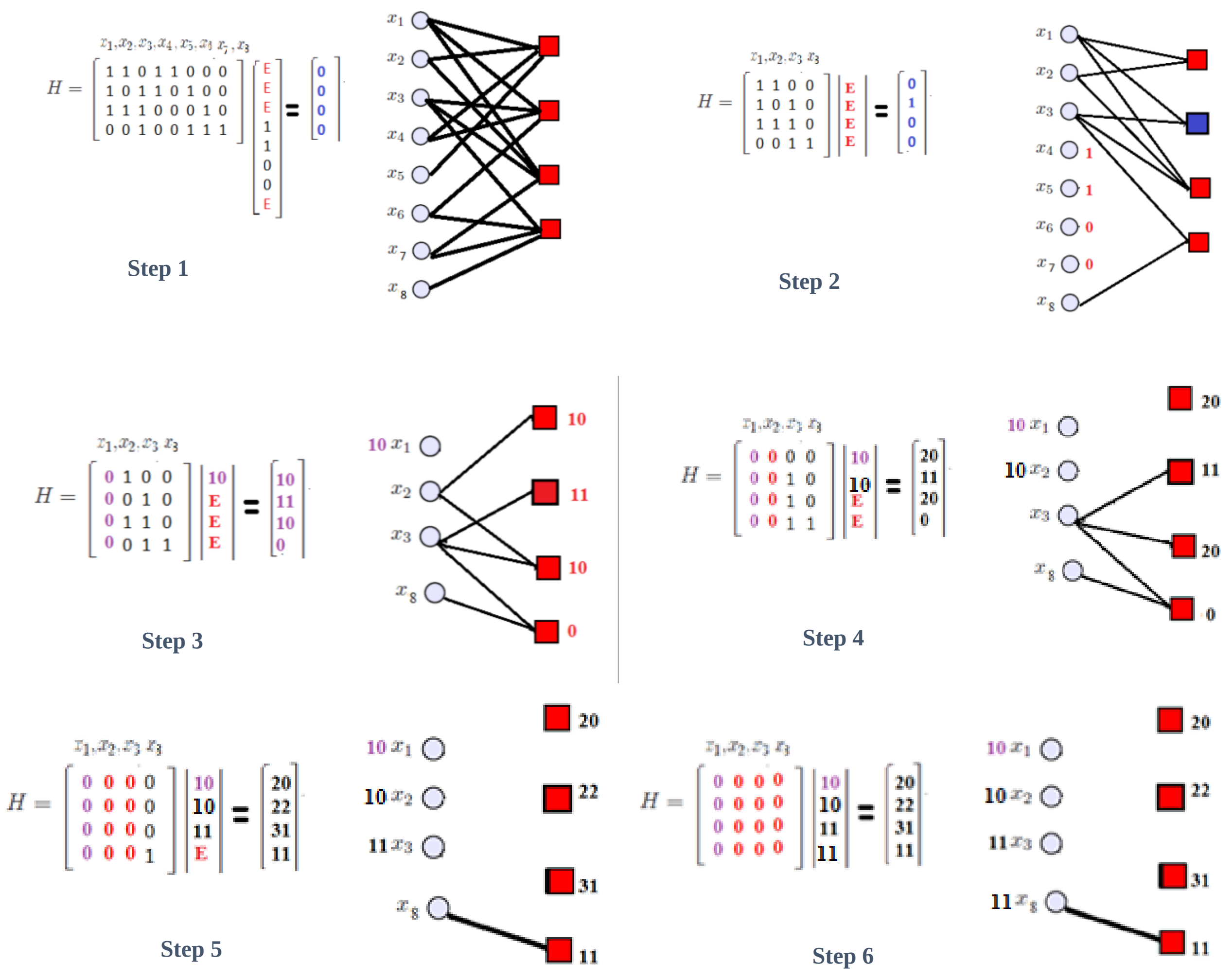

Fig. 5.1 Inactivation Decoder Steps

There is more elegant solution to this problem in which modulo operation does not take place. In this implementation technique, firstly, again PD is applied. If PD gets stuck during decoding, inactivated variable will pass a vector **K** as in the Figure 5.2 , instead of passsing value of coefficient. This **K** vector will be in the form of **K=B+P**. For the case v_i is inactivated, $\mathbf{B_i}$=*{bias 0 0 .... 0 0 }* and $\mathbf{P_i}$=*{ 0 0 ..1. 0 0 }* where 1 is located at the position of i+1, i ∈ *{1,2, .., number of variable nodes}*. If v_i is not the inactivated variable, $\mathbf{P_i}$ is all zero vector. Vector **B** will be updated similar to PD procedure. After all erased bits either inactivated or recovered, by using the Formula 3, we again equate check nodes to all-zero vector. From the information gathered by check nodes, the variable nodes can be resolved. This method does not require any modulo operations, so it works fast for short-size codes. On the other hand, vector implementation becomes costly for longer code implementations due to memory requirements.

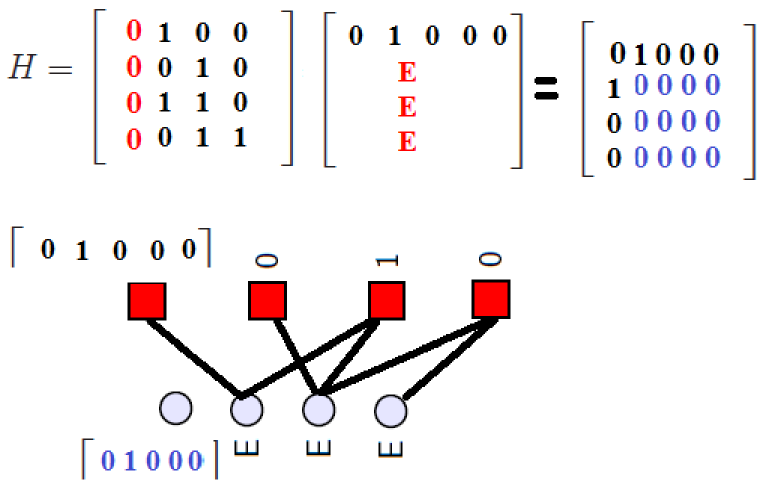

Fig. 5.2. Inactivation Decoding-Vector Implementation

This algorithm works for any code system whose rank is sufficient. If the code is rank-deficient, ML decoder cannot recover the erasure bits either.

The main motivation behind inactivation decoding is to provide a more powerful decoding algorithm. The key to the design of the linear system of equations is to ensure that the matrix is full rank with high probability to guarantee the successful decoding, while at the same time minimizing the total number of decoding symbol operations. For instance, a design is good if the average degree of a row symbol is constant and if the number of inactivated variables is proportional to the square root of the total number of variable nodes, as this means that the total number of symbol operations for inactivation decoding is linearly proportional to the number of variable nodes(i.e., number of columns). This is because of the fact that the number of symbol operations to solve for the inactivated variable nodes using the Gaussian elimination is bounded by the square of the number of inactivated nodes, and the number of symbol operations for the PD is linear in the number of non-zero entries in the original matrix, in other words number of edges[18].

Derivation of average number of inactivation is challenging. At step 5, the decision of which variable node will be inactivated, affects the complexity of decoding. It is worth noting that the set of basic equations depends on the choice of variable nodes we inactivated, so the number of inactivations. Intiutively, best choice for variable to inactivate, is the variable that is connected to the most check nodes. On the other hand, this does not always result in minimum number of inactivations. But still, by simulations,we showed that inactivating the variable that is connected to the most check nodes, shows better performance than the decoders where random variable or variable with minimum number of connections is inactivated.

CHAPTER 6
RESULTS

In this section, we provide some empirical results. We successfully implemented peeling, guessing, and inactivation decoders by using the algorithms described above. Throughout this study, we have worked and analyzed any quantity on the channel model BEC. Here in this part, we will illustrate performance results of different decoding algorithm (BLERs vs Erasure Probability) for RM and LDPC codes by simulations.

Fig. 6.1 shows the simulation results for BLERs over BEC on RM(3,7). This figure compares the performance of the PD with the proposed inactivation decoding algorithm. It is worth noting that inactivation decoding and ML decoding performance plots perfectly overlap, showing the same BLERs. This result proves the previously mentioned fact that inactivation decoding inhere the optimality of the GE. Also from the Monte-Carlo simulations, running time of ML decoder is more than the running time of proposed method,hence, showing efficiency of the proposed algorithm.

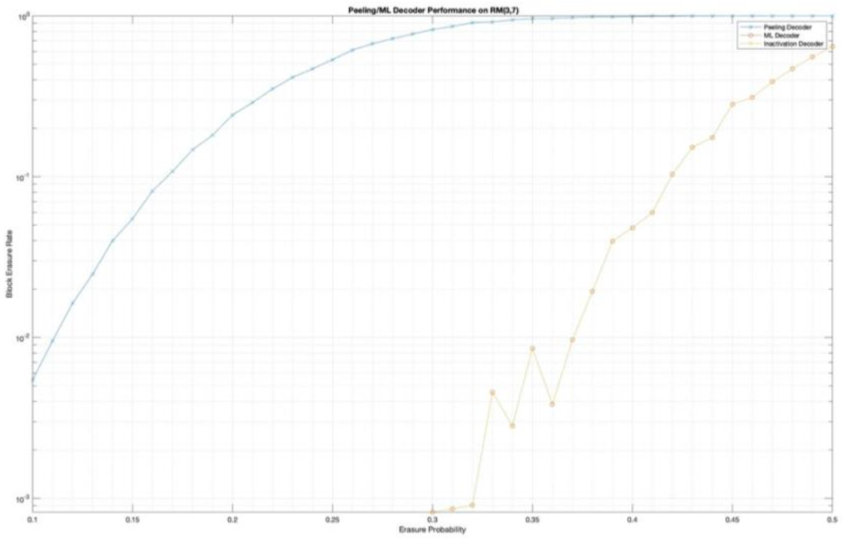

Fig. 6.1. Peeling/ML/Inactivation Performance on RM(3,7) over BEC

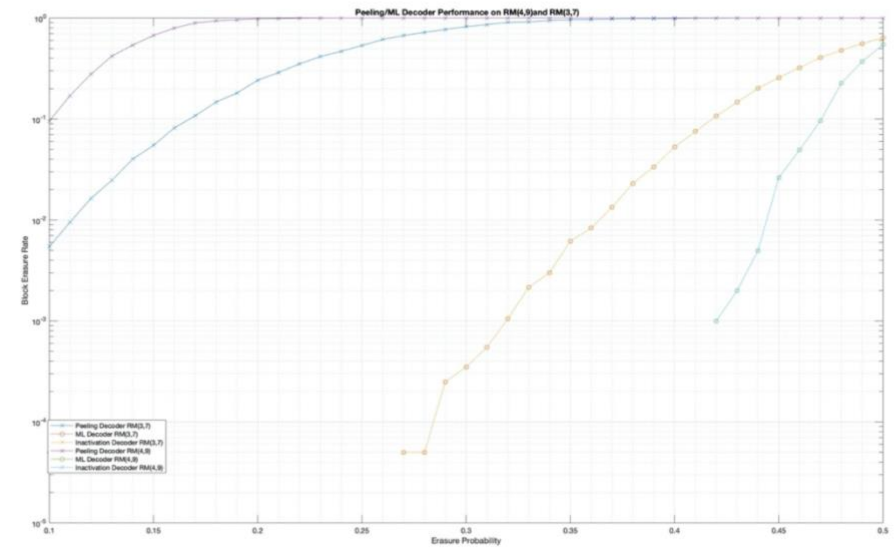

Fig. 6.2. Peeling/ML/Inactivation Performance on RM(3,7) and RM(4,9)over BEC

In Figure 6.2 above, BLERs vs erasure probability for different decodering procedures on RM(4,9) and RM(3,7), can be seen. As expected, again proposed inactivation decoders achieve the optimality of ML decoders,whereas peeling decoder performs suboptimally. PD applied to RM(4,9) gets stuck faster due to design of RM codes. H matrix of RM(4,9) code has more ones at every row, causing PD operation less possible. On the other hand, in terms ML performance, RM(4,9) codes provide better performance.

In this part, we experimentally verify the claim that a very few numbers of inactivation are enough to finish the decoding. We study the number of inactivations in proposed inactivation decoding when peeling decoding fails, where ML decoding is successful. We define $inac_{max}$ as upper bound of number of inactivations. To evaluate the number of required inactivations, we set $inac_{max}$ = infinite, hence, no constraint set on the maximum number of inactivations required.

We discussed previously in Chapter 5.2, the number of inactivation depends on the decision of variable node which will be inactivated. Intuitively, we propose starting inactivation procedure from the variable node which is connected to the highest number of edges.

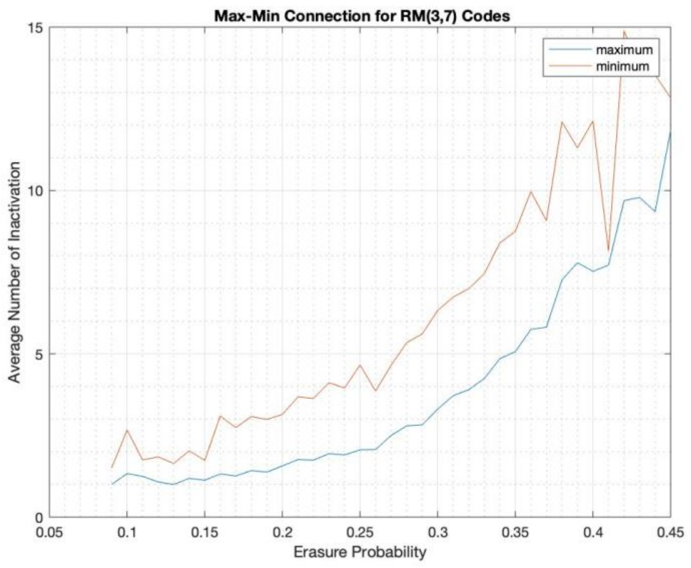

Fig. 6.3. Average Number of Inactivations for Max-Min for RM(3,7)

Effect of the selection of which variable to inactivate, on number of inactivations are shown numerically by Monte Carlo simulations in Fig. 6.3. At any erasure rate, average number of inactivations are fewer if we inactivate the node which has the most connection than the one with the least number of connections or randomly chosen variable to be inactivated. Since our intuition performs the best among the other variable selecting mechanisms, we continued with it. On the other hand, by randomly selecting variable nodes might also sometimes result in fewer inactivations than the maximum connected variable. Figure 6.3. shows that these results differ only 1 or 2 inactivations for RM(3,7) codes.

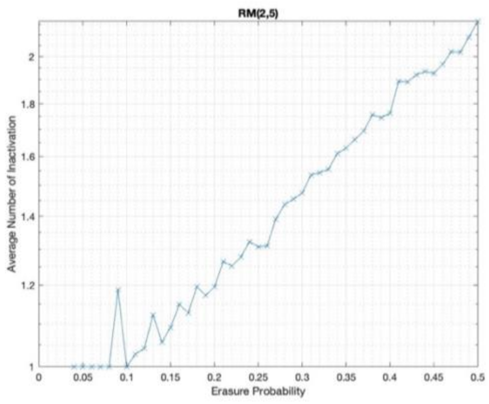

Fig. 6.4. Average Number of Inactivations vs Erasure Probability for RM(2,5)

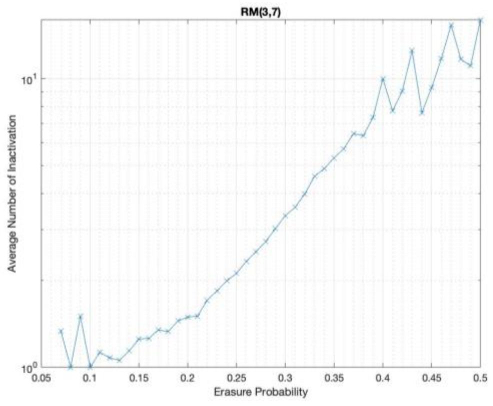

Fig. 6.5. Average Number of Inactivations vs Erasure Probability for RM(3,7)

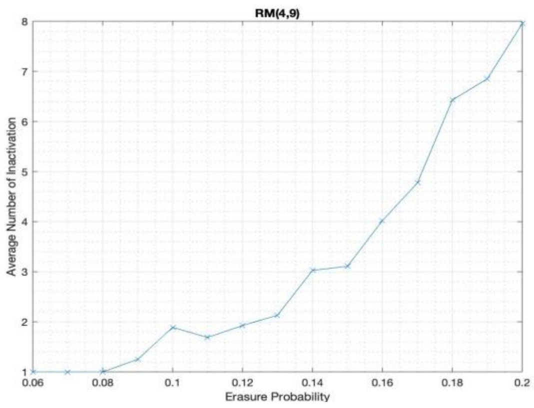

Fig. 6.6. Average Number of Inactivations vs Erasure Probability for RM(4,9)

Fig. 6.4, Fig. 6.5 and Fig. 6.6 show the Monte-Carlo simulation results for application of inactivation decoding algorithm to RM(2,5), RM(3,7), RM(4,9) codes over BEC. Figures show similar behaviors graphically , so the average number of inactivation to erasure probability as well. Here, from simulations, it can be seen that required inactivation number is proportional to the square root of code length. For instance, length of RM(2,5) code is 32 ,and let say at erasure probability 0.5, approximately 16 variable nodes will be erased and square root of this value is 4, which is more than number of inactivation at erasure rate 0.5 (2.2 inactivations). Length of RM(3,7) code is 128 and at erasure probability 0.4, approximately 51 variable nodes will be erased. Square root of this value is 7, which is close to the value of number of inactivation at erasure rate 0.4 (10 inactivations). Code size of RM(4,9) is 512 and at erasure probability 0.2, approximately 102 variable nodes will be erased. Square root of this value is 10, which is more than the value of number of inactivation at erasure rate 0.2 (8 inactivations).

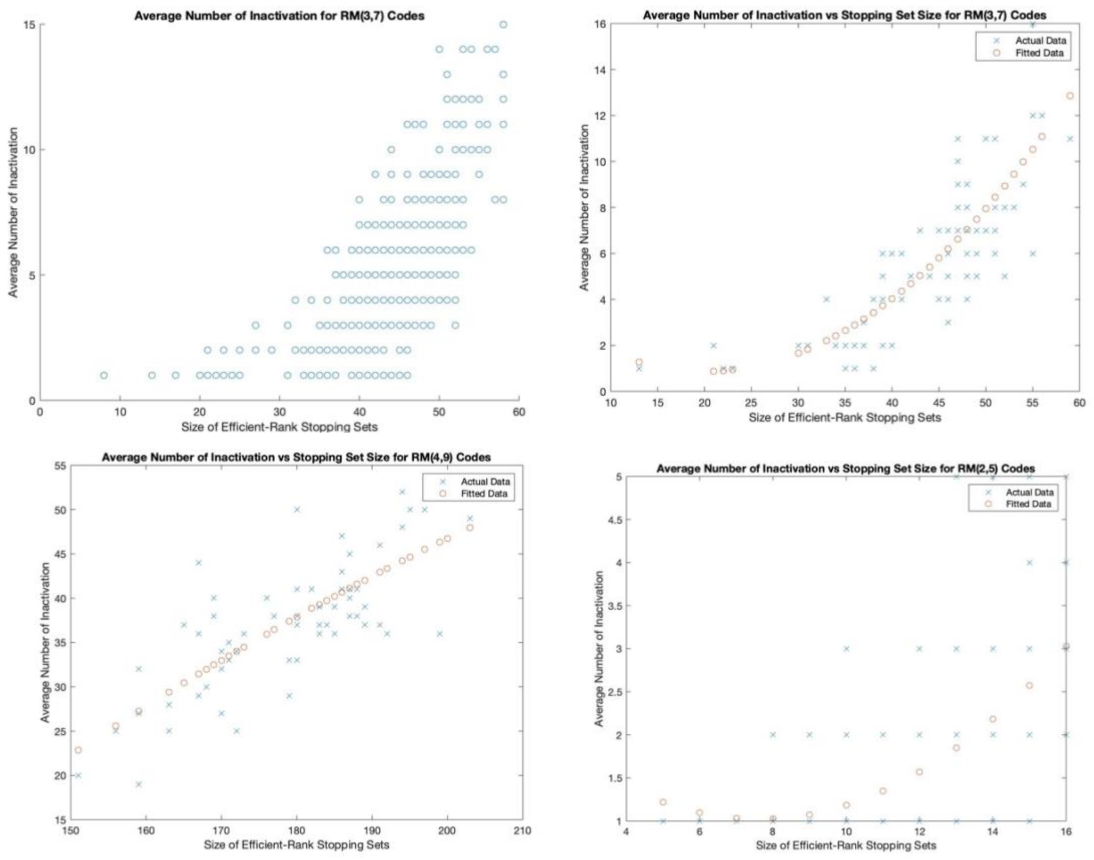

Fig. 6.7. Number of Inactivations vs Size of Full-Rank Stopping Sets at ε = 0.35

These results also show that ratio of RM(4,9)) code of length 512 to RM(2,5) code of length 32 is 16, whereas their number of inactivations at same erasure rates(0.2) is 6 (8/1.2). Then it can be commented as the size of code length is proportional to square of the number of inactivation.

Fig. 6.7 shows the Monte-Carlo simulation results of application of inactivation decoding algorithm to RM(2,5), RM(3,7), RM(4,9) codes over BEC. We have run our simulations for a rate 1/2 RM codes of length 32, 128 and 512 while we set no limit on inac$_{\max}$. To have more insight number of inactivations average number of guesses , we did our simulations at erasure rate ε = 0.35 and checked the number of inactivations and corresponding rank of stopping set. It can be seen from simulation, that for the full-rank stopping sets, average number of inactivations are linearly proportional or close to linear. We take into consideration only the cases whose rank is sufficient. After plotting the data, data points are fitted to the curve. For RM(3,7) codes, curve is quadratic polynomial, then average number of inactivation is proportional to square root of size of stopping set. On the other hand, for RM(2,5) codes, curve is first degree, hence, average number of inactivation is linearly proportional to size of stopping set. Then we can comment this degree change as code becomes less efficient for longer size codes (efficiency can be considered inversely proportional to ratio of inactivation number to code size).

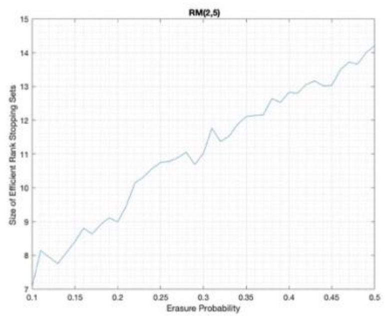

Fig. 6.8. Size of Full-Rank Stopping Sets vs Erasure Probability for RM(2,5)

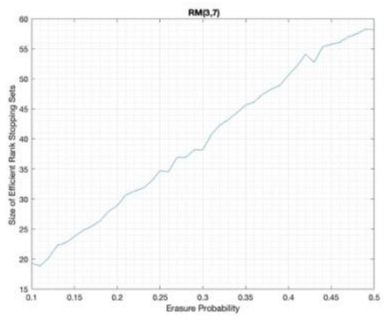

Fig. 6.9. Size of Full-Rank Stopping Sets vs Erasure Probability for RM(3,7)

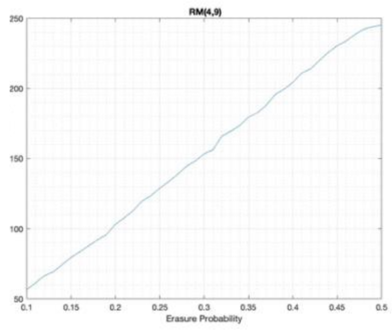

Fig. 6.10. Size of Full-Rank Stopping Sets vs Erasure Probability for RM(4,9)

From Fig. 6.8, Fig. 6.9 and Fig 6.10, it can be seen that average size of sufficient-rank stopping sets is proportional to code length. Both code length and size of sufficient-rank stopping sets quadruples from left graph to the right one.

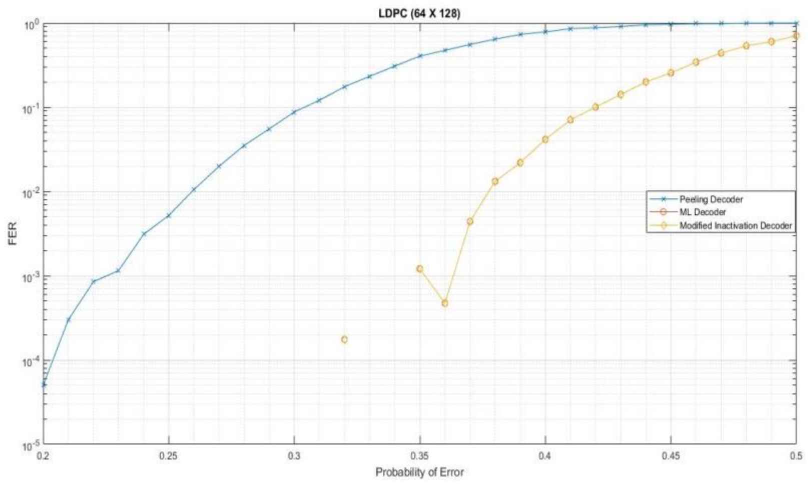

Fig. 6.11. Peeling/ML/Inactivation Performance on LPDC(64,128) over BEC

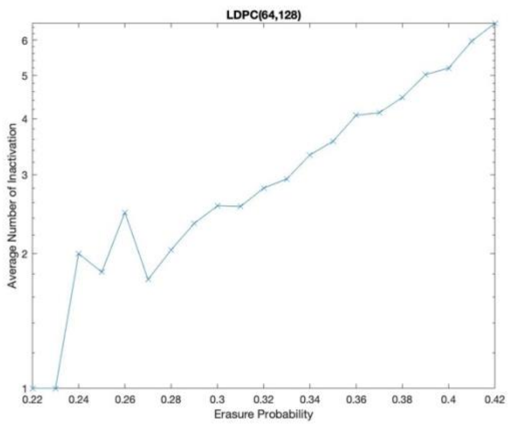

Fig. 6.12. Average Number of Inactivations vs Erasure Probability for LDPC(64,128)

For H of LDPC codes, we used permutation of identity matrix, and according to size we decided to size of identity matrix used. For example for size (64 x 128) H matrix, we used 32 size (16x16) random identity permutation matrices. Then, it can be seen that H is scarce having eight 1's in every row and four 1's in every column.

Fig. 6.11 shows the performance of peeling, ML and proposed inactivation decoding on LDPC(64,128) over BEC. Since it is the same code-length with RM(3,7), comparison with it would be illimunating and informative. Inactivation performance BLER of RM(3,7) falls to 10-3 at erasure rates 0.3, whereas BLER for LDPC code, falls to 10-3 at erasure rate 0.35. From the simulation, it can be said that BLER performance of inactivation decoder on LDPC codes is better. In terms of PD performances, BLER of LDPC code falls to 10-4 at erasure rate 0.2 , while BLER of RM(3,7) falls only to 1-10-1 at same erasure rate. Due to the fact that RM(3,7) is denser than LDPC(64,128), better perfomance of LDPC at peeling decoding is expected. Also LDPC code at this size requires fewer inactivation. But we know that RM codes achieve capacity at finite lenghts. Even though for short-size codes LDPC has better performance, RM will perform better when code-length increases.

Fig. 6.12 shows the average number of guessing of guessing decoder for low erasure probabilities. For higher rates, this method becomes inefficient in my implementatiton because it does not keep the guesses in memory to impove efficiency. Without any need to solve system of equations, for small erasure probabilities, it works faster and in this respect, it works better than PD. Fig. 6.5 shows the average number of inactivation for the same code. This way of implementation requires more guessing, but only for small erasure probabilities, even though makes more guesses , works way too much faster.

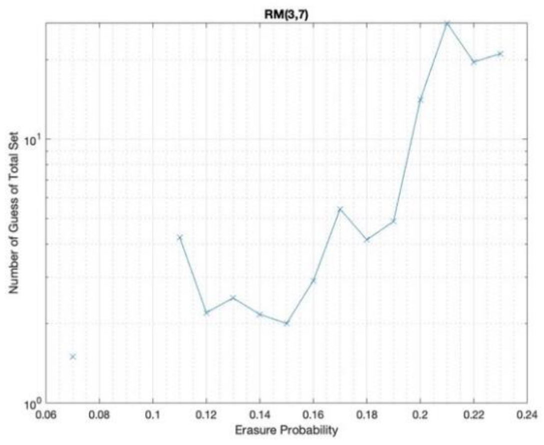

Fig. 6.13 Average Number of Guesses in Guessing Decoder

CHAPTER 7
CONCLUSION

7.1 Discussion

Throughout this study, the literature surrounding RM and LDPC codes over BEC, communication channels, and decoding techniques is covered. The nature of the decoding algorithms with the help of Tanner graph and stopping sets of the H matrix is discussed. An inactivation decoder is proposed and implemented for transmission over the BEC for some commonly used linear codes e.g. RM and LDPC. Peeling, Simple-Guessing and ML Decoder are implemented and their implementation algorithms are explained. With Monte-Carlo simulations, BLER performances over BEC are shown for different codes and sizes. It is shown in simulations that the proposed algorithm has same BLER as the ML decoder. The expected number of inactivations are illustrated numerically for various codes and various sizes. The relation between size of the codes and number of inactivation numbers are investigated. It is shown that for different selection procedure of variable node to be inactivated, number of required inactivation changes. We successfully implemented peeling, guessing, and inactivation decoders by using the algorithms described above.

7.2 Further Work

As a future work, comparison of number of inactivations for the different parity-check matrix of the same code can be worked on that might be used as a first-order proxy of decoder complexity. Analysis of stopping sets can be extended by using density of codes. By doing this, simulation results can be verified with the mathematical derivations. This result can be extended asymptotically as well for finite length or very long codes. Also to decrease number of symbol operation, minimizing number of inactivations can be futher worked. In order to find minimum number of inactivation, another variable selecting technique can be developed.

7.3 Realistic Constraints

a. Social, Environmental and Economic Impact

Digital communication is a daily life discipline, any improvement, achievement, or enlightenment will open new cutting edges. Eventually, better performance means less bandwidth and less power consumption. These are the beneficial impact of our project on an economical basis.

Not only for this project but in general for any project on Information and Communication Technology has an important role in the world, since we are now in the era of the information age. By utilizing technologies, the companies can make the business easier to happen with the client, supplier, and distributor. It is also very essential to our daily lives. The lack of access to appropriate information at the right time will result in low productivity, low-quality research works, and a waste of time to pursue information and even to do research which actually others had done or in other countries. Nowadays communication technologies cannot be considered separate from our daily lives anymore.

Any improvement in this era would contribute tremendously. For instance, specifically, Reed–Muller codes are error-correcting codes that are used in wireless communications applications, particularly in deep-space communication. Moreover, the proposed 5G standard relies on the closely related polar codes for error correction in the control channel.

Communication is a key component of relationships. In recent years, the merging of various kinds of technology has been increasing the number of options that people and institutions have for making contacts and keeping in touch.

b. Cost Analysis

- License cost of MATLAB = 940 USD

- Weekly cost of 1 associate engineer = 400 TRY
- Overhead costs = 400 USD

c. Standards

- Wi-Fi (IEEE 802.11n),
- WiMax (IEEE 802.16e),
- DVB-S2 standards.

BIBLIOGRAPHY

[1] Robert Gallager, course materials for 6.450 Principles of Digital Communications I, Fall 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on [01 07 2020]

[2] C. E. Shannon, "A mathematical theory of communication," in The Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, July 1948.

[3] Ryan, W. and Lin, S. (2009). Channel codes. Cambridge: Cambridge University Press, pp.1-5.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1,” Proceedings of ICC 93 - IEEE International Conference on Communications.

[5] David J.C. MacKay and Radford M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," Electronics Letters, July 1996

[6] D.E.Muller,“Application of boolean algebra to switching circuit design and to error detection,” Transactions of the IRE Professional Group on Electronic Computers, vol. EC-3, no. 3, pp. 6–12,Sep. 1954.

[7] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Sasoglu, and R. L. Urbanke, “Reed-Muller codes achieve capacity on erasure channels,” IEEE Trans. Inf. Theory, vol. 63, no. 7, pp. 42984316, Jul. 2017.

[8] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, “Finite-length analysis of low-density parity-check codes on the binary erasure channel,” IEEE Trans. Inf. Theory, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[9] E. R. Berlekamp, *Algerbraic Coding Theory*. McGraw-Hill, 1968

[10] B. A. LaMacchia and A. M. Odlyzko, “Solving large sparse linear systems over finite fields,”

[11] T. J. Richardson and R. L. Urbanke, “Efficient encoding of low-density parity-check codes,” IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 638– 656, Feb 2001

[12] Liva, G.,Yacoub E. B. (2019). Lecture Notes for Channel codes for iterative decoding.

[13] R. Michael Tanner Professor of Computer Science, School of Engineering University of California, Santa Cruz Testimony before Representatives of the United States Copyright Office February 10, 1999

[14] MacKay, D.J.C.; Neal, R.M.: Near Shannon limit performance of low density parity check codes. Electron. Letters, Vol.32, August 1996, 1645-1646 (reprinted with printing errors corrected in Vol.33, 457–458)

[15] Chen, L.; Xu, J.; Djurdjevic, I.; Lin, S.: Near-Shannon-Limit Quasi-Cyclic Low- Density Parity-Check Codes. IEEE Trans. on Communications, Vol.52, No.7, 2004, 1038–1042

[16] Liva, G.,Yacoub E. B. (2019). Lecture Notes for Channel codes for iterative decoding.

[17] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, “Efficient erasure correcting codes,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, Feb. 2001.

[18] Shokrollahi, Amin &Luby, Michael. (2009). Raptor Codes..Foundations and Trends in Communications and Information Theory. 6. 213-322.

[19] Pishro-Nik H., Fekri Faramarz , “On Decoding of Low-Density Parity-Check Codes Over the Binary Erasure Channel “, IEEE Trans. Inf. Theory, vol. 50, no. 3,March 2004

[20] E. Berlekamp, “Bounded distance+1 soft-decision Reed-Solomon decoding,” *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 704–720, 1996.