Detection and Removal of Cycles in LDPC Codes

By

Nafila Farheen

A Thesis

In

The Department

Of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirement for the Degree of

Master of Applied Science at

Concordia University

Montreal, Quebec, Canada.

April 2018

© Nafila Farheen, 2018

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Nafila Farheen

Entitled:

Detection and Removal of Cycles in LDPC Codes.

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

Complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________	Chair
Dr. R. Raut	
______________________________	Examiner, External To the Program
Dr. A. Youssef (CIISE)	
______________________________	Examiner
Dr. Y. Shayan	
______________________________	Supervisor
Dr. M. R. Soleymani	

Approved by: ______________________________

Dr. W. E. Lynch, Chair
Department of Electrical and Computer Engineering

__________20____   ______________________________

Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Acknowledgment

Firstly, I am wholeheartedly thankful to Almighty for his mercy and blessing.

It is with utmost sincerity that I would like to thank my supervisor, Prof. M.R. Soleymani, for his constant guidance and ardent inspiration during the whole course of study. I am thankful to him for letting me do research independently and being there when I needed his insight and expertise.

I would like to take this opportunity to thank all the members and colleagues of Wireless and Satellite Communication laboratory for assistance and guidance during difficult times.

I am grateful to my parents and my brother for limitless love and support. Thank you for supporting me throughout the journey of life. I am gratified to be such a fortunate daughter and sister.

Finally, I would like to express my gratitude to my husband for all the love and care, without you this journey would not have been possible.

I am genuinely obliged to all of you. Thank you all very much.

Abstract

Information technology, at present has thrived to great aspects and every day more beneficiary of this blessing is being connected to the modern invention and technology. With the swift growth of communication networks, there has been a high demand for efficient and reliable digital transmission and data storage system. LDPC is one of the channel codes for error correcting that have been developed for competent systems that require higher reliability. For low-end devices requiring a limited battery or computational power, low complexity decoders are useful. For LDPC codes, the presence of short cycle in the parity-check matrix lower the decoding threshold making it less efficient. In this research, a method has been developed from an existing algorithm that finds out the exact position of potential bits forming cycle-4 in the parity check matrix that might create decoding failure after transmission in binary erasure channel. Once the short cycles are detected, it can be removed by puncturing method to obtain capacity achieving codes. The code obtained by the method has a threshold 0.42 and rate 0.5, which is asymptotically close to the mother code. Simulations show that for less number of iterations and in the presence of same channel erasure the decoder block error probability close to 10^{-6} is achievable. As no other additional decoding algorithm is employed, the proposed scheme does not add additional computational complexity to the decoder. Furthermore, as an extension to the method a scheme has been proposed to generate rate-compatible LDPC codes using 0.5 rate regular code with puncturing method varying puncturing fractions. By the proposed method of generating different rate code, same amount of information can be sent with less parity.

Table of Contents

List of Figures

List of tables

List of Abbreviations and Symbols

ECC:	Error Correcting Codes
LDPC:	Low-Density Parity Check
GMS:	Global System for Mobile
CRC:	Cyclic Redundancy Check
CDMA:	Code Division Multiple Access
3G:	Third Generation
DVBS:	Digital Video Broadcasting System
CMMB:	China Multimedia Mobile Broadcast
DOCSIS:	Data Over Cable Service Interface Specification
WiMAX:	Worldwide Interoperability for Microwave Access
WLAN:	Wide Local Area Network
AWGN:	Additive White Gaussian Noise
RM Codes:	Reed-Mullar Codes
RS Codes:	Reed-Solomon Codes
SECDED:	Single Error Correcting-Double Error Detection
BER:	Bit Error Rate
SC-LDPC:	Spatially Coupled Low-Density Parity Check Code
LDPC-BC:	Low-Density Parity Check Block Codes
BEC:	Binary Erasure Channel
Gbps:	Gigabit per second
SPC:	Single Parity Check
H:	Parity Check Matrix
r :	no. of parity check equation
c:	codeword
v:	information bit
R:	density of the code
n:	total entries in the parity check matrix
J:	number of rows in the parity check matrix
λ:	column weight of parity check matrix
ρ:	row weight of parity check matrix
BSC:	Binary Symmetric Channel
μ:	Crossover probability

s: Syndrome

y: channel output

BEC: Binary Erasure Channel

ϵ : Channel Erasure Probability

PG: Projection Geometry

EG: Euclidean Geometry

m_{cv} : Message from check node to variable node

m_{vc} : Message from variable node to check node

e: Erased bit

WBF: Weighted Bit Flipping Decoding

MLD: Maximum Logic Decoder/ Majority Logic Decoding

SPA: Sum-Product Algorithm

MPA: Message Passing Algorithm

IDBP: Iterative Decoding Based on Belief Propagation

BP: Belief Propagation

APP: A posteriori Probability Decoding

β : Puncturing Fraction

1 Introduction

1.1 Structure of Modern Day Digital Communication System

Information technology, at present has thrived to great aspects and every day more beneficiary of this blessing is being connected to the modern invention and technology. The evolution of information technology started through the mass production and widespread use of digital computers, record keeping media, digital logic circuits, cellular phones but most importantly-the Internet. This modern-day communication system is the grace of digital revolution with its roots lying in the eminent Shannon's theory of channel capacity.

In 1948, C. E. Shannon stated in his paper that for any discrete memoryless channel, any data rate (R) that is below the capacity (C) of the channel, is achievable [1]. He proposed the use of ensemble of codes, which are vectors that can be transmitted to handle noise at any rate below channel capacity.

The simplest form of digital communication chain progression can be explained by Figure 1.1 that consists of a source, a channel and a sink or receiver. As the signal is transmitted over a channel, it is affected with undesired, various level of noise that creates difficulty in detection of the transmitted signal. The source behind such noise in the transmission channel could be of thermal origin, interference, crosstalk, intermodulation, industrial, atmospheric, extraterritorial noise and many more. When a transmitted signal goes through such noise, the data is either changed in such way that it is impossible to recover the information, or is the data lost forever. Utilizing the concept of channel coding in communication application, it is possible to design capacity achieving systems with lowest possible error rate. Channel coding provides controlled redundancy within the system to detect and possibly correct errors that signal encountered during transmission.

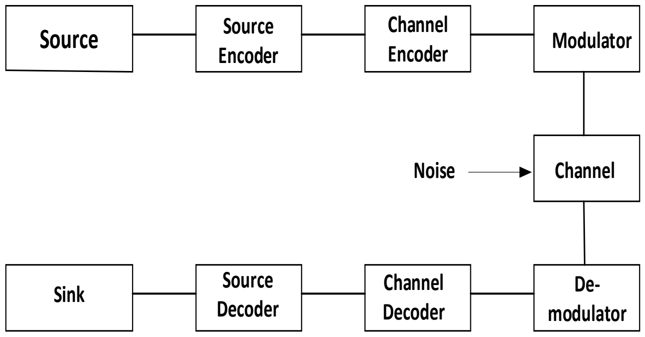


Figure 1.1: Block diagram of progression chain of modern day digital communication (Modified from [2]).

The information source, as shown in Figure 1.1 can be a human, machine or a digital computer which is encoded by the source encoder into a form of binary stream. The channel encoder transforms this binary data stream into discrete sequence by adding some redundancy to it and making it eligible to handle errors after transmission. Since discrete symbols are not suitable for transmitting over a physical channel, the modulator converts it to a waveform of certain duration, each representing a bit . Later on, the waveform is transmitted through the channel where it is introduced to and corrupted by noise. The demodulator reads the data from the channel as a waveform and then channel decoder decodes and corrects it. The decoding strategy is selected based on the channel encoding scheme and the type of channel model that has been used for communication. The source decoder transforms the corrected binary stream into estimation of the source and feeds it to the destination (sink).

In practical scenario, the systems are more complicated with lots of equipment and processing in between. The major concern that needs to be addressed for such systems is to design and adopt codes which are robust in handling and correcting errors feasibly below the capacity level of the transmission path in highly noisy channel. Such codes are known as error correcting codes (ECC) and mainly applied in the following field below [3]-

- Digital and mobile communication: In wireless communication ECC are used to cope with the interference caused by wide range of reasons including multipath fading, shadowing and other technical perturbations. GSM systems use cyclic redundancy check (CRC) codes for error detection. In CDMA and 3G systems, both convolutional and bock codes are used for error correction. Low density parity check (LDPC) codes are most widely adopted block codes that is used over different protocols such as – DVB-S2, G.hn home networking protocol over up to data rate 1Gbps, CMMB, DOCSIS, WiMAX (802.16e) , WLAN (802.11).
- Deep space communications: In space, the interference is modeled as additive white gaussian noise (AWGN) where, although the noise source is limited but the communication signal becomes significantly distorted because of propagation loss due to enormous distance. The first kind of ECC that has been used for mariner spacecraft is (32,6,16) Reed-Mullar (RM) codes in 1972. In recent times, Reed-Solomon (RS) codes, convolutional codes are used with improvement in gain, rate and performance.

- Satellite communication: RS codes and Turbo codes have been used for digital satellite TVs for a long time. Recently LDPC codes have been standardized as well for this purpose.
- Magnetic record keeping and data storage: RS codes are specifically used for data storage and magnetic record keeping. Single error correction double error detection (SECDED) which can correct one error and detect up to two errors are applied through RS codes for recording in CDs, DVDs and other media.

1.2 Motivation

As mentioned in previous section, the class of linear block codes such as LDPC provides a controlled redundancy within the system to control error. LDPC codes are the type of codes which are defined by the null space of the parity check matrix *(H)* [2] that has some specific structural properties. Due to the randomness in degree distribution, short cycles in such *H*-matrix is more likely to occur. The iterative decoder stops prematurely over BEC in the presence of these short cycles. This thesis focuses on the issue of such premature stopping of iterative decoding of LDPC code. An efficient way to detect and remove the unreliable transmission path in the parity check matrix created due to these short cycles is developed. This method proposes a trade-off between the additional decoder complexity and decoder error probability in order to achieve higher capacity in very noisy channels.

As stated in Shannon's paper, information rate, bit per channel use must be less than the channel capacity for the signal to be transmitted reliably [4]. The performance of error control codes was thought to be bounded by the Shannon's capacity theorem [2]. Practical error correcting codes could not reach the capacity up until the invention of turbo codes in 1993 [5]. After the invention of Turbo codes, LDPC code was reinvented again by Neal and McKay [6], and came into spotlight for its capacity achieving property.

LDPC code was originally invented by Robert Gallager in his PhD dissertation [7] in 1960 and was forgotten for decades because of its implementation complexity. This is a class of linear block codes with error correcting capability that reaches Shannon limit [4]. Robert G. Gallager proposed the layout of the code [7], a probabilistic decoding approach and a theory that goes beyond the performance of turbo code. Unfortunately, LDPC codes were ignored for decades due to the lack of computing power to fully implement them and afterwards it experienced an amazing comeback.

R. Gallager proposed the parity check matrix to have a fixed number of non-zero elements in rows and columns which is known as regular matrix. MacKay et al [6], M.G. et al [4], Richardson et al [8]are those eminent researchers who afterwards gave up this property of regularity by proposing irregular parity check matrix which outperforms regular codes in some cases. A. Shokrollahi mentioned in his paper [6] that the reason of inaccuracy of density evolution and its relation to the study of girth of a parity-check matrix which is an interesting topic to focus on for LDPC researchers. During the long course of research of this code, various researchers have mentioned methods to calculate the length of the girth (*g*) and higher cycles within the parity check matrix in [9] ,[10], [11],[12] .Among them Y.Mao et al in [13] have mentioned about calculation of *g* in a graph by initiating message passing but it does not calculate the higher cycles more than length *g*. In [14], authors have proposed a message passing algorithm for bipartite graph only that counts the number of cycle-4 but the computational complexity is higher since it requires precise matrix multiplication.

Among eminent researchers M. Karimi et al [9] have introduced a highly efficient message passing algorithm that can find out total number of cycles of length *2g,* where *g* is the length of the girth of the parity check matrix. The algorithm is applicable to bipartite and non-bipartite graph as well. This algorithm for counting number of cycles outperforms the tree search algorithm in terms of complexity. On the other hand, authors in [15] have mentioned designing of LDPC codes with large cycles for improving performance. Another paper by the same author of [9], have mentioned in [16] that BP for LDPC code can be improved by 'normalized BP' and 'offset BP' and the examples have been drawn specifically for soft decision channel.

During last two decades performance analysis and comparison have also been done for LDPC code under different scenario in binary input-output symmetric memoryless channel (BIOSM). Researchers in [17] have addressed the scenario on premature stopping of iterative decoding because of cycle. With an introduction of three different guessing algorithm an analysis has been made which shows that for a code length of n=10^3 and n=10^5, Algorithm A shows a BER= 0.00039 with $g_{avg.}$ =2.24. Other two algorithms, Algorithm B and Algorithm C shows a BER=10^{-2}, performing at the threshold of erasures. A rate compatible method to break cycles have been depicted in [18], [19] by insertion of dummy bits within the information bits while transmission. In this method a PEG-LDPC code with n=2304 and dummy bits ranging from 66.7-88.9% have been taken to observe the performance over AWGN channel. This approach showed a performance gap of 1.5 dB with BER less than 10^{-4} compared to the reference PEG with other previously used dummy bit insertion methods.

Another effort to design a capacity approaching codes called ‘Puncturing Methods’ have been introduced in [19], [21] for performance analysis in BEC, BISC and AWGN channel. Among these two papers, [21] has drawn more elaborate simulation results on BEC and BI-AWGN channel. Authors have considered codes with n=6 $\times 10^3$ with puncturing fraction α= 0, 0.26 and 0.4. These codes are drawn from asymptotically good SC-LDPC and LDPC-BC code and showed a BER not more than 10^{-7}. Considering different code rates and puncturing fraction, an in-depth analysis has been done in this paper for the above-mentioned approach. The paper also includes the depiction of the gap size to Shannon limit as a function of puncturing fraction (α) and cut-off rate (R_{max}). Experiment and derivation have been drawn from a random perspective with an aim to optimize different parameters to obtain capacity achieving codes. On the contrary, [20] have mentioned specifically that with a careful selection of punctured bit, cycles can be removed to gain highest capacity with iterative decoding.

1.3 Problem Statement

In LDPC codes, performance degradation occurs while there is a presence of short cycles or stopping sets in the H-matrix, which eventually causes early stopping of the iterative decoder. R. Gallager stated in his paper [7] that for the decoder to recover the erased bits or bits in error, the structure must be a tree. In other words, the information bits must be linearly independent to each other. The presence of stopping sets or cycles creates a correlation in the marginal probabilities passed by the sum-product decoder [22] making the information symbols dependent to each other. A cycle in a Tanner graph as shown in Figure 1.2 is a sequence of connected vertices which start and end in the same vertex in the graph, with vertices not more than once. The length of the cycle is the number of edges it contains, and the girth of a code is the smallest length of such path [22].

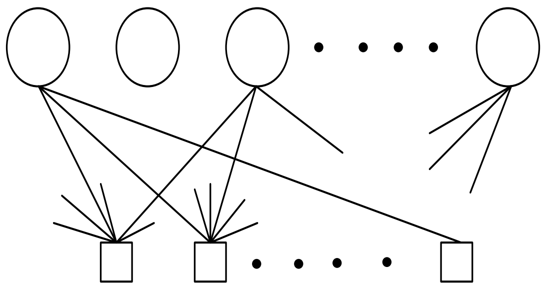


Figure 1.2 A Tanner graph containing a cycle of length-4

The smaller the length of girth is, the fewer is the number of iterations in the decoder that is correlation free. In case of bipartite graph, the length of smallest cycle is 4 that leaves detrimental effect in the decoding performance in error floor including premature stopping and deviation from theory of decoding threshold in BEC. Definite improvement can be obtained by avoiding cycle -4 and cycle-6 for finite length LDPC code.

The first aim of this research was to develop an algorithm to find out the exact potential information bits that might be associated with cycle -4 in the parity check matrix after transmission over BEC. We have modified the message passing algorithm [9] and applied it once offline in the parity check matrix before transmission to find out the potential bits for cycle formation. The algorithm takes two iterations to calculate associability of cycle-4 in the parity check matrix. Next, the performance analysis has been done after removing this unreliable transmission path over BEC of a finite length LDPC code.

1.4 Outline of the Thesis

In Chapter 2, an overview on the basic functionalities of error correcting codes are discussed. A brief review of the background of LDPC code have been given on the perspective of notation, expression and formation of the parity check matrix based on Robert G. Gallager's code. Furthermore, encoding [23]and hard decision decoding [2]schemes for LDPC code for BEC and Binary Symmetric Channel (BSC) are illustrated. The final part of this chapter addresses the problems that are associated with iterative decoding on BEC.

In Chapter 3, density-evolution of LDPC codes have been discussed as an important parameter to analyze code performance in the presence of erasures. The latter part of the chapter includes the calculation of threshold with an objective to find out the highest error correcting capability of such codes. Finally, the chapter concludes with simulation results showing the deviation of threshold from theory due to presence of short cycles in Tanner graph.

In Chapter 4, the methodology to overcome the problem stated in Chapter 3 have been discussed initially. A modification from [9] to find out the short cycle containing node in the *H*-matrix have been used. In this thesis, the main algorithm is adapted to locate the exact position of information bits in a parity check matrix that might be associated with a cycle-4 after transmission over BEC for regular code with ensemble (3,6). To do so, rather calculating total number of cycle passing through a node as stated in the paper, only the associability with cycle-4 have been calculated by sending monomials through each node for two iterations. In this way, further calculation of summing total number of cycles running through a certain node have been avoided. Once the node is calculated, it has been deactivated if it does not have any cycle-4 running through it. This practically, helps to reduce the complexity of the algorithm since the number of edges are reducing as the algorithm proceeds for the graph. Later, mathematical expression and simulation results have been illustrated to investigate the BER performance over BEC for regular (3,6) ensemble of code. Afterwards, removal method by puncturing of nodes containing short cycle from the transmission path of the parity check matrix have been proposed as a technique to achieve capacity. Elaborate description about the numerical analysis of maximum achievable rate and simulation results by removing this unreliable transmission path has been added to demonstrate the optimality of the scheme.

In Chapter 5, a method has been a proposed to achieve the rate compatible, capacity achieving LDPC code over BEC using 0.5 rate mother code. With the proposed method, decoder erasure probability is seen to have a value less than 10^{-5} for a code with length $n=10^3$. For generating rete compatible codes, parity bits have been considered to puncture out and the pattern is chosen from the result of Chapter 4 and additional pattern is chosen to match the target rate.

In Chapter 6, we provide all the summery of the research and some aspects of the possible future work.

1.5 Contribution of the Thesis

The main contributions of the thesis are stated as follows:

- Development of an existing iterative message passing algorithm [9] to find the exact position of the potential information bits that might create cycle after transmission occurred in BEC. This algorithm uses two iteration which can find out the associability of cycle of length *2g* with each node in the parity check matrix, *H*.
- An effective method has been proposed to design a capacity achieving LDPC code over BEC by removing short cycles through puncturing using ensemble (3,6) code.
- As an extension to the previous work, a method has been proposed to design rate compatible and capacity achieving LDPC code using a mother code of rate 0.5 to achieve higher rate code with ensemble (3,6).

2 Theoretical Background

This chapter provides an overview of the fundamentals and dynamics that includes the general idea for LDPC code, construction and various ways of code representation, which is important to develop the insight for the thesis. Additionally, the chapter includes the encoding and graphical representation of decoding methods of LDPC code over two different memoryless channels.

2.1 Error Detection Using Parity – Checks

The important idea for Forward Error Correction (FEC) is to augment the message bit with deliberately imposed and controlled redundancy called parity in the form of extra check bits to form a new codeword for the corresponding message. These parity check bits are imposed in such a way that the codewords are sufficiently distinguishable from one another and can be recovered at the receiver when changed after being transmitted through a channel.

The simplest possible redundancy that could be added in the message stream is Single Parity Check (SPC). The SPC involves adding only one single bit as redundancy to recover the data after transmission over the channel and the value of the parity depends on the value of the message bits. For example, in an even parity code, the additional bit added to each message ensures an even number of 1's in every codeword.

Example 2.1

We can consider an arbitrary seven-bit message v = [1 0 1 0 0 1 1] that is using SPC with even parity to correct error. The v already has an even number of 1s. Therefore, the value the eighth parity would be 0 and the codeword would be C= [1 0 1 0 0 1 1 0]. In general, the codeword has the following structure, C = [c_1 c_2 c_3 c_4 c_5 c_6 $c_{7....}$ c_i], where c_i is either 0 or 1 that satisfies the constraint,

$$c_1 \oplus c_2 \oplus c_3 \oplus c_4 \oplus c_5 \oplus c_6 \oplus c_7 \oplus c_8 = 0 \qquad (2.1)$$

Equation 2.1 is called a parity check equation and the symbol $\oplus$ represents modulo-2 operation in Galois Field, GF (2). The inversion of a single bit can be easily detected by SPC, but it is not feasible to handle large data with a higher probability of error. In this case, more than one redundancy is added and a message must satisfy a set of parity check equations to be a code.

Example 2.2

The code from the previous example is encoded and the resulting codeword was sent through a noisy channel where the codeword is changed and y = [1 0 0 1 0 0 1 0] was received. To check if y is a valid codeword we need to check with equation below-

$y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_8 = 1$

Since the modulo-2 sum is 1 and the constraint is not satisfied, y is not a valid codeword. Thus, by using parity we have detected that at least one error has occurred during transmission.

2.2 LDPC Codes: An overview

As the name suggests, low-density parity check codes are the class of linear block codes that have matrices with a very small number of non-zero entries. The sparseness of the parity-check matrix guarantees low decoding complexity and the minimum distance which also increases linearly with the length of the code [22]. LDPC codes are generally designed by designing a sparse parity check matrix. The main difference between LDPC code and other linear block code is the decoding methods. Classical block codes are decoded algebraically using Maximum Likelihood (ML) decoding where LDPC codes are decoded with iterative decoding algorithm allowing practical systems to have larger block lengths.

2.2.1 Definition and Notation

LDPC code *(C)* is defined by the null space of the H-matrix. This implies the definition as, an n-tuple $v = (v_0, v_1, \ldots, v_{n-1})$ of bits forms a codeword if and only if $v.H^T=0$, which means the codewords satisfies a set of parity check equations [7]. By sparsity, it means that the number of non-zero elements in the parity check matrix is less than the number of the zero elements. In this thesis, we have worked with binary LDPC code. So, the parity check matrix only consists of 0s and 1s.

The structure of the parity check matrix described by Gallager had the following properties [7]-

1. Each row consisting of ρ, 1s.
2. Each column consists of γ ,1s.
3. The number of 1's in common between any two columns is no greater than 1.
4. Both ρ and γ are smaller compared to the length of the code.

Property 1 and 2 make the H-matrix constant row and column weights, making it a regular code. If all the rows and columns do not have the same weight, the code is called irregular code. Throughout the thesis we have implemented and investigated the regular codes and their performances. The density of the code is defined as the ratio of the total number of 1's to the total number of entries in H [7].

$$R = \frac{\rho}{\lambda} = \frac{\lambda}{J} \tag{2.2}$$

2.2.2 Code Representation and Construction

The linear code takes input alphabet which has a structured field and encodes it by adding controlled redundancy as parity bits. It takes k bits as input and takes out n bits which is the code length. Such codes have a design rate of k/n and number of parity is $m= n\text{-}k$**.** For any code that takes k bits input, has 2^n total possible codewords and $2^{n-m} = 2^k$ valid codewords. LDPC code is graphically represented by bipartite Tanner graph with two sets of nodes. The H-matrix is analogous to this graph representation. The graph G, shown in Figure 2.1 with n left nodes called variable nodes, message nodes or information nodes and m right nodes called check node.

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

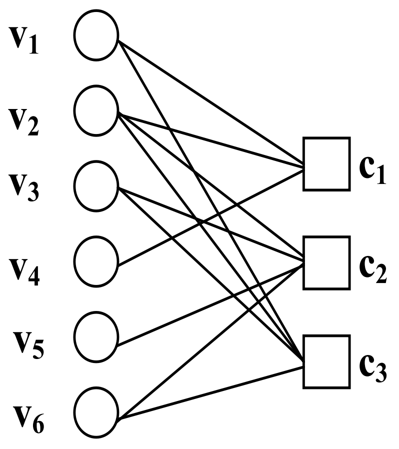


Figure 2.1 Parity check matrix (left) corresponding Tanner graph (right)

The same Tanner graphs are sometimes represented horizontally as variable nodes on the top and check nodes on the bottom Figure 2.2.

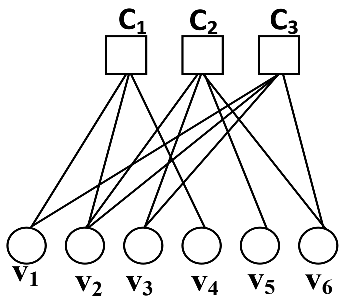


Figure 2.2 Alternative representation of Tanner graph

Another representation of Tanner graph is used where message nodes and parity nodes are kept separate considering the systematic formation of the matrix. In this case, generally the message nodes are drawn in below, parity bits are drawn on the top with the check nodes in between.

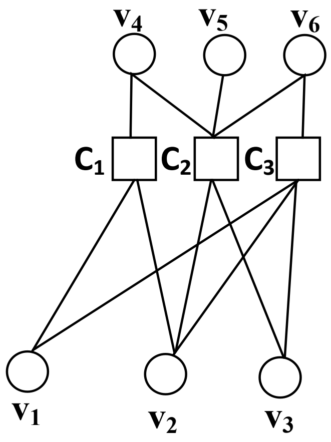


Figure 2.3 Alternative representation of Tanner graph

Figure 2.1,Figure 2.2, Figure 2.3 are all different representation of Tanner graph for LDPC code. *H*-matrix with dimension $m \times n$ in which the entry $(i\ ,\ j)$ is 1 if and only if the i-th check node is connected to the j-th variable node in the graph.

LDPC construction is done by assigning a very small number of non-zero elements compared to the zero-elements in the parity check matrix. The assignment of non-zero and zero elements are accomplished in such a way that the row and the column of the parity check matrix attain a particular and predefined degree distribution.

The original Gallager representation of LDPC code is a regular code defined by the tight structure in *H*-matrix [7], [22].

- The rows in the Gallager parity check matrix are divided into w_c sets with M/w_c rows in each set.

- The first set of rows contain w_r consecutive ONES from left to right.
- Rest of the set of rows are randomly chosen column permutation of the first set.
- Consequently, every column has a ONE entry once in every one of the w_c sets.

Example 2.3

Figure 2.4 shows a length 12 (3, 4) regular LDPC code parity check matrix based on Gallager construction.

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

Figure 2.4 Gallager construction of LDPC code

Example 2.4

Another common construction of LDPC parity-check matrix is Neal and McKay construction [6]. In this method, H is formed by adding one column at a time from left to right [22], [6]. Column weight is chosen to get the correct degree distribution and the location of the non-zero entries for each column is chosen randomly from the rows that have not been filled yet. The process checks if there are any rows that have not been filled, there must be columns left to be added to get the correct degree distribution. The process is then started again and backtracked to some few columns until the exact degree distribution is obtained.

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

Figure 2.5 Neal and McKay construction of LDPC code

Example 2.5

Another kind of construction of parity-check matrix is repeated accumulator (RA) code. In this kind of construction weight, 2 columns is chosen and arranged in a step pattern form for the last m columns of (*H*). This structure makes the whole construction systematic and allows low encoding complexity.

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

Figure 2.6 Construction of RA-LDPC code

LDPC codes can also be constructed algebraically based on the points and lines from the perspective of finite geometries, such as Euclidean geometries (EG) and Projective geometries (PG) which has been classed into four types of LDPC codes [2] –

1. Type-I EG-LDPC codes
2. Type-II EG- LDPC codes
3. Type-I PG-LDPC codes
4. Type-II PG-LDPC codes

In general, LDPC codes are formed pseudo-randomly based on some predefined criteria according to the objective of use. This is the reason while talking about construction, the emphasis is given on the set (or ensemble) of all possible codes with a choice of the parity-check matrix with those degree distributions [22].

2.3 Error Correction Using LDPC Codes

As we have mentioned earlier in section 2.1 that the code construction with augmented bits needs to satisfy a set of parity check equations. For LDPC code (C) must be the null space of the parity check matrix. This section demonstrates the methodology a parity check equation forms a valid code and the encoding scheme that enables to detect errors.

Example 2.6

Consider a code C consists of all length eight strings. C= [c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8] which satisfies a set of parity check equations:

$$c_2 \oplus c_4 \oplus c_5 \oplus c_6 = 0$$

$$c_1 \oplus c_2 \oplus c_3 \oplus c_6 \oplus c_7 = 0$$

$$c_3 \oplus c_6 \oplus c_7 \oplus c_8 = 0$$

$$c_1 \oplus c_4 \oplus c_5 \oplus c_8 = 0$$

The codeword constraints are written in matrix from as below

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}}_{H} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The matrix (H) is the parity check matrix. Every row in the matrix represents a parity check equation that is satisfied by the codeword. For example, if we take the codeword as of, C= [1 0 0 0 0 0 1 1] for the above parity check matrix, it is a valid codeword that satisfies every constraint. Thus, for binary parity check matrix, there is r parity-check constraints and length n codewords making an m × n binary matrix. Any codeword is valid if and only if it satisfies the constraint -

$$cH^T = 0 \tag{2.3}$$

Now that a codeword has been formed we would look at how this is used to detect and correct errors. We can consider that the codeword has been sent through a binary symmetric channel and some bits are flipped but we have no clue which ones are flipped. We will now see how using parity bits these incorrect bits are found and corrected.

2.4 Encoding of LDPC Code

Encoding of the linear block codes means adding the parity bits to the information bits in a controlled way by maintaining specifications to distinguish between the parity and message bits. One major concern that has been expressed in the implementation of LDPC code is the complexity of encoding. Authors in [24] suggested using cascaded graphs rather than a bipartite graph. One major drawback of this approach was consideration of stages in cascade containing smaller length which acted like sub-code. These stages had very small length compared to the length of the overall length of the code. This lead to overall performance loss of the code. In [25]authors suggested the idea of the almost lower triangular matrix for lowering encoding complexity to linear from quadratic. Richardson et. al in [26] published the efficient method for encoding LDPC code. The paper shows that encoding complexity is manageable in most cases with some pre-processing. For example, the paper[26] shows that for regular (3,6) code complexity can be of order n^2, with $0.017^2n^2 + O(n)$ operations before encoding starts.

This below section expresses the example that has been shown in [26]for a better understanding of the process of encoding. The proposed method of the encoder is obtained by performing row and column permutations to bring the H matrix in a lower triangular form. More precisely the matrix should in the form-

$$H=\begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \tag{2.4}$$

Where $A= (m\text{-}g)\ X\ (n\text{-}m)$, $B= (m\text{-}g)\ X\ g$, $T =(m\text{-}g)\ X\ (m\text{-}g)$, $C = g\ X\ (n\text{-}m)$, $D = g\ X\ g$, $E = gX\ (m\text{-}g)$.

All these matrices are sparse, and T is an approximately lower triangular matrix with ONE along the diagonal. Here, g is the gap of the matrix. For the next step, the matrix is multiplied from left by –

$$\begin{bmatrix} I & 0 \\ -ET^{-1} & I \end{bmatrix} \tag{2.5}$$

I = Identity Matrix

After multiplying we get-

$$\begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix} \tag{2.6}$$

Let x= [u, p_1, p_2] where s is the systematic part, p_1, p_2 gives the parity for encoding messages. p_1 has the length g and p_2 has the length $(m\text{-}g)$. The validating equation $Hx^T =0$ splits into two parts –

$$As^T + Bp_1^T + Tp_2^T=0 \tag{2.7}$$

$$(-ET^{-1}A+C)\ s^T + (-ET^{-1}B+D)\ p_1^T=0 \tag{2.8}$$

Authors have defined $\emptyset = -ET^{-1}B + D$ and assumption is made that $\emptyset$ is non-singular. Then from equation (2.8) we can conclude that –

$$P_1^T = -\emptyset^{-1}(-ET^{-1}A + C)s^T \tag{2.9}$$

The whole process has been shown in the below tables. Table 2.1 explains the preprocessing steps for encoding and Table 2.2 explains the operation for efficient computation of parity bits.

Preprocessing for Encoding

We have a non-singular parity check matrix H as input.

After the processing, we get a matric in the form $\begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}$ with $-ET^{-1}B + D$ in non-singular form. The steps that have been considered [23] in the processing are-

1. [Triangulation] Performing row and Column permutation to bring the parity check matrix in approximate lower triangular form with a small gap as possible.

$$H = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}$$

2. [Check Rank] Use Gaussian elimination to effectively perform the pre-elimination

$$\begin{bmatrix} I & 0 \\ -ET^{-1} & I \end{bmatrix} \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} = \begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix} \tag{2.10}$$

To check that $-ET^{-1}B + D$ in non-singular, performing further column permutations if necessary to ensure this property.

The next step is encoding by computing the parity bits.

Encoding:

After the pre-processing has been done on the matrix the encoding steps follows-

Input: The sparse parity check matrix in the format of (2.7). A vector $s \in F^{n-m.}$

Output: Vector $x = [u, p_1, p_2]$ such that $p_1 \in F^g$ and $p2 \in F^{m-g}$. Where x satisfies $Hx^T = 0$.

1. Calculate p_1 as shown in Table 2.1

2. Calculate p_2 as shown in Table 2.2

Table 2.1: Calculation of $p_1^T = -\emptyset^{-1}(-ET^{-1} + C)\, s^T$

Operation	Comment	Complexity
As^T	Multiplication by sparse matrix	$O(n)$
$T^{-1}[As^T]$	$T^{-1}[As^T]=y^T \leftrightarrow [As^T]=Ty^T$	$O(n)$
$-E\,[T^{-1}As^T]$	Multiplication by sparse matrix	$O(n)$
Cs^T	Multiplication by sparse matrix	$O(n)$
$[-E\,T^{-1}As^T] + [Cs^T]$	Addition	$O(n)$
$-\emptyset^{-1}[-ET^{-1} + Cs^T]$	Multiplication by $g{\times}g$ matrix	$O(g^2)$

Table 2.1 First set of Parity Bit Calculation

Table 2.2: Calculation of $p_2^T = -T^{-1}(As^T+Bp_1^T)$

Operation	Comment	Complexity
As^T	Multiplication by sparse matrix	O(n)
Bp_1^T	Multiplication by a sparse matrix	O(n)
$[As^T] +[Bp_1^T]$	Addition	O(n)
$-T^{-1}[As^T +Bp_1^T]$	$-T^{-1}[As^T +Bp_1^T] = y^T \leftrightarrow -[As^T+Bp_1^T] = Ty^T$	O(n)

Table 2.2 Second Set of Parity Bit Calculation

Example 2.7

We will conclude this section by demonstrating an example from [26] that illustrates an encoding example. We wish to encode a message of 6 with the *H*-matrix.

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

For efficient encoding the columns are re-ordered as 1,2,3,4,5,7,10,11,12,8,9 and brought into approximate lower triangular form with a gap two.

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

Now Gaussian elimination is used to clear E. This will result in-

$$H=\left[\begin{array}{cccccc|cc|cccc}1&1&1&0&0&1&1&0&1&0&0&0\\1&1&1&1&0&0&0&1&0&1&0&0\\0&0&0&0&1&1&1&0&1&1&1&0\\1&0&0&1&1&0&0&0&0&1&1&1\\\hline 0&1&0&1&0&0&1&1&0&0&1&1\\0&0&1&0&1&1&0&1&1&0&0&1\end{array}\right] \tag{2.11}$$

From Equation (2.11) we can see that $\emptyset = -ET^{-1}B + D = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$ which is singular. This singularity can be removed by exchanging some of the columns. In this case, we can exchange column 5 with column 8 that will result in $\emptyset = -ET^{-1}B + D = \begin{bmatrix}1 & 0\\1 & 1\end{bmatrix}$. So finally, after re-ordering again the matrix yields to-

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

Suppose we wish to encode u = [1 0 0 0 0 0] with this parity check matrix. To determine p_1 we need to follow the calculation mentioned earlier in Table 2.1.

$$As^T = [1\ 1\ 0\ 1]^T$$

$$T^{-1}[As^T] = [1\ 1\ 0\ 0]^T$$

$$-E\ [T^{-1}As^T] = [0\ 1]^T$$

$$Cs^T = [0\ 0]^T$$

$$[-ET^{-1} + As^T] + [\ Cs^T] = [0\ 1]^T$$

$$\text{And } -\emptyset^{-1}\ [-ET^{-1} + Cs^T] = [0\ 1]^T = p_1$$

For calculating p_2 we follow the similar steps mentioned in Table 2.2.

$$Bp_1^T = [0\ 1\ 0\ 0]^T$$

$$[As^T] + [Bp_1^T] = [1\ 0\ 0\ 1]^T$$

$$-T^{-1}\ [As^T + Bp_1^T] = [1\ 0\ 1\ 0]^T = p_2^T$$

Finally, we get the encoded message as x= [1 0 0 0 0 0 0 1 1 0 1 0] which verifies $Hx^T=0$ as per definition of the LDPC code.

2.5 Decoding of LDPC Code

The class of algorithm used to decode data with LDPC code is collectively known as message-passing algorithm because the class of this algorithm is initiated and propagated by passing messages considering any partition of the Tanner graph. Each node in progress works in isolation taking only into consideration the edges connected to it [22]. Message-passing algorithms are also termed as iterative decoding as the messages are passed and sent back forward until a desirable result is achieved or the decoder exhausts some pre-feed criteria. Messages passing algorithms are classified based on the type of the messages passed and the type of operation performed at the node. Based on the operation is performed in the node decoding classes can be categorized into-

1. Majority Logic Decoding (MLD): Hard decision
2. Bit Flipping (BF): Hard decision
3. Weighted Bit-Flipping (WBF): Combination of hard and soft decision
4. A posteriori Probability Decoding (APP): Soft decision
5. Iterative Decoding Based on Belief Propagation (IDBP): Soft decision

The IDBP is also known by the sum-product algorithm. MLD is the simplest form of decoding algorithm in terms of complexity. The other algorithms especially the fourth one gives the better performance but higher in complexity computation [2]. In hard decision ϵdecoding, messages are binary values. In case of soft decision, the messages are calculated in terms of probability and likelihood. It is often convenient to represent the probability values in log likelihood ratios. When likelihood ratios are used in belief propagation it is termed as sum-product algorithm.

2.5.1 Decoding in the BEC

As we have mentioned earlier in this chapter in the mechanism of BEC that a transmitted bit is either received correctly or erased in the channel with probability, ε. Since the received bit is always correct in this case, the task of the decoder is to determine the erased bit. If there is a corresponding parity-check equation for the erased bit, the erased bit can be found by choosing the value that satisfies the parity check equation.

Example 2.8

The code in Example 2.6

$$c_2 \oplus c_4 \oplus c_5 \oplus c_6=0$$

If the value of the bit $c_1=1$, $c_5=1$, $c_6=0$, and c_4 is erased, then the decoder would decide the value of c_4 as 0. We will start explaining about the decoding with message passing algorithm which is collectively known as BP. In the message passing or iterative decoder, the heck node determines the value of the erased bit in case of BEC in the parity check equation.

2.5.2 Decoding in BEC with BP

Decoding of LDPC code in BEC is straightforward. First, the message node *(v_i)* sends the outgoing message to the each of its connected check nodes *(c_j)*. Initially, it is sufficient to assume that all zero codewords were sent. Amin Shokrollahi in [4] has vividly explained about the mechanism of BP in BEC.

The log-likelihood of the message at round 0, m_c is $+\infty$ if the message has not been erased and is 0 if the message has been erased in the channel [4]. The check node equations are updated in every iteration. The update equation m_{cv} is $+\infty$ if and only if all the incident message node incident to check node c_j except v_i are not erased. In all other cases, m_{cv} is zero. On the other hand, m_{vc} is $+\infty$ if and only if there is some check node incident to node v_i other than c_j which was sending a message to this message node in the previous node.

Because of the binary character of the channel BP in the channel has been described as explained in [4]-

1. [Initialization]: Initialize all the values of the check node as zero.

2. [Direct recovery]: For all the message node v_i , if the node is received, then add its value to the values of the adjacent check nodes and remove v_i together with all edges emanating from it from the graph.

3.[Substitution recovery]: If there is a check node c_j of degree one, substitute its value into the value of unique neighbor among the message node, add that value into the values of all adjacent check nodes and remove the message nodes and all edges emanating from it from the graph.

Example 2.9

To demonstrate the decoding in BEC through BP we will again take the code from Example 2.6

The channel output is $M_i = [e\ e\ 0\ 0\ 0\ 0\ e\ 1]$.

Step 1:

At first, all the check nodes are initialized zero. Proceeding further, all the incoming message in the check node is calculated. For the parity-check matrix mentioned in and for the above codeword, the 1-st check node is connected to 2nd, 4th, 5th and 6th message nodes among which 2nd bit is erased. According to BP, the outgoing message from 1st check node to 2nd message node would be-

$$c_{1,2} = m_4 \oplus m_5 \oplus m_6 = 0 \oplus 0 \oplus 0 = 0$$

Since the first check node has only one incoming message that has been erased, the outgoing message on this edge $c_{1,2}$ will be the value second recovered bit. The 2nd check node is connected to 1st, 2nd, 3rd, 6th and 7th-bit node among which two are erased. So, this check node is not used to calculate any value for an update. The 3rd check node is connected to 1st,6th, 7th and 8th-bit node among them only one bit is erased. Like the first check node, the

outgoing message from this check node is-

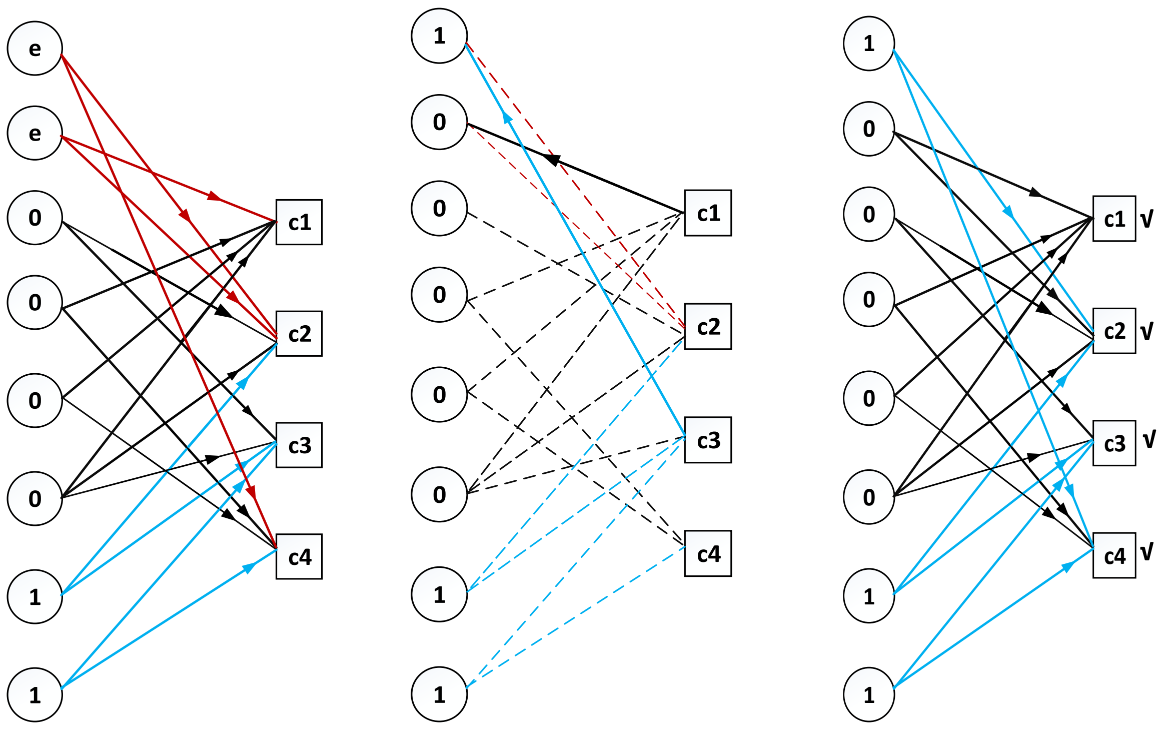
$$c_{4,1} = m_4 \oplus m_5 \oplus m_8 = 0 \oplus 0 \oplus 1 = 1$$

Step 2:

Substitution recovery. After first iteration, the codeword is C= [10 0 0 0 0 1 1].

At the end of this first iteration, the decoder will announce the codeword after updating C= [1 0 0 0 0 0 1 1] which satisfies the constraint in (2.3). There is no need additional iteration as the decoder successfully decoded all the erased bits. The received codeword string now has been successfully decoded after two of the bits being erased in the channel during transmission.

Since in case of BEC, the receiver only has the correct bit or the unknown bit, the messages passed between the nodes are either correct values or erasure. While introducing channels like BEC or AWGN, the messages passed in the nodes are the best guesses of the codeword bit values based on current information available.



(a) Message Passed (b) Message Update (c) Check Constraint

Figure 2.7 Message-passing decoding of the received string M= [*e e* 0 0 0 0 1 1]. Each sub-figure indicates the decision made at each step of the decoding based on the message passed from the previous step. A blue arrow indicated bit value '1' and black arrow indicates bit value '0'. The dashed arrow indicates the unchanged message bits.

The decoding scenario would have been different if the erasure positions were different in the codeword. Suppose we take the same codeword with erased bits in position 4th and 5th. For the erasure position in Figure 2.8 there is no check node with incoming messages among which only one message is unknown and can be calculated by module sum of the rest of the incoming message into that check node. This will result in decoder failure irrespective of the iteration number of the decoder. This type of combination is termed as cycle and leaves a detrimental effect on decoding by decoder failure, premature stopping and dropping down the threshold in BEC.

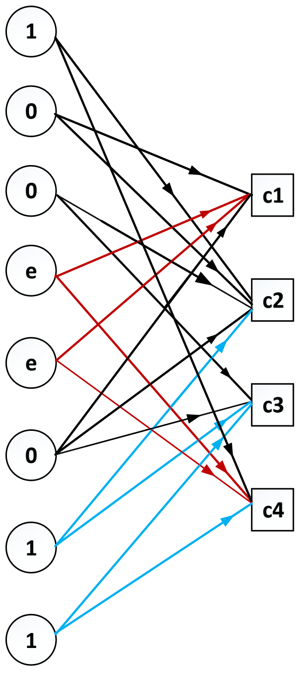


Figure 2.8 Erasure position in the cycle (red arrow) of the code causes early stopping of the iterative decoder

2.5.3 Decoding in BSC with Bit-Flipping Algorithm

Bit-flipping is a hard decision decoding algorithm for LDPC codes. A binary decision about each received bit is made at the decoder. For this algorithm, a bit node sends a message declaring ONE or ZERO, and each check node sends a message to connected bit node. The check node determines that if the parity-check equation is satisfied with the modulo-2 sum of the incoming bits. If the majority of the messages received by a bit node are different from the received value, the bit node changes or flips the current value. The algorithm keeps on repeating the iteration until a valid codeword is found. In fact, this is true for all the message passing decoding of LDPC codes and has two important advantages; firstly, additional iterations can be avoided once the solution is found and secondly the failure to converge to a codeword is always detected [22]. The bit-flipping algorithm is based on the principle that if a bit node to be checked into consideration is likely to be incorrect if it is involved with a majority number of incorrect check node equations[22].

In general, the bit-flipping algorithm can be explained as below:

1. [Initialization]: The check nodes are initialized as zero.

2. [Check messages]: Check nodes calculate the value by doing module-2 operation of the incoming message node but send the value to all the neighboring nodes but the incident one.

3. [Message Update]: If the majority of the check node gives the same value as the received one, the decoded bit remains same otherwise flipped.

The bit flipping algorithm is presented in the example below.

Example 2.10

We again take the parity-check equation with codeword of Example 2.6

Step 1: Check node calculation.

$m_{1,2} = c_4 \oplus c_5 \oplus c_6 = 0 \oplus 0 \oplus 1 = 1$

$m_{1,4} = c2 \oplus c5 \oplus c6 = 0 \oplus 0 \oplus 1 = 1$

$m_{1,5} = c2 \oplus c4 \oplus c6 = 0 \oplus 0 \oplus 1 = 1$

$m_{1,6} = c_2 \oplus c_4 \oplus c_5 = 0 \oplus 0 \oplus 0 = 0$

Repeating for the remaining check nodes the value gives:

Check node 2: $m_{2,1} = 0$, $m_{2,2} = 0$, $m_{2,3} = 0$, $m_{2,6} = 1$, $m_{2,7} = 1$

Check node 3: $m_{3,3} = 1$, $m_{3,6} = 0$, $m_{3,7} = 0$, $m_{3,8} = 0$

Check node 4: $m_{4,1} = 1$, $m_{4,4} = 1$, $m_{4,5} = 1$, $m_{4,8} = 0$

Step 2: The first message node receives a message from 2nd and 4th check nodes as of A_1= [2, 4]. One of the check nodes calculate '0' ($m_{2,1}$) and the other one '1'($m_{1,4}$) but the received symbol is '0'. Therefore, at this point, the decoder will decide this bit as zero as there is no maximum logic to follow and change the bit. The rest of the bit follows the same logic and there will be no update in the first three bits. The fourth message node receives '1' from both check nodes A_4= [1,4]. Thus, most of the messages coming to message node 4 is different from the value received from the channel. As a result, the decoder will change the value according to maximum logic and the bit will be flipped from '0' to '1'. Thus, a new bit of message update will be

M= [0 0 0 1 1 0 1 0]

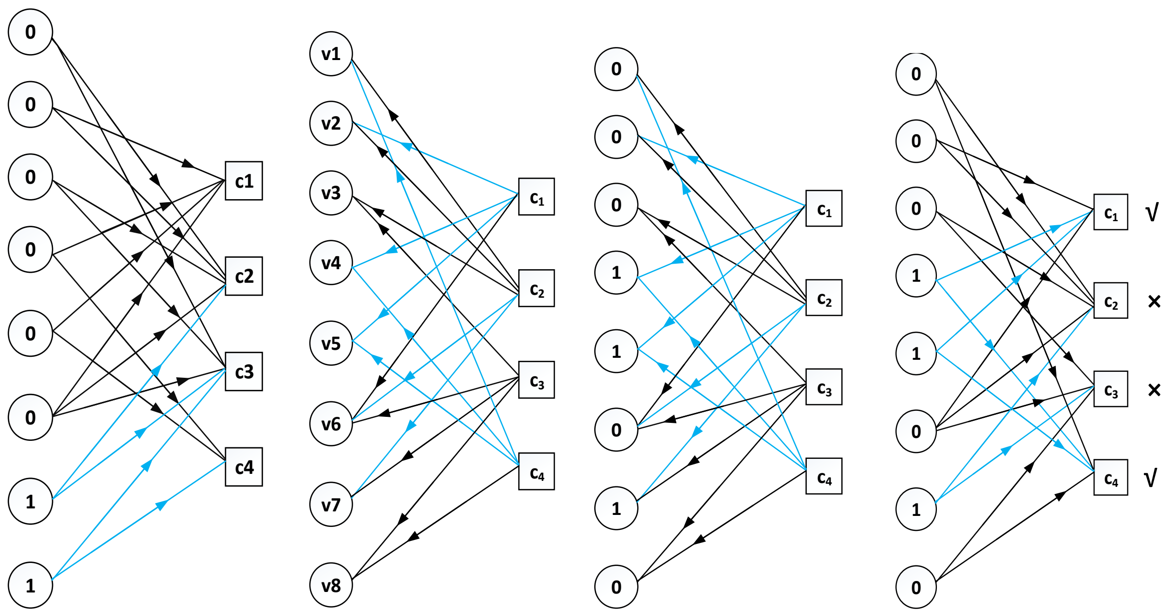


Figure 2.9 Bit-flipping decoding for the received sequence y= [0 0 0 0 0 1 1 1]. Each Sub-figure of the diagram indicates the decision made at every step. A cross (×) represents parity check is not satisfied and the tick (√) indicates that it is satisfied. For the messages, a blue line corresponds to '1' and black corresponds '0'.

For the test, the parity checks are calculated. For the first check node

$$L_1=m_2 \oplus m_4 \oplus m_5 \oplus m_6$$

$$=0 \oplus 1 \oplus 1 \oplus 0=0$$

Similarly, for the rest of the check nodes, $L_2=1$, $L_3=1$, $L_4=0$

This implies that the decoded codeword is not valid since it does not satisfy the check node constraints. The decoder announces (M) as the decoded message even though it could not solve all the check node constraint.

The existence of cycles in the Tanner graph of a code reduces the effectiveness of the iterative decoding process. The code in example

(2.10) has a cycle-4 consisting message node 6 and 7 that reduces the effectiveness of the decoding algorithm due to the occurrence of dependency in the marginal probability of the message bits.

2.6 Conclusion

Message-passing for decoding of LDPC codes was first introduced by R. Gallager in the same paper [7] he introduced the code. During its invention in the 1960s, due to limited computational resources, it prevented Gallager from implementing the algorithm for code more than 500 bits. It was re-discovered in 1993 [6], [27]with the wake of Turbo codes [5]and was recognized as an instance of sum-product algorithm. A cycle has a destructive effect on the performance of the decoder. The efficient way to design capacity approaching codes avoiding cycles has been discussed in next two chapters.

3 Density Evolution: A Parameter to Determine Decoder Capacity

This chapter emphasizes on the calculation method of density evolution as an important determining factor to analyze the performance of LDPC code over BEC. Performance of a code depends on its error detection and correction capability in the presence of erasure. For system modelling, it is crucial to know the highest level of noise the code can handle. Application of LDPC code requires to know the level of channel noise the iterative decoder can correct. This is an open problem which is still being researched on but what possible is to find out is how an ensemble of Tanner graph is likely to behave in the presence of noise in the channel under iterative decoding. To ensure this, probability density is tracked through message passing algorithm.

Density-evolution is the process of tracking the probability density function of Tanner graph for the purpose to analyze the behaviour. For density evolution of a code ensemble the assumption is necessary that the graph is cycle free [22], [4]. Section 3.1 introduces the concept of density evolution for regular and irregular ensemble of codes. Additionally, the concept of threshold has been introduced in this section. Proceeding further, section 3.2 discuss about the determination of threshold in regular and irregular ensemble of codes. Finally, section 3.3 provides the simulation result based on the discussion of the previous section. Additionally, the reason of deviation from theory has been discussed with simulation and analysis in section 3.4. The chapter is concluded by documenting a bibliographic note in section 3.5.

3.1 Density Evolution on BEC

Amin Shokrollahi stated in his paper [4] that most illustrative example of message passing algorithm is comprehensible in BEC. Chapter 2 has elaborated graphical explanation of BP in BEC that states, a parity check equation of a Tanner graph can correct an erasure if and only if only one information bit is erased among the neighboring nodes connected to it. However, the algorithm does not guarantee to decode all the erased codewords [4]. To analyze density-evolution on BEC it is strictly followed that the graph is cycle free which means the graph is a tree. In this case the bit-to-check message to check node in a lower layer of graph is determined by the check-to-bit messages from the neighboring bit nodes from the above level.

3.1.1 Density Evolution on Regular Ensemble

Density evolution has been illustrated in this chapter on the basis of explanation in [8], [24], [25].Given an ensemble of Tanner graph G (w_m, w_c) which consists of bit nodes of degree w_m and check nodes of degree w_c .The objective is to find out how the BP algorithm will perform on BEC having regular ensemble. Analysis done in this section have been discussed on the analysis from [4], [22].

For message passing decoding on BEC, the channel output is (0, 1, *e*) where *e=erased bit values.* The probability that at iteration *l* a check-to-message node is *e* is defined as *'q_l'* and probab*i*lity that probability that at iteration *l* a message-to-check node is erased is defined as *'p_l'*.

The check-to-message entity on an edge is *'e'* if one or more of the incoming messages on the other (w_c - *1)* edges into that check node is an *'e'*. To calculate the probability that *'e'* occurs at check-to-message at iteration *l* the two assumptions are considered-

Firstly, since the channel is memoryless, there is no correlation of the incoming messages. All the incoming messages coming to a check node are independent to each other.

Secondly, the Tanner graph of which the probability density function is determined about, is cycle free. It has no cycle of length *2l, l=* iteration number.

With the assumptions above, the probability that none of the other (w_c *-1)* messages to entering to check node is *'e'* is the product of probabilities, *(1- p_l),* that each individual message is not an *'e'*.

So, the probability that one or more than one other incoming messages from check-to-message is 'e' is-

$$q_l = 1 - (1 - p_l)^{(w_c - 1)} \tag{3.1}$$

Additionally, at iteration *l*, the message-to-check value will be 'e' if the original message received at channel output is an erasure. The probability of this message being an erasure is, Ɛ. Probability that all of the incoming messages to a check node at iteration *l-1* are erasures is, q_l . Considering the similar assumptions as above the probability that a message-to-check value is an *'e'* is the product of the probabilities that the other *(w_m-1)* incoming values to the message node, and the original message from the channel was erased.

$$p_l = \mathcal{E}(q_{l-1})^{(w_m - 1)} \tag{3.2}$$

Substituting, the value of q_{l-1} from equation (3.1) gives recursion as-

$$p_l = \mathcal{E}\left(1-(1-p_{l-1})^{(w_c-1)}\right)^{(w_m-1)} \tag{3.3}$$

Before the decoder starts decoding $p_o = \mathcal{E}$ is the probability that a codeword bit is erased in the channel.

Therefore, for a (w_r, w_c) regular ensemble –

$$p_o = \mathcal{E}\,,\, p_l = p_o\left(1-(1-p_{l-1})^{(w_c-1)}\right)^{(w_m-1)} \tag{3.4}$$

The recursion in (3.4) describes how the erasure probability of the message passing decoding evolves as a function of iteration number *l* for (w_r, w_c) regular ensemble LDPC code.

Example 3.1

Consider a (3,6) LDPC code to be transmitted over a BEC with channel erasure probability Ɛ=0.3 and decoded with message passing decoder. The probability that a codeword bit will remain erased after l iterations of the message passing decoder are given by-

$$p_o = 0.3 \qquad p_l = p_o\left(1-(1-p_{l-1})^{(5)}\right)^{(2)}$$

Applying recursion (3.4) for multiple iteration, we would like to observe at which point the probability that the decoded erased bit is zero occurs. After calculation it is seen Figure 3.1 that after 6 iterations the probability of error comes at zero. Which implies the iterative decoder is supposed to solve all the erased bits while the channel erasure probability is 0.3 for regular s(3,6) ensemble LDPC code.

Figure 3.1depicts the behaviour of a LDPC decoder in the presence of different erasure level over BEC. At the time channel erasure probability 0.3, the iterative decoder can correct all the erased bits after 6 iterations. The simulation has been done with channel erasure probability 0.38, 0.40,0.42, 0.45 and 0.50. As soon the channel erasure probability increases above the point 0.42, probability of error at the decoder also increases and the decoder does not produce satisfactory result in solving the erased bits during transmission. The simulation has been for 35 iterations and the erasure probability gets saturated after 0.42. Now we will shed a light on this critical point which is called he threshold level a code can correct.

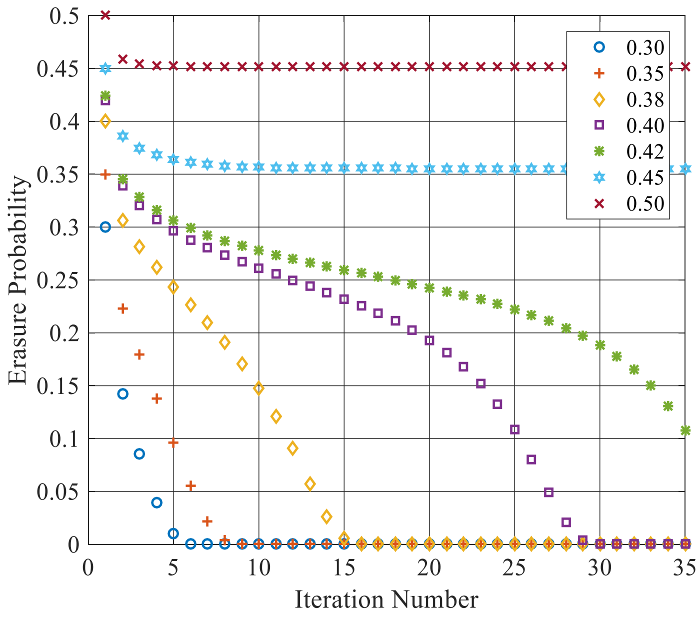


Figure 3.1 Probability of error of iterative decoder at different channel erasure level with (3,6) regular code

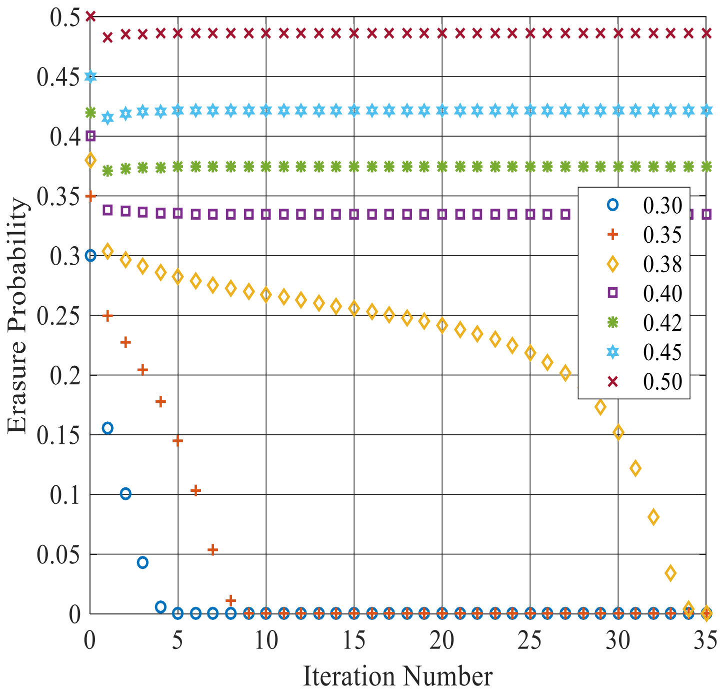


Figure 3.2 Probability of error of iterative decoder at different channel erasure level with $C_{(4,8)}$ regular code

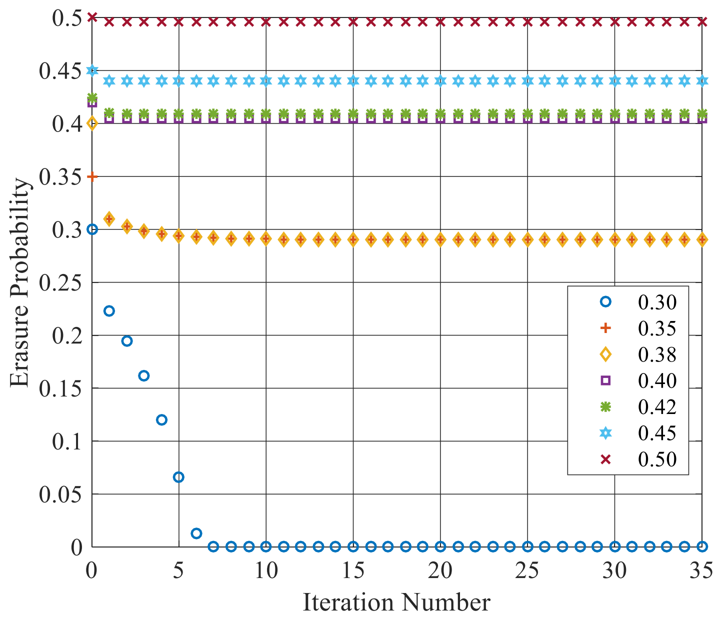


Figure 3.3 Probability of error of iterative decoder at different channel erasure level with C $_{(5,10)}$ regular code

To develop clear insight about the relation of decoder erasure probability to iteration number of regular ensemble, two more simulations have been added with two ensembles (4,8) and (5,10) using recursion equation (3.4). From Figure 3.2 and Figure 3.3, it is observable that at Ɛ=0.38, for (4,8) code 35 iterations are needed to recover all the erasures where as for (5,10) code 35 iterations are not enough to recover the erased bit, in fact the probability of erasure in the decoder gets saturated and decoder cannot recover the erased bits. This is because the recover capability of the decoder is different for different code ensemble.

Example 3.2

This example explains the determination of highest amount of channel erasures an iterative decoder can solve over BEC. From figure 3.1 it is seen that applying recursion (3.3) on a regular ensemble (3,6) LDPC code for channel erasure value ≥ 0.42 the probability of error does not decrease to zero even the iteration number l gets larger whereas, values ≤ 0.42 the probability of error decreases as l approaches to larger value. The transition value between above two outcome scenarios is called 'threshold'.

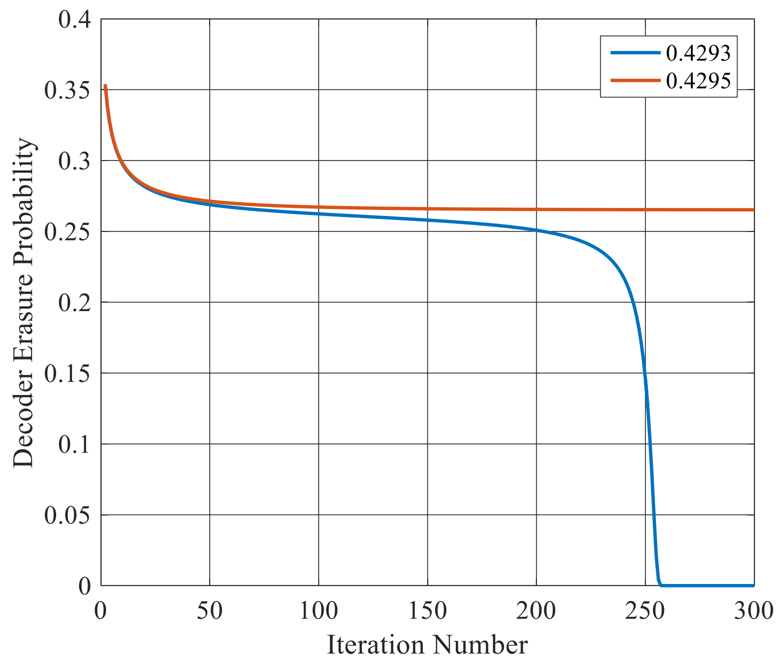


Figure 3.4 Erasures probabilities calculated for 250 iterations for C $_{(3,6)}$ ensemble

Figure 3.4 shows the effect of higher iteration values to decoder probability of error for (3,6) regular LDPC code. 250 iterations are needed to decode all the erased codeword bits at channel erasure probability 0.4295. Then again, the channel erasure probability level is increased to slightly higher than this current value to 0.4293. Unexpectedly, the decoder error probability gets saturated at 0.28 even with 300 iterations. This implies that there might be a point in between these two erasure values which is the actual *threshold* of the code.

3.1.2 Density Evolution on Irregular Ensemble

An irregular LDPC code have varying row and column degree distribution respectively check nodes and message nodes. The fraction of columns of weight i has been designated as v_i and the fraction of rows of weight i as h_i .

To derive the density evolution for irregular LDPC codes an alternative characterization of degree distribution is used which is based on Tanner graph perspective. The fraction of edges which are connected to degree- i message node are denoted as λ_i, and the fraction of edges that are connected to degree- i check node is termed as ρ_i. By definition of degree distribution probability-

$$\sum_i \lambda_i = 1 \qquad (3.5)$$

$$\sum_i \rho_i = 1 \tag{3.6}$$

The degree distribution generating functions are -

$$\lambda(x) = \lambda_2 x + \lambda_3 x^2 + \lambda_4^3 + \ldots + \lambda_i x^{i-1} + \ldots \tag{3.7}$$

$$\rho(x) = \rho_2 x + \rho_3 x^2 + \rho_4^3 + \ldots + \rho_i^{i-1} + \ldots \tag{3.8}$$

Translating between the node degrees and edge degree, the probability that an edge with degree I is connected to a message node is: $v_{i=\frac{\lambda_i/i}{\sum_j \lambda_j/j}}$ and probability that ab edge with degree *i* is connected to a check node is: $h_i = \frac{\rho_i/i}{\sum_j \rho_j/j}$.

In the beginning of this section it is mentioned that, at the *l*-th iteration of the iterative decoder, the probability that a check-to-message value is an erasure 'e', is all the incoming messages are independent, is

$$q_l = 1 - (1 - p_l)^{(w_c - 1)}$$

for an edge connected to a degree w_c check node. In case of irregular Tanner graph, the probability that an edge is connected to a check node with degree w_c is ρw_r. All the edges are averaged over the tanner graph to obtain the formula of probability of error of a check-to-bit message:

$$q_l = \sum_i p_i (1 - (1 - p_l)^{(i-1)}) = 1 - \sum_i p_i (1 - p_l)^{(i-1)}$$

Using the definition from the generating function from equation (3.8) , the above formula becomes

$$q_l = 1 - \rho(1 - p_l) \tag{3.9}$$

From the equation (3.8) we know that the probability a message-to-check message is 'e', at iteration *l* of message-passing decoding is all the incoming messages are independent, is

$$p_l = \epsilon(q_l - 1)^{(w_m - 1)} \tag{3.10}$$

The above formula for probability in check node is true if an edge is connected to a degree w_m message node. For irregular Tanner graph that the edge is connected to a degree of w_m, message node is λw_m. Averaging over all the nodes the probability that a check-to-message value is in error is-

$$p_l = \epsilon \sum_i \lambda_i (q_l - 1)^{(i-1)} \tag{3.11}$$

Again, the definition of degree distribution function from equation (3.7) (3.8) , the above formula in (3.11) becomes-

$$p_l = \epsilon\, \lambda(q_l - 1\,) \tag{3.12}$$

Finally, substituting for q_{l-1} ,(3.11) becomes

$$p_l = \epsilon\lambda(1 - \rho(1 - p_{l-1})) \tag{3.13}$$

Prior to decoding the value of p_0 is the probability that a bit has been erased in the channel is

$$p_0 = \epsilon$$

and for irregular LDPC code, the recursion for probability of decoding error is-

$$p_0 = \epsilon \;, \qquad p_l = p_o\lambda\,(1 - \rho(1 - p_{l-1})) \tag{3.14}$$

3.2 Threshold

The objective of density evolution is to determine the exact threshold for LDPC code in the presence of channel erasure. This section demonstrates the determination of highest value of the channel erasure a message-passing decoder or iterative decoder can correct. Recursion (3.14) gives a way to average the degree distribution of message nodes and check nodes over the whole Tanner graph. For density evolution, an assumption has been made that the graphs are cycle free regardless of row and column weight [4].

To observe the effect of channel erasure probability on decoder probability of error a function is defined [22].

$$f(p,\mathcal{E}) = \mathcal{E}\,\lambda(1 - \rho(1 - p)) \tag{3.15}$$

The erasure probability at iteration-*l* would be-

$$p_l(\mathcal{E}) = f(P_{l-1,}\ \mathcal{E}) \tag{3.16}$$

In (3.16) p and $\mathcal{E}$ are probabilities and can take values from 0 to 1. Here, the function in (3.15) is strictly an increasing function in p for $\mathcal{E}$> 0. Thus if $p_l > p_{l-1}$ then

$$p_{l+1} = f(p,\mathcal{E}) \geq f(p_{l-1},\mathcal{E}) = p_l \quad \text{for } \mathcal{E} \in [0,1] \tag{3.17}$$

so, p_l $(\mathcal{E})$ is a monotone sequence which is lower bounded at p=0 by

$$f(0,\mathcal{E}) = \mathcal{E}\,\lambda(1 - \big(1 - \rho(1)\big) = \mathcal{E}\,\lambda\,(1 - 1) = 0 \tag{3.18}$$

And upper bounded by p=1

$$f(1,\mathcal{E}) = \mathcal{E}\,\lambda\big(1 - \rho(1 - 1)\big) = \mathcal{E}\lambda(1 - 0) = \mathcal{E} \tag{3.19}$$

Since $f(p,\mathcal{E})$ is an increasing function in p,

$$0 \leq f(p,\mathcal{E}) \leq \mathcal{E} \quad \text{for} \quad p \in [0,1] \text{ and } \mathcal{E} \in [0,1] \tag{3.20}$$

Therefore, for a degree distribution pair (λ, ρ) and $\mathcal{E} \in [0,1]$ it can be proven that if

$p_l(\mathcal{E}) \to 0$ then $p_l\ (\mathcal{E}') \to 0$ for all $\mathcal{E} < \mathcal{E}'$. There is a value $\mathcal{E}^*$ that is called the threshold of the code such that for all values of $\mathcal{E}$ below $\mathcal{E}^*$, the decoder probability of error approaches to zero as the number of iterations goes infinity. The threshold for a degree distribution pair (λ, ρ) is defined as the supremum of erasure probability for which $p_l\ (\mathcal{E}) \to 0$:

$$\mathcal{E} * (\lambda, \rho) = \sup\{\mathcal{E} \in [0,1]: p_l(\mathcal{E})_{l\to\infty} \to 0\}$$

3.2.1 Threshold Calculation for Regular Ensemble

This section introduces example for determining the decoding threshold for regular and irregular LDPC code. Threshold provides a code designer the idea about the capacity of the code and as per that different ensemble of codes can be chosen to design the system.

Example 3.3

To calculate the threshold three regular ensemble – (3, 6), (4, 8) and (5, 10) of LDPC code has been considered. From recursion (3.15) and (3.19) the value of supremum of above mentioned code has been calculated. According to the calculation the highest number of channel erasure that is correctable is the minimum if the below mentioned functions-

- $P(x) = \frac{x}{(1-(1-x)^5)^2}$ on $(0,1)$ for ensemble (3,6)
- $P(x) = \frac{x}{(1-(1-x)^7)^3}$ on $(0,1)$ for ensemble (4,8)
- $P(x) = \frac{x}{(1-(1-x)^9)^4}$ on $(0,1)$ for ensemble (5, 10)

For the above-mentioned ensembles, the thresholds are-

Code	Threshold
(3,6)	0.4290
(4,8)	0.3882
(5,10)	0.3537

Table 3.1 Threshold Calculation for Different Ensemble of Codes

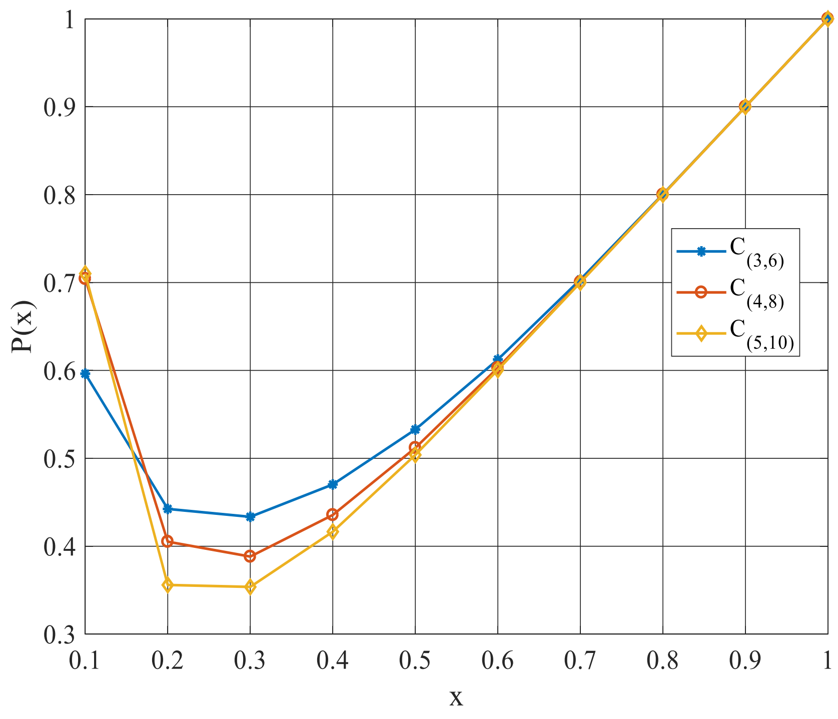


Figure 3.5 Decoding threshold for different regular codes

Figure 3.5 Decoding threshold for different regular codes display simulation results of the above-mentioned functions to determine the decoding threshold of different regular LDPC codes. From the simulation it is seen that the supremum of the function is attained at 0.4290, 0.3882, 0.3537 for (3,6), (4,8) and (5,10) code respectively. This result implied that a regular LDPC code with column weight-3 and row weight-6 is supposed to decode all the erasures if up to 42.9% of the information messages are erased. Similarly, for (4, 8) and (5, 10) codes, the decoder is supposed to correct all the erasures up to 38.82% and 35.37% of the information bits if they have been erased. From this analysis, it can be concluded that among the three regular ensembles above, (3,6) has highest error correcting capability.

3.2.2 Threshold Calculation for Irregular Codes

Example 3.4

An irregular LDPC code is considered with the below mentioned degree distribution have been considered from [22]

$$\lambda(x) = 0.1x + 0.4x^2 + 0.5x^{19}$$

$$\rho(x) = 0.5\, x^7 + 0.5\, x^8$$

Simulation result for the above example using equation (3.19) the supremum of the function has been shown in Figure 3.6.

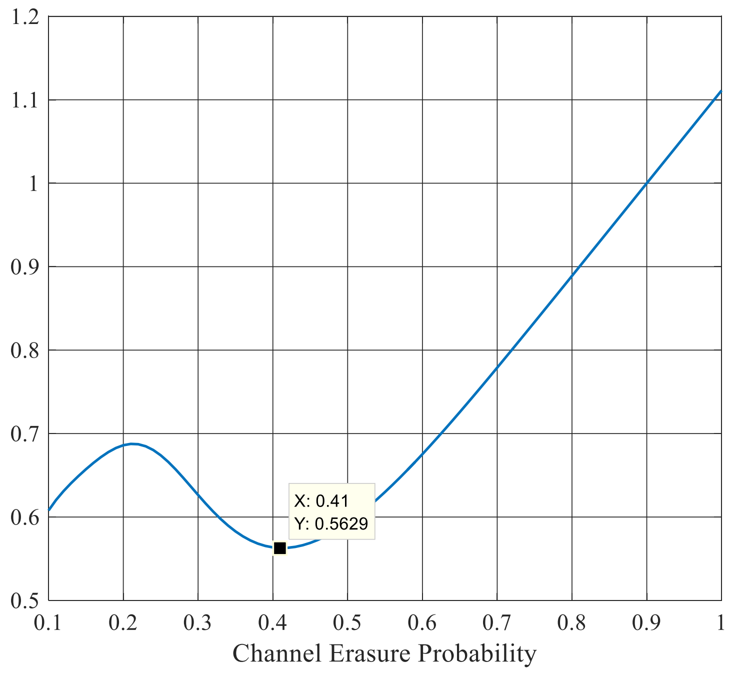


Figure 3.6 Threshold Calculation of Irregular Code

The simulation result in Figure 3.6 shows that the function attends its minimum at 0.41 which implies the code can correct highest if 56.29% of the message bits have been erased during transmission.

3.3 Simulation Results

In the previous sections, properties of regular and irregular code have been discussed based on the theories. In this section a comparison has been made between theory and experimental values. For decoding, three regular ensembles of LDPC code is encoded, transmitted over BEC channel and iteratively decoded using BP algorithm. The codes into consideration have following properties-

- Code length, n=1000
- Rate, R =0.5
- Ensemble $C_{(3,6)}$, $C_{(4,8)}$, $C_{(5,10)}$

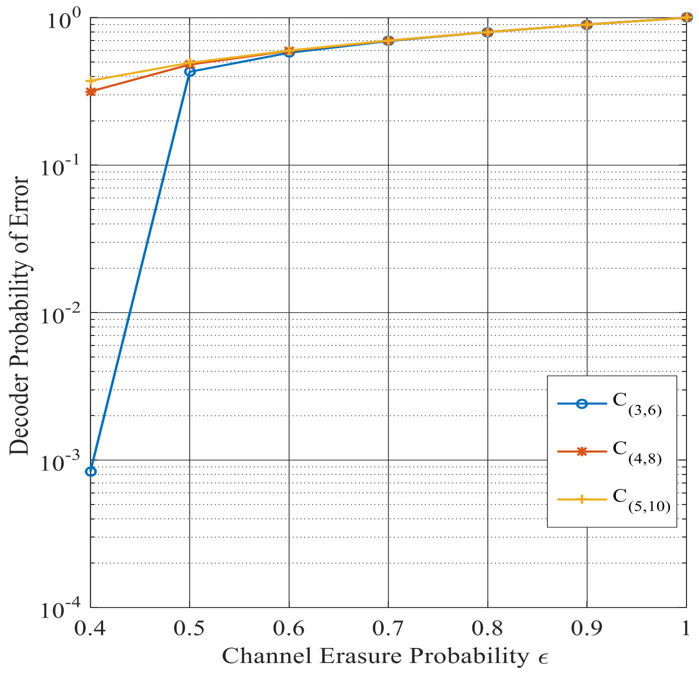


Figure 3.7 Block decoder Erasure probability for three regular LDPC code

From figure 3.7, the performance of three different regular LDPC code can be evaluated over binary erasure channel. The simulation has been run 1000 times for a for a block length of 1000 bit and over the time decoder erasure probability have been calculated. For code ensemble C $_{(3,6)}$ it is seen that the decoder cannot correct all the erasure beyond 40% hence erasure probability starts increasing. This result implies that the code cannot reach the threshold calculated in Figure 3.5. Next two simulations show the effect of iteration number on solving the erasure limit required to reach threshold.

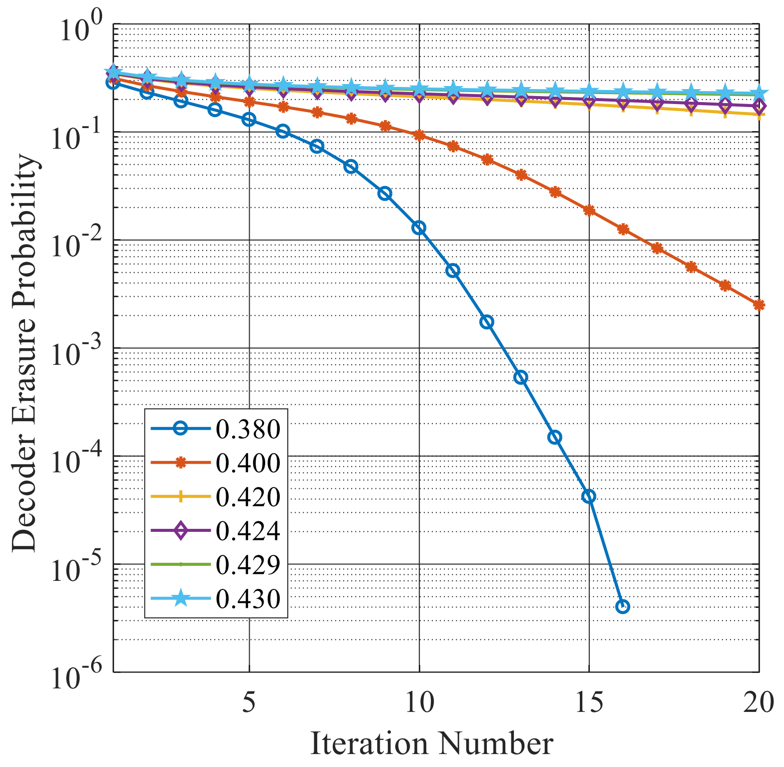


Figure 3.8 Simulation of (3, 6) code performance for 20 iterations

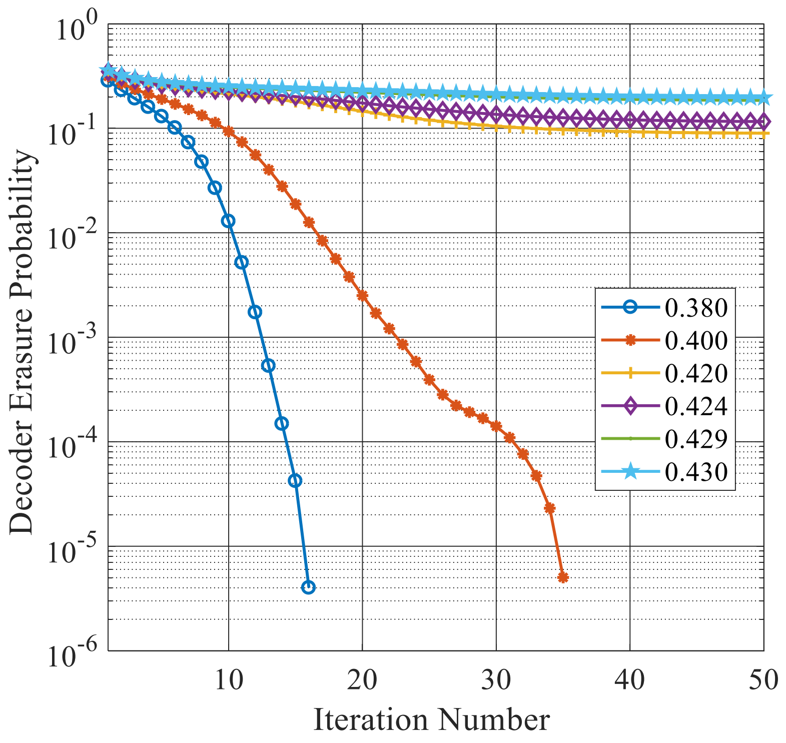


Figure 3.9 Simulation of (3, 6) code performance for 50 iterations

Figure 3.8 and Figure 3.9 provides a clear idea about the effect of iteration number of decoder in accuracy of decoding performance. In figure 3.8, simulations have been run with (3, 6) LDPC code over BEC for 20 iterations and it is observable that the iterations are not sufficient to recover all the erased bits in the channel. Decoder erasure probability cannot reach up the threshold point.

In Figure 3.9, the same simulation has been extended for 50 iterations to observe the effect. This time it is visible that at 38% channel erasure probability the decoder packet erasure rate does not make significant change but as increased to 40% channel erasure probability, at 35 iterations the decoder can recover all the erased bits making the decoder error rate below 10^{-4}. Finally, another important observation from Figure 3.9 can be made that the decoder cannot reach to the theoretical threshold for increasing number of iterations; error floor is seen at channel erasure probability at 42% which is below theoretical threshold. This is due to presence of short cycles in the parity check matrix which cause the decoder to stop prematurely. Elaborate discussion has been made on the effects of cycle in decoding performance and method to remove them have been discussed in next two chapters.

3.4 Conclusion

The type of analysis for evaluating the decoder performance of LDPC code first appeared in R. Gallager's paper [7]. Density evolution for irregular code was first presented [28] by three eminent researchers. Threshold determination with examples of regular codes considering iterative decoding using message-passing algorithm have been presented in [4]with examples for better understanding.

4 Detection of Short Cycles in LDPC Code

In the vast field of engineering and communications, factor graphs are studied very broadly due to its application in artificial intelligence, channel coding, computer science, statistics and wireless communications. To understand the dynamics of such systems, the study of structure of the graph plays a very vital role. For any factor graph or bipartite graph, sum-product algorithm is the generic name of the message passing algorithm which is applied to such graphs for decoding or other purposes [7].

Iterative decoding has showed remarkable performance near to Shannon limit for LDPC code in the presence of noise in BIOSM channel for factor graphs. Complexity of BP is less while considering performance in BEC but the probability of error is higher if there is an existence of short cycle in the Tanner graph [4]. However, since BP is faster than ML decoding, this is adopted by most of the practical system ranging from data storage to wireless communication [17]. For codes like LDPC or Turbo codes, the performance of message passing algorithm is closely related to the sparse structure of the graphs [7]. Girth or cycle distribution of such sparse structure have been associated over different times by scholar as the reason of degradation in the performance. Authors in [13] have explained the performance of such practical systems in the presence of cycles in the parity check matrix. Another paper [29] explains about BP for cycles in Tanner graph. However, dominant trapping sets and cycles are the unreliable transmission path that cause the iterative decoder to stop prematurely causing performance loss.

This chapter introduces the method of determining the unreliable transmission path associated with cycle by applying a class of message passing algorithm [9]. Section 4.1 discusses about cycle formation in parity check matrix and illustrates how exactly they cause the decoder to stop before reaching to the end of decoding. The next section 4.2, provides background about the development of the algorithm from the perspective of graph theory. Section 4.3 introduces the algorithm to find out the exact position in the parity check matrix that might cause cycle after transmission of data. The algorithm can calculate cycles in parity check matrix up to length of 2_g-2, where g is the girth (shortest cycle) of the graph. Additionally, the algorithm can also detect total number of cycles of different length that could be associated with an information node. Finally, section 4.4 concludes the chapter by providing Conclusion about message passing algorithm to find out cycles in Tanner graph.

4.1 Effects of Cycle in Tanner Graph

In a Tanner graph or a bipartite graph, a cycle refers to the finite set of connected edges that starts and ends at the same node and satisfies the condition that no node (except the initial and final node) appears more than once [30], [31] .The analysis tools like degree distribution or EXIT chart analysis also assumes that iterative decoding works best while the neighborhood is a tree [32]. On contrary, in [33]authors have showed that finite length code without cycle shows poor performance in terms of minimum distance. During designing a LDPC code the objective should be having larger girth as much as possible because smaller girth size decreases minimum distance ($d_{min}= g/2$) which is not desirable. The following theorem [32]gives an idea about how the degree distributions that are associated with formation of cycles.

Theorem 1: Assume that cycle is formed with degree two or three variable nodes. Likewise, assume that the message received from channel is also in error. If all the other incoming message in the check node is correct, then the variable node forming cycles cannot be corrected using BP algorithm or any other binary message passing algorithm [32].

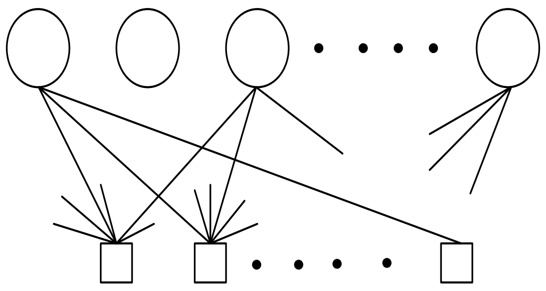


Figure 4.1 Cycle formed by variable nodes of degree two and three

An example of such cycle has been showed in the above figure, where circular edges indicate discrete bit of information from channel and the square edges are the check nodes. Two check nodes are connected to two set of variable nodes making it a cycle-4. In finite length code, the underlying tanner graph could also contain 4-cycle, 6-cycles, 8-cycles and 10-cycles.

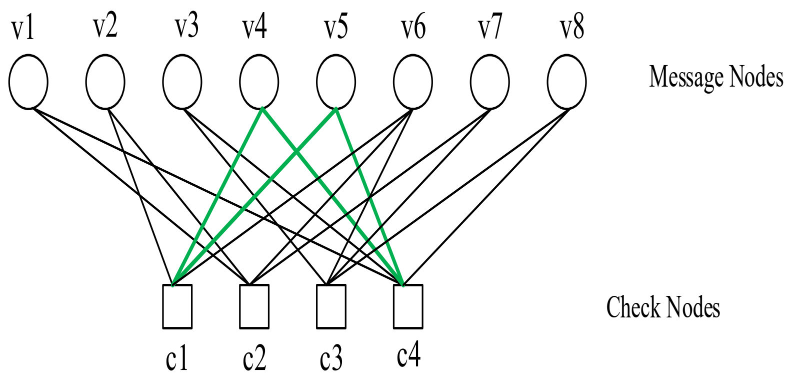


Figure 4.2 Cycle formed by variable node of degree two in BEC

Figure 4.2 shows a bipartite graph of rate=1/2 where message node v4 and v5 creates a cycle-4 which has been made prominent in green. Since every check node is connected twice to the set of variable nodes, the outgoing messages from the check nodes are also in error. In the first iteration, if message node v4 and v5 are erased they cannot be recovered in the next iteration since they are forming a cycle-4. Such conditions in BEC leads to decoding failure because of premature stopping of the iteration in decoder. As a result, outgoing messages from this check node to these two set of variable nodes cannot be recovered if it is erased.

Hence, if a factor graph contains cycles consisting of degree two and three, there is a non-zero probability that the involved channel messages are in error leading to a decoding failure [32]. According to the Theorem 1, to avoid premature stopping of decoder a factor graph must not have cycles. Theorem 2 [32] explains the necessary conditions for a factor graph to be cycle free.

Theorem 2: A factor graph with no cycles exists of variable nodes of degree two and three exists if and only if

$$3\lambda_2 + 4\lambda_3 \leq \frac{6}{d_c}\,\frac{(1-R)-\frac{1}{N}}{(1-R)} \approx \frac{6}{d_c} \qquad (4.1)$$

Where $\lambda_2 =$ node with degree two, $\lambda_3 =$ node with degree three

An important consequence of such condition lead to a conclusion that a regular LDPC code with, degree distribution of variable node, $d_v < 4$ cannot be decoded without an error floor using binary message passing decoder [32]. For regular codes with $d_v = 2\ or\ d_v = 3$, Theorem 2 is satisfied if the degree distribution of check node, $d_c \leq 2\ and\ d_C \leq 1.5$. For both cases the conditions are not possible for positive rates. Therefore, regular LDPC codes with $d_v = 3$ are important for cycle-free graph analysis, but not feasible from practical perspective. In [33] authors have stated that, cycle-free Tanner graphs cannot support good codes by proving and concluding that the number of cycles in Tanner graph must increase exponentially with length n for asymptotically good codes.

4.2 Message Passing Algorithm to Detect Cycles in Tanner Graph

The message passing algorithm depicts the tbc walk for cycle in a LDPC code performed by sending the message along a node and examining if the node has received a copy of the message [34]. This needs to be mentioned here that message passing algorithm for counting number of short cycles is not like BP for decoding with bipartite graph in anyway. Once the cycle forming bit position are detected, we can work on by removing them from the transmission path. This is referred as intentional puncturing of bits. By puncturing we mean removing bits from the codewords either in a random fashion or in some intentional and predetermined way [7].

The algorithm [34] explained can find out a bit associated with potential cycle of length g , $g+2, \ldots, 2_g-2$ where g= girth of the graph. The underlying mechanism is based on performing integer additions and subtractions in the nodes of the graph and passing extrinsic message to the adjacent nodes.

4.2.1 Definition and Notation

A graph G (U, E) is named bipartite if a set U can be partitioned in two disjoint subsets. These two disjoint subsets are V and C (U= V ∪ C and V ∩C = $\emptyset$) where every edge of E is connected from V to C [9]. Tanner graph is a bipartite graph in which two disjoint subsets are variable nodes and check nodes. For the algorithm n and m refer as code length and number of parity check equation respectively.

As mentioned earlier, in the beginning the girth is the length of the shortest cycle. For bipartite graphs all the girth has a length of even number due to two sets of disjoint subsets. The number of edges connected to a node is called degree. For a variable node we have defined degree as d_v and check node as d_c . We call the graph regular if all the nodes in a partition have same degree.

For a tanner graph G (V ∩ C, W) a message passing algorithm refers as passing a message from message node to check node or the opposite. This is referred as parallel scheduling and used in iterative algorithm [9]. In this extrinsic message passing mechanism a complete cycle from message node to check node and from check node to message node is called an iteration.

We assign a corresponding time index *t* where *t ≥ 0* associated with the iteration number. Iteration Number have been denoted by *itr = [t/2] +1 ≥ 1*. Other two-time indices *t=2*iteration-1* and *t=2*iteration-2* depicts the first half and second half of iteration, itr. We also define message passed at first time instance as *t=0*. Message passed from variable node to check node has been defined by m $_{c\leftarrow v}$ and the message from check node to variable node is termed as m $_{v\leftarrow c}$.

The message passing algorithm is operated in the graph by sending message considering any partition to the adjacent edges. The algorithm has a property that says a message sent along one edge e is not a function of the message previously received [9]. This property is termed as extrinsic message passing and the property has been used in the development of the algorithm to detect cycles.

4.2.2 Background for Developing Message Passing Algorithm

Now we focus on the problem of finding out the bits that are associated with potential cycles. In general, if we want to count the number of cycles of certain length *2k* < g/2, which pass through a certain variable node v in the bipartite graph it follows *lemma 1* [9]-

Lemma 1: *Consider a tanner graph with girth g and a node v* ∈ V. Initiate message passing algorithm by passing 1 on all the edges connected to node v ∈ V, v≠ c, while passing different monomials along the edges connected to v. For k ≤ g-1, we get

$$N_{2k}^{v} = \sum_{j=1}^{d_v} ex\,(m_{E,v\leftarrow e_j}^{(k)})/2 \tag{4.2}$$

where N_{2k}^{v} is the total number cycle of length *2k* associated with v.

This is the generalized algorithm for counting total number of cycles associated with each information bit. Now we develop the algorithm to find out the bit associated with cycle-4. We are particularly emphasizing removing cycle -4 because it has the most detrimental effect on the performance of iterative decoding.

To develop the algorithm, we consider that no node would receive a copy of a message it sent through any of the edge connected to it. For example, if a message is passed along any edge as *m*= $X_1^i X_2^j X_3^k X_4^l$, can say that this contains a i copy of X$_1$, j copies X$_2$ and so on.

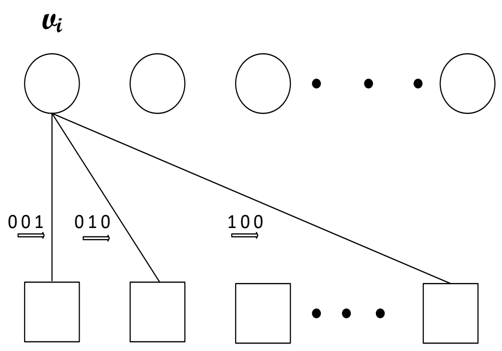


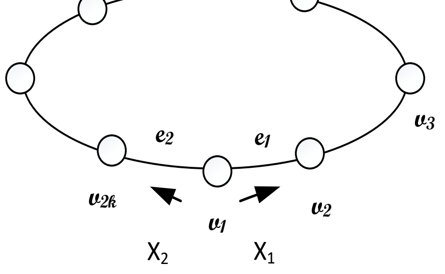
Figure 4.3 Extrinsic message passing at time

For Figure 4.3, suppose if we send all the monomials through each edge connected to this node v_i and this node receive a copy of these message along with the sent one after two iterations, this bit would be a part of the cycle.

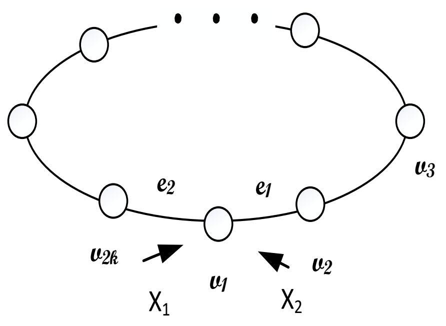
4.3 Detection of Short Cycles in LDPC code

In [9] authors have expressed the insight about a closed cycle path in the graphical modelling specially in bipartite graph. *Lemma 2* [9]exactly expresses the condition for a path to be a cycle.

Lemma 2: Let us consider that C is the cycle of length 2k in a bipartite graph $G = (U,E)$, and $v \in V$ is in C. Denote the two adjacent nodes in v by passing 1 along every edges in E except e*1* and *e2*. We pass monomials in these edges. For e_1 and e_2, the initial messages are monomials X_1 and X_2, respectively. Then, at iteration k, node v will receive one copy of X_2 and one copy of X_1 along e_1 and e_2, respectively, where both copies have travelled through all the edges of C.



(a)



(b)

Figure 4.4 Message Passing is initiated at the start of iteration at time t (a); At the end of iteration k, node v receives a copy of X_1 and X_2 in edge e_2 and e_1 respectively(modified [9]).

This property is applied to determine the associability with cycle of any bit in the parity check matrix. For example, if a message is passed along any edge as $m = X_1^i X_2^j X_3^k X_4^l$, can say that this contains an *i* copy of X_1, *j* copies X_2 and so on. To check this –

- First, the algorithm is initiated at t=0 by sending monomials (e.g.: binary bit stream) according to the degree of the corresponding node. Due to extrinsic property of message passing, the message will be passed to the neighbouring nodes at *t=1*.
- Other nodes will be kept inactive.
- Closed path is calculated for each associated node at *t=2k-1*
- If the node gets a copy of the message sent, it declares the corresponding node a part of cycle.

Example 4.1

An example has been depicted in Figure 4.6 ,that gives idea about the dynamic of the message passing algorithm to find out exact potential bit position as a part of cycle-4. For the graph below in Figure 4.5 [9], the algorithm is applied to find out the associability of cycle-4 for node m_1. The solid line shows the active node and dashed line shows the inactive node.

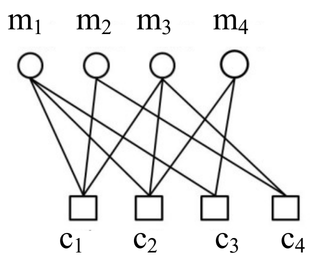
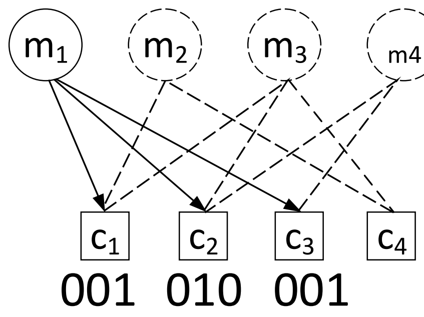
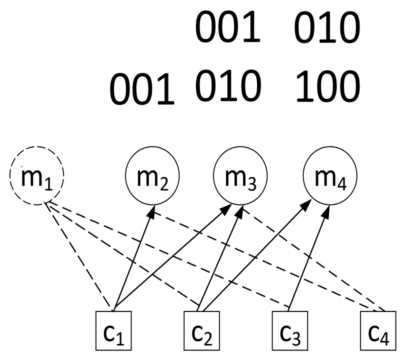


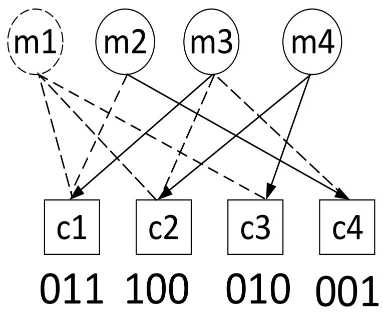
Figure 4.5 Tanner Graph for Cycle Detection



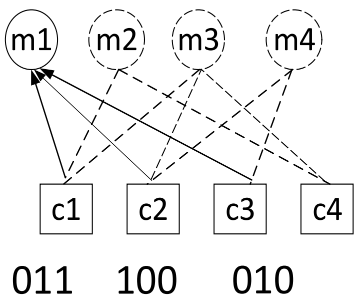
4.6 (I) Iteration 1 *(t=0)*



4.6 (II) Iteration 1 *(t=1)*



4.6 (III) Iteration 2 *(t=2)*



4.6 (IV) Iteration 2 *(t=3)*

Figure 4.6 Message passing algorithm to find out position of cycle forming bit

Iteration 1

Step 1:

- Message passing is initiated to a node starting at the bit string. The monomials are passed from message node to the neighboring check nodes connected to it.

Step 2:

- After the chock nodes received the messages, the send the corresponding messages the neighboring message node.

Iteration 2:

Step 1:

At this point message node receives messages from the check node. Now these message node sends the message again to the check node in a way

- If it receives from one node, sends back to the rest of neighboring check nodes
- If it receives from more than one nodes, the results are added up and the addition is sent to the rest but the incoming edges. Additionally, exchanging the incoming messages from the initial receiving nodes.

After two iterations, if a message node receives a copy of the message it had sent at the initialization of the first iteration, it announces that bit as a potential candidate for cycle-4. At the final stage, every message node where the algorithm was initialized is checked through the neighboring check node it is connected. For example, from Figure 4.6 (IV) we can see that, the node v_i receiving a message containing two ONES which implies a copy of the message from this node has been made and travelled though the cycle. So, we decide this bit as a potential candidate for cycle in the transmission path. The existing algorithm in [9] counts total number of cycle running through each node but in this thesis, we have modified the algorithm only to find out the associability with cycle-4 rather total number. Once the algorithm calculates for one node and if it does not find any associability with cycle, the node is deactivated. After that the algorithm proceeds for the next node. Additionally, we have considered only the calculation of associability with cycle-4 rather counting total number of cycle running through each node. By doing so, additional calculation of summing up number of all cycles has been avoided. Moreover, after one node has been deactivated after finding non-associability, it reduces the overall complexity of the algorithm since the number of edges reduces as the counting is proceed from one part to another of the graph.

4.3.1 Complexity of the algorithm

In the following section, complexity of the algorithm has been computed according to [9] and it has been explicitly compared with the existing literature to show optimality. For every node that is actively taking part in the calculation, it is needed to have $(2d_c - 1)d_c$ integer addition and subtraction for incoming messages and $(2d_c - 1)d_v$ integer subtraction is needed for outgoing messages. Since the algorithm performs for *g-1* iterations, the computational complexity for the algorithm is $O(g|E|^2)$, which is not added to the decoding complexity as the algorithm is only applied once advance and does not have any effect on decoding. On the other hand, the algorithm explained in [14] have computational complexity of $O(gn^3)$ where n= maximum (|V|, |C|). So, for the sparse graph, with |E| growing slower than $n^{3/2}$,the complexity of the algorithm is less than that of the proposed algorithm [9].

4.3.2 Removal of Cycle-4 Transmission Path in LDPC code

As discussed above, this is clear that cycle-4 in BEC leaves no room for the decoder to be able to decode the erased bits. As a result, this becomes the unreliable transmission path within the channel where the probability of decoding failure is higher. This section explains the gradational process for detecting and deducting unreliable transmission path in LDPC code.

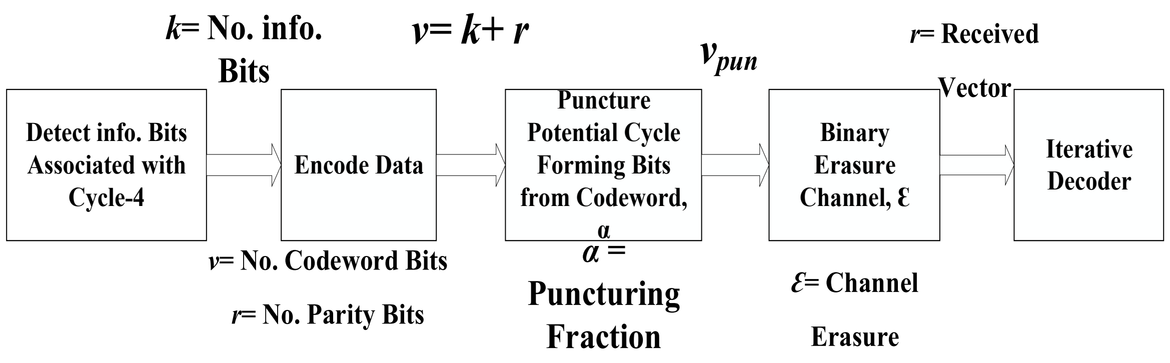


Figure 4.7 Methodology for Removal of Cycle

At this method, when the transmission is started the parity check matrix is examined for the potential information bits that might be associated with cycle-4. The encoder has the knowledge of the positions of the punctured bits. Then the data is encoded and the potential positions from the subset of codewords are punctured before the unpunctured bits are sent through the channel. In this method it is also assumed that the decoder also knows the position of the punctured bits and if needed this information can be used later. Transmitted code ensemble generated by this proposed method is asymptotically good to the mother code ensemble.

4.4 Rate and Threshold of Punctured Code Ensemble in BEC

In this section we consider the puncturing of bits following a predetermined pattern and investigate about the threshold of such code ensemble in BEC. After explaining the threshold of punctured code and puncturing range for code ensemble we focus on the versatility and optimality of the scheme discussed above. Then finally we focus on the decoding threshold for iterative decoding in BP of punctured LDPC code.

4.4.1 Puncturing Linear Codes and Code Rates

A linear code is punctured by removing a set of code word bits from the parity check matrix, it has its effect on reducing the codewords from n to n-p (p= removed columns). We define the puncturing fraction as β= p/n [21]. After deducting the unreliable cycle associated bits, the resulting transmission rate is described in [21] as-

$$R\,(\beta) = \frac{R}{1-\beta}\ , \quad \beta \in [0, 1) \tag{4.3}$$

Where *R (0)* is the rate of the mother code (unpunctured) code. The dimension of the code is preserved so flexible rate *R (β)* also could be achieved by deducting out the cycles from the transmission path.

4.4.2 Threshold of Punctured Codes

Now we concentrate our focus on the threshold of the punctured codes under any iterative decoding in BEC with erasure probability, Ɛ. Suppose we puncture a subset of codeword bit with length, *n* and a fraction of $\beta = p/n$ are punctured. The transmitted codeword is v^{pun} with a code length $n^{pun} = (1-\beta).n.$ After transmission the codeword will have on average $\mathcal{E}.n^{pun}$ erasure symbols and $(1-\mathcal{E})\ n^{pun}$ correct symbols [21]. Authors in [21]have described the puncturing threshold regardless of puncturing pattern by *Lemma 3*.

Lemma 3: *Considering any linear code ensemble whose BP decoding threshold can be computed by means of density evolution. Assume any fixed or random pattern has been used for puncturing. Then, on the BEC the BP decoding threshold will be same for puncturing strategies* [21]. *For this consideration the channel erasure probability is -*

$$\begin{aligned} \epsilon' &= \beta + (1-\beta).\epsilon \\ &= 1-(1-\epsilon).(1-\beta) \end{aligned} \tag{4.4}$$

Theorem 1 [20] describes the decoding threshold under any iterative message passing algorithm.

Theorem 1*:* The belief propagation threshold ϵ_{BP} (β) of a punctured LDPC code ensemble on BEC with puncturing fraction β is given by

$$\epsilon_{BP}(\beta) = 1-\frac{1-\epsilon}{R}.R(\beta) \tag{4.5}$$

Here Ɛ and R are the BP threshold and design rate of the punctured code respectively. *R(β)* is denoted as the achieved rate or changed rate after removing unreliable transmission path.

4.5 Simulation Results

For Simulation, regular LDPC mother code have been considered with the following theoretical specifications.

- Code length, n= 1000 bits
- Rate, R= 0.5
- Threshold = 0.429
- Degree distribution $(\lambda, \rho) = (3,6)$

The message passing algorithm for detecting short cycles calculates 0.007 % bits of each block of a packet containing 1000 blocks might be associated with cycles if those bits are erased in BEC during transmission. Therefore, the potential cycle forming bits have been punctured as a safeguard over the codeword yielding the transmission rate and threshold respectively as-

- Transmission rate, $R_{Tr} = 0.50$
- Transmitted code threshold, $\mathcal{E}_{BP}$ =0.42

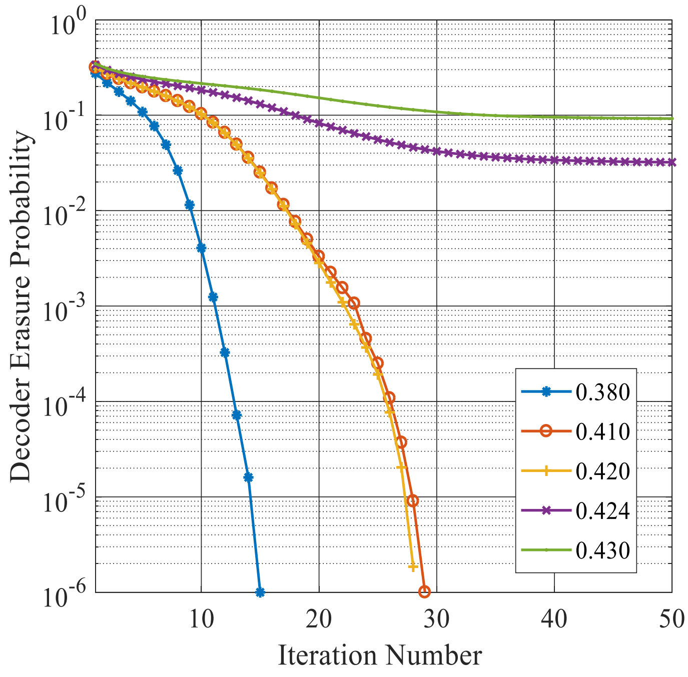


Figure 4.8 Decoder erasure probability for proposed scheme at different threshold

Figure 4.8 presents the decoder erasure probability after decoding for different level of channel erasure for a block of 10^3 bits. The figure above shows the decoder erasure probability gets below 10^{-5} after the code has been modified with the proposed method. Another important observation is that the effect of reduced decoder error is seen at higher erasure probability near threshold where the possibility of BP getting stuck is higher. For channel threshold 0.42, the decoder error probability goes close to 10^{-6}. Subsequently, at 0.429 channel erasure probability, the decoder error probability starts getting higher and after 40 iterations, it starts showing error floor as it is beyond the threshold of this code. Similar result is observed for any channel erasure beyond decoder threshold.

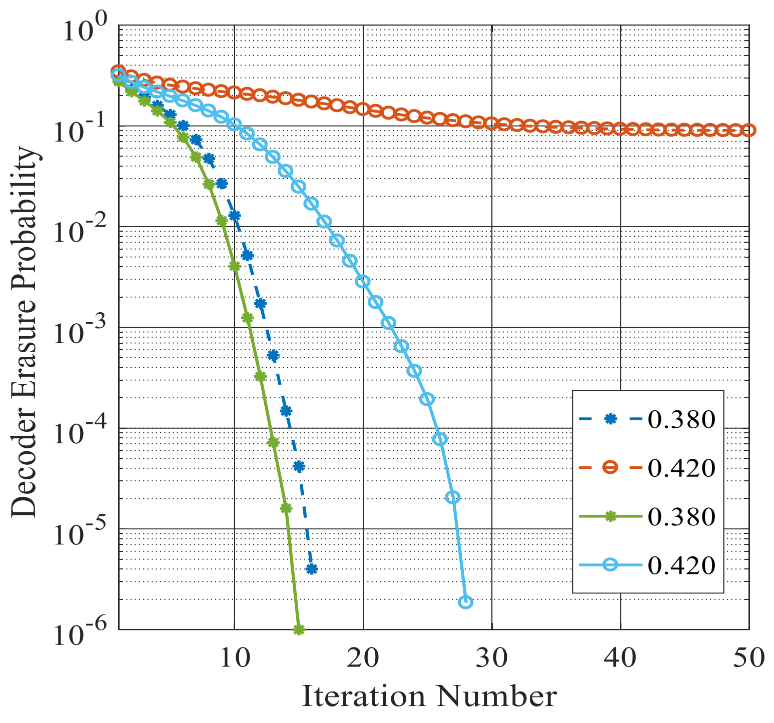


Figure 4.9 Comparison of the proposed scheme to codes with cycles

In figure 4.9, another simulation has been done to compare the proposed scheme with the existing conventional BP algorithm. The dashed line shows a random code with cyle-4 and the solid line shows a code whose short cycles have been taken out. The code that has been generated with the proposed scheme, has threshold 0.42 that is asymptotically close to the mother code. For conventional BP at erasure probability 0.42, the decoder cannot lower the bit erasure rate (BER) is 10^{-3} even the iteration number is increased. On the other hand, subsequently puncturing out cycles gives a bit erasure rate close to 10^{-6} at the threshold of the generated code at channel erasure 0.42 which validates the feasibility of the scheme.

4.6 Conclusion

Message passing algorithm was first introduced by H. Xiao et.el in [35]based on cycle and closed-walk distributions of the Tanner which eventually showed improvement in error correcting capability of the codes. Message passing algorithm for counting short cycles was developed in [9]by Mehdi Karimi et.el in 2013. Authors have mentioned in the paper about the problem of overlapping counting of cycles. In order to avoid the problem, the algorithm in this chapter have been developed, considers the fact that a single node has been neutralized after the algorithm has been applied to it. The algorithm is applied once offline before the transmission; as a result, it does not add additional complexity to the decoder.

5 Designing Rate Compatible Codes

LDPC code belongs to the most powerful channel coding techniques known today and have a broad range of applications. In modern wireless communication system, it is required to be able to adjust the rate of the code of the channel coding scheme to allow for a flexible strength of error protection for different services and to be able to adapt to the varying quality of the channel. Many systems in use these (e.g. WiMAX, WLAN) days require separate codes for varying code rates. In this chapter, rate compatible code generation methods have been discussed. For generation of rate compatible codes puncturing methods have been used with an intentional and random manner. For intentional puncturing to generate rate compatible codes, optimized bit position has been chosen which are part of cycle. In section 5.1, the puncturing technique of unreliable transmission path has been introduced. Afterwards section 5.2, discusses about the model for generating the codes based on the proposed scheme. In section 5.4, a method has been proposed to generate a rate compatible code by using a half rate regular mother code. Numerical calculation and simulation results have been included to develop clear idea about the proposed method. At the end, in section 5.5 Conclusion have been added about the essentials of puncturing method.

5.1 Puncturing for Generating Rate Compatible LDPC Code

The unreliable transmission path created by cycle-4 of LDPC codes in BEC have been punctured out to enhance the performance. This method can also be used to generate rate compatible LDPC code that approaches Shannon limit [21].

In chapter 4, equation 4.5 explicitly describes the BP threshold of the punctured code with puncturing fraction, (β). This BP threshold is expressed as a function of the target rate of R (β) ≥ R(0), by which we mean, the function ε_{BP} (β) depends only on the threshold $\varepsilon(0)$ and R of the mother code ensemble [21]. From this observation we can find out the highest achievable rate for an ensemble of LDPC code. Authors in [21] have described the highest achievable rate by the parameter, Θ

$$\Theta = \frac{1-\epsilon}{R} \geq 1 \tag{5.1}$$

Here equality holds if and only if the threshold of the mother code ensemble is equal to Shannon limit [21] and it follows the largest possible rate with non-negative threshold ϵ_{BP} (β) is given by the following formula-

$$R_{max} = R(\beta = \epsilon_{BP}(0)) = \frac{1}{\theta} \tag{5.2}$$

Here R_{max} is the maximum achievable rate.

At this point we can define the range of puncturing for achievable rate with non-negative BP threshold ε_{BP} of the mother code. The rate of the punctured code must be kept in the boundary as $R(0) \leq R(\beta) \leq R_{max}$ as maximum puncturing fraction with a non-negative BP threshold is equal to the BP threshold of the mother code [21].

Now we relate the parameter Θ to the maximum achievable rate as the gap to capacity for ensemble of punctured code. This has been well explained in [21]by the authors by *Corollary 1*.

Corollary 1*:* For BP decoding on the BEC, the gap to capacity $\Delta_{sh}(\beta)$ with randomly punctured LDPC code ensemble with puncturing fraction β and a given achievable rate $R(\beta)$ is

$$\Delta_{sh}(\beta) = \epsilon_{sh}(R(\beta)) - \epsilon(\beta) = (\theta - 1)R(\beta) \tag{5.3}$$

Here $\epsilon_{sh}(R(\beta)) = 1-R(\beta)$, is the Shannon limit for the randomly punctured code ensemble with a desired rate. A large value of Θ implies that mother code has a decoding threshold far from the Shannon capacity and the gap to capacity will be on the increasing line if the puncturing fraction is increased. On the other hand, for the Θ value close to 1, the mother code ensemble has a threshold close to Shannon limit and the gap to achieving capacity will grow slowly with increasing puncturing fraction. There might be some extreme cases where Θ=1 which implies threshold of the mother code ensemble is exactly equal to the Shannon limit and for any punctured rate $R(\beta) \geq R(0)$, capacity is achieved.

5.1.1 Numerical Analysis

In this section we calculate different values of Θ to understand the significance of the puncturing fraction and its effect on decoding threshold in BEC.

TABLE 5.1

Puncturing Fraction, *β*	**Rate, R(*β*)**	**ε_{BP} (β)**	**Δ_{sh} (β)**
0	0.5	0.4290	0.0710
0.10	0.5656	0.3541	0.1651
0.20	0.6250	0.2863	0.2672
0.30	0.7143	0.1257	0.4511
0.40	0.8333	0.0484	0.7527

Table 5.1 Threshold and corresponding gaps to capacity for punctured (3, 6) LDPC code

TABLE 5.2

Ensemble	**ϴ**	**Maximum Achievable Rate, R_{max}**
(3,6)	1.1420	0.87
(4,8)	1.2236	0.81
(5,10)	1.2926	0.77

Table 5.2 Values of ϴ for Different Ensemble Regular LDPC code

In Table 5.1 Threshold and corresponding gaps to capacity for punctured (3, 6) LDPC code, rate and threshold values have been calculated and compared for regular LDPC codes with different puncturing fraction. Regular LDPC ensemble (3, 6) has BP threshold of $\epsilon_{BP}(0) = 0.429$ and design rate R (0) =0.5, which results in ϴ=1.1420. Since, ϵ_{BP} $(0) = 0.429$ which is relatively close to capacity $\epsilon_{Sh} = 0.5$ that implies smaller value ϴ. From the value of ϴ, maximum achievable rate can be obtained using equation. Additionally, this is also visible that, the mother code has lowest gap to achieve capacity. Again, it is very vital to mention that the gap to capacity for a given target rate does not depend on the amount of puncturing rather it depends on the value of Θ obtained for the mother code ensemble.

In Table 5.2 Values of Θ for Different Ensemble Regular LDPC code shows the numerically calculated values of Θ for three different ensembles of LDPC code and maximum achievable rates caused by puncturing. The best value of Θ is obtained for (3, 6) code; therefore, for all achievable rate $R\ (\beta) \geq 0.5$, (3,6) is the best possible choice. On the other hand, lager value of Θ implies that the threshold of the mother code is far away from Shannon capacity and this gap will increase with increasing amount of puncturing fraction. In extreme case, if the value of Θ =1, then the mother code reaches capacity for all punctured code, $R(\beta)\ \geq R$.

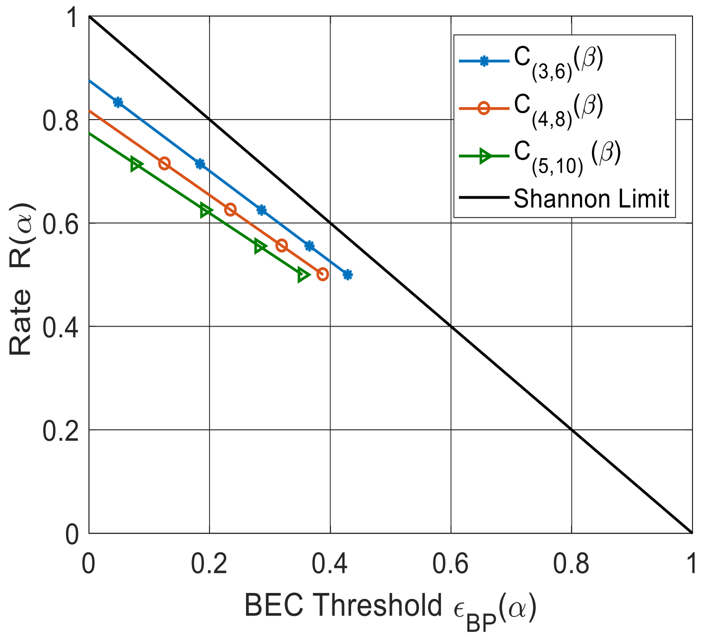


Figure 5.1 BEC BP threshold for three different ensemble codes with different puncturing fraction

Figure 5.1 shows numerically calculated BP threshold of punctured codes according to the proposed method with ensembles (3, 6), (4, 8) and (5, 10) for a variety of punctured fraction. The corresponding values of Θ have been mentioned in Table 5.2 Values of Θ for Different Ensemble Regular LDPC code. From $\epsilon_{BP}(\beta) = 1 - \theta . R(\beta)$, it is observable that Θ can be interpreted graphically as the slope of the parametrically defined line that determines the position of the threshold caused by puncturing.

5.1.2 Proposed Method to Generate Rate Compatible, Capacity Achieving LDPC Code Using Rate 0.5 Rate Mother Code

In this a method has been proposed to generate rate compatible LDPC code using (3,6) regular mother code with rate, *R*=0.5. One thing which has been mentioned in [20]is the method of splitting by which puncturing is done in a way so that we can have the same rate as the mother code. However, in this thesis, a way of generating higher rate code from a lower rate mother code have been studied and optimality have been investigated by doing simulations for (3,6) regular LDPC code over BEC.

Strategy: Let us consider first a LDPC code ensemble with degree distribution (λ, ρ) and a code length, *n.* Length of the encoded information bit sequence is v and the channel erasure probability are, ϵ respectively. It is acknowledged that the code has short cycles and it degrades the performance. At first, the bit position is found out that might be associated with cycle-4 when transmission is made through binary erasure channel. We assume –

β_1 = Punctured fraction associated with cycle-4= p_1/n

β_2 = Punctured fraction to achieve desired rate, $R\ (\beta) = p_2/n$

The resulting transmission rate is, $R\ (\beta) = \frac{R(0)}{1-(\beta 1+\beta 2)}\ ,\ \beta \in [0,\, 1)$ (5.4)

5.2 Simulation Results

From the discussion in 5.1.1, it can be concluded that, among all the regular code ensembles introduced in Figure 5.1, regular (3,6) code is most feasible for generating rate compatible codes since the decreasing gap to achieve Shannon limit. For simulation, a code with 1000 bits have been chosen with rate 0.5 and a packet of 1000 blocks have been considered to investigate about the achievable decoder packet erasure probability.

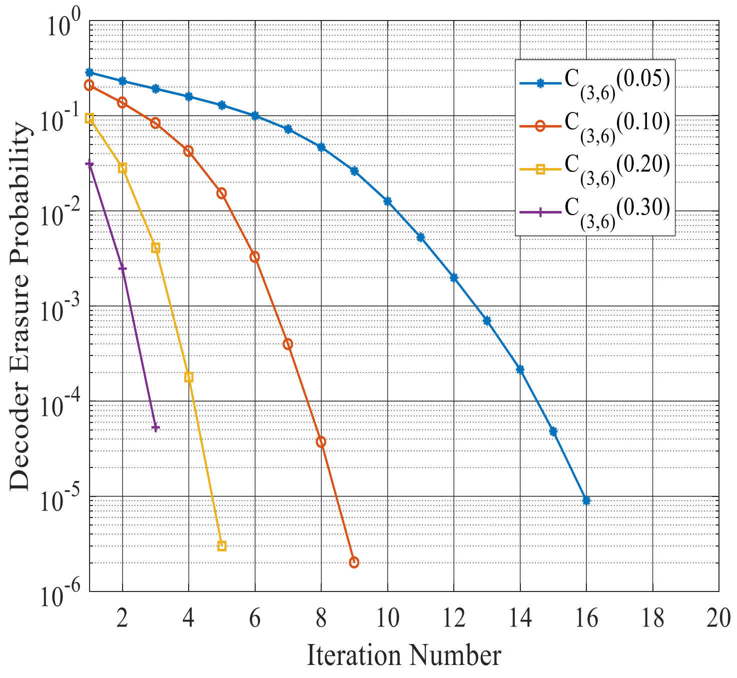


Figure 5.2 Performance comparison of regular (3,6) LDPC code with various level of puncturing

From the code ensemble 5%, 10% ,20% and 30% that generates code with following specification

- 5% Puncturing: $\mathrm{R}(\beta) = 0.52$, $\epsilon_{\mathrm{BP}}(\beta) = 0.4000$
- 10% Puncturing: $R(\beta) = 0.56$, $\epsilon_{BP}(\beta) = 0.3600$
- 20% Puncturing: $R(\beta) = 0.625$, $\epsilon_{BP}(\beta) = 0.2980$
- 30% Puncturing: $R(\beta) = 0.71$, $\epsilon_{BP}(\beta) = 0.1900$

For 5% puncturing, the code threshold is 0.40 which implies if 40 percent of the bits are erased in the channel, one simple iterative decoder should be able to recover that making erasure probabilities below 10^{-3}. From Figure 5.2, for channel erasure probability of 40%, the decoder solves all the erased with only 16 iterations which is low comparing to a code with cycles. The similar result is observed for three other puncturing fractions as well. For each case, the decoder solves all the erased bit with fewer number of iterations which validates the performance of the scheme.

Another Simulation has been done to compare the proposed method with an existing code without puncturing which is shown in Figure 5.3.

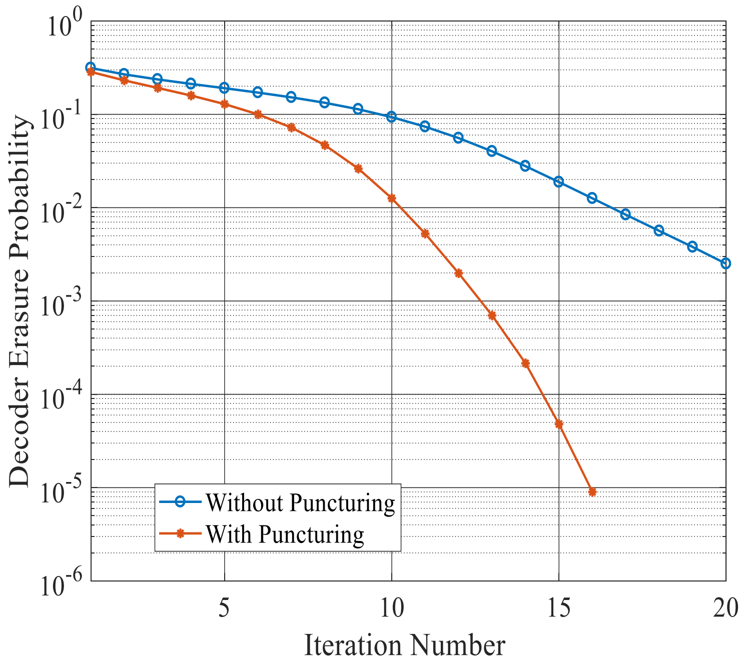


Figure 5.3 Comparison of code without puncturing and with puncturing at channel erasure probability 0.40

Figure 5.3 shows the comparison of two codes without puncturing and with puncturing at channel erasure probability 0.41 and rate of the code is 0.51. For the proposed scheme with puncturing out the short cycle forming bits the decoder shows a BER less than 10^{-5} for 16 iterations. Whereas, the code performing at same rate and threshold but without puncturing, cannot lower the BER even below 10^{-3} even if the iteration numbers are increased to 20. This simulation result shows the feasibility of the applied method to generate rate compatible codes. This rate-compatible code generation method can be applied to the systems with changing channel conditions and example of such systems are hybrid ARQ-FEC systems.

5.3 Conclusion

Puncturing is the method which is referred to taking out a set of variable nodes from the codeword to generate rate compatible codes. In this chapter puncturing has been done on the cycle forming bits in the parity part and the message have been left intentionally to keep the information bits intact. An accurate expression for predicting the BP threshold for the punctured code has been explained. The schemes have been used to design the code is based on [20].

6 Conclusion and Future Work

This thesis has the objective of addressing the problem of presence of cycle and stopping set formation in the parity check matrix of LDPC Codes over BEC. Since the random property of LDPC code, there is a presence of cycle in the parity check matrix that leads to decoding failure over BEC. Modification of an algorithm have been proposed which is run offline one time at the encoding side of the transmission system to find out the exact position that might be associated with cycles. For the purpose of the thesis, a design rate 0.5, with regular ensemble LDPC code have been chosen which supports efficient encoding using the methods explained [22]. Since no structures or particularities have been followed to generate the code, the parity check matrix has short cycles. To investigate the decoding performance of this 0.5 rate regular code, BP algorithm has been employed with message passing algorithm initially for 50 iterations. From the simulation results, it was visible that the decoder cannot reach the threshold at 42.9% and the probability of error was not below 10^{-3}, before reaching to the threshold. According to theoretical analysis of threshold, for (3, 6) LDPC code the decoder must have the capability of correcting all the erasures up to 42.9 % of the information bits. This decoding fall happens due to the presence of short cycles in the parity check matrix which creates dependency in the marginal probabilities passed by the decoder. The algorithm [9] which has been proposed is capable of finding out the exact bit position in the parity check matrix. The exact cycle forming bits have been punctured out from the codeword to generate capacity achieving codes over BEC making the BER less than 10^{-5} at channel erasure 0.42. Chapter 4 includes the simulation result for this proposed method over BEC which shows that for same number of iterations, the decoder this time is able to recover all the codewords. For this thesis, BEC has been chosen as it is seen as the channel model for real world problem like video broadcasting in noisy environment, space communication in the presence of propagation delay etc. Finally, Chapter 5, proposes a method to generate rate compatible LDPC code using a rate 0.5 regular mother code using puncturing method. To do so, 5%, 10%, 20% and 30% information bits have been punctured out to generate code with higher rate and threshold have been calculated. For each puncturing fraction the information is kept intact, and the BER goes below 10^{-5} for corresponding channel threshold. This time, the decoder is seen to be able to correct all the erasures with maximum of 15 iterations.

Possible future work is summarized below:

- Analysis the effect of cycle-4 over BSC to enhance code performance.
- Study the effect of short cycles on irregular LDPC code and its decoding threshold.
- Choosing an optimized parity bits pattern in irregular codes for puncturing out cycles with a purpose to generate rate compatible codes.
- Finally, to investigate the process where threshold can be kept same as the mother code by inserting new column after removing the columns that have cycles.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. July 1928, pp. 379–423, 1948.

[2] Shu Lin, Daniel J.Costello,"Error Control Coding" Second Edition .

[3] C. Members, "New Decoding Methods for LDPC Codes on Error and Error-Erasure Channels," *Technology*, 2010.

[4] A. Shokrollahi, "LDPC Codes: An Introduction," *Coding, Cryptogr. Comb.*, no. December 2002, pp. 85–110, 2004.

[5] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, 1996.

[6] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[7] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[8] L. P. Codes, T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of Capacity-Approaching Irregular," vol. 47, no. 2, pp. 619–637, 2001.

[9] M. Karimi and A. H. Banihashemi, "Message-passing algorithms for counting short cycles in a graph," *IEEE Trans. Commun.*, vol. 61, no. 2, pp. 485–495, 2013.

[10] J. Fan and Y. Xiao, "A method of counting the number of cycles in LDPC codes," *Int. Conf. Signal Process. Proceedings, ICSP*, vol. 3, pp. 4–7, 2007.

[11] H. Dehghani, M. Ahmadi, M. Alikhani and R. Hasni, "Calculation of Girth of Tanner Graph in LDPC Codes," *Trends in Applid Sciences Research*, vol. 7, no. 11. pp. 929–934, 2012.

[12] B. Li, G. Wang, and H. J. Yang, "A new method of detecting cycles in tanner graph of LDPC codes," *2009 Int. Conf. Wirel. Commun. Signal Process. WCSP 2009*, no. 1, pp. 9–11, 2009.

[13] Y. M. Y. Mao and a. H. Banihashemi, "A heuristic search for good low-density parity-check codes at short\nblock lengths," *ICC 2001. IEEE Int. Conf. Commun. Conf. Rec. (Cat. No.01CH37240)*, vol. 1, pp. 41–44, 2001.

[14] T. R. Halford and K. M. Chugg, "An algorithm for counting short cycles in bipartite graphs," *IEEE Trans. Inf. Theory*, vol. 52, no. 1, pp. 287–292, 2006.

[15] J. Camilo, S. Ripoll, and R. Barraza, "A new Algorithm to construct LDPC codes with large stopping sets," pp. 516–518, 2013.

[16] M. R. Yazdani, S. Hemati, and A. H. Banihashemi, “Improving Belief Propagation on Graphs with Cycles,” *IEEE Commun. Lett.*, vol. 8, no. 1, pp. 57–59, 2004.

[17] H. Pishro-nik, S. Member, F. Fekri, and S. Member, “On Decoding of Low-Density Parity-Check Codes Over the Binary Erasure Channel,” vol. 50, no. 3, pp. 439–454, 2004.

[18] M. Beermann and P. Vary, “Breaking Cycles with Dummy Bits: Improved Rate-Compatible LDPC Codes with Short Block Lengths,” *ITG-Fachbericht-SCC 2013*, vol. 9, 2013.

[19] M. Beermann, T. Breddermann, and P. Vary, “Rate-compatible LDPC codes using optimized dummy bit insertion,” *Proc. Int. Symp. Wirel. Commun. Syst.*, pp. 447–451, 2011.

[20] H. Pishro-Nik and F. Fekri, “Results on punctured low-density parity-check codes and improved iterative decoding techniques,” *IEEE Trans. Inf. Theory*, vol. 53, no. 2, pp. 599–614, 2007.

[21] D. G. M. Mitchell, M. Lentmaier, A. E. Pusane, and D. J. Costello, “Randomly Punctured LDPC Codes,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 408–421, 2016.

[22] S. J. Johnson, “Introducing Low-Density Parity-Check Codes”, A Master Thesis, The University of Newcastle, 2010.

[23] T. J. Richardson, “Efficient Encoding of Low-Density Parity-Check Codes,” vol. 47, no. 2, pp. 638–656, 2001.

[24] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, “Practical loss-resilient codes,” *Proc. twenty-ninth Annu. ACM Symp. Theory Comput. - STOC ’97*, pp. 150–159, 1997.

[25] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, “Comparison of constructions of irregular Gallager codes,” *IEEE Trans. Commun.*, vol. 47, no. 10, pp. 1449–1454, 1999.

[26] 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, 2001.

[27] D. J. C. MacKay, “Good Codes Based on Very Sparse Matrices,” pp. 100–111, 1995.

[28] M. G. Luby, M. Mitzenmacher, and M. A. Shokrollahi, “Analysis of Random Processes via And-Or Tree Evaluation,” *9th Annu. ACM-SIAM Symp. Discret. Algorithms*, pp. 364–373, 1998.

[29] A. T. Ihler, J. W. F. Iii, and A. S. Willsky, “Loopy belief propagation: Convergence and effects of message errors,” *J. Mach. Learn. Res.*, vol. 6, no. 1, pp. 905–936, 2005.

[30] Y. Xiao and M. H. Lee, “Low complexity MIMO-LDPC CDMA systems over multipath channels,” *IEICE Trans. Commun.*, vol. E89–B, no. 5, pp. 1713–1717, 2006.

[31] N. Wiberg, H. -A Loeliger, and R. Kotter, “Codes and iterative decoding on general graphs,” *Eur. Trans. Telecommun.*, vol. 6, no. 5, pp. 513–525, 1995.

[32] G. Lechner, “The effect of cycles on binary message-passing decoding of LDPC codes,” *2010 Aust. Commun. Theory Work. AusCTW 2010*, no. 2, pp. 43–47, 2010.

[33] T. Etzion, “Which codes have cycle-free tanner graphs?,” *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2173–2181, 1999.

[34] T. M. Cover, J. a Thomas, J. Bellamy, R. L. Freeman, and J. Liebowitz, *Elements of Information Theory WILEY SERIES IN Expert System Applications to Telecommunications*. 1991.

[35] H. Xiao and A. H. Banihashemi, “Graph-based message-passing schedules for decoding LDPC codes,” *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2098–2105, 2004.