INVESTIGATIONS ON BINARY AND NON-BINARY LDPC CODES

A

Thesis submitted

for the award of the degree of

DOCTOR OF PHILOSOPHY

By

KUNTAL DEKA

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781 039, ASSAM, INDIA

JANUARY 2016

INVESTIGATIONS ON BINARY AND NON-BINARY LDPC CODES

Kuntal Deka

Certificate

This is to certify that the thesis entitled "**INVESTIGATIONS ON BINARY AND NON-BINARY LDPC CODES**", submitted by **Kuntal Deka** (08610205), a research scholar in the *Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati*, for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under our supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the Institute and in our opinion, has reached the standard needed for submission. The results embodied in this thesis have not been submitted elsewhere for a degree.

Dated:

IIT Guwahati

A. Rajesh
Associate Professor

P. K. Bora
Professor

Deptartment of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
Guwahati - 781 039, Assam, India.

To

My dear parents

for their love and support

Acknowledgements

First and foremost I am grateful for the guidance and advice of my supervisors Dr. A. Rajesh and Prof. P. K. Bora. Their constant motivation and support have been instrumental to the completion of this thesis. I would particularly like to thank them for patiently and carefully correcting all my manuscripts.

I am also very thankful to other members of my doctoral committee, Prof. Ratnajit Bhattacharjee, Dr. Kalpesh Kapoor, Dr. Pravas Ranjan Sahu and Dr. Tony Jacob for sparing their precious time to evaluate the progress of my work. Their suggestions and assistance have been valuable. I would also like to thank other faculty members of the Department for their support and constant encouragement during this course of study.

I am grateful to all the members of the research and technical staff of the Department. My special thanks to Mr. Sanjib Das and Mr. Utpal Sarma for maintaining an excellent computing facility and providing various resources useful for the research work.

Thanks are also addressed to all my friends of the Department. Specially I wish to thank Mridupawan, Ramesh Mishra, Om Prakash Singh, Sanjoy Mondal, Shyam Anand, G. Aruna, Rajib, Shivanshu, Dibya, Vinay and Sam Darshi for their help and encouragement. They have always been around to provide useful suggestions, companionship and created a peaceful research environment. A special mention must be made of Mr. Dhruva Kartik with whom I have had many fruitful and intriguing discussions. I am grateful to him for introducing me to the field of high performance computing.

I wish to thank Indian Institute of Technology Guwahati for providing a beautiful and serene residential campus. The author also acknowledges the financial assistance provided by the Ministry of Human Resource Development, Government of India.

Finally, I would like to thank my family for their love and encouragement.

Kuntal Deka

Abstract

The binary low-density parity-check (LDPC) codes of short-to-medium lengths usually suffer from the error floor problem while decoded by an iterative algorithm. This problem is largely due to certain kinds of subgraphs known as the trapping sets present in the Tanner graph of the code. In addition to the good error floor behavior, the codes should preferably exhibit rate-compatibility. Puncturing is a popular technique to realize rate-compatible codes. The non-binary LDPC codes are more flexible and robust in the context of puncturing than their binary equivalents. The thesis primarily deals with two broad problems related to the LDPC codes: (1) to study the various aspects of trapping sets in the case of binary LDPC codes and identify them, (2) to design an optimum puncturing pattern to achieve rate-compatibility in the case of non-binary LDPC codes.

The identification of the dominant trapping sets of a binary LDPC code is an important task. The knowledge of these dominant trapping sets helps to design a robust coding system. A technique is proposed to find the dominant trapping sets in irregular LDPC codes. The proposed technique is based on the hierarchical approach where a smaller trapping set known as the predecessor is progressively expanded to obtain larger successor trapping sets. The technique considers six types of predecessors to find the trapping sets of a particular class.

The dominant trapping sets generally contain one or more short cycles. The extrinsic message degree (EMD) of a cycle in the Tanner graph measures the connectivity of the cycle. As it is difficult to calculate the EMD, the approximate cycle EMD (ACE) is generally used as a simpler alternative. The ACE of a cycle, however, is not equal to its EMD when the cycle contains sub-cycles. We investigate the possibility of the formation of any sub-cycle within a cycle for the ACE spectrum constrained LDPC codes. This investigation leads to the derivation of three sufficient conditions for the equality of the ACE and the EMD of a cycle.

The non-binary LDPC codes are suitable for rate-compatible puncturing. They provide one extra degree of freedom: a codeword symbol may be punctured bitwise or symbolwise. An extrinsic information transfer (EXIT) chart tool for the puncturing of the non-binary LDPC codes is formulated. With the help of this tool, the thresholds for various bitwise and symbolwise puncturing patterns can be found out. The grouping algorithm is a useful technique to find the recoverable variable nodes in the Tanner graph of the short-length codes. A method is devised to find the optimum recoverable puncturing pattern by the joint use of the EXIT chart model and the grouping algorithm.

The effect of cycles is less in the case of non-binary LDPC codes. A cycle is harmful for the non-binary iterative decoding if it satisfies a particular condition based on the edge-labels. Such a cycle is called a non-binary cycle. The thesis carries out a probabilistic analysis on the formation of a non-binary cycle for a given field size. Moreover, a technique is proposed for the rate-compatible puncturing of the non-binary LDPC codes. The proposed technique considers the lengths and the EMD values of the short non-binary cycles present in the Tanner graph.

Simulation studies are carried out to assess the performance of the proposed techniques on standard binary and non-binary LDPC codes.

Contents

List of Figures

List of Tables

List of Acronyms

ACE	Approximate cycle EMD
AS	Absorbing Set
AWGN	Additive White Gaussian Noise
BEC	Binary Erasure Channel
BER	Bit Error Rate
BI-AWGN	Binary Input AWGN
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CN	Check Node
CND	Check Node Decoder
DE	Density Evolution
EMD	Extrinsic Message Degree
EMS	Extended Min-sum
EXIT	Extrinsic Information Transfer
FAS	Fully Absorbing Set
FER	Frame Error Rate
FFT	Fast Fourier Transform
GF	Galois Field
k-SR	k-step recoverable
LDPC	Low-density Parity-check
LLR	Log-likelihood Ratio
MAP	Maximum a Posteriori
MS	Min-sum
NB	Non-binary

NB LDPC	Non-binary LDPC
PEG	Progressive Edge-Growth
QSPA	q-ary SPA
RC	Rate-compatible
SER	Symbol Error Rate
SNR	Signal-to-noise Ratio
SPA	Sum-product Algorithm
TS	Trapping Set
VN	Variable Node
VND	Variable Node Decoder

List of Symbols

$\lambda(x)$	Degree distribution polynomial for the VNs
$\rho(x)$	Degree distribution polynomial for the CNs
σ_{ch}^2	Variance of the channel LLR
σ_n^2	Variance of the channel noise for an AWGN channel
(a,b) TS T	A TS T with $\lvert T\rvert = a$ and $\mathcal{O}(T) = b$
b_p	Number of punctured bits per VN
c_j	ith CN
C	A codeword symbol
C_{2i}	Cycle of length $2i$
$\mathcal{C}_i$	A class of cycles
$d_c^{\max}$	Maximum degree of a CN in the Tanner graph
$d_c^{\min}$	Minimum degree of a CN in the Tanner graph
d_{v_j}	Degree of the VN v_j
d_{c_i}	Degree of the CN c_i
$d_v^{\max}$	Maximum degree of a VN in the Tanner graph
E	Set of edges in the Tanner graph
E_b/N_0	Energy per information bit to noise power spectral density ratio
$\frac{E_b}{N_0}^*$	Threshold for an AWGN channel
E_s	Energy per modulated symbol
$\mathcal{E}(T)$	Even-degree CNs in the subgraph induced by the VNs of T
f_j^a	Channel *a posteriori* probability of the jth symbol x_j being a
g	Girth of the Tanner graph
$G_{\mathbf{H}}$	Tanner graph corresponding to a parity-check matrix $\mathbf{H}$

$\mathrm{GF}(q)$	Galois field of size q
$\mathrm{GF}(q)^{-}$	$\mathrm{GF}(q) \setminus \{0\}$
$\mathbf{G}_i$	Group of i-SR VNs
$\mathbf{G}_R$	Set of recoverable VNs, i.e. $\mathbf{G}_R = \bigcup_{i=1}^{K} \mathbf{G}_i$
h_{ij}	Element of $\mathbf{H}$ in the ith row and the jth column
H_q	Hadamard transform of order $q = 2^s$
$\mathbf{h}_j$	jth column of $\mathbf{H}$
$\mathbf{H}$	Parity-check matrix
I_m	Maximum number of iterations for decoding
$I(X;L)$	Mutual information between the bit value X corresponding to any VN and an extrinsic message L regarding that VN
$I_{A,\mathrm{VND}}$	*a priori* input information for the VND curve plotted along the abscissa of the EXIT chart
$I_{E,\mathrm{VND}}$	Extrinsic output information for the VND curve plotted along the ordinate of the EXIT chart
$I_{A,\mathrm{CND}}$	*a priori* input information for the CND curve plotted along the ordinate of the EXIT chart
$I_{E,\mathrm{CND}}$	Extrinsic output information for the CND curve plotted along the abscissa of the EXIT chart
$J(\sigma)$	$I(X;L)$ when $L \sim \mathcal{N}\left(\frac{\sigma^2}{2}, \sigma^2\right)$
$J_q(\cdot)$	$I(C;\mathbf{W})$, when $\mathbf{W}$ is any symmetric and permutation-invariant Gaussian distributed message for a code defined over $\mathrm{GF}(q)$
$J_V(\cdot)$	$I(C;\mathbf{V})$
$J_V^0(\cdot)$	Mutual information for a VN→CN message $\mathbf{V}$ of an unpunctured VN
$J_V^{b_p}(\cdot)$	Mutual information for a VN→CN message $\mathbf{V}$ of a VN containing b_p punctured bits
L_{ch_j}	Channel LLR for x_j
L_j	*a posteriori* LLR for x_j
$\mathbf{L}_{\mathrm{ch}_j}$	Channel LLR for x_j for the NB LDPC codes
M	Total number of CNs in the Tanner graph
$\mathbf{M}(i)$	Set of neighboring VNs of c_i

η_{2i}	ACE constraint on the cycles of length $2i$
$\boldsymbol{\eta}_{\text{ACE}}^{\text{exp}}$	Expanded ACE spectrum
$\boldsymbol{\eta}_{2i}^{\text{exp}}$	Component of expanded ACE spectrum corresponding to the cycles of length $2i$
$n_{(2i,j)}$	Component of $\boldsymbol{\eta}_{2i}^{\text{exp}}$ which denotes the number of VNs involved in the length-2i cycles with the minimal ACE values j
N	Total number of VNs in the Tanner graph or the block-length
N_0	Noise power spectral density for an AWGN channel
N_E	Total number of edges in the Tanner graph
N_l^p	Number of VNs required to be punctured to attain a rate r_l from r_0
N_{d_i}	Total numbers of degree-d_i VNs
$N_{d_i}^g$	Number of VNs of degree d_i in $\mathbf{G}_R$
N_{d_i,b_p}	Total number of degree-d_i VNs having b_p punctured bits
$\mathcal{N}(\mathcal{T})$	Set of CNs connected to the VNs in the TS T
$\mathcal{N}_{v_j}^l$	Neighborhood from v_j within depth l
$\mathbf{N}(j)$	Set of neighboring CNs of v_j
$\mathcal{O}(T)$	Odd-degree CNs in the subgraph induced by the VNs of T
p_v	Probability that a VN is punctured
$p_{\mathcal{N}(\tau)}$	Probability that a set $\mathcal{N}$ contains τ number of punctured VNs
P_{nb}^q	Probability of a cycle being a legitimate NB cycle
P_l	Set of the VNs selected for puncturing to achieve rate r_l
q	Field size
$\mathbf{q}_{ji}$	Message sent by v_j to c_i during FFT-QSPA
$\mathbf{q}_j$	*a posteriori* probability term for the jth symbol x_j
r	Rate of the code
r_0	Rate of the mother code
r_l	Rate of a punctured code
$r_{\max}$	Maximum rate of a punctured code according to the grouping algorithm
$\mathbf{r}_{ij}$	Message sent by c_i to v_j during FFT-QSPA
s	$q = 2^s$
S_T	Subgraph induced by the VNs in the TS T

t	Mean column weight
t_c	Cross-over point for puncturing of quasi-regular NB LDPC codes
t_{opt}	Optimum mean-column weight
T	A TS
$\mathcal{T}_a$	Set of TS classes of the form $(a, *)$
$\mathbf{T}$	Intermediate message used in the EXIT chart analysis
U	Set of the CNs in the Tanner graph, i.e., $U = \{c_i, i = 1, \ldots, M\}$
U_{ij}	Message sent by c_i to v_j during SPA
$\mathbf{U}_{ij}$	Message sent by c_i to v_j during LLR-domain QSPA
$\mathbf{U}$	A random variable corresponding to the CN→VN messages used in the EXIT chart analysis
v_j	jth variable node
V	Set of the VNs in the Tanner graph, i.e., $V = \{v_j, j = 1, \ldots, N\}$
V_{ji}	Message sent by v_j to c_i during SPA
$\mathbf{V}_{ji}$	Message sent by v_j to c_i during LLR-domain QSPA
$\mathbf{V}$	A random variable corresponding to the VN→CN messages used in the EXIT chart analysis
$\mathbb{Z}^+$	Set of integers greater than zero
x_j	Codeword bit/symbol value corresponding to v_j

1

Introduction

Contents

1.1 Background

The area of information and coding theory started with the remarkable and insightful work of Claude E. Shannon in 1948 [63]. Shannon's channel capacity theorem states that there exists a coding system with the help of which an arbitrarily small probability of error can be attained with any rate up to the capacity of the channel for a given noise level. For an additive white Gaussian noise (AWGN) channel, the channel capacity is represented in terms of the signal-to-noise ratio (SNR) E_b/N_0, where E_b is the energy per bit and $\sigma_n^2 = \frac{N_0}{2}$ is the variance of the channel noise. As an implication of the capacity theorem, reliable communication at a particular rate requires a minimum E_b/N_0 value. This minimum E_b/N_0 is considered as the *Shannon limit* for that particular rate. Although the Shannon theorem guarantees the existence of an error-free coding system at an SNR above the Shannon limit, its non-constructive proof did not reveal the process of designing the optimal codes. The scope of achieving the Shannon limit has triggered an intense research to design optimal coding systems. However, the coding community has experienced a slow progress since then as far as the achievement of the Shannon limit is concerned. In 1993, Berrou *et al.* have introduced turbo codes [11], which were the first practical codes to achieve near-Shannon-limit performance when decoded with a sub-optimal *iterative message-passing* algorithm. This discovery has revolutionized the apparently saturated field of coding theory. The idea of the iterative message-passing decoding has become a core ingredient of any modern coding system proposed thereafter. The outstanding success of the turbo codes has brought light to the *low-density parity-check* (LDPC) codes which were proposed by Gallager way back in 1962 [27]. The unique feature of the LDPC codes is the sparsity of the parity-check matrices. This feature provides the scope for employing iterative decoding algorithms. Mackay *et al.* have shown that long LDPC codes can yield near-Shannon-limit performance with the iterative decoding [49, 50]. This finding has instilled a lot of interest in the researchers since late 90s.

Regular and Irregular Codes

The iterative message-passing decoding algorithms for the LDPC codes operate over a bipartite graph known as *Tanner graph* which is a graphical representation of the parity-check matrix [66]. The Tanner graph comprises of the variable nodes (VNs), the check nodes (CNs) and the edges corresponding respectively to the columns, the rows and the 1s of the parity-check matrix. During iterative decoding, the VNs and the CNs propagate messages over the edges. Gallager investigated

the ensemble of *regular* codes where the degrees of the VNs and the CNs are fixed at the respective constant values. In [47], Luby *et al.* devised an improved class of codes namely the *irregular* codes where the degrees of the VNs and the CNs can vary widely. An ensemble of irregular codes is specified with the help of the degree-distributions for the nodes [47, 60].

Error Rate Curves

The performance of an LDPC code can be evaluated by means of the plot of the *bit error rate* (BER) or *frame error rate* (FER) versus the channel quality. The commonly used measure of the channel quality for an AWGN channel is the SNR E_b/N_0. Typical BER and FER curves for an LDPC code decoded by an iterative algorithm can be interpreted as shown in Figure 1.1. The shape of these

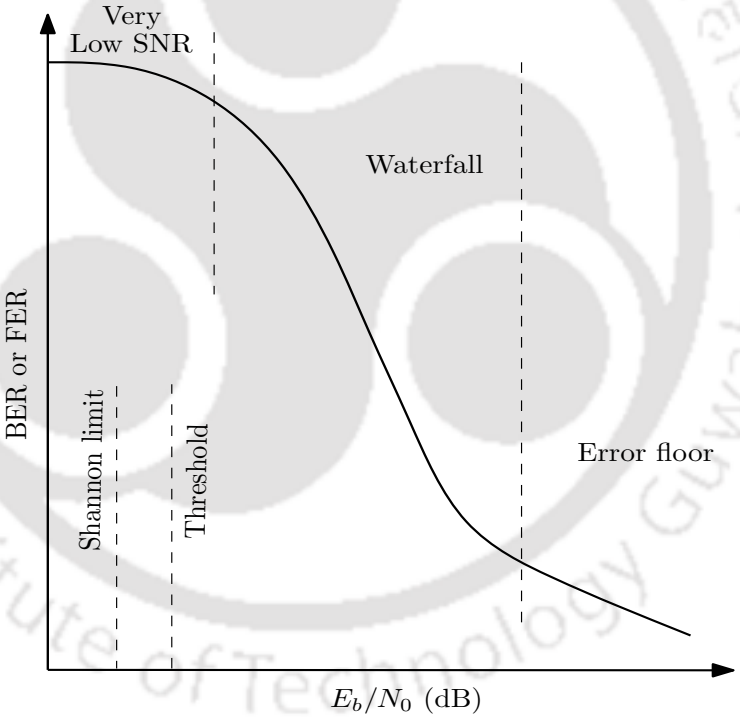


Figure 1.1: Typical BER or FER curve for the LDPC codes

curves can be divided into three regions: *very low SNR*, *waterfall* and *error floor*. Figure 1.1 also shows the typical locations of the Shannon limit and the threshold value (to be discussed shortly). In the very low SNR region, the error rates are too high for any meaningful application. In the waterfall region, the BER and the FER curves fall steeply as the SNR increases. The situation becomes different in the error floor region. In this region, the error rate curves do not fall as rapidly as in the waterfall region. The existence of a proper demarcation between the waterfall and the error

floor regions depends on various factors like the degree distributions and the length of the code, the channel, the kind of iterative decoder and its realization, etc.. The contrast of the slopes of the error rate curves in these two regions becomes more pronounced for irregular codes which exhibit excellent waterfall performance. The behavior of the iterative decoder is different in the waterfall and the error floor regions. The performance in the waterfall region is usually characterized and analyzed in an asymptotic condition as the blocklength tends to infinity. On the contrary, the error floor problem emerges from the finite-length effects of the code. The most challenging task for a code designer is to ensure the optimum performance in both the waterfall and the error floor regions.

Asymptotic Analysis

In the waterfall region, the LDPC codes exhibit a threshold phenomenon in the asymptotic sense. Reliable communication is possible only when the SNR is higher than the threshold. Richardson and Urbanke have proposed an algorithm known as the *density evolution* (DE) to asymptotically design and analyze an ensemble of LDPC codes [59, 60]. In DE, the evolution of the density of the messages in the iterative decoder is monitored in order to compute the threshold. The objective is to design capacity-approaching codes exhibiting threshold values close to the Shannon limit. A simpler alternative approach for the evaluation of the threshold is the *extrinsic information transfer* (EXIT) chart method [67,68]. It tracks the mutual information between the bit value and an extrinsic message regarding that bit in order to compute the threshold. The DE and the EXIT chart are the two popular tools to optimize LDPC codes in an asymptotic sense.

Finite-length Analysis

The design of the LDPC codes from an asymptotic viewpoint generally yields the optimum degree distributions for the nodes in the Tanner graph. However, in practice, the LDPC codes must be of finite-length. In that case, the interconnections between the VNs and the CNs in the Tanner graph become very crucial in addition to their degree distributions. The nodes in the Tanner graph may form short cycles . Because of the presence of the short cycles, the messages flowing between the nodes in the Tanner graph become correlated after a small number of iterations. Consequently, the performance of the iterative decoder gets degraded. The progressive edge-growth (PEG)[1] and the approximate cycle extrinsic message degree (ACE)[1] algorithms are two popular techniques to construct Tanner graphs

[1] discussed in detail in Section 2.5

containing less harmful short cycles. In the PEG algorithm, the edges are assigned between the VNs and the CNs in such a way that the length of the cycles are maximized. In the ACE algorithm, the connectivity of the cycles are considered in addition to the lengths while constructing the Tanner graph. However, even after a careful construction, the Tanner graph may contain harmful short cycles. The superposition of one or more short cycles may create some specific type of subgraphs in the Tanner graph which contributes to the occurrence of the error floor in the case of the short to medium-length LDPC codes. Depending on the channel and the approach of analysis, different authors have come up with different names for these subgraphs. For the binary erasure channel (BEC), the error floor is attributed to the *stopping sets* [22]. For the other channels like the binary symmetric channel (BSC) and the AWGN channel, the concept of the stopping set is replaced by that of the *trapping set*[2] (TS) [58]. Zhang *et al.* [80] have come up with the notion of a subset of the TSs known as *absorbing sets*[2] (AS). The ASs have been found to be the most detrimental. Nevertheless, from the point of view of the graph theoretic definition, all other terms may be classified as some special types of TSs.

Non-binary LDPC Codes

A class of LDPC codes less susceptible to the error floor problem is the non-binary (NB) LDPC codes. The NB LDPC codes, constructed over GF(q) ($q > 2$), were first studied by Davey and MacKay in 1998 [18]. They have shown that at short to moderate block-lengths, the NB LDPC codes can achieve higher coding gains than the equivalent binary LDPC codes. In addition to the superior finite-length performance, the NB LDPC codes have been found to exhibit thresholds close to the Shannon limit [10, 46]. These codes are naturally suitable for the communication systems involving higher order modulation schemes [46]. They are capable of correcting symbolwise errors. Therefore, they are particularly advantageous for the burst-noise channels where multiple consecutive bits get corrupted. The main disadvantage of the NB LDPC codes is their higher decoding complexity. During iterative decoding, the message passing through each edge is a vector of length equal to the field size q. The decoding complexity increases significantly as the field size increases.

Rate-compatible Codes

Rate-adaptability is an essential criterion of an error correcting code in addition to the good high SNR performance. In some communication systems, the channel condition may vary frequently

[2]discussed in detail in Section 3.2

with time. In such a situation, adhering to a fix-rate code results in unnecessary consumption of power. Whenever the channel improves, the data should be transmitted with fewer parity bits. Rate-compatible (RC) error correcting codes become useful in such situations. An RC code family is a set of codes with different rates where a higher-rate code is embedded into a lower-rate code. This fact, in turn, enables the coding system to conveniently adapt to the varying channel conditions. Usually, the family of RC codes are designed using a code-modification technique known as *puncturing*. The investigations of the RC puncturing schemes for the finite-length binary LDPC codes started with the work of Ha *et al.* [32]. They have introduced the concept of the recoverability of a punctured VN. A VN is said to be recovered if it receives a non-zero message from any of its neighboring CNs. They have proposed *grouping algorithm*[3] to identify the recoverable VNs. The class of NB LDPC codes is suitable for the RC schemes because of less vulnerability to the error floor problem. For the NB LDPC codes defined over a binary extension field, $\mathrm{GF}(2^s), s \in \mathbb{Z}^+ \setminus \{1\}$, the codeword symbols are transmitted in bit-level representations over a binary input (BI) AWGN channel. In such a condition, the VNs can be punctured either *symbolwise* or *bitwise* [43]. In a symbolwise scheme, all the bits corresponding to a VN are punctured. On the other hand, in a bitwise scheme, only a certain number of bits per VN are punctured. The feasibility of having different numbers of punctured bits per VN accommodates one extra design specification. This feature makes the NB-LDPC codes more flexible and robust to the puncturing operations than their binary counterparts.

1.2 Motivation and Objectives

The investigation broadly centers around the following two important areas of research:

1.2.1 Trapping Sets and Short Cycles in Binary LDPC Codes

The Tanner graph of a code may contain a large number of TSs. However, only a few of them contribute to the emergence of the error floor. These TSs are commonly known as the *dominant* TSs. For designing a coding system robust to the error floor problem, the preliminary task is to find the dominant TSs of the given LDPC code. A list of dominant TSs helps to design a better coding system in three ways:

(1) Using the knowledge of the dominant TSs, the parity-check matrix or the Tanner graph can be

[3] discussed in detail in Section 5.2.1

modified to avoid the presence of some of the dominant TSs in it [7, 37, 42, 53].

(2) Exploiting the knowledge of the dominant TSs, the decoding rules may be adapted to improve the performance in the high SNR region [33, 80].

(3) The list of the dominant TSs is useful for estimating the high SNR performance of the LDPC codes [13, 16, 25].

Thus devising an efficient technique to find the dominant TSs of a particular LDPC code is an important job. For the irregular codes, the dominant TSs of the same class may have various graphical structures. All these structures must be taken into account while finding the TSs of the particular class. The above task may be exhausting for certain irregular codes. This fact motivated us to devise an efficient method for finding the dominant TSs of an irregular LDPC code.

Almost each dominant TS is formed by the union of one or multiple short cycles [41, 79]. The length is an important parameter for a cycle. The *girth* is the length of the shortest cycle/cycles and a vital parameter of a code. In addition to the length, the connectivity of the cycle with the rest of the Tanner graph also plays a key role in ascertaining the harmfulness. The connectivity can be measured by the *extrinsic message degree*[4] (EMD) [70]. An alternative and approximate measure of the connectivity of a cycle is the *approximate cycle EMD*[4] (ACE) [70]. Because of simplicity, the ACE has been incorporated in code design methods such as the ACE spectrum constrained LDPC codes [72, 73]. Although the ACE is frequently used as a substitute, it may not be equal to the EMD. In that case, the ACE does not express the true connectivity of a cycle. This fact motivated us to investigate the conditions for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained codes.

1.2.2 Puncturing of Non-binary LDPC Codes

In [30,31], the authors proposed a technique to optimize puncturing distributions for the NB LDPC codes by calculating the thresholds via Monte-carlo based DE. For this optimization, the grouping algorithm was not taken into consideration. This fact motivated us to devise a method to determine optimum puncturing pattern compatible with the grouping algorithm. In order to accomplish this task, we propose an EXIT chart model in the context of puncturing. We plan to compute the thresholds

[4] discussed in detail in Section 2.5.2

for different puncturing patterns for the NB LDPC codes with the help of the EXIT chart. The puncturing pattern which yields the minimum threshold can be considered as the optimum one.

Next we consider the design of an RC puncturing scheme for the NB LDPC codes by taking into account the connectivity of the harmful short cycles. The effect of short cycles is less in the case of NB LDPC codes in comparison to the binary LDPC codes. The cycles which satisfy a particular condition [4] based on the non-binary edge labels, can create a problem to the iterative decoders. These cycles can be considered as the *non-binary (NB) cycles*. Some NB LDPC codes having higher degree VNs and defined over a smaller field may contain some detrimental NB cycles. For such codes, the VNs to be punctured can be judiciously selected by considering the EMD values of the NB cycles. The short NB cycles with low EMD are harmful and degrade the performance of iterative decoders. Puncturing of the VNs present in these cycles will make the cycles even more susceptible to noise. Consequently, the performance will degrade significantly. We propose to find a cycle-based rule for the selection of the punctured VNs in the case of NB LDPC codes.

With the above backgrounds and motivations, the objectives of the thesis may be stated as follows:

(a) Devise an efficient method of finding a list of dominant TSs for an irregular LDPC code. This list should be as exhaustive as possible. Investigate the connectivity of the short cycles for the ACE spectrum constrained LDPC codes.

(b) Formulate an EXIT chart model for the puncturing of NB LDPC codes. Find an optimum puncturing pattern using the EXIT chart model. This optimum puncturing pattern must be compatible with the grouping algorithm. Considering the harmful cycles, devise a selection rule for the punctured VNs.

1.3 Thesis Contribution

The major contributions of this research work can be summarized as follows:

(a) Proposed an improved technique to find the dominant TSs of an irregular LDPC code.

(b) Derived three sufficient conditions for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained LDPC codes.

(c) Proposed an EXIT chart model for the puncturing of NB-LDPC codes. Devised a method to obtain an optimum puncturing pattern with the selection of only recoverable VNs for puncturing.

(d) Proposed an improved RC puncturing technique for the NB-LDPC codes. This technique is based on the lengths and the EMD values of the harmful cycles present in the code.

1.4 Organization of the Thesis

The rest of the thesis is organized into six chapters. A brief outline of each of these chapters is as follows:

Chapter 2

This chapter describes the basic tools and terminologies regarding the binary and the NB-LDPC codes relevant to the works described in the subsequent chapters. First the graphical representation of a parity-check matrix in terms of the Tanner graph is described. Then we describe the sum-product algorithm (SPA) formulated in the log-likelihood ratio (LLR) domain. Two popular heuristic algorithms for constructing the parity-check matrices, namely the progressive edge-growth (PEG) algorithm [34] and the ACE algorithm [73], are described. These two algorithms construct parity-check matrices with an objective of reducing the number of short and harmful cycles. The error floor problem and the reason of its occurrence are discussed. The later part of the chapter discusses the NB-LDPC codes. A decoding algorithm for the NB-LDPC codes, namely the Fast Fourier Transform (FFT) based q-ary SPA (FFT-QSPA) is described.

Chapter 3

This chapter proposes a technique to find the dominant TSs of an irregular LDPC code. A brief review of different approaches to find the dominant TSs is presented. We adopt the hierarchical approach of finding the TSs. In this approach, the smaller TSs are progressively expanded to obtain the larger TSs. The proposed method for finding out the dominant TSs is presented next. The efficiency of the proposed technique over other similar techniques available in the literature is highlighted. The proposed technique is applied on several commonly considered irregular codes and their numerical results are presented.

Chapter 4

The chapter presents three sufficient conditions for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained LDPC codes. The equivalent TS representation of a cycle with equal ACE and EMD is described. We also present some simulation results to examine the effectiveness of the sufficient conditions.

Chapter 5

This chapter proposes an EXIT chart model to analyze different puncturing patterns for the NB-LDPC codes. First, we consider the case of random selection of the VNs and regular puncturing (the number of punctured bits per VN is constant). The thresholds for the punctured quasi-regular codes are computed using the proposed EXIT chart model. Next we extend the proposed model to irregular puncturing. In irregular puncturing, different number of bits are punctured for different VNs. The selection of the VNs to be punctured is done by the grouping algorithm. By considering the constraints imposed by the grouping algorithm, we propose a method to find the optimum recoverable puncturing pattern. Simulation results for different rates are provided.

Chapter 6

This chapter proposes an improved RC puncturing technique for the NB-LDPC codes. This technique is based on the lengths and the EMD values of the short NB cycles present in the NB-LDPC codes. The proposed criterion is to select those VNs which are involved in a lower number of short NB cycles with low EMD. Simulation results are shown for this technique.

Chapter 7

This chapter provides a summary of the contributions made in the thesis and some suggestions for future work.

2

Low-density Parity-check Codes

Contents

2.1 Introduction

The LDPC codes form a special class of linear block codes. The parity-check matrix of an LDPC code is sparse, i.e. the number 1s is very less than the number of 0s. The sparseness of the parity-check matrix widens the scope of employing the sub-optimal iterative decoding algorithms. Carefully designed long LDPC codes have been able to perform close to the Shannon limit when decoded with the iterative message-passing algorithms [14]. Davey and Mackay came up with the idea of considering the LDPC codes defined over a higher order field GF(q), $q > 2$ [18]. This class of codes, often called the non-binary (NB) LDPC codes, performs significantly better than the binary counterparts, specifically for short block-lengths.

This chapter reviews some of the popular construction and decoding algorithms for the LDPC codes. The rest of the chapter is organized as follows. Section 2.2 describes the graphical representation of the parity-check matrix of an LDPC code. Section 2.3 reviews the decoding algorithms usually employed for LDPC codes. In Section 2.4, the concept of the decoding threshold is explained. Two popular algorithms for the construction of short-length LDPC codes are presented in Section 2.5. In Section 2.6, we briefly describe the error floor problem arising in iterative decoding. Section 2.7 reviews the NB-LDPC codes and the decoding algorithm. This chapter is concluded in Section 2.8.

2.2 Graphical Representation of LDPC Codes

As the parity-check matrix $\mathbf{H} = [h_{ij}]$ of an LDPC code is sparse, it can be efficiently represented by a bipartite graph known as the Tanner graph [66]. The Tanner graph of an $M \times N$ parity-check matrix $\mathbf{H}$ can be denoted by $G_{\mathbf{H}} = (V \cup U, E)$. Here $V = \{v_j : j \in \{1, \ldots, N\}\}$ is the set of N nodes (known as *variable nodes* (VNs)) corresponding to the columns of $\mathbf{H}$, $U = \{c_i : i \in \{1, \ldots, M\}\}$ is the set of M nodes (known as *check nodes* (CNs)) corresponding to the rows of $\mathbf{H}$ and $E = \{(v_j, c_i) : h_{ij} = 1\}$ is the set of edges.

We illustrate the graphical representation of the LDPC codes with the following example.

Example 2.1. *Consider an LDPC code with the following* 6×12 *parity-check matrix:*

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{2.1}$$

The Tanner graph for this parity-check matrix is shown in Figure 2.1, where ◯ *and* □ *denote a VN and a CN respectively.*

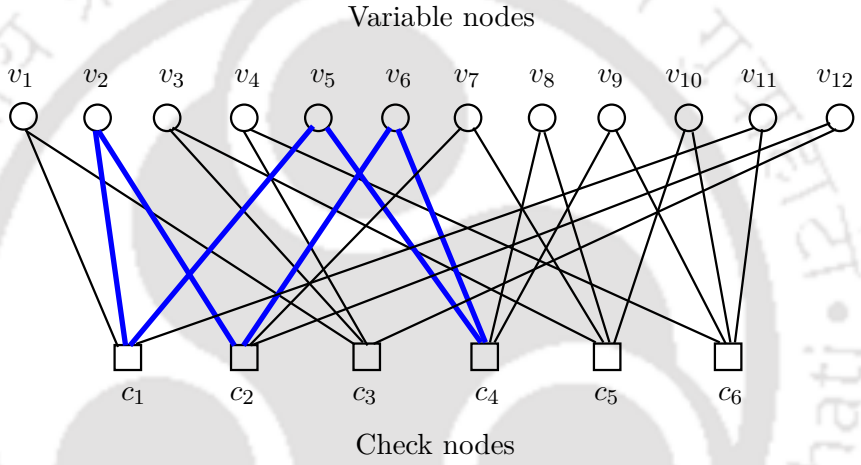


Figure 2.1: Tanner graph for the parity-check matrix shown in Example 2.1

The structure of the Tanner graph determines the performance of an LDPC code under iterative decoding. The Tanner graph can be characterized by various parameters. The most important specification is the degrees of the VNs and the CNs. The degree of a node is the number of edges emanating from that node. The degree of a VN v_j and a CN c_i can be denoted by d_{v_j} and d_{c_i} respectively. If $d_{v_j} = d_v, \forall j \in \{1, \ldots, N\}$ and $d_{c_i} = d_c, \forall i \in \{1, \ldots, M\}$, then that code is called a (d_v, d_c)-*regular* code. The LDPC code considered in Example 2.1 is a $(2, 4)$-regular code.

If either of the VNs or the CNs do not maintain regularity in the degrees, then the code is called an *irregular* code. In [47], it was shown that suitably designed irregular LDPC codes can surpass the performance of the regular codes of the same block-length. Irregular codes can be specified in a compact form with the help of the *degree distribution* polynomials. The degree distribution polynomial $\lambda(x)$ for the VNs is given by

$$\lambda(x) = \sum_{j=1}^{d_v^{\max}} \lambda_j x^{j-1}, \tag{2.2}$$

where λ_j is the fraction of edges coming out of the degree-j VNs and $d_v^{\max}$ is the maximum degree of the VNs present in Tanner graph.

Similarly, the degree distribution polynomial $\rho(x)$ for the CNs is given by

$$\rho(x) = \sum_{i=1}^{d_c^{\max}} \rho_i x^{i-1}, \tag{2.3}$$

where ρ_i is the fraction of edges coming out of the degree-i CNs and $d_c^{\max}$ is the maximum degree of the CNs in the Tanner graph. For a (d_v, d_c)-regular code, the degree distribution polynomials reduce to $\lambda(x) = x^{d_v-1}$ and $\rho(x) = x^{d_c-1}$.

The rate r of an LDPC code can be lower-bounded in terms of the degree distributions. Let $N_E = |E|$ be the number of edges in the Tanner graph. Then, we have

$$\begin{aligned} N =& N_E \sum_{j=1}^{d_v^{\max}} \frac{\lambda_j}{j} \\ =& N_E \int_0^1 \lambda(x)\, dx \\ \text{and } M =& N_E \sum_{i=1}^{d_c^{\max}} \frac{\rho_i}{i} \\ =& N_E \int_0^1 \rho(x)\, dx. \end{aligned}$$

The rate r is lower-bounded as

$$\begin{aligned} r \geq& 1 - \frac{M}{N} \\ =& 1 - \frac{\int_0^1 \rho(x)\, dx}{\int_0^1 \lambda(x)\, dx}. \end{aligned}$$

The degree distributions shown in (2.2) and (2.3) are defined from the *edge-perspective* as the coefficients denote some fractions of edges. The *node-perspective* degree distributions are denoted by $\hat{\lambda}(x) = \sum_{j=1}^{d_v^{\max}} \hat{\lambda}_j x^j$ and $\hat{\rho}(x) = \sum_{i=1}^{d_c^{\max}} \hat{\rho}_i x^i$, where $\hat{\lambda}_j$ is the fraction of degree-j VNs and $\hat{\rho}_i$ is the fraction of degree-i CNs. The node-perspective coefficients $\hat{\lambda}_j$ and $\hat{\rho}_i$ are related to their corresponding

edge-perspective coefficients by

$$\hat{\lambda}_j = \frac{\lambda_j / j}{\int_0^1 \lambda(x)\, dx} \qquad \text{and} \qquad \hat{\rho}_i = \frac{\rho_i / i}{\int_0^1 \rho(x)\, dx}.$$

An irregular LDPC code is demonstrated in the following example.

Example 2.2. *Consider an irregular LDPC code with the following* 6×12 *parity-check matrix:*

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

The Tanner graph for this parity-check matrix is shown in Figure 2.2. The parity-check matrix in this example contains a significant number of 1's and may not be considered as a sparse one. However, the intention behind this example is only to illustrate the degree-distribution polynomials for irregular codes. The edge-perspective degree distributions are given by

$$\lambda(x) = 0.4375x + 0.1875x^2 + 0.3750x^3$$
$$\textit{and} \quad \rho(x) = 0.6250x^4 + 0.3750x^5.$$

The node-perspective degree distributions are given by

$$\hat{\lambda}(x) = 0.5833x + 0.1667x^2 + 0.2500x^3$$
$$\textit{and} \quad \hat{\rho}(x) = 0.6667x^4 + 0.3333x^5.$$

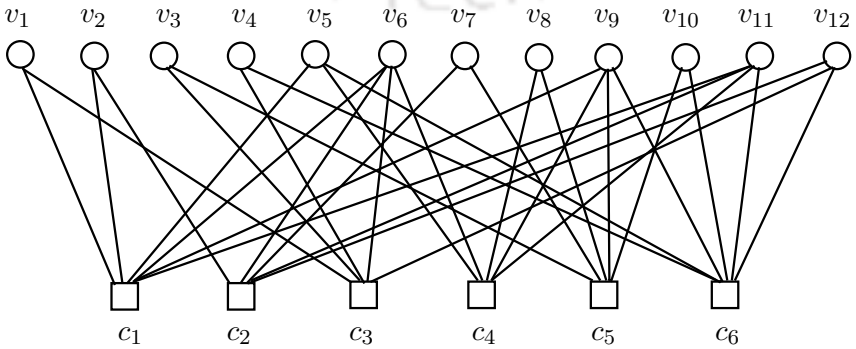


Figure 2.2: Tanner graph for the parity-check matrix shown in Example 2.2

A decisive structure of the Tanner graph is the cycle. A cycle is a closed path consisting of a

sequence of vertices starting and ending at the same vertex with no repetitions of vertices and edges allowed. The presence of short cycles in the Tanner graph significantly degrades the performance of iterative decoders specifically for short-length LDPC codes [34, 51, 70]. The length of the shortest cycle/cycles is known as the *girth* which partially reflects the severity of the performance degradation. In Section 2.5, we shall review some well-established strategies to construct a Tanner graph by avoiding harmful short cycles. As an example of cycle, consider the Tanner graph shown in Figure 2.1. A cycle of length 6 is shown in thick blue lines. The blue-colored 1s of $\mathbf{H}$ in (2.1) correspond to the edges present in the cycle. There is no cycle of length 4 in the Tanner graph. Thus the girth is 6.

2.3 Decoding of LDPC Codes

There is a variety of hard and soft-decision decoding algorithms for the LDPC codes in the literature [12, 61, 64]. The hard-decision decoding techniques like the majority-logic and the bit-flipping algorithms [28] have low computational complexity. The error correcting performances of these decoders are not satisfactory. The soft-decision decoding algorithms for the LDPC codes follow the message-passing principle of Pearl's belief propagation algorithm [55]. In such a decoding scheme, the VNs and the CNs in the Tanner graph iteratively exchange messages regarding the belief or the confidence level of the values taken by the bit variables. The soft-decision decoding algorithms offer much better error correction capability than the hard-decision algorithms. However, this improvement takes place at the expense of a higher decoding complexity. The most commonly used version of the soft-decision decoding techniques is the *sum-product algorithm* (SPA) defined in the log-likelihood ratio (LLR) domain [28, 49].

2.3.1 Sum-Product Algorithm

The underlying foundation for the SPA is the maximum *a posteriori* probability (MAP) rule [61]. The MAP decoding rule can be stated as

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \Pr\left(\mathbf{x}|\mathbf{y}, \underbrace{\text{all checks/syndromes are satisfied}}_{\mathbf{Hx=0}}\right), \tag{2.4}$$

where $\mathbf{x} = [x_1 \dots x_N]^T$ is a codeword, $\mathbf{y} = [y_1 \dots y_N]^T$ is the received sequence, $\hat{\mathbf{x}} = [\hat{x}_1 \dots \hat{x}_N]^T$ is the decoder output and $\Pr(\mathbf{x}|\mathbf{y})$ is the *a posteriori* probability.

The MAP decoding is performed block-wise. Although optimal, the MAP decoding is not feasible for the LDPC codes of practical lengths because of the enormous number of codewords involved. In the SPA, the decoding is carried out in a bitwise manner as shown below. For the bit x_j, the decoder output is given by

$$\hat{x}_j = \arg\max_{x_j} \Pr\left(x_j | \mathbf{y}, \text{ all checks involving } x_j \text{ are satisfied}\right). \tag{2.5}$$

As $x_j \in \{0, 1\}$, the decision about x_j can be made from the *a posteriori* LLR L_j defined by

$$L_j \triangleq \log \frac{\Pr\left(x_j = 0 | \mathbf{y}, \text{all checks involving } x_j \text{ are satisfied}\right)}{\Pr\left(x_j = 1 | \mathbf{y}, \text{all checks involving } x_j \text{ are satisfied}\right)}. \tag{2.6}$$

In the above expression, $\Pr\left(x_j = u | \mathbf{y}, \text{all checks involving } x_j \text{ are satisfied}\right)$ is the conditional probability that $x_j = u$, $u \in \{0, 1\}$ given that (i) all the CNs c_i, $i \in \mathbf{N}(j)$ are satisfied (i.e all such i^{th} components of the syndrome vector are zero) and (ii) the received sequence is $\mathbf{y}$.

Now, (2.5) can be alternatively expressed as

$$\hat{x}_j = \frac{1 - \text{sgn}\left(L_j\right)}{2}. \tag{2.7}$$

In the SPA, L_js are updated iteratively by exchanging messages between the VNs v_js and the CNs c_is. The extrinsic messages from v_j are initialized with the channel LLR (intrinsic message) L_{ch_j} for x_j given by

$$L_{\text{ch}_j} = \log \frac{\Pr\left(x_j = 0 | y_j\right)}{\Pr\left(x_j = 1 | y_j\right)}. \tag{2.8}$$

We consider the transmission of the codewords over an AWGN channel with mean zero and variance σ_n^2. The codeword sequence $\mathbf{x} = [x_1 \dots x_N]^T$ is BPSK modulated to $\mathbf{t_x} = [t_{x_1} \dots t_{x_N}]^T$ according to $t_{x_j} = 1 - 2x_j$. After the transmission of $\mathbf{t_x}$ over an AWGN channel, the received sequence is given by $\mathbf{y} = \mathbf{t_x} + \mathbf{n}$, where $\mathbf{n} = [n_1 \dots n_N]^T$ and n_js are independent and identically distributed Gaussian random variables with mean zero and variance σ_n^2. Assuming x_js to be equally likely to take 0 and 1,

L_{ch_j} in (2.8) can now be expressed as

$$L_{\text{ch}_j} = \log \frac{\exp\left(-\frac{(y_j-1)^2}{2\sigma_n^2}\right)}{\exp\left(-\frac{(y_j+1)^2}{2\sigma_n^2}\right)} = \frac{2y_j}{\sigma_n^2}. \tag{2.9}$$

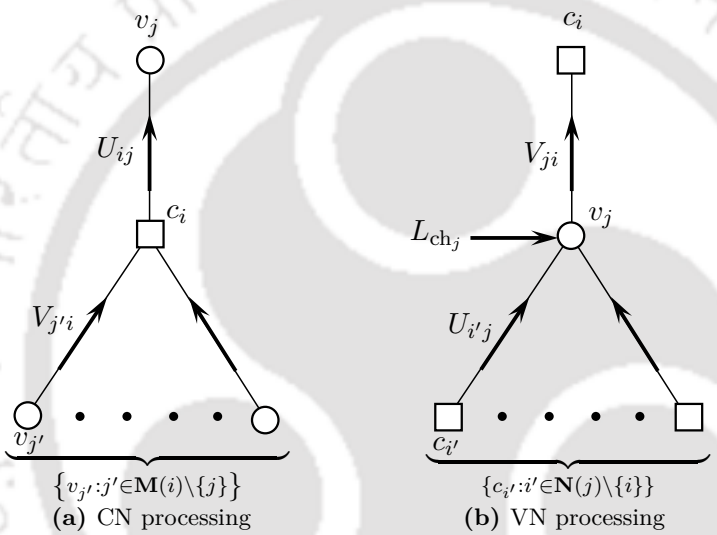


Figure 2.3: Diagrammatic representation of the message passing in the SPA

The SPA in the LLR domain is described in Algorithm 2.1. At the beginning, the VN→CN messages are initialized with the channel LLR. The rest of the SPA can be presented in the following steps:

(i) **CN processing**: The message U_{ij} sent from a CN c_i towards its neighboring VN v_j is defined as

$$U_{ij} \triangleq \log \frac{\Pr\left(\text{check } c_i \text{ is satisfied}|x_j = 0, \mathbf{y}\right)}{\Pr\left(\text{check } c_i \text{ is satisfied}|x_j = 1, \mathbf{y}\right)}.$$

In the above expression, $\Pr\left(\text{check } c_i \text{ is satisfied}|x_j = u, \mathbf{y}\right)$ is the conditional probability that the i^{th} CN is satisfied (i.e. the i^{th} element of the syndrome vector is zero) given that (i) the value of x_j is u and (ii) the received sequence is $\mathbf{y}$. With the assumption that all the VNs connected

Algorithm 2.1: Sum-Product Algorithm

input : A received sequence $\mathbf{y} = [y_1 \dots y_N]^T$, the parity-check matrix $\mathbf{H}$ and the maximum number of iterations I_m

output: Estimated codeword $\hat{\mathbf{x}} = [\hat{x}_1 \dots \hat{x}_N]^T$

Find the set of neighbors for each VN $j \in \{1, \dots, N\}$, $\mathbf{N}(j) = \{i : h_{ij} = 1\}$;
Find the set of neighbors for each CN $i \in \{1, \dots, M\}$, $\mathbf{M}(i) = \{j : h_{ij} = 1\}$;
for $j \leftarrow 1$ **to** N **do** // initiliazation of the VN→CN messages
  $L_{\mathrm{ch}_j} = \frac{2y_j}{\sigma_n^2}$;
  Set $V_{ji} = L_{\mathrm{ch}_j}, \forall i \in \mathbf{N}(j)$;
end
for $I \leftarrow 1$ **to** I_m **do** // Loop for iterations
  for $i \leftarrow 1$ **to** M **do** // CN processing
    Compute U_{ij} $\forall j \in \mathbf{M}(i)$:
    $$U_{ij} = \log \frac{1 + \prod_{j' \in \mathbf{M}(i) \setminus \{j\}} \tanh\left(\frac{V_{j'i}}{2}\right)}{1 - \prod_{j' \in \mathbf{M}(i) \setminus \{j\}} \tanh\left(\frac{V_{j'i}}{2}\right)}$$
  end
  for $j \leftarrow 1$ **to** N **do** // VN processing
    Compute V_{ji} $\forall i \in \mathbf{N}(j)$:
    $$V_{ji} = \sum_{i' \in \mathbf{N}(j) \setminus \{i\}} U_{i'j} + L_{\mathrm{ch}_j}$$
  end
  for $j \leftarrow 1$ **to** N **do** // Computation of *a posteriori* LLR
    $$L_j = \sum_{i' \in \mathbf{N}(j)} U_{i'j} + L_{\mathrm{ch}_j}$$
  end
  for $j \leftarrow 1$ **to** N **do** // Hard decision
    $\hat{x}_j = \frac{(1 - \mathrm{sgn}(L_j))}{2}$
  end
  if $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$ **then** // Stopping criterion
    Stop;
  end
end

to c_i send independent messages towards c_i, U_{ij} can be computed as

$$U_{ij} = \log \frac{1 + \prod_{j' \in \mathbf{M}(i) \setminus \{j\}} \tanh\left(\frac{V_{j'i}}{2}\right)}{1 - \prod_{j' \in \mathbf{M}(i) \setminus \{j\}} \tanh\left(\frac{V_{j'i}}{2}\right)}, \tag{2.10}$$

where $\mathbf{M}(i) = \{j : h_{ij} = 1\}$ is the set of the indices of the neighboring VNs connected to c_i and $V_{j'i}$ is the message sent from $v_{j'}$ to c_i as described in the next item. The graphical representation of the CN processing at c_i is shown in Figure 2.3(a).

(ii) **VN processing**: The message V_{ji} sent from a VN v_j towards its neighboring CN c_i is defined as

$$V_{ji} \triangleq \log \frac{\Pr\left(x_j = 0 | \mathbf{y}, \text{all checks involving } x_j \text{ except } c_i \text{ are satisfied}\right)}{\Pr\left(x_j = 1 | \mathbf{y}, \text{all checks involving } x_j \text{ except } c_i \text{ are satisfied}\right)}.$$

With the assumption that all the checks connected to v_j are conditionally independent given the value of x_j, V_{ji} can be computed as

$$V_{ji} = \sum_{i' \in \mathbf{N}(j) \setminus \{i\}} U_{i'j} + L_{\mathrm{ch}_j}, \tag{2.11}$$

where $\mathbf{N}(j)$ is the set of the indices of the neighboring CNs connected to v_j, i.e., $\mathbf{N}(j) = \{i : h_{ij} = 1\}$. The graphical representation of the VN processing at v_j is shown in Figure 2.3(b).

(iii) **Computation of *a posteriori* LLR**: The *a posteriori* LLR L_j for the VN v_j as formulated in (2.6) can be computed as

$$L_j = \sum_{i' \in \mathbf{N}(j)} U_{i'j} + L_{\mathrm{ch}_j}. \tag{2.12}$$

(iv) **Hard decision and stopping criterion**: Based on the sign of the *a posteriori* LLR, the hard decision is made according to (2.7). If a valid codeword is obtained, i.e., $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then the decoding process is stopped, otherwise it is continued till the maximum number of iterations are executed.

2.4 Decoding Threshold for an Ensemble of LDPC Codes

A degree-distribution pair $(\lambda(x), \rho(x))$ usually specifies an ensemble of LDPC codes. As $N \to \infty$, an ensemble exhibits a threshold phenomenon. Richardson and Urbanke [60] have shown that, in the asymptotic sense, there exists a critical value of the channel condition parameter which partitions the parameter space into two regions. In one region, the probability of decoding error converges to zero as the number of iterations tends to ∞. In the other, it remains non-negligible even after a large number of iterations. This critical value of the channel parameter is known as the *decoding threshold* or *threshold* in short. The channel parameter for an AWGN channel may be specified in terms of either the noise standard deviation $\sigma_n = \sqrt{\frac{N_0}{2}}$ or the energy per information bit to noise power spectral density ratio $\frac{E_b}{N_0}$. In the case of the BPSK modulation, $\frac{E_b}{N_0}$ is equal to $r^{-1}\frac{E_s}{N_0}$, where E_s is the energy per modulated symbol and r is the rate of the code. Assuming that the signal energy

has been normalized (i.e., $E_s = 1$), we have $\frac{E_b}{N_0} = \frac{1}{2r\sigma_n^2}$. The threshold can be denoted by either σ_n^* or $\left(\frac{E_b}{N_0}\right)^*$. Reliable communication can be accomplished only when $\sigma_n < \sigma_n^*$ or $\frac{E_b}{N_0} > \left(\frac{E_b}{N_0}\right)^*$. If the threshold $\left(\frac{E_b}{N_0}\right)^*$ for an ensemble is close to the Shannon limit, then that ensemble is considered as *capacity-approaching*.

Richardson and Urbanke [60] have devised a technique known as *density evolution* (DE) to analyze the iterative decoding process from an asymptotic perspective. The DE tracks the variations of the probability density function (density in short) for the extrinsic messages flowing during iterative decoding. Such knowledge of the density can be utilized to compute $\left(\frac{E_b}{N_0}\right)^*$. By minimizing $\left(\frac{E_b}{N_0}\right)^*$ over all possible degree-distribution pairs by a method known as the *differential evolution* [57], the optimum pair can be found out. Brink *et al.* have proposed an alternative approach for the threshold evaluation known as the *extrinsic information transfer* (EXIT) chart method [67,68]. In this approach, the growth of the mutual information between the bit value and a related extrinsic message is monitored. The detailed procedure of the threshold computation via the EXIT chart is described in Chapter 5.

2.5 Finite-length Construction of LDPC Codes

Long LDPC codes designed according to the optimized degree distributions can perform close to the Shannon limit [14]. However, for the short-to-moderate length codes (commonly referred as the *finite-length* codes), the degree distributions alone cannot guarantee a good performance. The interconnections of the VNs and the CNs in the Tanner graph play a key role in the performance of the iterative decoders like the SPA. All the incoming messages to a node (VN or CN) are assumed to be independent in the SPA. Although this assumption holds asymptotically, it becomes invalid for the finite-length codes after a certain number of iterations [28]. The reason for this is the presence of cycles in the Tanner graph. In a Tanner graph of girth g, the independence assumption becomes invalid after m iterations where $m < g/4 < m+1$ [28]. The girth of the Tanner graph should, therefore, be increased in order to delay the violation of the independence assumption.

In the following, we describe two popular greedy algorithms for the construction of finite-length LDPC codes avoiding harmful short cycles.

2.5.1 Progressive Edge-Growth Algorithm

To construct a Tanner graph with a large girth, Hu *et al.* proposed a greedy algorithm known as the *progressive edge-growth* (PEG) algorithm [34]. The PEG algorithm starts with a graph consisting of only the nodes (VNs and CNs) where there are no edges present, i.e., $G_{\mathbf{H}} = (V \cup U, E = \{\})$. The VNs are processed one by one sequentially by assigning the edges to them. While assigning an edge to a VN, it is ensured that the local girth at that VN gets maximized. This maximization is accomplished by putting the edge between the VN and a properly chosen CN which is at the maximum distance from that VN. The *distance* between two nodes is the number of edges in a shortest path connecting them. After the required number of edges are assigned to the VN, the next VN is processed. The PEG algorithm as presented in [34] is described in Algorithm 2.2.

Algorithm 2.2: Progressive Edge-Growth Algorithm [34]

input : N, M and the VN degree sequence $D_v = \{d_{v_1}, \ldots, d_{v_N}\}$
output: Tanner graph or parity-check matrix $\mathbf{H}$

for $j \leftarrow 1$ **to** N **do** `// for each VN`
  for $k \leftarrow 1$ **to** d_{v_j} **do** `// for each edge`
    if $k = 1$ **then** `// the first edge from a VN`
      Set $E_{v_j}^k$=edge(v_j, c_i), where c_i is a CN having the lowest degree under the current graph setting $E_{v_1} \cup E_{v_2} \cup \cdots \cup E_{v_{j-1}}$. If multiple such CNs are present, then select one of them uniformly at random.
    else
      Expand a subgraph from v_j up to depth l under the current graph setting such that either of the following two events occurs:
      (i) $\left|\mathcal{N}_{v_j}^{l+1}\right| = \left|\mathcal{N}_{v_j}^{l}\right| < M$ `// not all CNs are reachable from` v_j
      (ii) $\left|\mathcal{N}_{v_j}^{l}\right| < M$ and $\left|\mathcal{N}_{v_j}^{l+1}\right| = M$ `// all the CNs are reachable from` v_j
      Set $E_{v_j}^k$=edge(v_j, c_i), where c_i is the CN with the lowest degree present in $\bar{\mathcal{N}}_{v_j}^{l}$. If multiple such CNs exist, select one of them uniformly at random.
    end
  end
end

The inputs to the PEG algorithm are the number of columns (N) and rows (M) of the parity-check matrix to be constructed and the degree sequence for all the VNs given by $D_v = \{d_{v_1}, \ldots, d_{v_N}\}$, where d_{v_j} is the degree of v_j. It is assumed that the elements of D_v are arranged in a non-decreasing way, i.e., $d_{v_j} \leq d_{v_i}$ for $j < i$. D_v can be obtained from the degree-distribution $\lambda(x)$ and N. The set of edges in the Tanner graph can be expressed as $E = \bigcup_{j=1}^{N} E_{v_j}$, where E_{v_j} is the set of edges connected

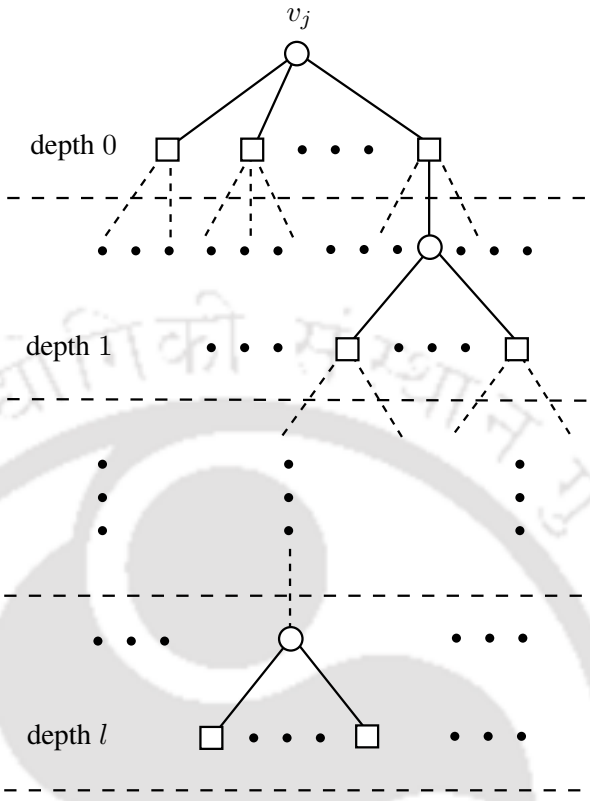


Figure 2.4: Visualization of $\mathcal{N}_{v_j}^{l}$

to v_j. E_{v_j}, in turn, can be expressed as $E_{v_j} = \bigcup_{k=1}^{d_{v_j}} E_{v_j}^{k}$, where $E_{v_j}^{k}$ is the kth edge connected to v_j. For a given VN v_j, the neighborhood $\mathcal{N}_{v_j}^{l}$ within depth l, is defined as the set of CNs which are connected to v_j with a maximum distance of $2l+1$. $\mathcal{N}_{v_j}^{l}$ can be visualized as shown in Figure 2.4.

In PEG, one VN is considered at a time and the edges are assigned sequentially to it. For the assignment of the *first* edge to any VN v_j, a CN c_i of the lowest degree under the current graph setting $E_{v_1} \cup E_{v_2} \cup \cdots \cup E_{v_{j-1}}$ is selected. If more than one choices are available, then one CN is selected uniformly at random. The assignment of the first edge to a VN does not create new cycles. For *other* edges, a subgraph is expanded from v_j up to a depth of l, where l is such that either (1) the cardinality of $\mathcal{N}_{v_j}^{l}$ stops increasing and is less than M or (2) the cardinality of $\mathcal{N}_{v_j}^{l}$ is less than M, but the cardinality of $\mathcal{N}_{v_j}^{l+1}$ is M. In either case, a CN c_i which has the lowest degree amongst the CNs in $\bar{\mathcal{N}}_{v_j}^{l} = U \setminus \mathcal{N}_{v_j}^{l}$ is selected. If multiple candidates exist, any one of them is selected uniformly at random. In the first case, some CNs are not reachable from v_j. Thus the assignment of any edge to such a CN does not create new cycles. However, in the second case, the assignment of an edge creates new cycles as all the CNs are already reachable from v_j.

Note that the CN degree distribution $\rho(x)$ is not used in the original setting of the PEG algorithm.

As the CNs with the lowest degree are given more preference, the resulting CN degree profile becomes uniform at a particular degree or concentrated around two consecutive degrees. Although $\rho(x)$ can be included in Algorithm 2.2 with some modifications, it might be unnecessary as the optimum CN degree distribution usually contains a concentrated degree sequence [59].

2.5.2 Approximate Cycle Extrinsic Message Degree Algorithm

In the PEG algorithm, utmost importance is given to the lengths of the cycles. Tian *et al.* [70] have shown that all the cycles of the same length are not equally harmful. It is emphasized that the connectivity of the cycles is also crucial in addition to the lengths. A new measure of the connectivity of the cycles, the *extrinsic message degree* (EMD) is proposed.

Definition 2.1. *The EMD of a cycle C_{2i},* $\mathrm{EMD}(C_{2i})$ *is defined as the number of CNs singly connected to the VNs involved in C_{2i}.*

To calculate the EMD, one has to examine the subgraph induced by the VNs involved in the cycle. In order to facilitate an easy calculation of the EMD, the *approximate cycle EMD* (ACE) [70] is devised.

Definition 2.2. *The ACE of a cycle C_{2i} of length $2i$ is given by*

$$\mathrm{ACE}(C_{2i}) = \sum_{k=1}^{i} (d_{v_k} - 2), \tag{2.13}$$

where d_{v_k} is the degree of the VN v_k and the summation is over all the VNs in the cycle.

Although primarily devised for the cycles, the ACE can also be similarly defined for a node and a path. The ACE of a VN v_k is given by $d_{v_k} - 2$. The ACE of a CN is zero. The ACE of a path between any two nodes is the sum of the ACE values for all the nodes present in the path including the two terminal nodes.

A construction method commonly referred as the ACE algorithm is proposed in [70] to enhance the connectivity of the cycles by improving the ACE values. This method constructs a $(d_{\mathrm{ACE}}, \eta_{\mathrm{ACE}})$ LDPC code which signifies that all the cycles of length $2d_{\mathrm{ACE}}$ or less have ACE values at least η_{ACE}. The ACE algorithm as presented in [70] is described in Algorithm 2.3. The inputs to the algorithm are N, M, $\lambda(x)$, $\rho(x)$, d_{ACE} and η_{ACE}. It constructs a parity-check matrix given by $\mathbf{H} = [\mathbf{H}_1; \mathbf{H}_2]$, where $\mathbf{H}_1$ and $\mathbf{H}_2$ correspond to the information and the parity bits respectively. The information

bits should be provided with more protection. Thus the VN degrees are arranged in a non-increasing manner, i.e., $d_{v_j} \geq d_{v_i}$ for $j < i$.

Algorithm 2.3: ACE algorithm [70]

input : N, M, $\lambda(x)$, $\rho(x)$, d_{ACE} and η_{ACE}
output: The parity-check matrix $\mathbf{H}$ for a $(d_{\text{ACE}}, \eta_{\text{ACE}})$ LDPC code

```
Initialize flag = 0, H ← 0_{M×N};
for j ← N to 1 do                                                    // for each VN
    Randomly generate the jth column h_j according to λ(x) and ρ(x);
    if j > N − M then                                                // if j is a parity bit
        Gaussian elimination (GE) on H_2;
        if h_j ∈ SPAN(h'_{j+1}, ..., h'_N) then                      // if h_j is linearly dependent
            h_j ← 0;
            flag = 1;
            j ← j + 1 ;                                              // Repeat the random generation of h_j
        else
            h'_j is the residue for h_j after GE;
            flag = 0;
        end
    end
    if flag = 0 then                                                 // if h_j is linearly independent
        success =ACE detection(v_j, H, d_ACE, η_ACE);
        if success = 0 then                                          // if the ACE criterion is not satisfied
            h_j ← 0;
            j ← j + 1 ;                                              // Repeat the random generation of h_j
        end
    end
end
```

In the ACE algorithm, any particular column $\mathbf{h}_j$ is generated randomly satisfying the degree-distributions pair. For the parity bits, the columns are assigned such that they are linearly independent. This is carried out by performing the Gaussian elimination on $\mathbf{H}_2$. In such a situation, the parity-check matrix $\mathbf{H}$ will be of full rank. Moreover, if the number of VNs of degree 2 is less than M, there will be no cycle involving all degree-2 VNs. If the randomly generated column becomes linearly dependent on the previously assigned parity columns, that column is generated again randomly.

Upon the fulfillment of the linear-independence condition, the currently developed parity-check matrix is fed to the ACE detection procedure. If each cycle of length $2d_{\text{ACE}}$ or lower has the ACE value at least η_{ACE}, then the next column is generated, otherwise the same procedure is repeated with a new randomly generated $\mathbf{h}_j$. The ACE detection procedure during the construction of the jth column is presented in Algorithm 2.4.

Algorithm 2.4: ACE detection for a VN v_j [70]

Function `ACE detection`$(v_j, \mathbf{H}, d_{\text{ACE}}, \eta_{\text{ACE}})$
  Initialize $success = 0$;
  Initialize $p(v_j) \leftarrow \text{ACE}(v_j)$ for the root node v_j in level 0;
  Initialize $p(\mu_t) \leftarrow \infty$ for all nodes μ_t ; `// Initially unvisited`
  for $l \leftarrow 1$ **to** d_{ACE} **do** `// for each level`
    foreach *node* w_s *in level* $l-1$ **do**
      Find its children set $Ch(w_s)$;
      foreach *child* $\mu_t \in Ch(w_s)$ **do**
        $p_{\text{temp}} \leftarrow p(w_s) + \text{ACE}(\mu_t)$; `// ACE for the current temporary path`
        if $(p_{\text{temp}} + p(\mu_t) - \text{ACE}(v_j) - \text{ACE}(\mu_t)) < \eta_{\text{ACE}}$ **then** `// ACE condition`
          exit with $success = 0$;
        else if $p_{\text{temp}} < p(\mu_t)$ **then**
          $p(\mu_t) \leftarrow p_{\text{temp}}$; `// update the path metric for` μ_t
        end
      end
    end
  end
  return with $success = 1$;
end

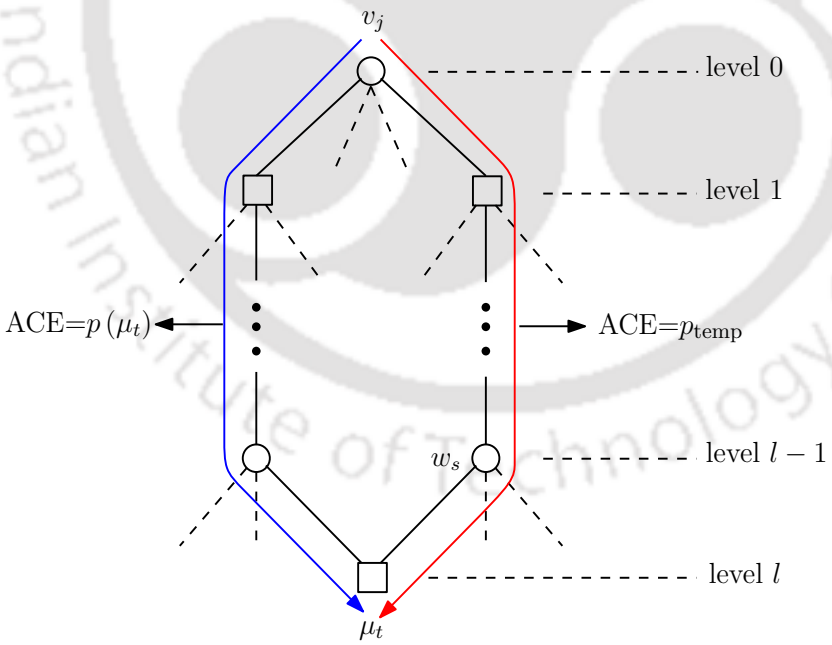


Figure 2.5: Identification of a cycle in the ACE detection procedure

In Algorithm 2.4, $p(\mu_t)$ denotes the minimum ACE value for any path linking the current VN v_j and any node μ_t. It can be considered as the path metric for the node μ_t. At the start of the algorithm, $p(\mu_t)$ for all nodes (VNs and CNs) except v_j are initialized to ∞ (unvisited). $p(v_j)$ is

initialized to ACE(v_j). The presently developed Tanner graph is expanded as a tree-like structure by considering v_j as the root node. Suppose for any node w_s in level $l-1$, there is a child μ_t in level l. The ACE for this temporary path via w_s is given by $p_{\text{temp}} = p(w_s) + \text{ACE}(\mu_t)$, where $p(w_s)$ is the minimum ACE for a path up to w_s. Suppose there is already a path to μ_t from v_j, i.e., μ_t has been visited before ($p(\mu_t) \neq \infty$). In that case, a cycle exists. This situation is depicted pictorially in Figure 2.5. The ACE of this cycle is $p_{\text{temp}} + p(\mu_t) - \text{ACE}(v_j) - \text{ACE}(\mu_t)$ as the ACE values for v_j and μ_t are counted both in $p(\mu_t)$ and p_{temp}. If this ACE value does not meet the design criterion, then the algorithm exits with a failure. Otherwise, p_{temp} is considered as the new path metric for μ_t if it is less than $p(\mu_t)$.

In the ACE algorithm [70], all the cycles of length up to $2d_{\text{ACE}}$ are treated similarly with the same constraint η_{ACE} applied on them. Vukobratovic and Senk [72, 73] have suggested that instead of using the same ACE constraint for the cycles of different lengths, higher ACE constraints should be used for the shorter cycles as they are more detrimental. They have advocated for the use of an ACE spectrum constraint, i.e., $\boldsymbol{\eta}_{\text{ACE}} = (\eta_2, \eta_4, \cdots, \eta_{2d_{\text{ACE}}})$, where η_{2i} is the ACE constraint for all the cycles of length $2i$. As shorter cycles are more harmful, $\eta_2 \geq \eta_4 \geq \cdots \geq \eta_{2d_{\text{ACE}}}$ in general. The notation $\eta_{2i} = \infty$ means that there is no cycle of length $2i$. Since d_{ACE} is obvious from $\boldsymbol{\eta}_{\text{ACE}}$, a $(d_{\text{ACE}}, \boldsymbol{\eta}_{\text{ACE}})$ code is generally designated as an $(\eta_2, \eta_4, \cdots, \eta_{2d_{\text{ACE}}})$ code. This type of codes, whose cycles are constrained by an ACE spectrum, are known as the *ACE spectrum constrained codes* [72, 73].

Note that the ACE spectrum provides an assessment regarding the connectivity of the cycles for an ensemble of codes. In order to obtain further insight about the cycles present in a specific Tanner graph, the authors in [72, 73], have devised the *expanded ACE spectrum* as defined below.

Definition 2.3. *Let η_{2i}^{v} be the minimum ACE value of all the cycles of length $2i$ which involve the VN v. Then, for a given d_{ACE}, the expanded ACE spectrum is given by the d_{ACE}-tuple:*

$$\boldsymbol{\eta}_{\text{ACE}}^{\text{exp}} = \left(\boldsymbol{\eta}_{2}^{\text{exp}}, \boldsymbol{\eta}_{4}^{\text{exp}}, \cdots, \boldsymbol{\eta}_{2d_{\text{ACE}}}^{\text{exp}}\right), \tag{2.14}$$

where $\boldsymbol{\eta}_{2i}^{\text{exp}}$ is an $(\eta_{2i}^{\max}+1)$-tuple :

$$\boldsymbol{\eta}_{2i}^{\text{exp}} = \left(n_{(2i,0)}, \ldots, n_{\left(2i,\eta_{2i}^{\max}\right)}\right). \tag{2.15}$$

The component $n_{(2i,j)}$ of $\boldsymbol{\eta}_{2i}^{\text{exp}}$ is the number of VNs whose η_{2i}^{v} value equals j and $\eta_{2i}^{\max}$ is the maximum possible ACE value of a length-$2i$ cycle.

Observe that $\boldsymbol{\eta}_{\mathrm{ACE}}$ can be obtained from $\boldsymbol{\eta}_{\mathrm{ACE}}^{\mathrm{exp}}$. The component η_{2i} of $\boldsymbol{\eta}_{\mathrm{ACE}}$ is equal to the smallest index j such that the component $n_{(2i,j)}$ of $\boldsymbol{\eta}_{2i}^{\mathrm{exp}}$ is non-zero. Two different Tanner graphs with the same $\boldsymbol{\eta}_{\mathrm{ACE}}$ can be distinguished with the help of $\boldsymbol{\eta}_{\mathrm{ACE}}^{\mathrm{exp}}$. Useful information about the amount of involvement of the VNs in the formation of the cycles can be obtained from $\boldsymbol{\eta}_{\mathrm{ACE}}^{\mathrm{exp}}$.

2.6 Error Floor of LDPC Codes

The error floor problem stems from the sub-optimality of the iterative decoders due to the presence of cycles in the Tanner graph of the finite-length LDPC codes. The combinations of one or more short cycles give rise to some harmful subgraphs in the Tanner graph. The set of the VNs present in such a subgraph has been referred by various terms depending on the channel. For a binary erasure channel (BEC), the error floor is attributed to the *stopping sets* [22]. The error floor for an ensemble of LDPC codes over a BEC has been completely characterized in terms of the stopping sets [22]. However, for other channels like the binary symmetric channel (BSC) or the AWGN channel, the understanding of the error floor problem has not been as complete as in the case of a BEC. Richardson has proposed the notion of the *trapping sets* (TSs) and formulated a technique to predict the error floor performance of an LDPC code [58]. Dolecek *et al.* have devised another structure named the *absorbing sets* (ASs) [24]. The ASs are the subsets of the TSs and have been found to be more detrimental [23,80]. We describe the TSs and the ASs in detail and with proper terminology in Chapter 3.

So far, we have concentrated on the binary LDPC codes which are defined over GF(2). The class of non-binary (NB) LDPC codes, defined over a higher order field, are comparatively less affected by the error floor issue. The next section discusses the NB-LDPC codes.

2.7 Non-binary LDPC Codes

Davey and Mackay have initiated the study on NB LDPC codes defined over GF(q), $q > 2$ [18]. They have shown that the NB-LDPC codes can perform significantly better than their binary counterparts. The salient feature of the NB-LDPC codes is that they do not suffer from the error floor problem unlike the binary LDPC codes and consequently perform much better in the finite-length regime. This feature makes them reasonable candidates for practical communication systems. Recently the Consultative Committee for Space Data Systems (CCSDS) have recommended the use of some short-length NB-LDPC codes for telecommand applications [2]. The main drawback of the

NB-LDPC codes hindering their applications is the higher decoding complexity.

The NB-LDPC codes can be represented in terms of the parity-check matrices as in the case of the binary codes. The only difference is that a non-zero element of the parity-check matrix for an NB-LDPC code can be any member from $\mathrm{GF}(q)\setminus\{0\}$. Throughout the thesis, $\mathrm{GF}(q)^{-} \triangleq \mathrm{GF}(q)\setminus\{0\}$. The Tanner graph for an NB-LDPC code contains a label for each edge corresponding to the element from $\mathrm{GF}(q)^{-}$. We illustrate an example of NB-LDPC code below.

Example 2.3. *Consider an NB-LDPC code with the following* 5×10 *parity-check matrix over* $\mathrm{GF}(2^3)$*:*

$$\mathbf{H} = \begin{bmatrix} 3 & 1 & 0 & 1 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 5 & 1 \\ 0 & 0 & 2 & 0 & 7 & 7 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 5 & 6 & 0 & 7 \\ 0 & 6 & 0 & 0 & 7 & 0 & 6 & 0 & 6 & 0 \end{bmatrix}.$$

The primitive polynomial for $GF(2^3)$ *is* $f(x) = x^3 + x + 1$*. The numbers in* $\mathbf{H}$ *correspond to the coefficients in the polynomial representations of the elements of* $GF(2^3)$*. The Tanner graph for* $\mathbf{H}$ *is shown in Figure 2.6.*

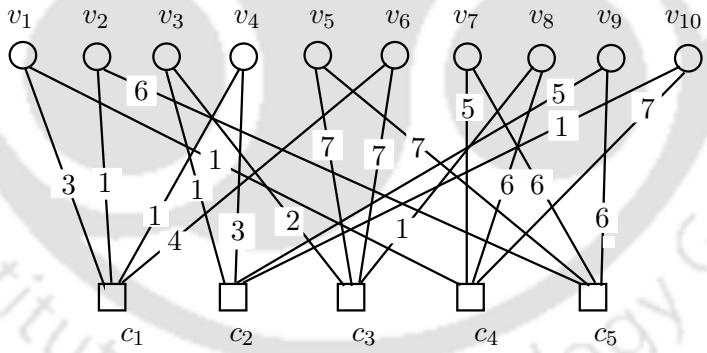


Figure 2.6: Tanner graph for the parity-check matrix shown in Example 2.3

2.7.1 Decoding by FFT-based Q-ary Sum-product Algorithm

Davey and Mackay have extended the probability-domain SPA to decode the NB-LDPC codes over $\mathrm{GF}(q), q > 2$ [18]. This algorithm is commonly known as the q-ary SPA (QSPA). They have considered mainly the NB-LDPC codes defined over binary extension fields, i.e., $\mathrm{GF}(q = 2^s), s \in \mathbb{Z}^{+} \setminus \{1\}$. Any message flowing through an edge in the QSPA is a vector of length of q. The ath component of such a message corresponds to the probability of the concerned symbol (VN) being $a, a \in \mathrm{GF}(q)$. Because of the vector messages, the complexity of the QSPA increases significantly compared to the binary

analogue. The most computationally-intensive part is the CN processing. The complexity of the CN processing for the QSPA in the rudimentary form is $O\left(q^{d_c}\right)$ for a degree-d_c CN. Davey and Mackay have themselves shown that the CN processing can be realized in a forward-backward manner because of which the complexity gets reduced to $O\left(q^2\right)$ [18]. It can be shown that the CN processing is actually a convolution operation [60]. Consequently it can be accomplished in the frequency domain via s-dimensional 2-point *fast Fourier transforms* (FFT) [9, 19]. Such a realization of the CN processing reduces its complexity further to $O\left(q \log_2 q\right)$. This version of the QSPA is known as the FFT-QSPA.

The FFT-QSPA is described in Algorithm 2.5. Before explaining the algorithm, we describe the channel model as considered in [18]. The codeword symbols are transmitted over a BI-AWGN channel. Each codeword symbol $x_j, j \in \{1, \cdots, N\}$ is represented in a bit-level form $x_j = [x_{j,1} \ldots x_{j,s}]$. x_j is BPSK modulated to obtain $t_{x_j} = \left[t_{x_{j,1}} \ldots t_{x_{j,s}}\right]$, where $t_{x_{j,b}} = 1 - 2x_{j,b}$. The t_{x_j} transmitted over the BI-AWGN channel yields $y_j = [y_{j,1} \ldots y_{j,s}]$, where $y_{j,b} = t_{x_{j,b}} + n_{j,b}$ and $n_{j,b}$s are independent and identically distributed Gaussian random variables with mean zero and variance σ_n^2.

The *a posteriori* probability of $x_{j,b}$ being 0 given the channel output $y_{j,b}$ is defined by

$$g_{j,b}^0 \triangleq \Pr\left(x_{j,b} = 0 | y_{j,b}\right).$$

Therefore, for a BI-AWGN channel, we have

$$g_{j,b}^0 = \frac{1}{\left(1 + \exp\left(-2y_{j,b}/\sigma^2\right)\right)}$$
$$\text{and} \quad g_{j,b}^1 = 1 - g_{j,b}^0.$$

The channel *a posteriori* probability f_j^a of the jth symbol x_j being a, $a \in \mathrm{GF}\left(2^s\right)$ for a given y_j is defined as

$$f_j^a \triangleq \Pr\left(x_j = a | y_j\right). \tag{2.16}$$

Therefore, we have

$$f_j^a = \prod_{b=1}^{s} g_{j,b}^{a_b}, \tag{2.17}$$

where a_b is the bth bit of the binary representation of a.

Now we explain the main steps of Algorithm 2.5. As in the case of SPA, the VN→CN messages are initialized with the channel *a posteriori* probability vector. The remaining steps are given by:

Algorithm 2.5: FFT-QSPA

input : A received sequence $\mathbf{y} = [y_{1,1} \dots y_{1,s} \dots y_{N,1} \dots y_{N,s}]^T$, the parity-check matrix $\mathbf{H}$ over GF(2^s) and the maximum number of iterations I_m

output: Estimated codeword $\hat{\mathbf{x}} = [\hat{x}_1 \dots \hat{x}_N]^T$

Find the set of neighbors for each VN $j \in \{1, \dots, N\}$, $\mathbf{N}(j) = \{i : h_{ij} \neq 0\}$;
Find the set of neighbors for each CN $i \in \{1, \dots, M\}$, $\mathbf{M}(i) = \{j : h_{ij} \neq 0\}$;
for $j \leftarrow 1$ **to** N **do** `// initiliazation of the VN→CN messages`
  Compute f_j^a, $\forall a \in \text{GF}(2^s)$ according to (2.17) ;
  Initialize $\mathbf{q}_{ji}$, $\forall i \in \mathbf{N}(j)$:
  $q_{ji}(a) = f_j^a$, $\forall a \in \text{GF}(2^s)$;
end
for $I \leftarrow 1$ **to** I_m **do** `// Loop for iterations`
  for $j \leftarrow 1$ **to** N **do** `// permutation`
    Compute $\tilde{\mathbf{q}}_{ji}$, $\forall i \in \mathbf{N}(j)$:
    $\tilde{q}_{ji}(a) = q_{ji}\left(h_{ij}^{-1} a\right) \quad \forall a \in \text{GF}(2^s)$;
  end
  for $i \leftarrow 1$ **to** M **do** `// CN processing`
    Compute $\tilde{\mathbf{r}}_{ij}$, $\forall j \in \mathbf{M}(i)$:
    $\tilde{\mathbf{r}}_{ij} = H_q \overset{\bullet}{\prod_{j' \in \mathbf{M}(i) \setminus \{j\}}} H_q \tilde{\mathbf{q}}_{j'i}$;
  end
  for $i \leftarrow 1$ **to** M **do** `// depermutation`
    Compute $\mathbf{r}_{ij}$, $\forall j \in \mathbf{M}(i)$:
    $r_{ij}(a) = \tilde{r}_{ij}(h_{ij} a) \quad \forall a \in \text{GF}(2^s)$;
  end
  for $j \leftarrow 1$ **to** N **do** `// VN processing`
    Compute $\mathbf{q}_{ji}$, $\forall i \in \mathbf{N}(j)$:
    $q_{ji}(a) = \alpha_{ji} f_j^a \prod_{i' \in \mathbf{N}(j) \setminus \{i\}} r_{i'j}(a) \quad \forall a \in \text{GF}(2^s)$
    where, α_{ji} is chosen such that $\sum_{a \in \text{GF}(2^s)} q_{ji}(a) = 1$.
  end
  for $j \leftarrow 1$ **to** N **do** `// Computation of` *a posteriori* `probability`
    $q_j(a) = \alpha_j f_j^a \prod_{i \in \mathbf{N}(j)} r_{ij}(a) \quad \forall a \in \text{GF}(2^s)$
    where, α_j is chosen such that $\sum_{a \in \text{GF}(2^s)} q_j(a) = 1$.
  end
  for $j \leftarrow 1$ **to** N **do** `// Hard decision`
    $\hat{x}_j = \arg \max_{a \in \text{GF}(2^s)} q_j(a)$
  end
  if $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$ **then** `// Stopping criterion`
    Stop;
  end
end

(i) **CN processing**: The message sent by c_i towards v_j is denoted by $\mathbf{r}_{ij} = [r_{ij}(a)]_{a \in \text{GF}(2^s)}$. It

depends on the incoming messages $\mathbf{q}_{j'i} = \left[q_{j'i}\left(x_{j'}\right)\right]_{x_{j'}\in \mathrm{GF}(2^s)}$ from the VNs $v_{j'}$, $j' \in \mathbf{M}\left(i\right) \setminus \{j\}$. Suppose the values taken by these VNs is denoted by $\mathbf{x}_{ij} = \left[x_{j'}\right]_{j'\in \mathbf{M}(i)\setminus\{j\}}$. The set of combinations of $x_{j'}$s resulting in the satisfaction of the ith check/syndrome with $x_j = a$ is known as the *configurations set* $\mathrm{Conf}_{ij}\left(a\right)$ [17, 20]. It is given by

$$\mathrm{Conf}_{ij}\left(a\right) = \left\{ \left[x_{j'}\right]_{j'\in \mathbf{M}(i)\setminus\{j\}} : \sum_{j'\in \mathbf{M}(i)\setminus\{j\}} h_{ij'}x_{j'} = h_{ij}a \right\}.$$

$r_{ij}\left(a\right)$ is given by

$$r_{ij}\left(a\right) = \sum_{\mathbf{x}_{ij}\in \mathrm{Conf}_{ij}(a)} \prod_{j'\in \mathbf{M}(i)\setminus\{j\}} q_{j'i}\left(x_{j'}\right). \tag{2.18}$$

With a little bit of arrangements, the CN output can be put in a simplified expression. Let $\tilde{x}_{j'} = h_{ij'}x_{j'}$, $\tilde{\mathbf{x}}_{ij} = \left[\tilde{x}_{j'}\right]_{j'\in \mathbf{M}(i)\setminus\{j\}}$ and $\tilde{a} = h_{ij}a$. Consider two new vectors $\tilde{\mathbf{q}}_{j'i}=\left[\tilde{q}_{j'i}\left(\tilde{x}_{j'}\right)\right]_{\tilde{x}_{j'}\in \mathrm{GF}(2^s)}$ and $\tilde{\mathbf{r}}_{ij} = \left[\tilde{r}_{ij}\left(\tilde{a}\right)\right]_{\tilde{a}\in \mathrm{GF}(2^s)}$, where $\tilde{q}_{j'i}\left(\tilde{x}_{j'}\right) = q_{j'i}\left(x_{j'}\right)$ and $\tilde{r}_{ij}\left(\tilde{a}\right) = r_{ij}\left(a\right)$. That means the messages incoming to the CN undergoes the following arrangement known as the *permutation* step:

$$\tilde{q}_{j'i}\left(a\right) = q_{j'i}\left(h_{ij'}^{-1}a\right) \quad \forall a \in \mathrm{GF}\left(2^s\right) \tag{2.19}$$

(2.18) can now be rewritten as

$$\tilde{r}_{ij}\left(\tilde{a}\right) = \sum_{\tilde{\mathbf{x}}_{ij}\in \mathrm{Conf}^*_{ij}(a)} \prod_{j'\in \mathbf{M}(i)\setminus\{j\}} \tilde{q}_{j'i}\left(\tilde{x}_{j'}\right), \tag{2.20}$$

where $\mathrm{Conf}^*_{ij}\left(a\right) = \left\{ \left[\tilde{x}_{j'}\right]_{j'\in \mathbf{M}(i)\setminus\{j\}} : \sum_{j'\in \mathbf{M}(i)\setminus\{j\}} \tilde{x}_{j'} = \tilde{a} \right\}$.

The CN processing step expressed above is a circular convolution operation and can be written in a compact form as in the following:

$$\tilde{\mathbf{r}}_{ij} = \bigodot_{j'\in \mathbf{M}(i)\setminus\{j\}} \tilde{\mathbf{q}}_{j'i}, \tag{2.21}$$

where $\bigodot$ is denotes the circular convolution.

Computation of $\tilde{\mathbf{r}}_{ij}$ can be efficiently done in the frequency domain through s-dimensional 2-point FFT. Thus, we have

$$\tilde{\mathbf{r}}_{ij} = \mathrm{FFT}^{-1} \bigodot_{j'\in \mathbf{M}(i)\setminus\{j\}} \mathrm{FFT}\left(\tilde{\mathbf{q}}_{j'i}\right), \tag{2.22}$$

where $\overset{\bullet}{\prod}$ denotes the term-by-term product operation.

An s-dimensional 2-point FFT is equivalent to the Hadamard transform H_q of order $q = 2^s$ [44]. Moreover, $H_q^{-1} = H_q$. Thus (2.22) can be written as

$$\tilde{\mathbf{r}}_{ij} = H_q \overset{\bullet}{\prod_{j' \in \mathbf{M}(i) \setminus \{j\}}} H_q \tilde{\mathbf{q}}_{j'i}. \tag{2.23}$$

To convert $\tilde{\mathbf{r}}_{ij}$ back into $\mathbf{r}_{ij}$, the following operation known as the *depermutation* step is carried out:

$$r_{ij}(a) = \tilde{r}_{ij}(h_{ij}a) \quad \forall a \in \mathrm{GF}(2^s). \tag{2.24}$$

(ii) **VN processing**: The message $\mathbf{q}_{ji} = [q_{ji}(a)]_{a \in \mathrm{GF}(2^s)}$ sent by v_j to c_i is updated as

$$q_{ji}(a) = \alpha_{ji} f_j^a \prod_{i' \in \mathbf{N}(j) \setminus \{i\}} r_{i'j}(a) \quad \forall a \in \mathrm{GF}(2^s), \tag{2.25}$$

where α_{ji} is a normalizing factor such that $\sum_{a \in \mathrm{GF}(2^s)} q_{ji}(a) = 1$.

(iii) **Computation of *a posteriori* probability**: The *a posteriori* probability term $\mathbf{q}_j = [q_j(a)]_{a \in \mathrm{GF}(2^s)}$ for v_j is computed as

$$q_j(a) = \alpha_j f_j^a \prod_{i \in \mathbf{N}(j)} r_{ij}(a) \quad \forall a \in \mathrm{GF}(2^s), \tag{2.26}$$

where α_j is a normalizing factor such that $\sum_{a \in \mathrm{GF}(2^s)} q_j(a) = 1$.

(iv) **Hard decision and stopping criterion**: Based on the *a posteriori* probability, the following hard decision is made:

$$\hat{x}_j = \arg \max_{a \in \mathrm{GF}(2^s)} q_j(a).$$

If a valid codeword is obtained i.e. $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then the decoding process is stopped, otherwise it is continued till the maximum number of iterations are executed.

Discussion

The iterative decoding of the NB-LDPC codes is usually described in the probability domain because of the feasibility of accomplishing the CN processing via the FFT. In the LLR domain, the FFT can no longer be directly utilized for the CN processing. LLR-domain decoding algorithms are

preferable for practical purposes as these are less susceptible to the quantization noise. One popular and low-complexity LLR domain decoding scheme is the min-sum (MS) algorithm [77]. The complexity of the CN processing in the MS decoding remains at $O\left(q^2\right)$. In order to reduce the complexity of the MS algorithm, Declercq and Fossorier have proposed the extended MS (EMS) algorithm [20]. The EMS decoding algorithm considers only a few number (n_m) of elements out of the q elements of a vector message incident at a CN. This adjustment reduces the complexity of the CN processing to $O\left(n_m q\right)$. Since a limited number of elements are considered, the accuracy of the EMS decoding may be less than that of the FFT-QSPA.

2.8 Conclusions

This chapter reviews the basic concepts, the construction and the decoding algorithms for the binary LDPC codes and a decoding algorithm for the NB-LDPC codes relevant to the works presented in the thesis. First, the Tanner graph representation of a parity-check matrix for an LDPC code is illustrated. The steps of the SPA, the commonly used algorithm for the decoding of the binary LDPC codes, are discussed. We describe the PEG and the ACE algorithms which are very useful to construct short-to-medium length LDPC codes having good cycle properties. The error floor problem and its causes are also briefly reviewed. The NB-LDPC codes along with their decoding process using the FFT-QSPA are described thereafter.

3

Finding Dominant Trapping Sets of Irregular LDPC codes

Contents

3.1 Introduction

As mentioned in the previous chapter, the error floor problem of the LDPC codes for an AWGN channel is attributed to the subgraphs of the Tanner graph associated with the so-called trapping sets (TSs) [58]. Although the Tanner graph may contain a large number of TSs, only a few of them are relevant in the context of the error floor. These TSs are commonly referred as the *dominant* TSs. The absorbing sets (ASs) [25] are the special subsets of the TSs and found to contribute significantly to the error floor. For tackling the error floor problem of the LDPC codes, the most crucial task is to find a list of the dominant TSs. A list of dominant TSs may help to design a better coding system in three ways:

- Using the knowledge of the dominant TSs, the Tanner graph can be modified to avoid the presence of some of the dominant TSs. For example, the authors in [7, 37] have performed edge swapping between the specific number of copies of the Tanner graph to obtain a larger Tanner graph which is devoid of some of the dominant TSs. The authors in [42] have proposed a modified PEG algorithm where each edge is carefully assigned by examining the newly formed dominant TSs. In [53], the authors have proposed a technique to construct structured regular codes of VN-degree 3 which are free of certain small and dominant TSs. The most important part of this technique is the method of finding the TSs.

- A list of the dominant TSs can be used to improve the decoding performance. Exploiting the knowledge of the dominant TSs, Han and Ryan [33] have proposed three decoders to lower the error floor. In order to improve the high SNR performance of a particular structured code, Zhang *et al.* [80] have devised a post-processing decoding step by considering the structures of the most dominant ASs.

- The list of the dominant TSs is useful for estimating the high SNR performance of the LDPC codes. Different importance sampling methods [13, 16, 25] have been proposed to estimate the high SNR performance by biasing the noise towards the dominant TSs.

The above discussion shows that devising an efficient technique to find the dominant TSs of a particular LDPC code is an important research task. This chapter proposes a method for the identification of the dominant TSs for the irregular LDPC codes.

The rest of the chapter is organized as follows. We present the definitions of various types of the TSs and the ASs in Section 3.2. Section 3.3 reviews some of the popular techniques to obtain the dominant TSs. One popular approach to find the dominant TSs known as the hierarchical approach, is further illustrated in detail in this section. Section 3.4 presents the proposed method of finding the dominant TSs of an irregular LDPC code. The results of the enumerations of the TSs and the ASs of various commonly considered irregular codes are presented in Section 3.5. The chapter is concluded with Section 3.6.

3.2 Definitions

Suppose $\mathbf{H}$ is the parity-check matrix of an LDPC code. The corresponding Tanner graph $G_{\mathbf{H}} = (V \cup U, E)$ contains the sets V of the VNs, U of the CNs and E of the edges. Consider a subset $T \subset V$. Let $\mathcal{N}(T) \subset U$ denote the set of the neighboring CNs connected to the VNs in T. The subgraph S_T induced by the VNs in T contains the nodes $T \cup \mathcal{N}(T)$ and the edges $\{(v,c) \in E : v \in T, c \in \mathcal{N}(T)\}$. Let $\mathcal{E}(T)$ and $\mathcal{O}(T)$ denote the even-degree and odd-degree CNs respectively present in S_T. With these notations, we first present the commonly considered definition of a TS [33, 41, 52].

Definition 3.1. *A subset T of V is an (a,b) trapping set (TS) if $|T| = a$ and $|\mathcal{O}(T)| = b$. T is called elementary if all CNs in $\mathcal{O}(T)$ and $\mathcal{E}(T)$ are of degree 1 and 2, respectively (the degrees being measured in the context of the subgraph S_T).*

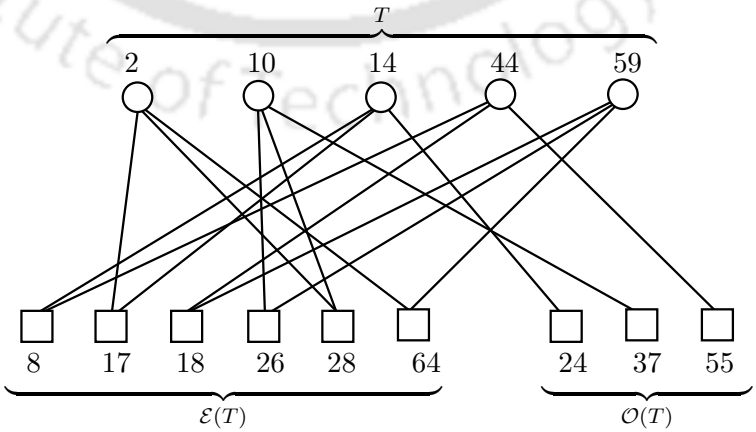


Figure 3.1: A (5,3) TS from the $N = 120, M = 64$ code (120.64.3.109) [48]

A $(5,3)$ TS from the $N = 120, M = 64$ code (120.64.3.109) available at [48] is shown in Figure 3.1.

The sets T, $\mathcal{E}(T)$ and $\mathcal{O}(T)$ are also shown in the figure. This TS is of elementary type as all the CNs in $\mathcal{O}(T)$ and $\mathcal{E}(T)$ are of degree 1 and 2, respectively. The number of VNs present in a TS is usually referred to as the size of the TS. For example, the size of the $(5,3)$ TS is 5. Although a TS denotes a set of VNs, we will use the term "trapping set" for both the set of VNs and the subgraph induced by the VNs interchangeably.

If all the bits or VNs in the TS are detected incorrectly then the even-degree CNs will be *mis-satisfied* and the odd-degree CNs will be *unsatisfied*. During the decoding process, the mis-satisfied CNs will send wrong-signed messages and the unsatisfied CNs will send correct-signed messages towards the neighboring VNs. If the number b of unsatisfied CNs is very small compared to the number of mis-satisfied CNs, then a large number of incorrect messages will be flowing within the TS. In that case, the decoder will never be able to come out of the incorrect state even if the number of iterations is increased. In this way, if a and b are small and all the VNs of the TS are detected incorrectly, then it is almost impossible for the iterative decoder to correct the decoding decisions [33,39,40]. Such TSs with relatively small values of a and b are usually considered as the dominant ones.

Definition 3.1 specifies only the number of VNs $|T| = a$ and the number of odd-degree CNs $|\mathcal{O}(T)| = b$. The even-degree CNs in $\mathcal{E}(T)$ also play a crucial role in determining the harmfulness of a TS. In order to consider the effect of the even-degree CNs, a modified definition of a TS has been used in [41].

Definition 3.2. *A subset T of V is an (a,b) TS if the following conditions are satisfied:*

(i) $|T| = a$.

(ii) $|\mathcal{O}(T)| = b$.

(iii) Each VN of T with a degree more than 2 is connected to at least two CNs in $\mathcal{E}(T)$.

(iv) Each VN of T with a degree equal to 2 is connected to at least one CN in $\mathcal{E}(T)$.

The (5,3) TS shown in Figure 3.1 is a legitimate TS according to Definition 3.2 as all the VNs in T are of degree 3 and are connected to at least two CNs in $\mathcal{E}(T)$. In the rest of the chapter, Definition 3.2 will be used for a TS unless otherwise specified.

Dolecek *et al.* have formulated the concept of the absorbing set (AS) [24] which is defined in the following:

Definition 3.3. *An (a, b) trapping set T is called an absorbing set (AS) if each VN in T is connected to strictly more CNs in $\mathcal{E}(T)$ than in $\mathcal{O}(T)$. An absorbing set T is called a fully absorbing set (FAS) if each VN in $V \setminus T$ is connected to strictly more CNs in $U \setminus \mathcal{O}(T)$ than in $\mathcal{O}(T)$.*

The (5,3) TS in Figure 3.1 is an AS as each VN is connected to strictly more even-degree CNs than odd-degree CNs. To check whether an AS is an FAS or not, the whole Tanner graph needs to be examined. Note that any AS is a TS but the converse is not always true. The ASs form a subset of the collection of TSs. Each VN in an AS receives strictly more incorrect messages than correct messages. Consequently, the chances of the reversal of the erroneous decisions at those VNs become low. Hence, the ASs are the most detrimental classes of TSs.

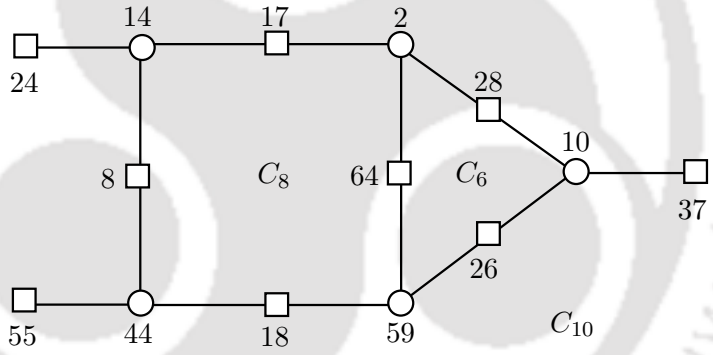


Figure 3.2: A different representation of the (5,3) TS in Figure 3.1

Almost each dominant TS contains one or more short cycles [41, 79]. For example, consider the (5,3) TS shown in Figure 3.1. For better visualization of the cycles, the TS is redrawn in Figure 3.2. It can be observed that the (5,3) TS contains two cycles: C_6 of length 6 (2-28-10-26-59-64) and C_8 of length 8 (14-17-2-64-59-18-44-8). Observe that the subgraph in Figure 3.2 contains an outer cycle C_{10} of length 10 (14-17-2-28-10-26-59-18-44-8). However, the nodes and the edges in C_{10} are already included in C_6 and C_8. Therefore, the TS is considered to be constituted by only C_6 and C_8.

Note that any codeword of weight a can be considered as an $(a, 0)$ TS as there is no odd-degree CN involved. Therefore, the codewords can be considered as special type of TSs. The next section briefly reviews the various methods available in the literature to find the dominant TSs.

3.3 A Brief Literature Review on the Methods to Find Trapping Sets in LDPC Codes

Several methods have been proposed so far to obtain the dominant TSs of an LDPC code. These methods can be classified into two broad categories: *simulation based* and *graph search based*.

In the simulation based methods, the decoding algorithm is simulated to obtain the TSs. In [16], a deterministic error impulse is applied to some specific bits and the resultant vector is fed to the decoder. The set of the erroneous VNs at the decoder output after the maximum number of iterations is considered as a TS and the process is repeated several times to collect more TSs. Abu-Surra *et al.* [3] have devised a graph-transformation strategy in order to convert the TSs of the original Tanner graph to the codewords in the transformed Tanner graph. Then, a simulation-based low-weight codeword enumeration technique [21] is performed over the transformed graph in order to find the dominant TSs of the original graph. We refer this technique as the "*ADDR* method" (after the authors Abu-Surra, DeClercq, Divsalar, and Ryan). In [13], the authors have proposed a scheme where the noise densities corresponding to the VNs involved in the shortest cycles are biased and the resulting decoder failures are accumulated as TSs. The advantage of the simulation based techniques is that they find the TSs where the decoder actually fails. However, this approach is less likely to produce an exhaustive list of the dominant TSs and also the time required for finding the TSs is very high.

In the graph search approach, specific subgraphs of the Tanner graph which correspond to the TSs are found. These techniques may be further classified into two groups: *brute force* and *hierarchical*. In [74] and [45], the authors have proposed several brute force algorithms to exhaustively find the TSs and the ASs, respectively. The complexities of these algorithms are very high. Therefore, these algorithms can find only the small TSs/ASs and that too only of the small codes. In the hierarchical techniques, smaller TSs are expanded by adding some specific VN/VNs to obtain larger TSs. Recently, the hierarchical approach has been considered by a number of authors [41, 53].

We focus on the hierarchical approach which is described in detail below.

3.3.1 Hierarchical Approach to Find the Trapping Sets

In the hierarchical approach, the smaller TSs are recursively enlarged by adding some specific VNs to obtain the larger TSs [41, 53]. A larger TS obtained from a smaller TS is called the *successor* of

the smaller one and the smaller one is called the *predecessor* of the larger one. For example, consider the (5,3) TS of the $N = 120, M = 64$ code shown in Figure 3.2. This TS can be obtained by adding one VN to the (4,4) TS as shown in Figure 3.3. The predecessor TS is shown in solid lines and the subgraph appended is drawn in dashed lines. All the dominant TSs are assumed to be of elementary type and therefore, the VN/VNs to be added must connect only to the degree-1 (odd-degree) CNs of the predecessor TS.

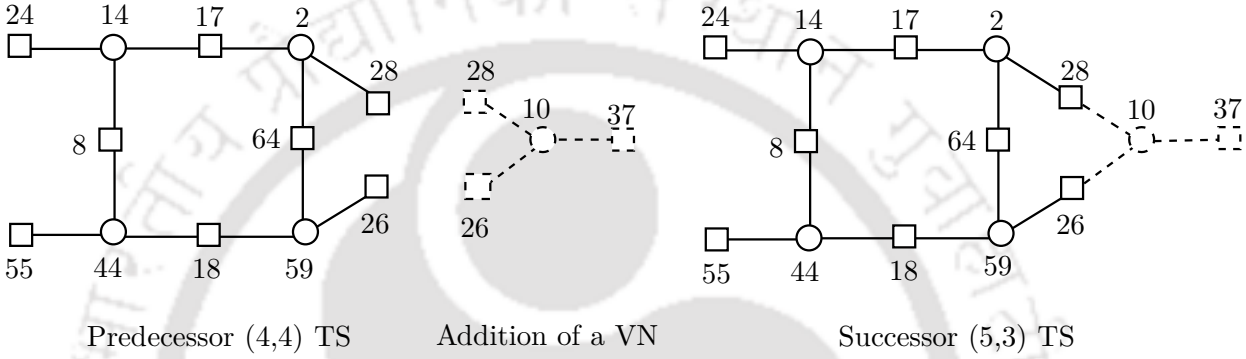
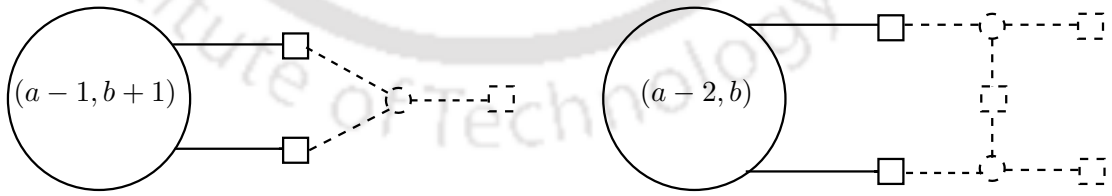


Figure 3.3: Expansion of a (4,4) TS to a (5,3) TS

Nguyen *et al.* [53] have presented a method based on the hierarchical approach to find the dominant TSs of the regular codes of VN-degree 3. They have carried out the expansion by adding at most 2 VNs to the predecessor TS. As the VNs are of degree 3, they have considered two kinds of predecessor TSs $(a-1, b+1)$ and $(a-2, b)$ to obtain the (a, b) TSs. These expansions are pictorially depicted in Figure 3.4.

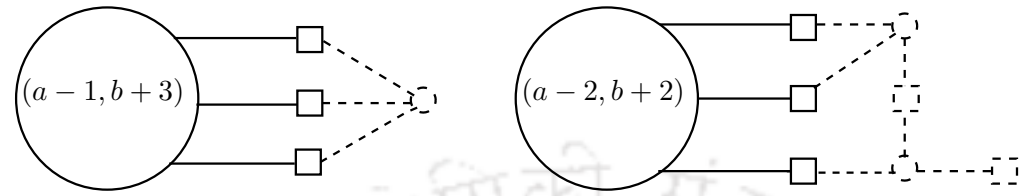


(a) An (a, b) TS from an $(a-1, b+1)$ TS **(b)** An (a, b) TS from an $(a-2, b)$ TS

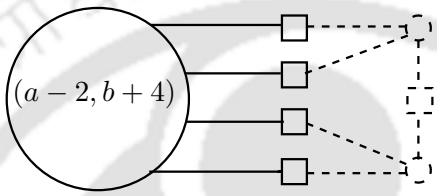
Figure 3.4: Types of expansions considered for the regular codes of VN-degree 3 in [53]

The sufficiency of the two types of expansions shown in Figure 3.4 is verified in the following. The expansion of a predecessor TS involves the following two constraints: (1) the expanded subgraph must satisfy Definition 3.2 for it to be a legitimate TS and (2) a predecessor TS is expanded by adding a maximum of 2 VNs at a time. By keeping these constraints in mind, we figure out the other feasible expansions. Figure 3.5 depicts three such expansions. Unlike those in Figure 3.4, these expansions

involve more than two connections between the subgraph to be added and the odd-degree CNs in the predecessor TS. The redundancy of these expansions are validated in the following.



(a) An (a, b) TS from an $(a-1, b+3)$ TS **(b)** An (a, b) TS from an $(a-2, b+2)$ TS



(c) An (a, b) TS from an $(a-2, b+4)$ TS

Figure 3.5: Additional expansions for the regular codes of VN-degree 3 which are found to be redundant

First consider the expansion shown in Figure 3.5(a). This case is redundant as it is taken into account by the two expansions shown in Figure 3.4. As an example, consider the (10,2) TS of a girth-8 code shown in Figure 3.6.

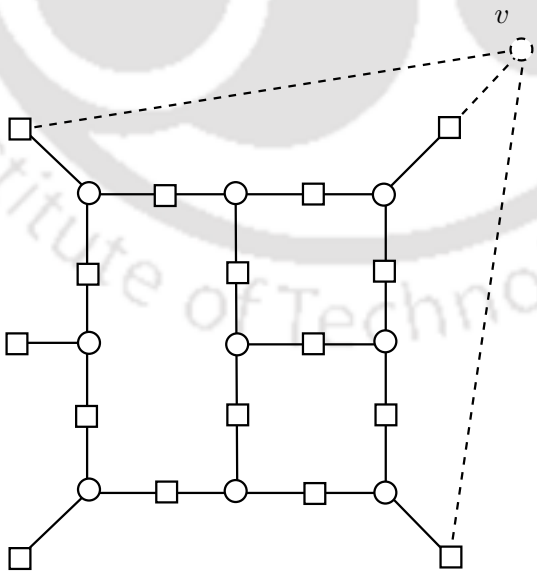


Figure 3.6: A (10,2) TS that can be obtained from a (9,5) TS according to the expansion shown in Figure 3.5(a)

This TS can be obtained from the (9,5) TS by adding the VN v following the expansion type shown in Figure 3.5(a). Note that the VN v has three links to the predecessor (9,5) TS. However, after a

careful examination of the structure of the TS, it can be observed that the same (10,2) TS can be reached from a different TS by employing only those expansions having two links to the predecessor. One possible sequence of expansions is shown in Figure 3.7. The sequence starts from the (4,4) TS which is generated by a cycle of length 8.

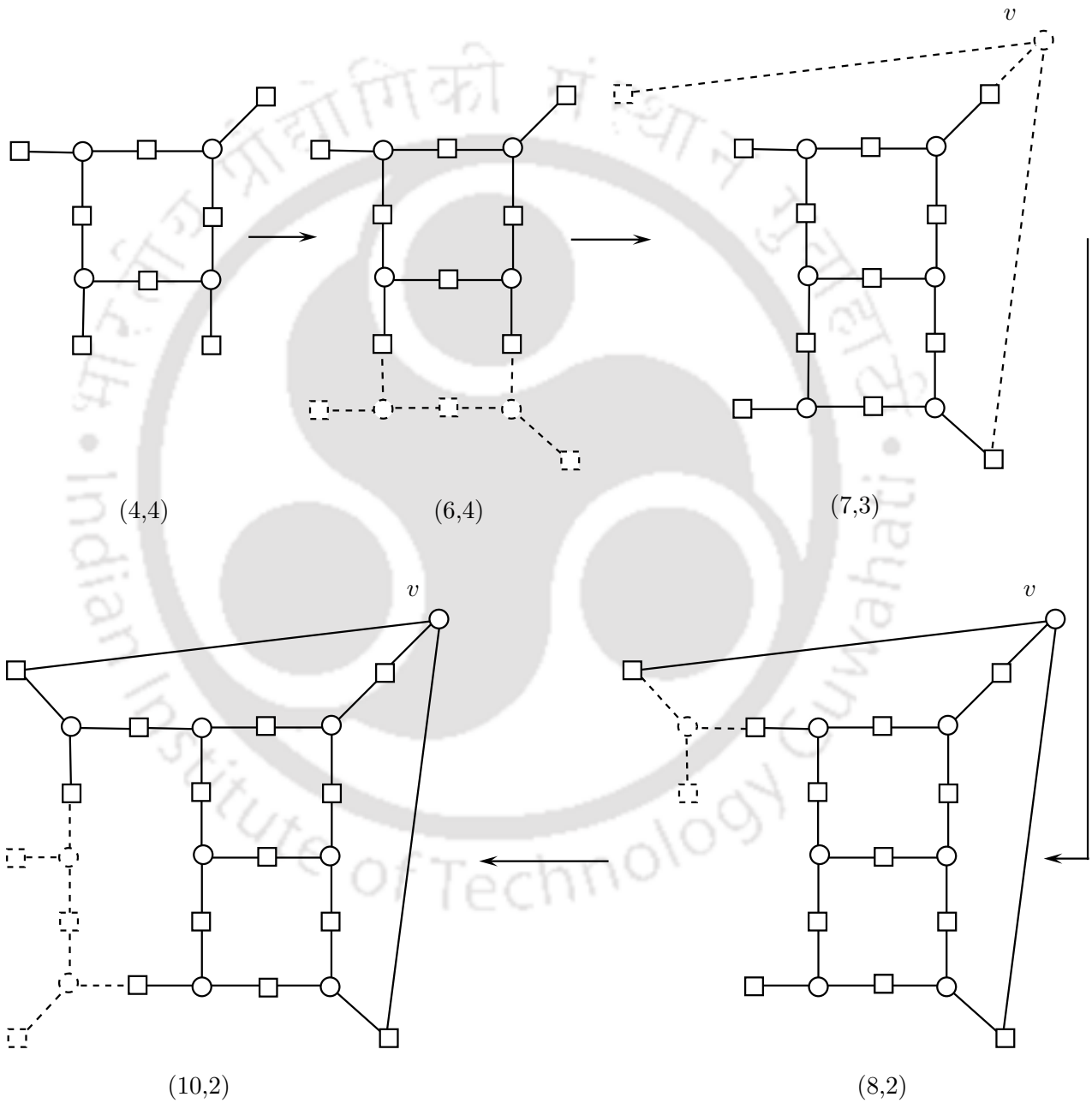


Figure 3.7: A sequence of expansions that leads to the (10,2) TS where only the expansions shown in Figure 3.4 are considered

Since, the (10,2) TS can be obtained in this way, the expansion of the (9,5) TS as shown in Figure 3.6 is not required. Simulations of several codes have confirmed that the expansion shown

in Figure 3.5(a) is redundant. However, this expansion is very useful for finding the codewords as explained in [53]. Recall from Section 3.2 that any codeword of weight a can be considered as an $(a, 0)$ TS. It is shown in [53] that by adding a VN to an $(a-1, 3)$ TS according to the expansion shown in Figure 3.5(a), a codeword of weight a can be obtained.

The redundancy of the expansion shown in Figure 3.5(b) can be established easily. This particular type of expansion can be realized by performing the expansion shown in Figure 3.4(a) sequentially for each of the two VNs to be added. Similarly, the expansion shown in Figure 3.5(c) is also redundant as it can be realized via the concatenation of the expansions shown in Figure 3.4(a) and Figure 3.5(a) sequentially.

Therefore, by considering only the two types of expansions shown in Figure 3.4, Nguyen *et al.* have been able to find almost as many TSs as the exhaustive methods can yield in a very less amount of time for the regular codes of VN-degree 3 [53].

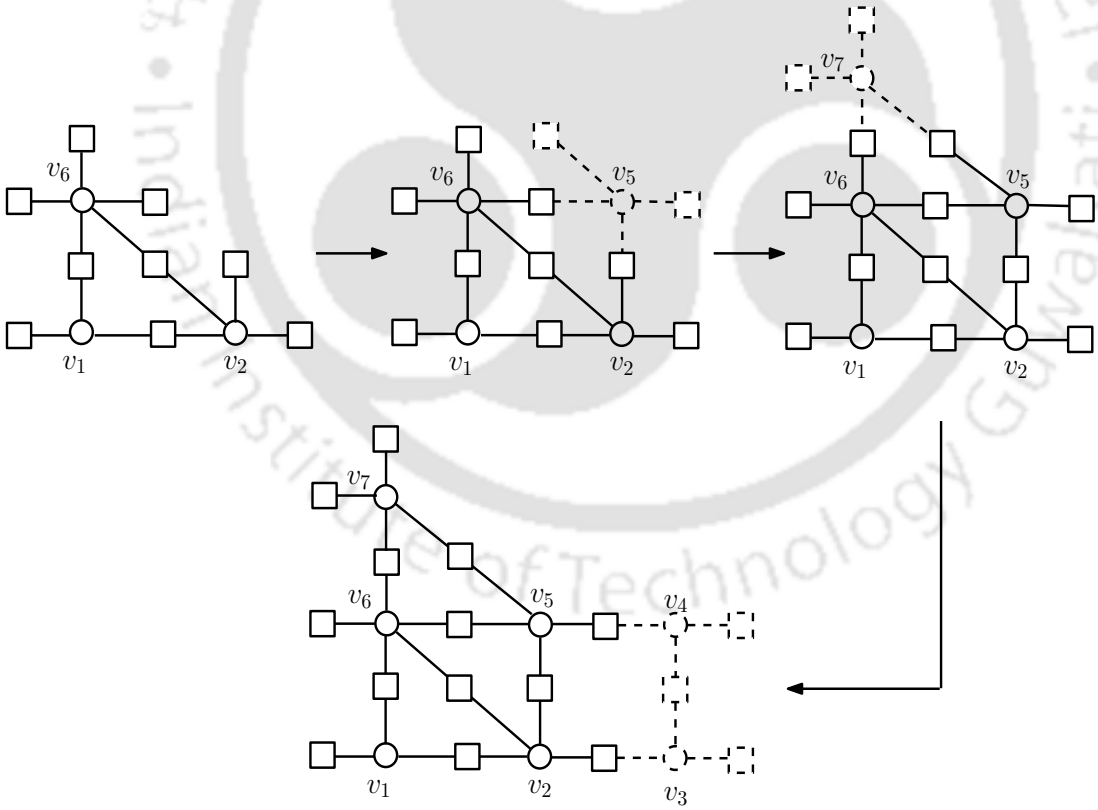


Figure 3.8: Expansion of a $(3, 6)$ TS $\{v_1, v_2, v_6\}$ to a $(7, 6)$ TS $\{v_1, v_2, v_6, v_5, v_7, v_3, v_4\}$ [41]: the (6,6) TS $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ is missed

Karimi and Banihashemi have proposed an algorithm based on the hierarchical approach which is applicable to any regular or irregular code [41]. This algorithm will be referred by "*KB* algorithm" in the rest of this chapter. The *KB* algorithm has been able to find almost all the dominant TSs and ASs

of some popular codes available in the literature. In this algorithm, out of various candidate subgraphs, the one which contains the minimum number of VNs, is selected for appending to the predecessor. Because of this, a specific type of intermediate TS is missed while expanding a specific smaller TS to a larger one. The authors in [41] have themselves presented one such example which is reproduced in Figure 3.8. The algorithm will reach the $(7, 6)$ TS $\{v_1, v_2, v_6, v_5, v_7, v_3, v_4\}$ from the $(3, 6)$ TS $\{v_1, v_2, v_6\}$ in the following sequence $\{v_1, v_2, v_6\} \rightarrow \{v_1, v_2, v_6, v_5\} \rightarrow \{v_1, v_2, v_6, v_5, v_7\} \rightarrow \{v_1, v_2, v_6, v_5, v_7, v_3, v_4\}$. This sequence implies that the algorithm missed the (6,6) TS $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ which is specifically mentioned in [41].

We propose a hierarchical method to find the dominant TSs and ASs of the irregular codes addressing all the issues. The method is presented in the next section.

3.4 Proposed Method of Finding the Dominant Trapping Sets

The proposed hierarchical method generalizes the technique in [53] to irregular codes. We also consider the expansion by adding at most 2 VNs at a time. For the irregular codes, the patterns of the VN/VNs to be added to the predecessor TS may be diverse because of the different kinds of possible degrees of the VNs. In other words, one may have to consider several types of expansions in order to find the dominant TSs of an irregular code.

In order to devise a systematic approach of finding the dominant TSs of an irregular code, we consider the following important points:

- The dominant TSs of the irregular codes are usually formed by the low-degree VNs, in particular, mostly by the degree-2 VNs [41]. The low-degree VNs contribute more to the error floor as they are more vulnerable because of the reception of less number of extrinsic messages during the decoding process. As the low-degree VNs are more likely to be part of the dominant TSs, we consider the expansion by adding only the degree-2 and degree-3 VNs.

- As discussed in Section 3.3.1, the expansions involving more than two links between the predecessor and the subgraph to be appended to it, are usually redundant. Therefore, we consider to expand a predecessor TS through a maximum of two links.

- The successor must be a valid TS as per Definition 3.2 and should be of elementary type.

With the constraints mentioned above, we analyze all possible expansions of the following two types resulting in an (a, b) TS:

(a) **Expansions via the Addition of One VN**

The following $(a-1, *)$ predecessors can be expanded to obtain (a, b) TSs by adding one VN:

(i) $(a-1, b)$ *TSs:*

An $(a-1, b)$ TS can be expanded to an (a, b) TS by adding a VN of degree 2 sharing one odd-degree CN of the $(a-1, b)$ TS as shown in Figure 3.9. Note that the resultant

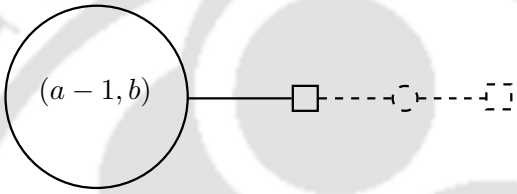


Figure 3.9: An (a, b) TS from an $(a-1, b)$ TS

TS is a valid one since a degree-2 VN may be connected to only one even-degree CN as per Definition 3.2. For the predecessor TS, we show only those odd-degree CNs which are involved in the expansion.

(ii) $(a-1, b+2)$ *TSs:*

An $(a-1, b+2)$ TS can be expanded to an (a, b) TS by adding a VN of degree 2 sharing two odd-degree CNs of the $(a-1, b+2)$ TS as shown in Figure 3.10.

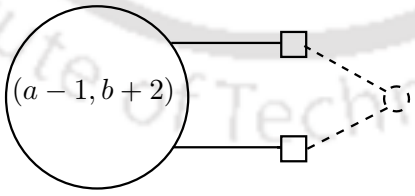


Figure 3.10: An (a, b) TS from an $(a-1, b+2)$ TS

(iii) $(a-1, b+1)$ *TSs:*

An $(a-1, b+1)$ TS can be expanded to an (a, b) TS by adding a VN of degree 3 sharing two odd-degree CNs of the $(a-1, b+1)$ TS as shown in Figure 3.11. Note that this type of expansion has been considered in [53] for the regular codes of VN-degree 3 as mentioned in Section 3.3.1.

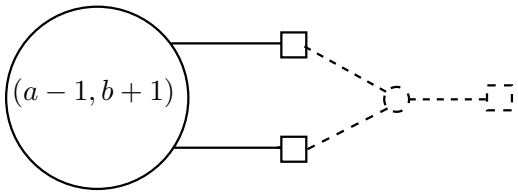


Figure 3.11: An (a, b) TS from an $(a-1, b+1)$ TS

(b) **Expansions via the Addition of Two VNs**

The following $(a-2, *)$ predecessors can be enlarged by adding two VNs to obtain (a, b) TSs:

(iv) $(a-2, b)$ *TSs:*

An $(a-2, b)$ TS can be expanded to an (a, b) TS by adding two VNs of degree 2 such that they share one CN among each other and any one of them is connected to one odd-degree CN of the $(a-2, b)$ TS as shown in Figure 3.12.

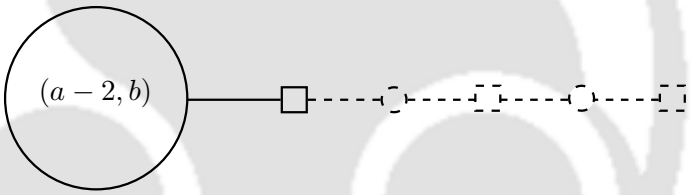


Figure 3.12: An (a, b) TS from an $(a-2, b)$ TS

(v) $(a-2, b-1)$ *TSs:*

An $(a-2, b-1)$ TS can be expanded to an (a, b) TS by adding a degree-3 and a degree-2 VN such that they share one CN between them and the degree-3 VN shares one odd-degree CN of the $(a-2, b-1)$ TS. This situation is described in Figure 3.13.

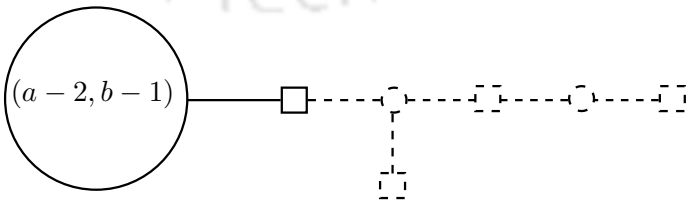


Figure 3.13: An (a, b) TS from an $(a-2, b-1)$ TS

(vi) $(a-2, b+1)$ *TSs:*

An $(a-2, b+1)$ TS can be expanded to an (a, b) TS by adding a degree-3 and a degree-2 VN, such that these two VNs share one CN among them and the degree-3 VN shares two

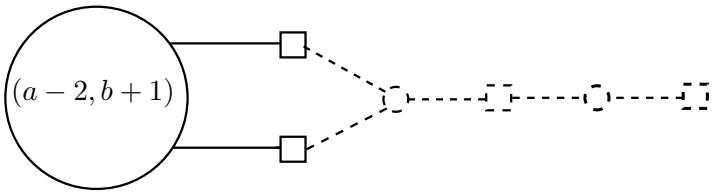


Figure 3.14: An (a, b) TS from an $(a-2, b+1)$ TS

odd-degree CNs of the $(a-2, b+1)$ TS as shown in Figure 3.14.

(vii) $(a-2, b+2)$ *TSs:*

An $(a-2, b+2)$ TS can be expanded to an (a, b) TS by adding two VNs of degree 2 such that they share one CN among them and each of them is connected to one odd-degree CN of the $(a-2, b+2)$ TS as shown in Figure 3.15.

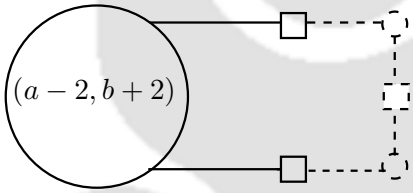


Figure 3.15: An (a, b) TS from an $(a-2, b+2)$ TS

(viii) $(a-2, b+1)$ *TSs:*

An $(a-2, b+1)$ TS can be expanded to an (a, b) TS by adding a degree-2 and a degree-3 VN, such that these two VNs share one CN among them and each of them shares one odd degree CN of the $(a-2, b+1)$ TS as shown in Figure 3.16.

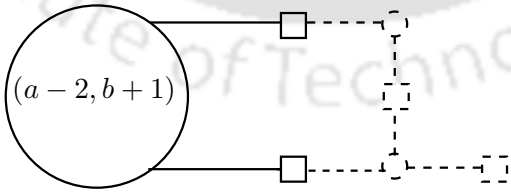


Figure 3.16: An (a, b) TS from an $(a-2, b+1)$ TS

(ix) $(a-2, b)$ *TSs:*

An $(a-2, b)$ TS can be expanded to an (a, b) TS by adding two VNs of degree 3 sharing one CN among them and each of them shares one odd-degree CN of the $(a-2, b)$ TS as shown in Figure 3.17. Note that this type of expansion has been considered in [53] for the regular codes of VN-degree 3 as mentioned in Section 3.3.1.

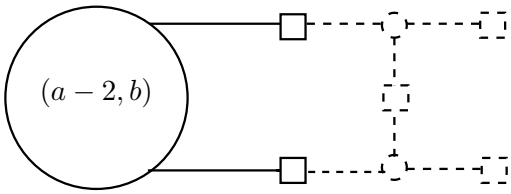
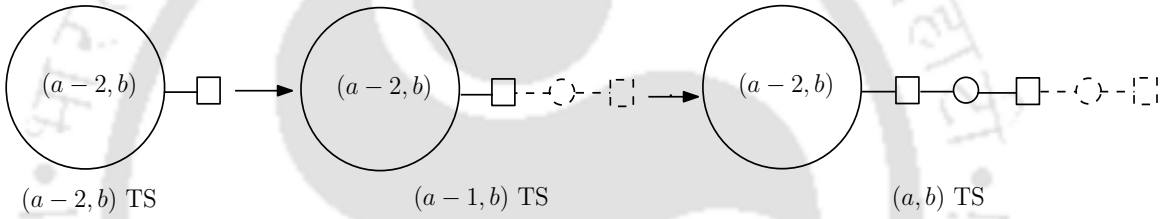
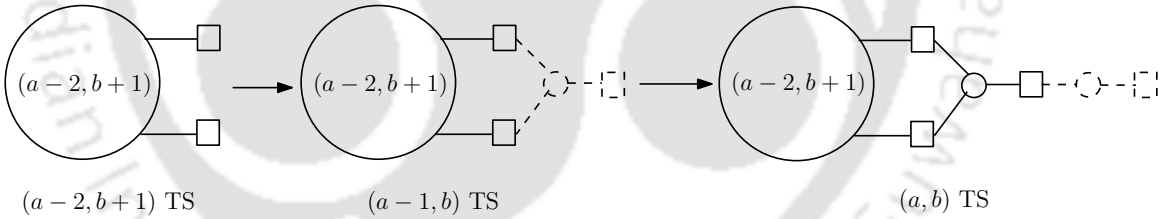


Figure 3.17: An (a,b) TS from an $(a-2,b)$ TS

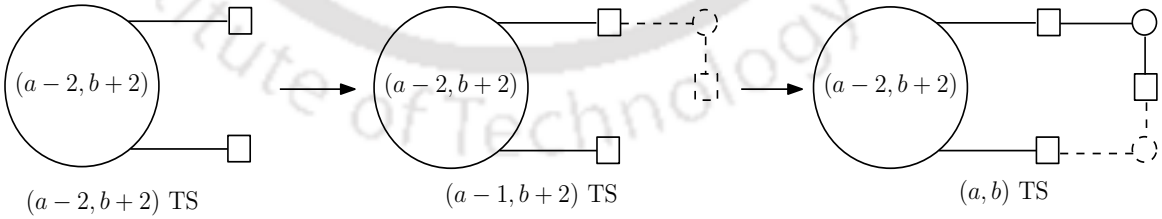
We have enlisted three expansions in (a) and six expansions in (b). Some of the expansions via the addition of two VNs can be realized by means of the concatenation of two expansions via the addition of one VN. These expansions will not produce any new TSs and therefore, are redundant. Out of the six types of expansions in (b), (iv), (vi), (vii) and (viii) are redundant. The expansion (iv)



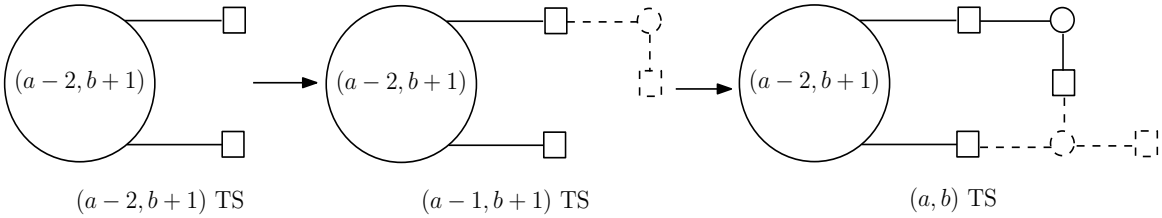
(a) An (a,b) TS from an $(a-2,b)$ TS via an $(a-1,b)$ TS



(b) An (a,b) TS from an $(a-2,b)$ TS via an $(a-1,b)$ TS



(c) An (a,b) TS from an $(a-2,b+2)$ TS via an $(a-1,b+2)$ TS



(d) An (a,b) TS from an $(a-2,b+1)$ TS via an $(a-1,b+1)$ TS

Figure 3.18: Pictorial description of the redundant expansions

can be realized by sequentially performing the expansion (i) twice as shown in Figure 3.18(a). The expansion (vi) can be obtained by performing the expansions (iii) and (i) sequentially and is shown in Figure 3.18(b). The expansion (vii) is equivalent to the concatenation of the expansions (i) and (ii) as shown in Figure 3.18(c). Similarly, the expansion (viii) can be realized by the concatenation of the expansions (i) and (iii) sequentially as shown in Figure 3.18(d).

The TSs formed by a single cycle, sometimes, may not be obtained by the expansions shown above. Any length-$2a$ cycle containing b degree-1 (odd-degree) CNs is an (a, b) TS. One such example is a (3,3) TS formed by a cycle of length 6 involving three degree-3 VNs as shown in Figure 3.19.

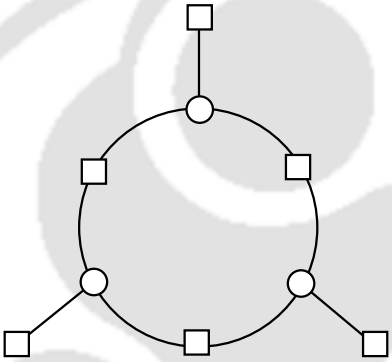


Figure 3.19: A (3,3) TS formed by a cycle of length 6

Necessary Sources for (a, b) TSs

Based on the above discussion, we consider a total of six types of sources for (a, b) TSs :

$\mathcal{S}_1^{(a,b)}$: $(a-1, b)$ TSs as shown in Figure 3.9

$\mathcal{S}_2^{(a,b)}$: $(a-1, b+2)$ TSs as shown in Figure 3.10

$\mathcal{S}_3^{(a,b)}$: $(a-1, b+1)$ TSs as shown in Figure 3.11

$\mathcal{S}_4^{(a,b)}$: $(a-2, b-1)$ TSs as shown in Figure 3.13

$\mathcal{S}_5^{(a,b)}$: $(a-2, b)$ TSs as shown in Figure 3.17

$\mathcal{S}_6^{(a,b)}$: Length-$2a$ cycles involving b degree-1 (odd-degree) CNs

These sources can be expanded to (a, b) TSs as explained in (a) and (b) above. To illustrate different types of sources for a particular class of TSs, several possible configurations of (10,2) TSs

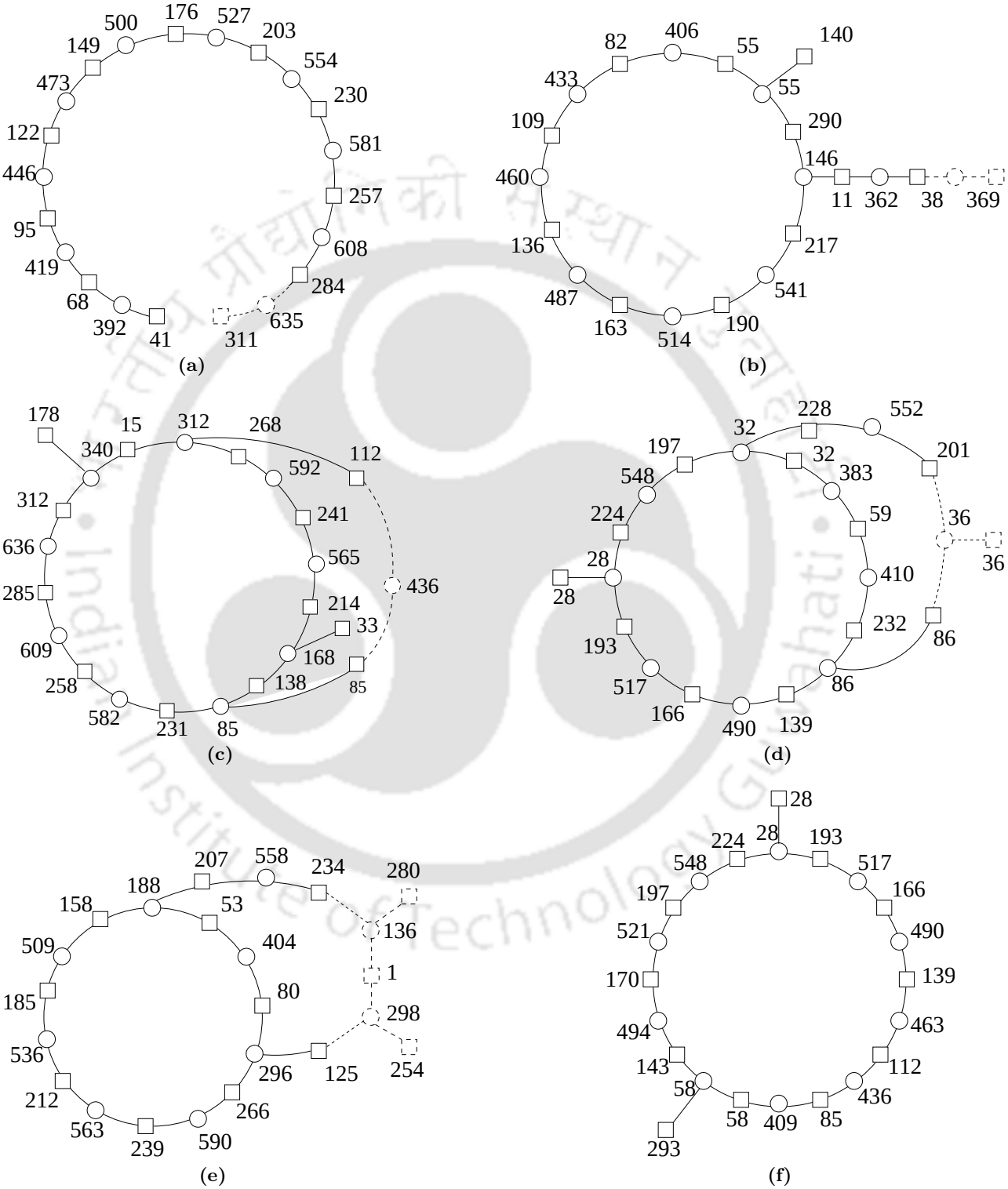
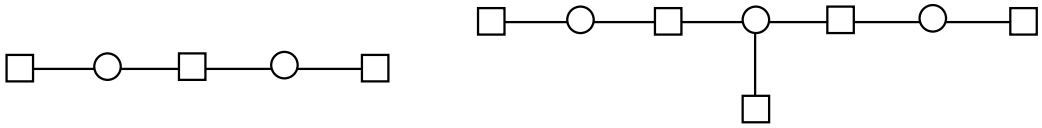


Figure 3.20: Different types of (10,2) TSs of the IEEE 802.11n code with $N = 648, M = 324$ [1]

of the IEEE 802.11n code [1] with $N = 648, M = 324$ are shown in Figure 3.20. Figure 3.20(a) and 3.20(b) show the expansions of two different forms of (9,2) TSs as per the type $(a-1, b) \rightarrow (a, b)$ shown in Figure 3.9. Figure 3.20(c) illustrates the expansion of a (9,4) TS according to the type $(a-1, b+2) \rightarrow (a, b)$ depicted in Figure 3.10. The (10,2) TS in Figure 3.20(d) can be obtained from a (9,3) predecessor TS as per the expansion of the type $(a-1, b+1) \rightarrow (a, b)$ shown in Figure 3.11. This code does not contain any $(8, 1)$ TS. Therefore, there is no $(10, 2)$ TS that can be obtained as per the expansion $(a-2, b-1) \rightarrow (a, b)$. Figure 3.20(e) shows the expansion of an (8,2) TS in accordance with the expansion $(a-2, b) \rightarrow (a, b)$ illustrated in Figure 3.17. Finally, Figure 3.20(f) shows a (10,2) TS formed by a cycle of length 20.

Algorithmic Steps

The procedure of finding the dominant TSs of an irregular LDPC code is described in Algorithm 3.1. The (a, b) TSs with small a and b are usually dominant. Suppose, we are interested in finding the (a, b) TSs with $a \leq a_m$ and $b \leq b_m$, where a_m and b_m are pre-determined. The major prior task before the execution of Algorithm 3.1 is the identification of the cycles of length up to $2a_m$. These cycles are found out in a brute-force manner. The higher-degree VNs are less probable to be involved in the dominant TSs. Therefore, in order to reduce the complexity of the identification of the cycles and the dominant TSs, the columns of higher weights are removed from the parity-check matrix. Let $\mathcal{T}_a$ denote the set of the TS classes of the form $(a, *)$. The process of obtaining the larger TSs from the smaller ones is initiated from the smallest predecessor TSs. As per Definition 3.2, the smallest possible TS for an irregular code is of the class (2,2). It is formed by two degree-2 VNs sharing one common CN between them as shown Figure 3.21(a). It can be verified that the only legitimate TS class of the form $(2, *)$ is (2,2) provided the Tanner graph does not contain any 4-cycle. Thus, we have $\mathcal{T}_2 = \{(2, 2)\}$. Algorithm 3.1 takes the list of TSs of the class (2,2) as one of the inputs. Apart from



(a) A $(2, 2)$ TS **(b)** A $(3, 3)$ TS

Figure 3.21: Initial trapping sets

this class, the (3,3) TSs of the structure shown in Figure 3.21(b) are also considered as the initial TSs.

Algorithm 3.1: Finding the Dominant TSs of an Irregular Code

input : Tanner graph, the maximum VN size a_m and the maximum number b_m of odd-degree CNs of the TSs to be found out, the list of cycles of length up to $2a_m$, the list of $(2,2)$ and $(3,3)$ TSs of the structures shown in Figure 3.21

output: The list of (a,b) TSs with $a \leq a_m$ and $b \leq b_m$

`// Phase 1 : Identification of the necessary TS classes`

• Initialize $\mathcal{T}_a = \phi$, for $a = 3, \ldots, a_m - 1$, $\mathcal{T}_2 = \{(2,2)\}$ and $\mathcal{T}_{a_m} = \{(a_m, 1), \cdots, (a_m, b_m)\}$;

for $a \leftarrow a_m$ **down to** 4 **do** `// a_m is usually larger than 4`

  • Construct the set $\mathcal{L}^1$ of the classes of the form $(a-1, *)$ needed for finding the TSs of each class (a,b) in $\mathcal{T}_a$. For that, consider the three expansions involving the addition of only one VN, i.e., the sources $\mathcal{S}_1^{(a,b)}$, $\mathcal{S}_2^{(a,b)}$ and $\mathcal{S}_3^{(a,b)}$;

  • Set $\mathcal{T}_{a-1} = \mathcal{T}_{a-1} \cup \mathcal{L}^1$;

  if $a > 4$ **then** `// as T_2 is already fixed`

    • Construct the set $\mathcal{L}^2$ of the classes of the form $(a-2, *)$ needed for finding the TSs of each class (a,b) in $\mathcal{T}_a$. For that, consider the two expansions involving the addition of two VNs as per $\mathcal{S}_4^{(a,b)}$ and $\mathcal{S}_5^{(a,b)}$;

    • Set $\mathcal{T}_{a-2} = \mathcal{L}^2$;

  end

end

`// Phase 2 : Finding the TSs of the necessary classes`

• Include the $(2,2)$ and $(3,3)$ TSs having the structures shown in Figure 3.21 in the respective classes;

for $a \leftarrow 3$ **to** a_m **do**

  foreach *class* (a,b) *in* $\mathcal{T}_a$ **do**

    for $i = 1$ **to** 5 **do** `// look for possible expansions`

      • Consider each TS T of the predecessor class $\mathcal{S}_i^{(a,b)}$; `// already found out`

      if $i \leq 3$ **then** `// if S_i^(a,b) is of the form (a-1, *)`

        • Search for every lone VN such that the addition of it to the predecessor T produces an (a,b) TS. The expansion must follow the type associated with $\mathcal{S}_i^{(a,b)}$. Include the resulting successor TS in the list of (a,b) class.

      else `// if S_i^(a,b) is of the form (a-2, *)`

        • Search for every pair of two linked VNs of the pattern associated with $\mathcal{S}_i^{(a,b)}$ such that the appending of it to the predecessor T generates an (a,b) TS. Include the resulting successor in the list of (a,b) class.

      end

    end

    • Examine each cycle of length $2a$. If it involves b degree-1 (odd-degree) CNs, then include it in the list of (a,b) TS class as per the source type $\mathcal{S}_6^{(a,b)}$.

  end

end

• Enlist the (a,b) TSs with $a \leq a_m$ and $b \leq b_m$; `// dominant TSs`

These (2,2) and (3,3) TSs are exhaustively found out. The maximum size of the TSs to be identified is a_m and we consider $\mathcal{T}_{a_m} = \{(a_m, 1), \ldots, (a_m, b_m)\}$. In order to progressively find the TSs of all the

dominant classes in the range $\mathcal{T}_2 \rightarrow \mathcal{T}_{a_m}$, we may have to find the TSs of many intermediate classes apart from the dominant ones. That means several (a, b) TS classes with $2 < a < a_m$, $b > b_m$ may be necessary to build up the entire list of (a, b) TSs with $a \leq a_m, b \leq b_m$. The algorithm operates in two phases. In the *first* phase, all the necessary TS classes are identified for given a_m and b_m and included in the respective $\mathcal{T}_a$, $a = 3, \ldots, a_m - 1$. In the *second* phase, the TSs of all the necessary classes in $\mathcal{T}_a$ are found out starting from $a = 3$ to $a = a_m$. The six types of sources $\mathcal{S}_1^{(a,b)}, \ldots, \mathcal{S}_6^{(a,b)}$ are considered while finding the TSs of a particular (a, b) class. Finally, the TSs of the dominant classes are enlisted. Note that each TS considered in all the stages of the algorithm must be a valid elementary TS as per Definition 3.2. Because of this restriction, the TSs of many intermediate classes do not exist and thus the complexity of the search procedure remains low.

Remark 3.1. *For the degree-distributions* [59] *optimized for the rate-0.5 LDPC codes over the AWGN channel, the minimum degree of a VN is 2. The degree-2 VNs form 43-55% of the total number of VNs. If the minimum degree of a VN for a particular Tanner graph is not 2, then the above procedure needs to be adjusted accordingly.*

Remark 3.2. *The KB algorithm finds the dominant TSs at one stretch. During this process, the TSs of different classes are encountered at different times and they are included in the respective classes accordingly. On the other hand, the proposed method focuses on only one target class at a time. This*

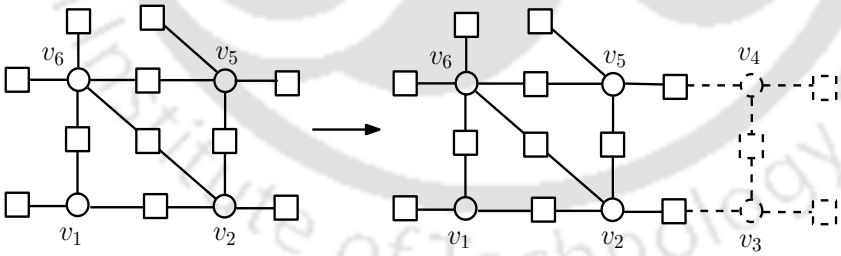


Figure 3.22: (6,6) TS $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ from (4,6) TS $\{v_1, v_2, v_6, v_5\}$ according to the expansion of the type $(a-2, b) \rightarrow (a, b)$ shown in Figure 3.17

aspect makes the method systematic. Specifically, we want to emphasize that the particular issue of the KB algorithm can be resolved with the proposed method. Recall from Figure 3.8 that the KB algorithm was unable to detect the (6,6) TS $\{v_1, v_2, v_3, v_4, v_5, v_6\}$. However, the proposed algorithm will be able to find it from the (4,6) TS $\{v_1, v_2, v_6, v_5\}$ as per the source $\mathcal{S}_5^{(6,6)}$. This process is pictorially depicted in Figure 3.22.

In the KB algorithm, the maximum number of VNs considered for appending to a predecessor TS may vary from code to code. It has been reported that in most of the cases, this number is 3 [41].

For the proposed algorithm, we fix this number at 2 for all codes. Therefore, the expansion process in the proposed algorithm is less complex than that in the KB algorithm. Although the method involves two phases, the complexity of the first phase is very less. Thus, the proposed method is simple and sufficient in the sense that it considers only six types of sources in an organized manner and yet finds almost all the dominant TSs.

3.5 Numerical Results for Enumeration of Trapping Sets

We apply the proposed technique to find the dominant TSs of three irregular codes: (a) $N = 504$, $M = 252$ PEG code (PEGirReg252x504), (b) $N = 1008$, $M = 504$ PEG code (PEGirReg504x1008) available in [48] and (c) $N = 648, M = 324$ code used in IEEE 802.11n standard [1]. The enumeration results are presented below:

(a) $N = 504, M = 252$ PEG code

This code is constructed by the PEG algorithm [34] with the degree distribution corresponding to the maximum VN degree of 15 [59]. The girth of the code is 6. We have discarded the columns

Table 3.1: The numbers of dominant trapping sets, absorbing sets and fully absorbing sets of $N = 504$, $M = 252$ PEG code obtained by the proposed algorithm and the *KB* algorithm [41]

Class	Proposed algorithm			*KB* algorithm [41]
	TS	AS	FAS	FAS
(3,2)	219	219	219	219
(4,2)	208	208	208	208
(5,2)	198	198	198	198
(6,2)	207	207	205	205
(7,1)	2	2	2	2
(7,2)	276	276	271	271
(8,1)	8	8	8	8
(8,2)	466	463	458	458
(9,1)	16	16	16	16
(9,2)	863	859	855	855
(10,1)	22	22	22	22
(10,2)	1596	1590	1565	1533

of the parity-check matrix which have more than five 1s per column. We consider $a_m = 10, b_m = 2$. The number of different classes of dominant TSs, ASs and FASs found out by the proposed technique along with the results of the *KB* algorithm are shown in the Table 3.1. The identification of the ASs and the FASs are done in accordance with Definition 3.3. However, as per this definition, a degree-2 VN must be connected to two even-degree CNs for it to be a constituent of an AS. That means a degree-2 VN, connected to one even-degree CN, cannot be a part of an AS. Many harmful structures may be omitted due to this strong condition. Therefore, the definition of an AS is slightly relaxed to include the degree-2 VNs connected to one even-degree CN as also done in the *KB* algorithm. All the enumerated TSs, ASs and FASs are of elementary type. It can be observed from Table 3.1 that the proposed technique is able to find all the dominant FASs as provided by the *KB* algorithm. Note that we have obtained more number of (10,2) FASs than that by the *KB* algorithm.

(*b*) $N = 1008, M = 504$ **PEG code**

This code is constructed by the PEG algorithm with the degree distribution corresponding to the maximum VN degree of 15 [59]. The girth of the code is 6. As above, we have discarded the columns

Table 3.2: The numbers of dominant trapping sets, absorbing sets and fully absorbing sets of $N = 1008$, $M = 504$ PEG code obtained by the proposed algorithm

Class	Proposed algorithm		
	TS	AS	FAS
(3,2)	439	439	439
(4,2)	420	420	420
(5,2)	404	404	404
(6,2)	388	388	388
(7,1)	1	1	1
(7,2)	406	406	403
(8,1)	4	4	4
(8,2)	524	522	519
(9,1)	8	8	8
(9,2)	803	799	795
(10,1)	14	14	14
(10,2)	1283	1282	1278

of the parity-check matrix which have more than five 1s per column.The number of different classes of dominant TSs, ASs and FASs found out by the proposed technique are shown in Table 3.2. All the enumerated TSs, ASs and FASs are of elementary type. In this example also, we relax the condition on the degree-2 VNs in the definitions of the ASs and the FASs.

(*c*) $N = 648, M = 324$ **IEEE 802.11n code**

This code is an example of a quasi-cyclic code. For such codes, the parity-check matrix can be divided into sub-blocks. Each sub-block or sub-matrix is obtained by shifting the columns of an identity matrix by a carefully selected value. For this example code, the size of a sub-block is 27×27. The degree distributions for this code are $\lambda(x) = 0.25x + 0.3409x^2 + 0.4091x^{11}$ and $\rho(x) = 0.6364x^6 + 0.3636x^7$. The girth of the code is 6. To find the cycles and the dominant TSs, we have discarded the columns of weight 12 from the parity-check matrix.

Table 3.3: The numbers of dominant trapping sets of $N = 648$, $M = 324$ IEEE 802.11n standard code obtained by the proposed algorithm and by the *ADDR* method [3]

Class	**Proposed algorithm**	*ADDR* **method [3]**
	TS	TS
(3,2)	243	-
(4,2)	216	216
(5,2)	189	189
(5,3)	10557 (11610)[1]	11301
(6,2)	189	189
(6,3)	15498 (16443)[1]	14200
(7,2)	297	297
(7,3)	22518 (23436)[1]	14324
(8,2)	621	590
(9,2)	1161	-
(10,2)	2133	1810

The numbers of dominant TSs of different classes found by the proposed technique along with the results of the *ADDR* method [3] are shown in Table 3.3. Note that we used Definition 3.2 for finding

[1]The number shown in the bracket is the number of TSs when the Definition 3.2 is relaxed to allow degree-3 VNs sharing at least one even-degree CN, to be part of the TS.

the TSs, whereas, the *ADDR* method employs Definition 3.1. For, (5,3), (6,3) and (7,3) TSs, we have also shown the results when Definition 3.2 is relaxed to include degree-3 VNs sharing at least one even-degree CN. It can be seen that the proposed method is able to find significantly more number TSs than the *ADDR* method. Note that the enumerated TSs are of elementary type.

One interesting observation from Table 3.3 is that all the numbers of different TS classes found by the proposed algorithm, are multiples of 27. It is not accidental, because the code is quasi-cyclic and the size of each cyclic sub-matrix is 27×27. Owing to the cyclic nature of the sub-matrices of the parity-check matrix, a TS with a specific subgraph has at least 26 other isomorphic copies. This fact also suggests that all TSs of each class have been found out by the proposed algorithm.

3.6 Conclusions

In this chapter, we presented an improved technique based on the recently reported hierarchical approach to find the dominant TSs of irregular LDPC codes. In the hierarchical approach, smaller TSs known as predecessors are progressively expanded to obtain the larger ones known as successors. We have proposed to consider only six types of sources to obtain all the TSs of any particular class. Numerical results for several popular irregular codes are presented. These results have shown that consideration of mere six types of sources are sufficient to find the dominant TSs of a specific class. Various comparative studies substantiate the efficiency and the accuracy of the technique over other similar techniques available in the literature.

4

On the Equality of the ACE and the EMD of a Cycle for ACE Spectrum Constrained LDPC Codes

Contents

4.1 Introduction

As described in the previous chapter, a dominant TS of an LDPC code contains one or more short cycles, which degrade the performance of the iterative decoders. Besides the length of a cycle, the connectivity of the cycle with the rest of the Tanner graph plays a key role in ascertaining the harmfulness. The cycles which are short as well as poorly connected, cause severe performance degradation. The connectivity can be measured by the *extrinsic message degree* (EMD) [70] as already discussed in Section 2.5.2. The EMD of a cycle is defined as the number of CNs singly connected to the VNs in the cycle. In the rest of this chapter, the phrase "singly connected CNs" is used to refer to the CNs that are singly connected to the VNs in the cycle unless otherwise specified. The concept of EMD is illustrated with the following example.

Example 4.1. *Consider the cycle of length 6 in Figure 4.1. The singly connected CNs are c_4 and c_5. Therefore, the EMD of the cycle is 2. We are interested only in the external connections from the VNs involved in the cycle. The irrelevant edges from the CNs are not shown.*

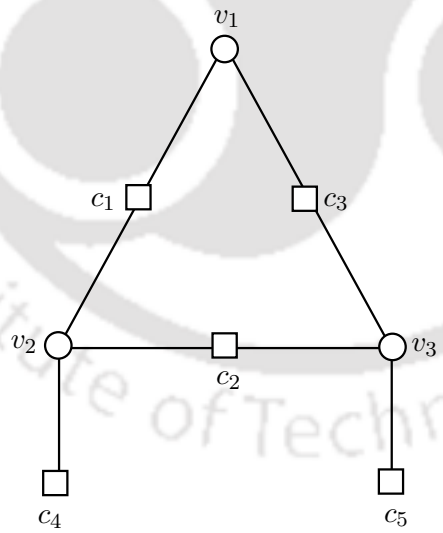


Figure 4.1: A cycle of length 6 with EMD=2

In order to compute the EMD, one needs to identify the singly connected CNs in the subgraph which may be a tedious job for certain cycles [70]. An alternative and approximate measure of the connectivity of a cycle is the *approximate cycle EMD* (ACE) [70]. Recall the definition of the ACE of a cycle from (2.13) of Section 2.5.2 which is reproduced in the following. Consider a cycle C_{2i} of length $2i$ containing i VNs $v_1, v_2, \ldots, v_i$. Suppose, the degree of v_k is d_{v_k}, $k = 1, \ldots, i$. Then the ACE

of C_{2i} is given by

$$\mathrm{ACE}\,(C_{2i}) = \sum_{k=1}^{i} (d_{v_k} - 2). \tag{4.1}$$

For example, the ACE of the cycle shown in Figure 4.1 is 2. The definition of the ACE of a cycle is entirely based on the degrees of the VNs involved in the cycle and thus easy to compute. As the ACE provides an easy assessment of the connectivity of a cycle, it has been incorporated in many code design methods. One such example is the ACE spectrum constrained LDPC codes [72, 73]. Recall from Section 2.5.2 that an ACE spectrum constrained LDPC code is designated by an ACE spectrum $\boldsymbol{\eta}_{\mathrm{ACE}} = (\eta_2, \eta_4, \cdots, \eta_{2d_{\mathrm{ACE}}})$, where η_{2i} is the minimum ACE value for any legitimate cycle of length $2i$ present in the Tanner graph. The ACE spectrum has become one of the important criteria for the construction of the LDPC codes [8, 38].

Although the ACE was proposed in order to easily calculate the EMD, it needs not be equal to the EMD in general. It does not reveal the true connectivity of a cycle in many cases. For a (d_v, d_c)-regular code, the ACE of a cycle C_{2i} of length $2i$ is equal to $i(d_v - 2)$. The ACE in such a case is just a scaled version of the length. All the cycles of the same length will have the same ACE. This implies that the consideration of the ACE for regular codes, is not useful. In such a situation, the EMD only provides meaningful information about the connectivity of the cycle. Therefore, it is important to study the relation between the ACE and the EMD of a cycle in detail. In this chapter, we derive three sufficient conditions for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained codes.

The rest of the chapter is organized as follows. Section 4.2 illustrates the inequality of the ACE and the EMD of a cycle. This section also shows that a cycle with equal ACE and EMD can be considered as an elementary TS. In Section 4.3, we present the sufficient conditions for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained LDPC codes. Simulation results regarding the effectiveness of the sufficient conditions for various codes are presented in Section 4.4. Section 4.5 concludes this chapter.

4.2 Inequality of the ACE and the EMD in the Presence of Sub-cycles

In this section, we analyze the inequality of the ACE and the EMD of a cycle present in the Tanner graph of an LDPC code. For this, the *sub-cycle* is a useful notion and is defined below.

Definition 4.1. *Let V_1 and V_2 be the sets of the VNs involved in the cycle C_{2i_1} and C_{2i_2}, respectively. We call C_{2i_1} to be a sub-cycle of C_{2i_2} if $V_1 \subseteq V_2$.*

The inequality of the ACE and the EMD of a cycle can be studied in terms of the presence of any plausible sub-cycle as explained below.

Lemma 4.1. *For a cycle C_{2i}, $\mathrm{ACE}(C_{2i}) \geq \mathrm{EMD}(C_{2i})$. The equality holds if and only if the cycle C_{2i} does not contain any sub-cycle.*

Proof. First we prove the inequality.

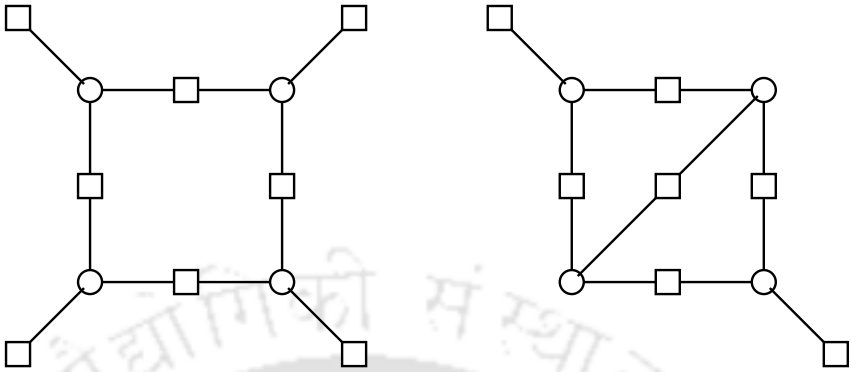
Assume that $\mathrm{ACE}(C_{2i}) < \mathrm{EMD}(C_{2i})$. From the calculation of the ACE by (4.1), it is clear that there exists at least one VN v_1 present in the cycle such that more than $d_{v_1} - 2$ edges from v_1 are linked to the singly connected CNs. This means that less than two edges coming out of v_1 are involved in the cycle. It contradicts the fact that v_1 is present in the cycle. Hence, we must have $\mathrm{ACE}(C_{2i}) \geq \mathrm{EMD}(C_{2i})$.

Next we prove the equality part.

First consider that $\mathrm{ACE}(C_{2i}) = \mathrm{EMD}(C_{2i})$. It implies that the calculation of the ACE from (4.1) correctly gives the number of singly connected CNs. This implies that for each VN v_k, exactly 2 edges are used up in the formation of the cycle C_{2i} and the remaining $d_{v_k} - 2$ edges link to some of the singly-connected CNs. Therefore, there cannot be any formation of a cycle involving the subset of the VNs present in C_{2i}.

Next consider that C_{2i} does not contain any sub-cycle. It means that no subset of the VNs in C_{2i} form a cycle. This can happen only when no VNs in C_{2i} share a CN outside C_{2i}. This means that for any VN v_k, only 2 edges are involved in the cycle and thus $d_{v_k} - 2$ edges correctly give the number of singly connected CNs linked to v_k. Thus the ACE as computed from (4.1) is exactly equal to the EMD. □

Example 4.2. *Consider the two cycles of length 8 in Figure 4.2. The cycle in Figure 4.2(a) does not contain any sub-cycle and hence the ACE is equal to the EMD. On the other hand, the outer cycle in Figure 4.2(b) contains two sub-cycles and therefore, its ACE is not equal to the EMD.*



(a) EMD=4 (ACE=EMD) **(b)** EMD=2 (ACE≠EMD)

Figure 4.2: Two cycles of length 8 with ACE=4

On the basis of the above analysis, we discuss the equivalent TS representation of a cycle whose ACE is equal to its EMD.

Equivalent Trapping Set Representation of a Cycle with Equal ACE and EMD

A cycle of length $2i$ and having an ACE value of η is usually denoted by the 2-tuple notation $(2i, \eta)$. The number of VNs in an $(2i, \eta)$ cycle is i. Since η is equal to the EMD, there are exactly η number

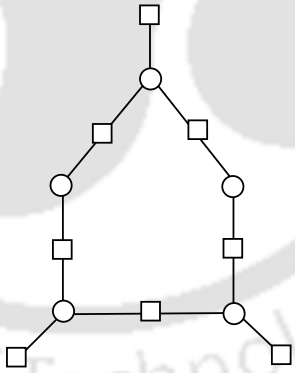


Figure 4.3: (10,3) cycle or (5,3) TS

of singly-connected CNs. Hence the subgraph induced by the VNs in the cycle contains η number of odd-degree CNs as other CNs are connected exactly twice to the VNs in the cycle. Therefore, the set of i VNs involved in the cycle form an (i, η) TS. Recall from Definition 3.1 that an elementary TS involves only degree-1 and degree-2 CNs. Thus, a TS formed by a cycle with ACE=EMD, is of elementary type. For example, the $(10, 3)$ cycle shown in Figure 4.3, can be treated as an elementary (5, 3) TS. The elementary TSs have been found to contribute significantly to the occurrence of the error floor [52]. A sizable number of dominant TSs can be found out by identifying the cycles with equal ACE and EMD.

4.3 Sufficient Conditions for the Equality of the ACE and the EMD of a Cycle for the ACE Spectrum Constrained Codes

As described in Section 4.1, for the ACE spectrum constrained LDPC codes, the cycles of length $2i$ ($i = 1, \ldots, d_{\text{ACE}}$) are constrained to have the ACE values at least η_{2i}. These constraints prevent certain cycles from containing sub-cycles. Analyzing these constraints, we derive three sufficient conditions for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained codes.

A cycle with its ACE not equal to EMD, contains some sub-cycles. These sub-cycles have the following two constraints:

(i) *Girth constraint:* The length of each sub-cycle must be greater than or equal to the girth.

(ii) *ACE constraint:* The ACE values of the sub-cycles must be greater than or equal to their corresponding ACE constraints.

Using the girth constraint first, a sufficient condition for the equality of the ACE and the EMD of a cycle is derived as follows.

Theorem 4.1. *For an LDPC code with girth g, any cycle C_{2i} of length $2i$ has equal ACE and EMD if $i < g - 2$.*

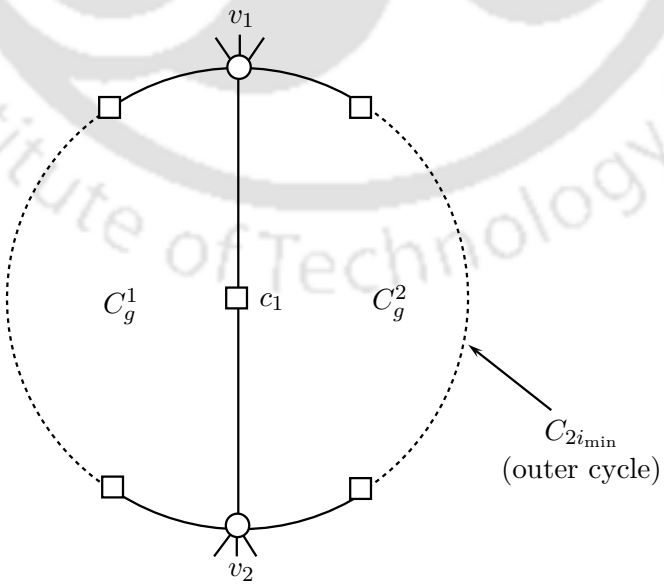


Figure 4.4: A cycle containing two sub-cycles of minimum possible length, i.e., girth g

Proof. Consider a cycle $C_{2i_{\min}}$ involving the minimum number of VNs such that it contains two legitimate sub-cycles. The minimum number of VNs that can be present in a legitimate cycle is $g/2$,

where g is the girth of the Tanner graph. This situation is depicted in Figure 4.4. The two VNs v_1 and v_2 share a CN c_1 which is not present in $C_{2i_{\min}}$. This results in the formation of two sub-cycles C_g^1 and C_g^2 of length g each. The number of VNs present in each of C_g^1 and C_g^2 is $g/2$. The total number of VNs present in the outer cycle $C_{2i_{\min}}$ is now given by

$$\begin{aligned} i_{\min} &= \frac{g}{2} + \frac{g}{2} - 2 \\ &= g - 2 \end{aligned}$$

Therefore, a cycle C_{2i} of length $2i$, cannot contain any sub-cycle if $i < i_{\min} = g - 2$. In other words, the ACE and EMD of C_{2i} will be equal if $i < g - 2$.

□

Note that the sufficient condition in Theorem 4.1 is valid for any general LDPC code as it is related to the girth only. Recall from Section 2.5.2 that, for the ACE constrained codes, if the number of degree-2 VNs is less than the total number of the CNs, then any cycle must contain at least one VN with degree more than two [70]. In that case, the minimum possible ACE of any cycle for the ACE constrained codes is 1. The following theorem says that a cycle with the minimum possible ACE has the same ACE and EMD.

Theorem 4.2. *For any cycle C_{2i} of length $2i$,* $\mathrm{ACE}(C_{2i})$ *is equal to* $\mathrm{EMD}(C_{2i})$ *if* $\mathrm{ACE}(C_{2i}) = 1$.

Proof. Figure 4.5 shows a cycle C_{2i} whose ACE value is equal to 1. Observe that C_{2i} contains $(i-1)$

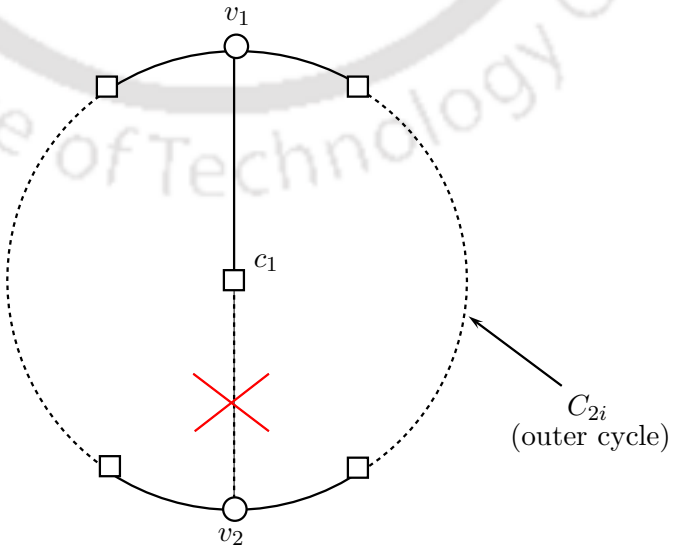


Figure 4.5: A cycle with ACE equal to 1

number of degree-2 VNs and only one VN v_1 of degree 3. In that case, the CN c_1 connected to v_1

(but not involved in the cycle) cannot be shared by any of the remaining VNs like v_2 as they are of degree 2. Thus, C_{2i} cannot contain any sub-cycle which, in turn, implies that $\mathrm{ACE}\,(C_{2i})$ is equal to the $\mathrm{EMD}\,(C_{2i})$. □

Next we derive the sufficient condition for ACE=EMD of a cycle for the ACE spectrum constrained LDPC codes using the ACE constraints of the sub-cycles. Let $\mathrm{ACE}^{(s)}_{\min}(C_{2i})$ denote the minimum ACE required for C_{2i} to have some VNs sharing s number of CNs outside C_{2i}. $\mathrm{ACE}^{(s)}_{\min}(C_{2i})$ satisfies the following lemma which says that $\mathrm{ACE}^{(s)}_{\min}(C_{2i})$ is a non-decreasing function of s.

Lemma 4.2. $\mathrm{ACE}^{(s)}_{\min}(C_{2i}) \leq \mathrm{ACE}^{(s+1)}_{\min}(C_{2i})$

Proof. Consider the cycle C_{2i} drawn in the left of Figure 4.6. It does not show the CNs present in the cycle. Therefore, the entire cycle is drawn in a dashed form. The VNs in C_{2i} share $(s+1)$

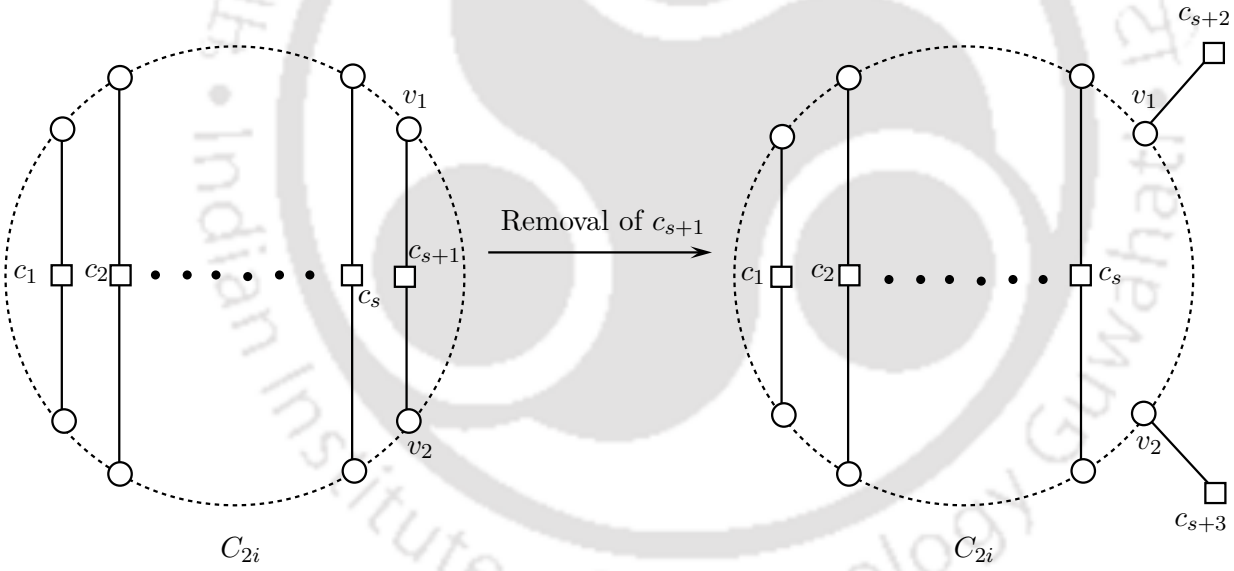


Figure 4.6: Removal of a CN shared outside of a cycle by two VNs involved in the cycle

number of CNs $(c_1, c_2, \ldots, c_{s+1})$ outside C_{2i} with the minimum possible ACE value, $\mathrm{ACE}^{(s+1)}_{\min}(C_{2i})$. The configuration of C_{2i} can be changed to have s number of CNs shared outside the cycle by deleting any one CN out of the $(s+1)$ CNs. Let this CN be c_{s+1} and let it be originally connected to the VNs v_1 and v_2 involved in C_{2i}. The removal of c_{s+1} can be done as follows: delete c_{s+1} and put the edges from v_1 to c_{s+2} and from v_2 to c_{s+3}, where c_{s+2} and c_{s+3} are any two CNs not present in the subgraph induced by the VNs in C_{2i}. After the removal of c_{s+1}, we get a modified subgraph as shown in the right of Figure 4.6. The removal of c_{s+1} will delete some of the sub-cycles with no further creation of new cycles in the subgraph. Moreover, as the number of edges coming out of each VN involved

in the cycle is unaltered, the degree of each of the VNs remains fixed. Hence, the ACE values of all the remaining cycles in the modified subgraph will remain the same according to (4.1). Consequently, they will continue to satisfy their corresponding ACE constraints even after the removal of c_{s+1}. This fact ensures that the cycle C_{2i} with the modified subgraph is a valid one for the code with the given ACE spectrum. The ACE of C_{2i} with the modified subgraph is the same as that of the cycle with the original subgraph, i.e., $\text{ACE}_{\min}^{(s+1)}(C_{2i})$. Hence, $\text{ACE}_{\min}^{(s)}(C_{2i})$ has to be less than or equal to the ACE of the cycle with the present configuration, i.e.,

$$\text{ACE}_{\min}^{(s)}(C_{2i}) \leq \text{ACE}_{\min}^{(s+1)}(C_{2i})$$

□

Using Lemma 4.2, a sufficient condition for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained codes is derived in the following. This condition will be relevant only when the sufficient conditions in Theorem 4.1 and Theorem 4.2 are not satisfied.

Theorem 4.3. *For an ACE spectrum constrained code with the ACE spectrum* $(\eta_2, \eta_4, \cdots, \eta_{2d_{\text{ACE}}})$ *and girth* g, $\text{ACE}(C_{2i})$ *is equal to* $\text{EMD}(C_{2i})$, $i \geq g-2$, *if*

$$\text{ACE}(C_{2i}) < \min_{j \in \left\{\frac{g}{2}, \frac{g}{2}+1, \cdots, \left\lceil \frac{i-1}{2} \right\rceil + 1\right\}} \frac{\eta_{2j} + \eta_{2(i-j+2)}}{2}. \tag{4.2}$$

Proof. The ACE of a cycle is exactly equal to its EMD if the cycle does not contain any sub-cycle as per Lemma 4.1. In that case, no VNs in the cycle share any CN outside the cycle. According to Lemma 4.2, if $\text{ACE}(C_{2i}) < \text{ACE}_{\min}^{(1)}(C_{2i})$, then $\text{ACE}(C_{2i}) < \text{ACE}_{\min}^{(s)}(C_{2i}), \forall s \geq 1$. It means that if $\text{ACE}(C_{2i}) < \text{ACE}_{\min}^{(1)}(C_{2i})$, then the VNs in the cycle cannot share any CN outside the cycle and consequently we must have $\text{ACE}(C_{2i}) = \text{EMD}(C_{2i})$. Hence, the sufficient condition for the equality of the ACE and the EMD of C_{2i} is

$$\text{ACE}(C_{2i}) < \text{ACE}_{\min}^{(1)}(C_{2i}). \tag{4.3}$$

In the following, the expression for $\text{ACE}_{\min}^{(1)}(C_{2i})$ will be derived.

Let the VNs involved in the cycle C_{2i} be denoted by $v_1, v_2, \ldots, v_{i-1}$ and v_i. Consider a situation where two VNs in C_{2i} share a CN outside C_{2i} and other VNs do not. Without loss of generality, let these two VNs be v_1 and v_j. The situation is shown in Figure 4.7. The two sub-cycles produced in this case can have lengths $2j$ and $2(i-j+2)$ for $j \in \left\{\frac{g}{2}, \frac{g}{2}+1, \cdots, \left\lceil \frac{i-1}{2} \right\rceil + 1\right\}$. Let these two

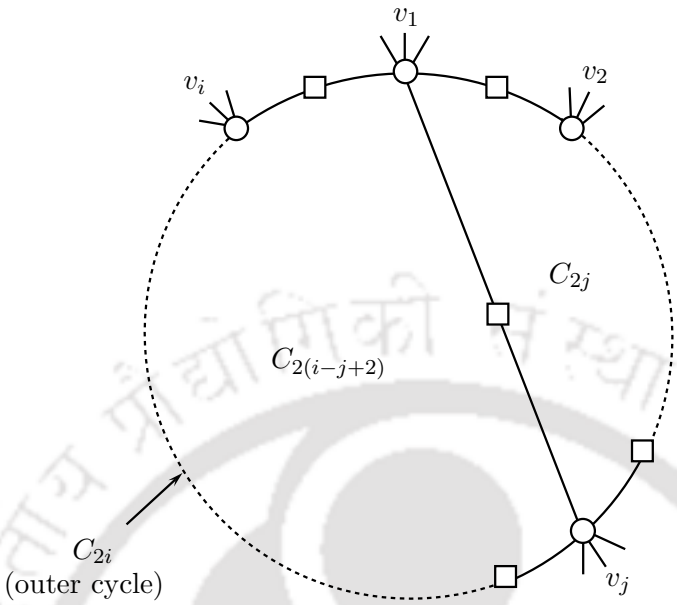


Figure 4.7: Formation of two sub-cycles due to the sharing of one CN outside a cycle of length $2i$

cycles be denoted by C_{2j} and $C_{2(i-j+2)}$. C_{2j} and $C_{2(i-j+2)}$ contain the VNs $\{v_1, v_2, \ldots, v_j\}$ and $\{v_1, v_j, v_{j+1}, \ldots, v_i\}$ respectively.

Using (4.1), the ACE values of the cycles C_{2j} and $C_{2(i-j+2)}$ can be written as

$$\text{ACE}\left(C_{2j}\right) = \sum_{k=1}^{j} \left(d_{v_k} - 2\right) \tag{4.4}$$

$$\text{and} \quad \text{ACE}\left(C_{2(i-j+2)}\right) = \left(d_{v_1} - 2\right) + \sum_{k=j}^{i} \left(d_{v_k} - 2\right). \tag{4.5}$$

The cycles C_{2j} and $C_{2(i-j+2)}$ must satisfy their corresponding ACE constraints. Therefore,

$$\sum_{k=1}^{j} \left(d_{v_k} - 2\right) \geq \eta_{2j} \tag{4.6}$$

$$\text{and} \quad \left(d_{v_1} - 2\right) + \sum_{k=j}^{i} \left(d_{v_k} - 2\right) \geq \eta_{2(i-j+2)}. \tag{4.7}$$

The VNs v_1 and v_j involved in C_{2i} share a CN outside C_{2i}. Therefore, the degrees of v_1 and v_j are at least 3. For the other VNs not sharing any CN outside C_{2i}, the degree is greater than or equal to 2. Therefore,

$$\left(d_{v_r} - 2\right) \geq 0 \qquad \forall r \in \{2, \ldots, i\} \setminus \{j\}. \tag{4.8}$$

Adding (4.6), (4.7) and (4.8) gives

$$2\sum_{k=1}^{i}(d_{v_k}-2) \geq \eta_{2j}+\eta_{2(i-j+2)}$$
$$\Rightarrow \mathrm{ACE}\,(C_{2i}) \geq \left\lceil \frac{\eta_{2j}+\eta_{2(i-j+2)}}{2} \right\rceil. \tag{4.9}$$

Therefore, any two VNs possibly can share a CN outside C_{2i} only if (4.9) is satisfied for any $j \in \left\{\frac{g}{2}, \frac{g}{2}+1, \cdots, \left\lceil \frac{i-1}{2} \right\rceil + 1\right\}$.

Thus, $\mathrm{ACE}^{(1)}_{\min}(C_{2i})$ is given by

$$\mathrm{ACE}^{(1)}_{\min}(C_{2i}) = \min_{j\in\left\{\frac{g}{2},\frac{g}{2}+1,\cdots,\left\lceil\frac{i-1}{2}\right\rceil+1\right\}} \left\lceil \frac{\eta_{2j}+\eta_{2(i-j+2)}}{2} \right\rceil. \tag{4.10}$$

Using (4.10) in (4.3), the sufficient condition for the equality of the ACE and the EMD can be written as

$$\mathrm{ACE}\,(C_{2i}) < \min_{j\in\left\{\frac{g}{2},\frac{g}{2}+1,\cdots,\left\lceil\frac{i-1}{2}\right\rceil+1\right\}} \left\lceil \frac{\eta_{2j}+\eta_{2(i-j+2)}}{2} \right\rceil. \tag{4.11}$$

As the ACE values are integers only, we have equivalently

$$\mathrm{ACE}\,(C_{2i}) < \min_{j\in\left\{\frac{g}{2},\frac{g}{2}+1,\cdots,\left\lceil\frac{i-1}{2}\right\rceil+1\right\}} \frac{\eta_{2j}+\eta_{2(i-j+2)}}{2}. \tag{4.12}$$

□

Using Theorem 4.3, the following result is derived.

Result 4.1. *For an* $(\infty, \infty, \eta_6, \eta_8, \eta_{10})$ *code,* $\mathrm{ACE}\,(C_8)$ *and* $\mathrm{ACE}\,(C_{10})$ *will be equal to the corresponding EMDs if* $\mathrm{ACE}\,(C_8) < \eta_6$ *and* $\mathrm{ACE}\,(C_{10}) < (\eta_6+\eta_8)/2$.

Proof. In this case, the girth $g = 6$. Therefore, in (4.2), j can take only one value which is $g/2 = 3$. Then, the proof directly follows from (4.2). □

In the following, we explain the sufficient conditions with the help of some examples.

Example 4.3. *Consider an LDPC code of girth,* $g = 10$. *All the cycles of length* 10 $(i = 5)$, 12 $(i = 6)$ *and* 14 $(i = 7)$ *satisfy* $i < g - 2$. *Hence, these cycles have equal respective ACE and EMD according to Theorem 4.1.*

Example 4.4. *Suppose, for a cycle* C_{20} *of length* 20, $\mathrm{ACE}\,(C_{20}) = 1$. *Then, by Theorem 4.2,* $\mathrm{ACE}\,(C_{20}) = \mathrm{EMD}\,(C_{2i})$.

Example 4.5. *Consider an* $(\infty, \infty, 13, 9, 4)$ *code. All the cycles of length* 8 *with the ACE value less than* 13*, will have equal ACE and EMD according to Result 4.1. For all the cycles of length* 10 *having ACE value less than* 11 $((13+9)/2)$*, the ACE will be equal to the EMD.*

Remark 4.1. *The EMD of a cycle measures its connectivity with the rest of the Tanner graph. If the EMD is low, the cycle is harmful to iterative decoding. The calculation of the EMD is not easy as we have to go for the extra step of examining the entire subgraph induced by the VNs in the cycle. On the other hand, the ACE can be easily computed from the degrees of the VNs in the cycle. Sometimes, the ACE is not equal to the EMD and in those cases, ACE may not reflect the detrimental effect correctly. Thus, the sufficient conditions presented in this section can help one to assertively asses the detrimental effect of a cycle based on its ACE value.*

4.4 Simulation Results and Discussions

We consider two examples of ACE spectrum constrained LDPC codes: (a) $(\infty, \infty, 10, 4, 3)$ code with $N = 504, M = 252$ and (b) $(\infty, \infty, 13, 5, 4)$ code with $N = 816, M = 408$. The rate of each code is 0.5. The degree distribution corresponding to $d_v^{\max} = 11$ [59] is considered for both the codes.

(a) $(\infty, \infty, 10, 4, 3)$ code with $N = 504, M = 252$

The expanded ACE spectrum of the particular realization of the code ensemble is shown in Figure 4.8. Recall from Section 2.5.2 that the expanded ACE spectrum gives an idea about the involvement of the VNs in the cycles with low ACE values for a particular parity-check matrix. The expanded ACE spectrum for the 6-cycles is denoted by $\boldsymbol{\eta}_6^{\text{exp}} = \left(n_{(6,0)}, \ldots, n_{(6,27)}\right)$. The ACE constraint on the 6-cycles is 10. Therefore, for $\boldsymbol{\eta}_6^{\text{exp}}$, we have $n_{(6,j)} = 0$, $\forall j < 10$. The values of the nonzero components of $\boldsymbol{\eta}_6^{\text{exp}}$ are $n_{(6,10)} = 367, n_{(6,11)} = 71, n_{(6,12)} = 4, n_{(6,17)} = 7, n_{(6,18)} = 47$ and $n_{(6,19)} = 2$. Similarly, as the ACE constraint on the 8-cycles is 4, for $\boldsymbol{\eta}_8^{\text{exp}}$, we have $n_{(8,j)} = 0$, $\forall j < 4$. The values of the nonzero components of $\boldsymbol{\eta}_8^{\text{exp}}$ are $n_{(8,4)} = 155, n_{(8,5)} = 20, n_{(8,6)} = 1, n_{(8,8)} = 10, n_{(8,9)} = 154, n_{(8,10)} = 153$ and $n_{(8,11)} = 11$. As the ACE constraint on the 10-cycles is 3, we have $n_{(10,j)} = 0$, $\forall j < 3$. The values of the nonzero components of $\boldsymbol{\eta}_{10}^{\text{exp}}$ are $n_{(10,3)} = 331$, $n_{(10,4)} = 72$, $n_{(10,5)} = 7$, $n_{(10,8)} = 5, n_{(10,9)} = 74$ and $n_{(10,10)} = 15$.

For the $(\infty, \infty, 10, 4, 3)$ code, all 6-cycles satisfy the sufficient condition for the equality of the ACE and the EMD according to Theorem 4.1. All the 8-cycles of ACE less than 10 and all the 10-cycles

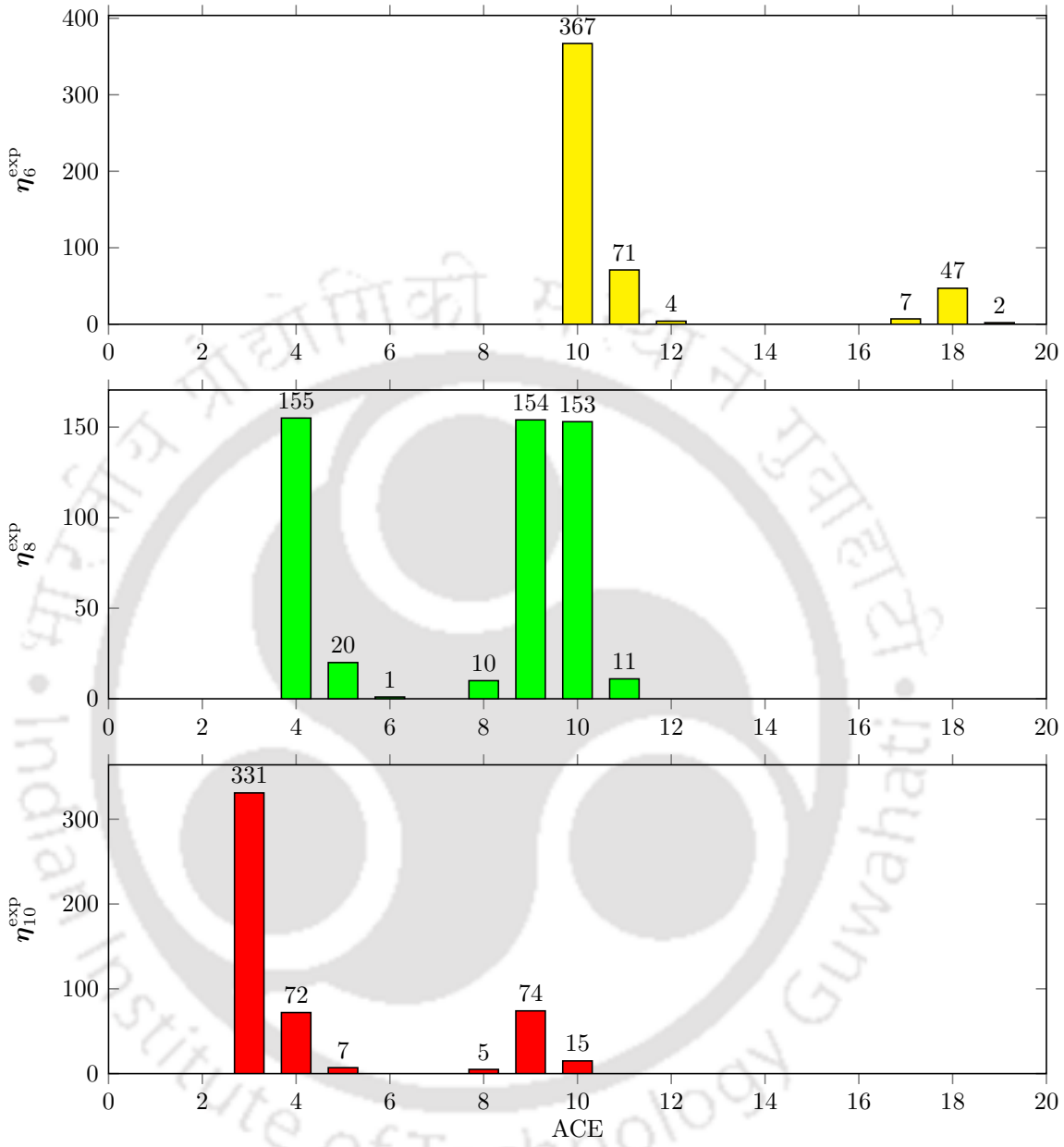


Figure 4.8: Expanded ACE spectrum of the $(\infty, \infty, 10, 4, 3)$ code with $N = 504, M = 252$

with ACE less than 7 have equal respective ACE and EMD according to Theorem 4.3. We find out all the cycles of lengths 6, 8 and 10. For each of these cycles, we calculate the ACE and the EMD. The total numbers of cycles with equal ACE and EMD are also enumerated. Then, for each of the cycles of lengths 6, 8 and 10, we check for the satisfaction of the appropriate sufficient condition for the equality of the ACE and the EMD. The total numbers of the cycles, those with ACE=EMD and those satisfying the sufficient conditions are reported in Table 4.1. All the cycles of length 6 have

Table 4.1: Numbers of cycles for the $(\infty, \infty, 10, 4, 3)$ code with $N = 504$, $M = 252$

Number of cycles	6-cycles	8-cycles	10-cycles
All	6953	159650	6254699
With ACE=EMD	6953	105409	1378709
Satisfying the sufficient conditions for ACE=EMD	6953	165	1329

equal ACE and EMD. This is due to the fact that the 6-cycles cannot contain sub-cycles as the girth of the code is 6. Observe from Table 4.1 that there are a large number of cycles of length 8 and 10 with unequal respective ACE and EMD. It means that the approximation regarding the measure of the external connectivity of cycle fails quite frequently. Furthermore the fraction of the 10-cycles with ACE equal to EMD is significantly less than that for the 8-cycles. It is because of the fact that a 10-cycle has a higher chance of containing sub-cycles compared to an 8-cycle.

Observe from Table 4.1 that the number of cycles satisfying the sufficient conditions for the equality of the ACE and the EMD is less than the number of cycles with ACE equal to the EMD. This indicates that there are a lot of cycles which do not satisfy the sufficient conditions but actually having equal ACE and EMD. This fact illustrates that there is a scope of finding more tighter conditions for the equality of the ACE and the EMD of a cycle.

(b) $(\infty, \infty, 13, 5, 4)$ code with $N = 816, M = 408$

The expanded ACE spectrum of the particular realization of the code ensemble is shown in Figure 4.9. For this example, the ACE constraint on the 6-cycles is 13. Therefore, all the $n_{(6,j)}$ components of $\boldsymbol{\eta}_6^{\text{exp}}$ are 0 if $j < 13$. The nonzero components of $\boldsymbol{\eta}_6^{\text{exp}}$ are $n_{(6,13)} = 26$, $n_{(6,18)} = 427$, $n_{(6,19)} = 299$ and $n_{(6,20)} = 10$. The nonzero components of $\boldsymbol{\eta}_8^{\text{exp}}$ are found to be $n_{(8,5)} = 66$, $n_{(8,6)} = 6$, $n_{(8,9)} = 207$, $n_{(8,10)} = 463$ and $n_{(8,11)} = 74$. The nonzero components of $\boldsymbol{\eta}_{10}^{\text{exp}}$ are $n_{(10,4)} = 452$, $n_{(10,5)} = 66$, $n_{(10,6)} = 6$, $n_{(10,9)} = 158$ and $n_{(10,10)} = 134$.

For the $(\infty, \infty, 13, 5, 4)$ code also, all 6-cycles satisfy the sufficient condition for the equality of the ACE and EMD. All the 8-cycles of ACE less than 13 and all the 10-cycles with ACE less than 9 have equal respective ACE and EMD. The total number of cycles of lengths 6, 8 and 10 are shown in Table 4.2. It also shows the numbers of cycles with ACE equal to EMD and the numbers of cycles satisfying the sufficient conditions for the equality of the ACE and the EMD. Observe that in this case

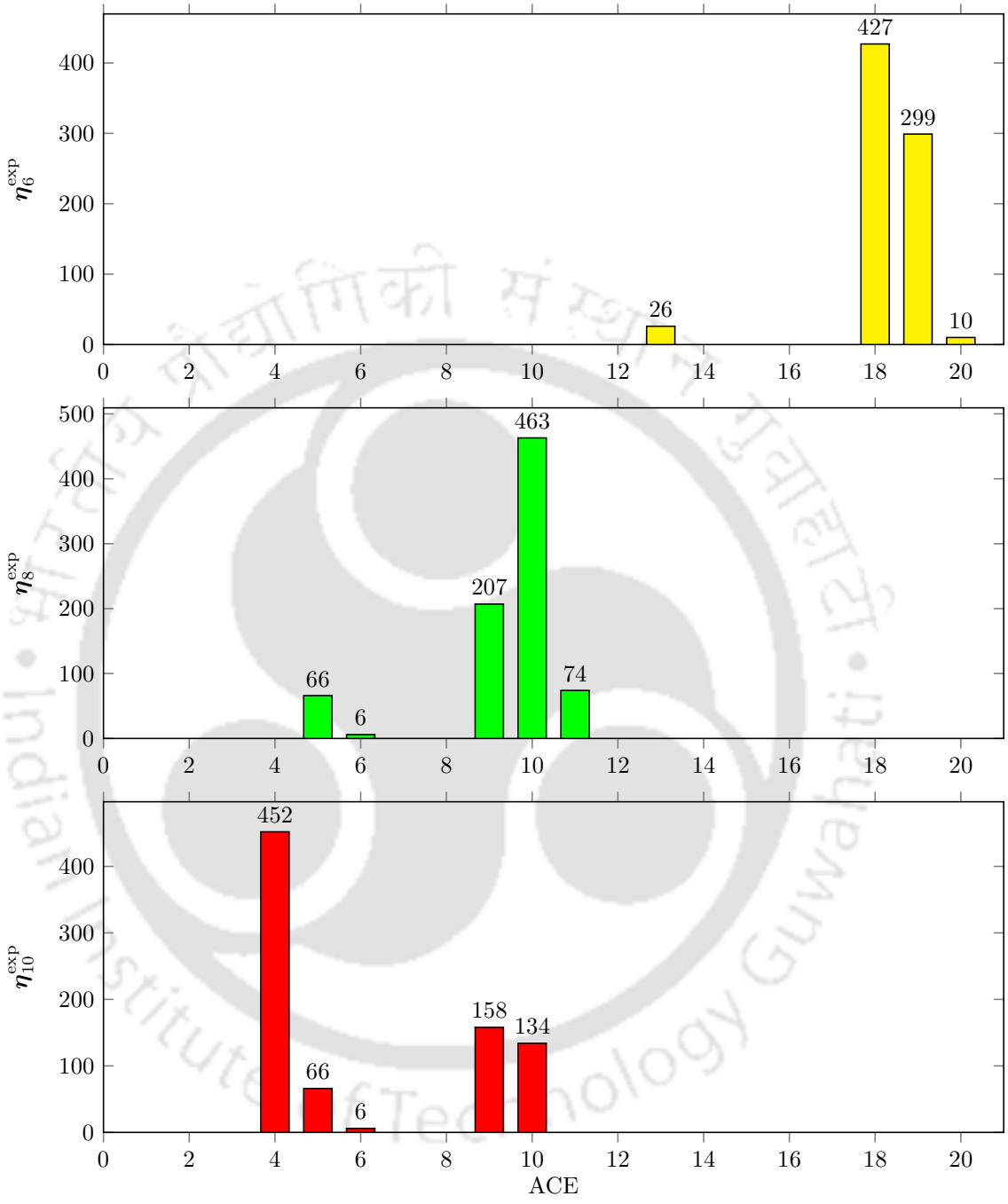


Figure 4.9: Expanded ACE spectrum of the $(\infty, \infty, 13, 5, 4)$ code with $N = 816, M = 408$

also a large number of 8-cycles and 10-cycles do not have equal respective ACE and EMD. Moreover, the numbers of cycles satisfying the sufficient conditions for the equality of the ACE and the EMD are less than those actually having equal ACE and EMD.

Table 4.2: Numbers of cycles for the $(\infty, \infty, 13, 5, 4)$ code with $N = 816, M = 408$

Number of cycles	6-cycles	8-cycles	10-cycles
All	6070	156278	4113047
With ACE=EMD	6070	53238	2083399
Satisfying the sufficient conditions for ACE=EMD	6070	140	448

4.5 Conclusions

We have derived three sufficient conditions for the equality of the ACE and the EMD of a cycle for the ACE spectrum constrained LDPC codes. The first sufficient condition is satisfied when the length of the particular cycle is less than a threshold determined by the girth. The second sufficient condition is a simple one and it says that when the ACE of the cycle is equal to the minimum possible value, i.e., 1, the ACE has to be equal to the EMD. According to the third condition, the ACE of a cycle is guaranteed to be equal to its EMD if it is less than a threshold determined by the ACE constraints of the sub-cycles. Simulation results on two codes show that for a large number of cycles, ACE and EMD are not equal and the sufficient conditions are able to predict a small portion of the total number of cycles with ACE=EMD.

5

EXIT Chart Analysis of Puncturing for Non-binary LDPC Codes

Contents

5.1 Introduction

For the communication systems with time-varying channel conditions, the codes employed should be preferably rate-adaptive. Whenever the channel improves, one should switch to a higher-rate code and vice versa. One important class of rate-adaptive codes comprises of the rate-compatible (RC) codes. In the RC codes, a higher-rate code is embedded into a lower-rate code. This aspect ensures convenient switching between different rates with a single pair of an encoder and a decoder. RC coding schemes can be accomplished by different techniques like puncturing, pruning, extending, shortening, and different combinations of them [17, 32, 75, 78]. However, puncturing has been most frequently considered in the design of RC codes. In addition to the channel adaptation, puncturing is also useful for adaptation to source coding schemes. Specifically, for scalable video and audio codecs, the source encoding results in data of different importance levels [17, 76]. The less important data may be punctured to maintain a healthy target rate and overall reliability of the highly-sensitive data. In puncturing, a low-rate code is considered as a mother code and higher-rate codes are obtained by puncturing carefully selected bits. The punctured parity bits in a lower-rate code form a subset of the punctured parity bits of a higher-rate code. The challenge in designing such puncturing patterns is to guarantee good performance at diverse rates.

Ha *et al.* initiated the study on the RC puncturing schemes for the short-length binary LDPC codes [32]. They have proposed *grouping* and *sorting* algorithms to select the puncturing bits. The grouping algorithm helps to identify the VNs which can be recovered after puncturing. These recoverable VNs can be sorted in the decreasing order of preference with the help of the sorting algorithm. The grouping and sorting algorithms are further described in Section 5.2.1.

As discussed in the previous chapters, the short-to-moderate length LDPC codes usually suffer from the error floor problem when decoded by the sub-optimal iterative algorithms. The design of finite-length LDPC codes with good error floor performance has inspired the study of the NB-LDPC codes. The NB-LDPC codes, defined over a higher order field (GF(q), $q > 2$), were introduced by Davey and Mackay [18]. They have reported that the NB-LDPC codes can yield significant performance improvement over their binary counterparts. The NB-LDPC codes are less susceptible to the error floor problem and consequently perform very well at short block-lengths. Moreover, the NB-LDPC codes are more flexible than their binary counterparts for the design of the optimum puncturing scheme. This chapter focuses on the design of optimum RC puncturing scheme for the NB-LDPC

codes.

Klinc *et al.* [43] proposed a puncturing scheme for the NB-LDPC codes. They have considered codes defined over binary extension fields, $\mathrm{GF}(2^s), s \in \mathbb{Z}^+ \setminus \{1\}$. The codeword symbols are transmitted in bit-level representations over a BI-AWGN channel. In such scenarios, puncturing can be done either *bitwise* or *symbolwise*. In a bitwise scheme, only a certain number of bits per VN are punctured. On the other hand, all the bits corresponding to a VN are punctured in a symbolwise scheme. Bitwise and symbolwise puncturing are sometimes referred as *partial* and *complete* puncturing respectively. The authors in [43] have observed that the grouping algorithm can be applied for the NB-LDPC codes also. They have recommended to use the grouping algorithm for selecting the recoverable VNs and then perform bitwise puncturing on the recoverable VNs.

Gorgoglione *et al.* have investigated the asymptotic performance of the puncturing schemes for NB-LDPC codes [30,31]. They have formulated a Monte-carlo based DE method to compute the threshold for a puncturing pattern. It has been observed that the performances of different puncturing patterns depend massively on the degrees of the VNs. They have considered an irregular code over $\mathrm{GF}(2^4)$ and found the optimum puncturing distributions for different rates (A puncturing distribution refers to the fractions of VNs of a particular degree with different numbers of punctured bits per symbol). The optimization of the puncturing pattern was done entirely in the asymptotic sense. The grouping algorithm, which is a standard tool to find the recoverable VNs, was not taken into consideration. The grouping algorithm usually selects low-degree VNs for puncturing. If a higher degree VN gets punctured, its effect will propagate to a larger part of the Tanner graph. The grouping algorithm, therefore, does not allow the higher-degree VNs to be punctured. We have verified this observation by different simulations for short-length codes. In such a scenario, the optimized patterns in [30, 31] cannot be directly used along with the grouping algorithm as these patterns take into account all the VNs including the higher-degree ones.

The EXIT chart is an alternative to the DE for analyzing and designing iterative codes for optimum asymptotic performance. It has been used successfully to design LDPC codes for various scenarios [26, 62, 68]. The EXIT chart model for the NB-LDPC codes is developed by Bennatan and Burshtein [10] for any arbitrary discrete-memoryless channel. In this chapter, we propose an EXIT chart model to analyze bitwise and symbolwise puncturing schemes for the NB-LDPC codes. By considering the grouping algorithm in the EXIT chart model, we devise a procedure of finding the optimum puncturing

pattern with the recoverable VNs.

The rest of the chapter is organized as follows. Section 5.2 provides a background on the RC schemes for the binary and the NB-LDPC codes. We also describe the EXIT chart models for the binary and the NB-LDPC codes. In Section 5.3, we propose the EXIT chart model of puncturing for the NB-LDPC codes. Simulation results are presented in Section 5.4 and Section 5.5 concludes the chapter.

5.2 Background

In this section, we present the background concepts on the RC puncturing of the NB-LDPC codes and the EXIT chart tool for analyzing these codes. This section begins with a discussion on the RC puncturing for the binary LDPC codes.

5.2.1 Rate-compatible Puncturing Scheme for Binary LDPC Codes

The concept of the recoverability of a punctured VN is an important concept [32]. A punctured VN v_j does not receive any signal from the channel, i.e., $y_j = 0$. From (2.9), the channel LLR is given by $L_{\text{ch}_j} = \frac{2y_j}{\sigma_n^2}$. As $y_j = 0$, we have $L_{\text{ch}_j} = 0$. In other words, the decoder is initially unbiased about the value of a punctured VN. From (2.12), the *a posteriori* LLR for such a VN can be written as

$$L_j = \sum_{i \in \mathbf{N}(j)} U_{ij}. \tag{5.1}$$

Thus L_j would become non-zero if any of the U_{ij} values is non-zero. A punctured VN is said to be *recovered* if it receives a non-zero message from at least one neighboring CN. Note that the recovery of a VN does not necessarily imply that the corresponding non-zero message is of correct sign.

Now consider the processing for a CN c_i in (2.10) which is reproduced below:

$$U_{ij} = \log \frac{1 + \prod\limits_{j' \in \mathbf{M}(i) \setminus \{j\}} \tanh\left(\frac{V_{j'i}}{2}\right)}{1 - \prod\limits_{j' \in \mathbf{M}(i) \setminus \{j\}} \tanh\left(\frac{V_{j'i}}{2}\right)}. \tag{5.2}$$

Observe that a CN c_i is capable of sending some non-zero message to a VN v_j if all the messages from the neighboring VNs in $\mathbf{M}(i) \setminus \{j\}$ towards c_i are non-zero. Such a CN is called a *survived* CN with respect to v_j. If at least one VN in $\mathbf{M}(i) \setminus \{j\}$ sends a zero-valued message, then c_i is considered

dead with respect to v_j.

A punctured VN v_j is said to be one-step recoverable (1-SR) if it is connected to at least one CN c_i such that all the neighboring VNs in $\mathbf{M}(i) \setminus \{j\}$ are unpunctured. In general, a punctured VN v_j is called *k-step recoverable* (k-SR) if v_j has at least one neighboring CN c_i such that the set $\mathbf{M}(i) \setminus \{j\}$ contains at least one $(k-1)$-SR node and the others are m-SR, where $0 \leq m \leq k-1$ [32]. A k-SR VN will get a non-zero message exactly after k iterations.

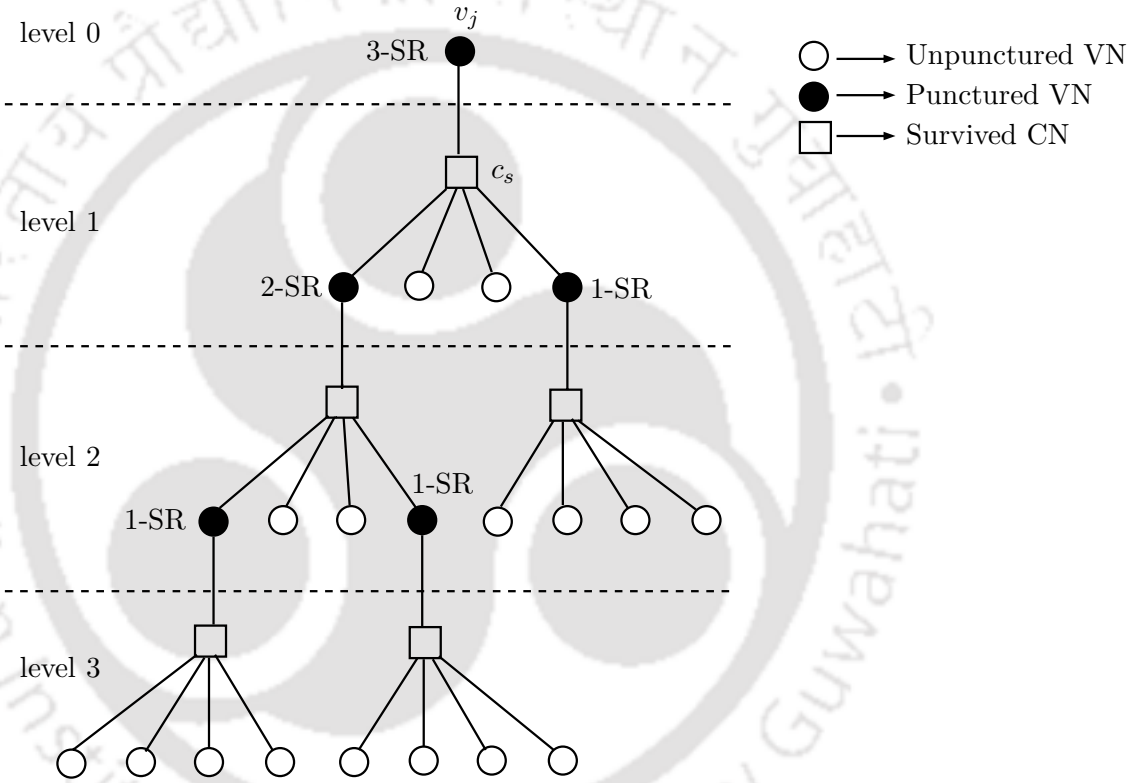


Figure 5.1: Recovery tree of the VN v_j

The concept of recoverability can be visualized through a graph known as the *recovery tree* [32]. Figure 5.1 shows the recovery tree for a VN v_j. The level 0 of the tree includes the lone VN v_j as the root node. In order to grow the recovery tree, v_j is first joined to the neighboring guaranteed survived CN c_s. Then all the other neighboring VNs of c_s are included in the level 1 of the tree. The same procedure is carried out for all the punctured VNs in level 1. The unpunctured VNs are not joined to any CN in the subsequent levels. Only the punctured VNs are linked with the survived CN in the next level. Moreover, the dead CNs are not considered as they do not contribute to the recovery of the punctured VNs. This process of expanding the tree is continued until every branch terminates with an unpuntured VN. The resultant tree reflects the recovery of v_j through the propagation of various

non-zero messages and thus named as the recovery tree of v_j. It can be observed from Figure 5.1 that v_j will receive some non-zero message via c_s after 3 iterations of the SPA and thus it is a 3-SR node. Figure 5.1 shows some 1-SR and 2-SR nodes, too.

In [32], the authors have devised a grouping algorithm to partition the set V of the VNs according to their recoverability. V is partitioned into the groups $\mathbf{G}_0, \mathbf{G}_1, \cdots, \mathbf{G}_K$, where $\mathbf{G}_0$ is the set of unpunctured nodes, $\mathbf{G}_i$ is the set of the VNs guaranteed to recover exactly after i iterations and K is the maximum number of iterations required to recover all the punctured VNs. The VNs in $\mathbf{G}_i$ with a lower i ($i \neq 0$) are expected to recover faster. Therefore, the partitioning starts with the objective of maximizing the number of VNs in $\mathbf{G}_1$, then $\mathbf{G}_2$ and so on. The VNs in $\mathbf{G}_0$ should not be punctured. The maximum rate $r_{\max}$ that can be achieved by puncturing nodes in the other groups is given by

$$r_{\max} = \frac{r_0 N}{|\mathbf{G}_0|}, \tag{5.3}$$

where r_0 is the rate of the mother code and N is the total number of VNs.

The number of VNs required to be punctured to attain a rate r_l is given by

$$N_l^p = \left[\frac{N \times (r_l - r_0)}{r_l}\right],$$

where $[\cdot]$ is the nearest integer function.

While selecting N_l^p number of VNs for puncturing, the aim is to include those VNs which can be recovered after a small number of iterations. For that, initially the VNs from $\mathbf{G}_1$ are considered. If all the VNs in $\mathbf{G}_1$ are included for puncturing, then the VNs in $\mathbf{G}_2$ are considered. The process is repeated until N_l^p VNs are selected.

In order to fix the selection of the VNs within a particular group $\mathbf{G}_i$, the authors in [32] have proposed a sorting algorithm. This sorting algorithm helps to distinguish different VNs of the same recoverability. It sorts the VNs of $\mathbf{G}_i$ in the decreasing order of preference for puncturing. The selection of a VN for puncturing is done mainly in two sequential steps:

Step 1

This step is driven by the idea of the survived CNs. The VNs which are connected to a larger number of survived CNs, are expected to sustain better in puncturing. A subset $\mathcal{G}$ of $\mathbf{G}_i$ is formed such that each VN in $\mathcal{G}$ is connected to the same maximum number s_m of survived CNs. If $\mathcal{G}$ contains only

one VN, then that VN is selected for puncturing, otherwise, Step 2 is performed.

Step 2

This step is based on the concept of the dead CNs. The motive is to select the VNs which are connected to the minimum number of dead CNs. Each VN v_k in $\mathcal{G}$ is connected to s_m survived CNs. Therefore, $d_{v_k} - s_m$ is the number of dead CNs connected to v_k, where d_{v_k} is the degree of v_k. Thus the degree d_{v_k} alone reflects the relative number of dead CNs linked to v_k. A subset $\mathcal{G}'$ of $\mathcal{G}$ is formed such that each VN in $\mathcal{G}'$ has the same minimum degree. If $\mathcal{G}'$ contains only one VN, then it is selected for puncturing, otherwise a VN is selected randomly from $\mathcal{G}'$.

The details of the grouping and the sorting algorithms can be found out in the reference [32].

5.2.2 Rate-compatible Puncturing Scheme for Non-binary LDPC Codes [43]

We first discuss the recoverability of a punctured VN in the context of the FFT-QSPA described in Section 2.7.1. Suppose $y_{j,b}$ denotes the bth received bit of the VN v_j. From (2.17), the channel *a posteriori* probability f_j^a of v_j acquiring the value $a \in \mathrm{GF}(2^s)$ is given by

$$f_j^a = \prod_{b=1}^{s} g_{j,b}^{a_b}, \tag{5.4}$$

where a_b is the bth bit of the binary representation of a and $g_{j,b}^0$ and $g_{j,b}^1$ are given by

$$g_{j,b}^0 = \frac{1}{(1+\exp(-2y_{j,b}/\sigma^2))}$$
$$\text{and} \quad g_{j,b}^1 = 1 - g_{j,b}^0.$$

If the bth bit of v_j is punctured, then $y_{j,b} = 0$ and consequently

$$g_{j,b}^0 = g_{j,b}^1 = 0.5. \tag{5.5}$$

Klinc *et al.* [43] have pointed out that the notion of symbol (VN) recovery is best understood under the assumption of complete puncturing. If all the bits of v_j are punctured, then from (5.4) and (5.5), we have

$$f_j^a = (0.5)^s \quad \forall a \in \mathrm{GF}(2^s). \tag{5.6}$$

Therefore, the channel *a posteriori* probability vector $\mathbf{f}_j$ for v_j is given by

$$\mathbf{f}_j = (0.5)^s \, \mathbf{1}_{2^s \times 1}, \tag{5.7}$$

where $\mathbf{1}_{2^s \times 1}$ is a vector of length 2^s containing all 1s. We designate such a vector with equal components as the *constant* vector.

Using (2.26), the *a posteriori* probability term $\mathbf{q}_j$ can be expressed as

$$\mathbf{q}_j = \alpha \prod_{i \in \mathbf{N}(j)}^{\bullet} \mathbf{r}_{ij} \quad , \tag{5.8}$$

where α is the normalizing constant, $\prod^{\bullet}$ is the term-by-term product operator and $\mathbf{r}_{ij}$ is the message sent by the CN c_i to the VN v_j. Observe that if any of the $\mathbf{r}_{ij}$ messages has all distinct components, then the components of $\mathbf{q}_j$ will be distinct. Therefore, a VN is considered to be recovered if any of the neighboring CNs sends a message having all distinct components.

Next we examine the notions of survived and dead CNs. Ignoring the permutation, the CN processing from (2.21) can be expressed as

$$\mathbf{r}_{ij} = \bigodot_{j' \in \mathbf{M}(i) \setminus \{j\}} \mathbf{q}_{j'i} \quad , \tag{5.9}$$

where $\odot$ denotes the circular convolution.

The circular convolution of a constant vector with any other vector results in a constant vector. If any of the $\mathbf{q}_{j'i}$ messages in (5.9) is a constant vector, then $\mathbf{r}_{ij}$ will become a constant vector. In that situation we consider the CN c_i to be dead with respect to the VN v_j. If all the components of $\mathbf{r}_{ij}$ are distinct then c_i is considered as a survived CN with respect to v_j.

Under these notions of symbol recovery and survived/dead CNs, the grouping algorithm from [32] can be applied to an NB-LDPC code in the same way as in the binary case [43]. The incidence matrix of the NB Tanner graph is considered as the parity-check matrix. The grouping algorithm is applied over this parity-check matrix. The RC puncturing scheme [43] for the NB-LDPC codes involves the following two steps:

(a) **Symbol Grouping**

The set of the VNs in the Tanner graph is partitioned into various groups with different levels of

recoverability using the grouping algorithm as discussed above.

(b) **Bitwise Spreading**

On the basis of the observations from numerous experiments, the authors of [43] have advocated for bitwise puncturing whenever it is feasible depending on the rate to be attained. At lower and intermediate rates, the VNs are punctured bitwise. If the recoverability level of a VN is lower, then one should puncture more number of bits per symbol and vice versa. Puncturing should be spread uniformly over all VNs of the same level of recoverability. This procedure of performing bitwise puncturing is referred to as *bitwise spreading* [43]. However, at a higher rate, complete bitwise puncturing may not be feasible. In that case some of the VNs with low recoverability level undergo symbolwise puncturing and the remaining VNs are punctured bitwise.

Note that in this puncturing scheme for the NB-LDPC codes, the sorting algorithm is not used.

5.2.3 EXIT Chart for Binary LDPC Codes [68]

In EXIT chart, the mutual information between the transmitted bit/symbol at an average VN and each related message is computed. The EXIT chart is basically a superposition of the input/output mutual information transfer curves for the VN decoder (VND) and the CN decoder (CND). Since the output of a neighboring VN acts as one of the inputs to a CN, the CND transfer curve is plotted with the reverse axes. For the convergence of the iterative decoding, the mutual information should increase with each message processing. This condition boils down to a nice and intuitive graphical interpretation: *the VND curve should lie above the reverse-axed CND curve for convergence.* In other words, the decoder trajectory should be able to sneak through the tunnel between the two transfer curves [10,67,68]. Figure 5.2 shows the typical EXIT curves and the decoder trajectory at a successful convergence.

With the help of the EXIT chart, the threshold for a particular code ensemble can be predicted. The threshold $\left(\frac{E_b}{N_0}\right)^*$ is the E_b/N_0 value for which the reverse-axed CND curve just touches the VND curve. The EXIT chart analysis assumes a cycle-free Tanner graph, an infinite codeword length and an infinite number of decoding iterations. The decoding steps are represented in the LLR domain, thereby providing the scope of modeling the messages as Gaussian random variables. Without loss of generality, an all-zero codeword is assumed to be transmitted.

For the binary LDPC codes over an AWGN channel with BPSK modulation, the channel LLR

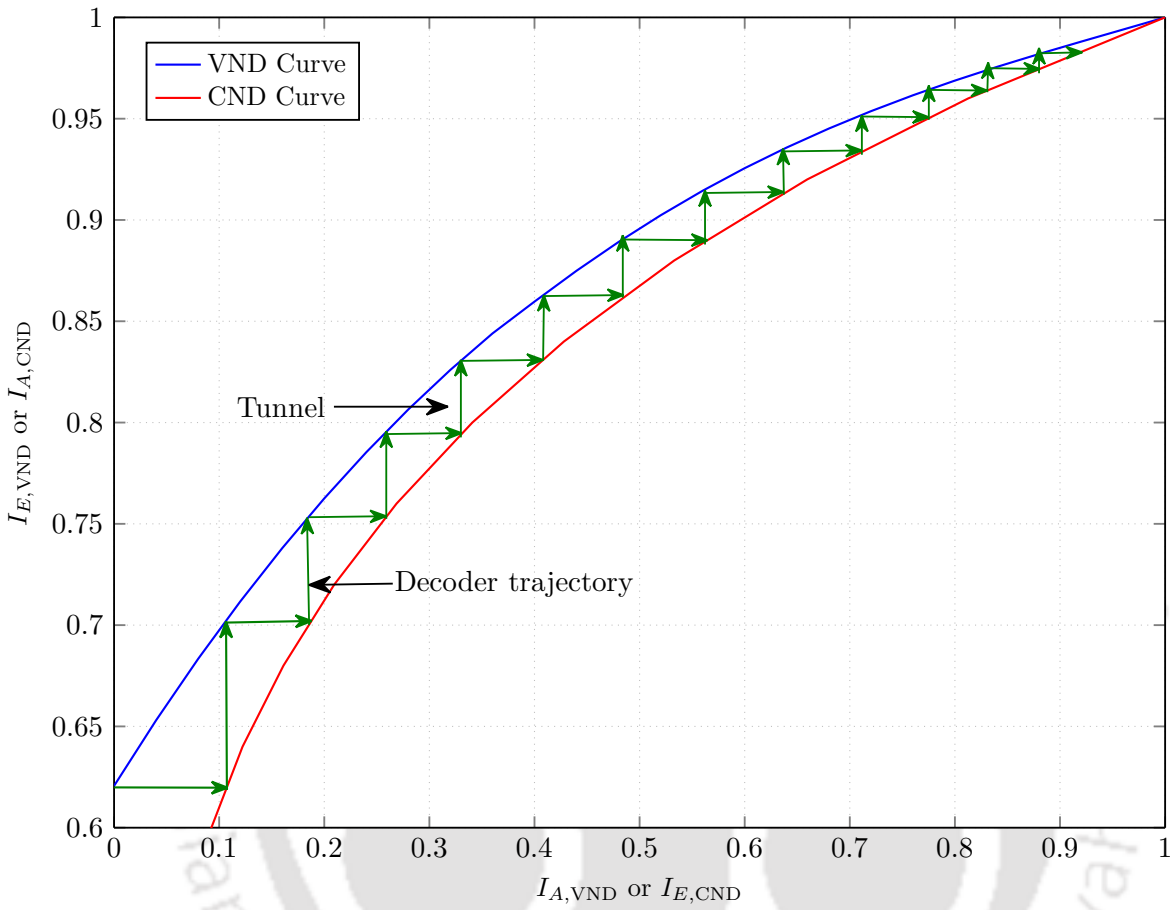


Figure 5.2: A typical EXIT chart for the LDPC codes

is exactly Gaussian distributed [15]. Many authors have assumed the Gaussian approximation for the VN and the CN outputs [15, 65], although the approximation for the CN output is less accurate. Nevertheless, the Gaussian approximation makes the EXIT chart analysis for the binary LDPC codes very simple.

For brevity, we shall refer the mutual information between the bit value X and any related extrinsic message L as the mutual information for L in the remaining part of the chapter. The steps of the EXIT chart for the binary LDPC codes are summarized below [68]:

(a) **Approximation of the Mutual Information for any Symmetric Gaussian Distributed Message**

Let $I(X;L)$ denote the mutual information between the bit value X and any extrinsic message L at the corresponding VN. Suppose the distribution of L is given by a symmetric Gaussian, i.e.,

$L \sim \mathcal{N}\left(\frac{\sigma^2}{2}, \sigma^2\right)$. In that case, $I(X;L)$ will be a function of a single parameter σ and denoted by $J(\sigma)$. The function $J(\sigma)$ is given by

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{\exp\left(-\frac{\left(l-\frac{\sigma^2}{2}\right)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}} . \log_2\left(1+\exp(-l)\right) dl.$$

It can be approximated using the Marquardt-Levenberg algorithm as [68]

$$J(\sigma) \approx \begin{cases} a_1\sigma^3 + b_1\sigma^2 + c_1\sigma, & 0 \leqslant \sigma \leqslant 1.6363 \\ 1 - \exp\left(a_2\sigma^3 + b_2\sigma^2 + c_2\sigma + d_2\right), & 1.6363 < \sigma < 10 \\ 1, & \sigma \geqslant 10, \end{cases}$$

where $a_1 = -0.0421062$, $b_1 = 0.209252$, $c_1 = -0.00640081$, $a_2 = 0.00181491$, $b_2 = -0.142675$, $c_2 = -0.0822054$, $d_2 = 0.0549608$.

The inverse J^{-1} is similarly approximated as

$$J^{-1}(I) \approx \begin{cases} \alpha_1 I^2 + \beta_1 I + \gamma_1\sqrt{I}, & 0 \leqslant I \leqslant 0.3646 \\ -\alpha_2 \ln\left[\beta_2(1-I)\right] - \gamma_2 I, & 0.3646 < I < 1, \end{cases}$$

where $\alpha_1 = 1.09542, \beta_1 = 0.214217, \gamma_1 = 2.33727, \alpha_2 = 0.706692, \beta_2 = 0.386013, \gamma_2 = -1.75017$.

(b) **Computation of the VND Curve**

Let $I_{A,\mathrm{VND}}$ denote the *a priori* information input to the VND. It represents the mutual information for the CN→ VN messages. For a given $I_{A,\mathrm{VND}}$, the extrinsic information output $I_{E,\mathrm{VND}}(I_{A,\mathrm{VND}}, i)$ for a degree-i VN is given by

$$I_{E,\mathrm{VND}}(I_{A,\mathrm{VND}}, i) = J\left(\sqrt{(i-1)\left[J^{-1}(I_{A,\mathrm{VND}})\right]^2 + \sigma_{\mathrm{ch}}^2}\right),$$

where $\sigma_{\mathrm{ch}}^2 = \frac{4}{\sigma_n^2}$ is the variance of the channel LLR and σ_n^2 is the variance of the channel noise. As $\sigma_n^2 = \frac{N_0}{2}$, σ_{ch}^2 is related to the $\frac{E_b}{N_0}$ value as

$$\sigma_{\mathrm{ch}}^2 = 8\left(\frac{E_b}{N_0}\right).$$

The effective VND EXIT function is given by

$$I_{E,\mathrm{VND}}\left(I_{A,\mathrm{VND}}\right)=\sum_{i=1}^{d_v^{\max}}\lambda_i I_{E,\mathrm{VND}}\left(I_{A,\mathrm{VND}},i\right), \tag{5.10}$$

where $(\lambda_1,\cdots,\lambda_{d_v^{\max}})$ is the edge-perspective degree distribution for the VNs.

The VND EXIT curve can be obtained by plotting the $I_{E,\mathrm{VND}}$ values over a fine grid of $I_{A,\mathrm{VND}}$ in the interval $[0,1]$.

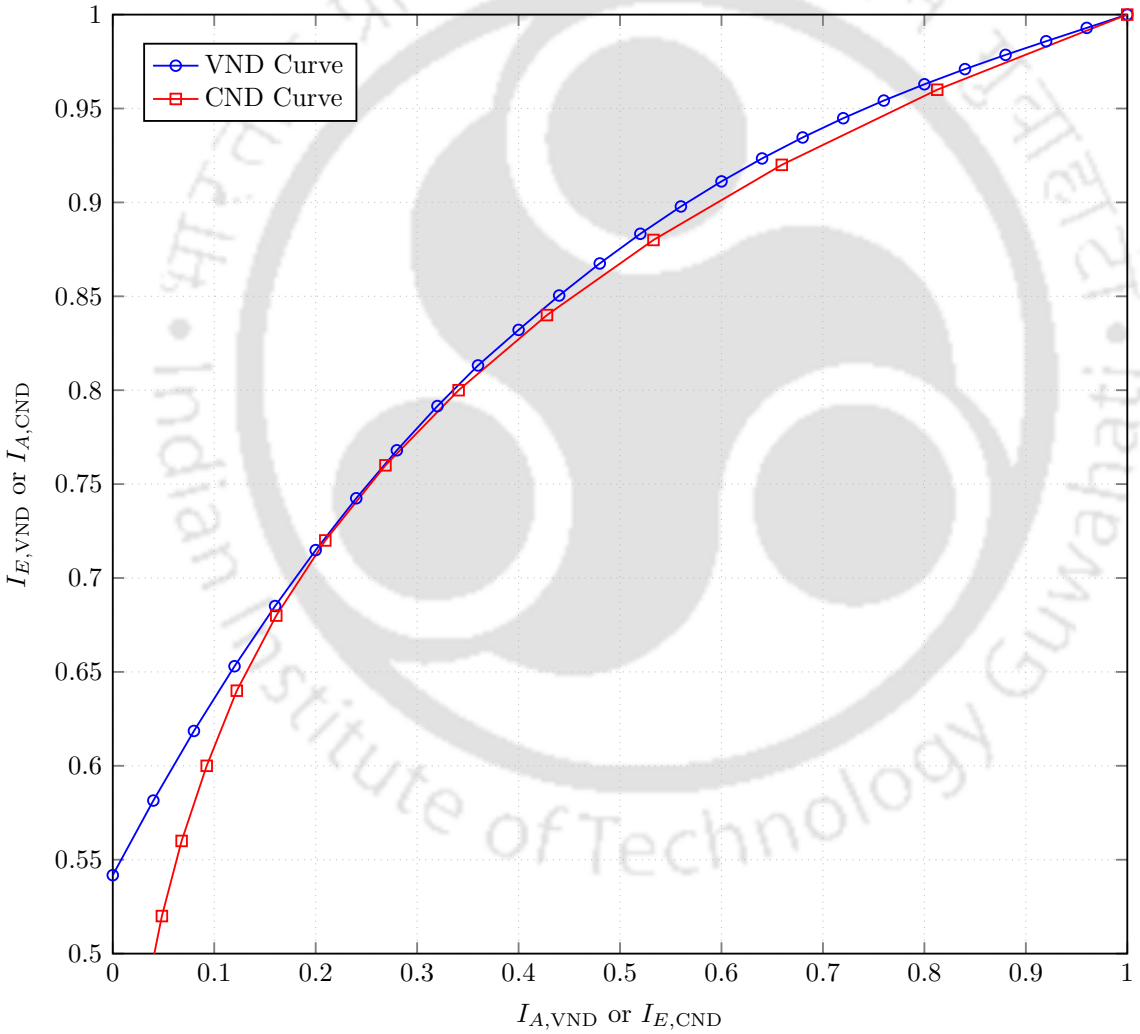


Figure 5.3: EXIT chart for the rate-0.5 irregular code ensemble corresponding to $d_v^{\max}=5$ [59] at $E_b/N_0=0.73$ dB

(c) Computation of the CND Curve

Let $I_{A,\mathrm{CND}}$ denote *a priori* information input to the CND. It represents the mutual information for the VN→ CN messages. By exploiting a duality property of the repetition codes and the single

parity-check codes, the extrinsic information output $I_{E,\mathrm{CND}}\left(I_{A,\mathrm{CND}}, j\right)$ for a degree-j CN can be approximated as

$$I_{E,\mathrm{CND}}\left(I_{A,\mathrm{CND}}, j\right) \approx 1 - J\left(\sqrt{j-1}.J^{-1}\left(1 - I_{A,\mathrm{CND}}\right)\right).$$

The effective CND EXIT function is given by

$$I_{E,\mathrm{CND}}\left(I_{A,\mathrm{CND}}\right) = \sum_{j=1}^{d_c^{\max}} \rho_j I_{E,\mathrm{CND}}\left(I_{A,\mathrm{CND}}, j\right), \tag{5.11}$$

where $\left(\rho_1, \cdots, \rho_{d_c^{\max}}\right)$ is edge-perspective degree distribution for the CNs.

The CND EXIT curve can be obtained by plotting the $I_{E,\mathrm{CND}}$ values over a fine grid of $I_{A,\mathrm{CND}} \in [0, 1]$. Unlike the VND curve, it does not depend on the operating $\frac{E_b}{N_0}$ value. Superimposing the VND curve and the reverse-axed CND curve on the same plot, we obtain the EXIT chart.

Example 5.1. *Consider the following optimized degree distribution pair corresponding to* $d_v^{\max} = 5$ *for rate-0.5* [59]*:*

$$\lambda(x) = 0.32660x + 0.11960x^2 + 0.18393x^3 + 0.36988x^4$$
$$\textit{and} \quad \rho(x) = 0.7855x^5 + 0.2144x^6.$$

Figure 5.3 shows the EXIT chart for this degree distribution at $E_b/N_0 = 0.73$ *dB. Observe that the VND curve lies just slightly above the CND curve. Therefore, the threshold for the code ensemble can be considered as* 0.73 *dB.*

5.2.4 EXIT Chart for NB-LDPC Codes [10]

Bennatan and Burshtein [10] have proposed an accurate model of the EXIT chart for the NB-LDPC codes over any arbitrary discrete memoryless channel. They have derived many interesting results by analyzing the NB-LDPC codes in the framework of random coset vectors and random selection of the nonzero elements of GF(q) in the parity-check matrix. Two key outcomes of their analysis are the *symmetry* and the *permutation-invariance* properties associated with the LLR vector messages. Together with the Gaussian approximation, the $(q-1)$-dimensional distribution of the CN→VN vector messages can be described by a single scalar parameter. This property facilitated the formulation of the EXIT chart for the NB-LDPC codes. This EXIT chart method is described in the following. The

discussion starts with the formulation of the decoding steps in the LLR domain.

5.2.4.1 Formulation of the Decoding Steps in the LLR Domain

For the decoding of the NB-LDPC codes over GF($q = 2^s$) , $s \in \mathbb{Z}^+ \backslash \{1\}$, we have described the FFT-QSPA in Section 2.7.1. Note that the FFT-QSPA is expressed in the probability domain. However, the decoding steps preferably should be represented in the LLR domain for the EXIT chart analysis.

Given a probability vector $\mathbf{x} = [x_0 \dots x_{q-1}]^T$, the LLR w_i is defined as

$$w_i \triangleq \log\left(\frac{x_0}{x_i}\right), \quad i = 0, \dots, q-1. \tag{5.12}$$

Clearly $w_0 = 0$. Therefore, any LLR message for a code defined over GF(q) is represented by a $(q-1)$-dimensional vector $\mathbf{w} = [w_1 \dots w_{q-1}]^T$. Given an LLR vector $\mathbf{w} = [w_1 \dots w_{q-1}]^T$, the corresponding probability vector can be obtained as

$$x_i = \frac{\exp(-w_i)}{1 + \sum_{k=1}^{q-1} \exp(-w_k)}, \quad i = 0, \dots, q-1. \tag{5.13}$$

With the definition provided in (5.12), the computation of the channel *a posteriori* probability, the VN processing and the CN processing described in Section 2.7.1 can be expressed in the LLR domain as follows:

(a) **Computation of the Channel LLR**

The channel LLR vector for the VN v_j is represented as $\mathbf{L}_{\text{ch}_j} = \left[L_{\text{ch}_j}(1) \dots L_{\text{ch}_j}(q-1)\right]^T$. According to (5.12), $L_{\text{ch}_j}(a)$, $a \in \text{GF}(q)^-$ is given by

$$L_{\text{ch}_j}(a) = \log\left(\frac{f_j^0}{f_j^a}\right),$$

where f_j^a is the channel *a posteriori* probability of the jth symbol being a. Using (2.17), $L_{\text{ch}_j}(a)$ can be expressed as

$$L_{\text{ch}_j}(a) = \sum_{b=1}^{s} a_b \frac{2y_{j,b}}{\sigma_n^2}, \tag{5.14}$$

where $[a_1 \dots a_s]$ is the bit-level representation of a, $y_{j,b}$ is the channel output for the bth bit of the jth symbol and σ_n^2 is the variance of the channel noise.

(b) **VN Processing**

The LLR-domain message sent by the VN v_j to the CN c_i is represented as $\mathbf{V}_{ji} = [V_{ji}(1) \dots V_{ji}(q-1)]^T$. As per (5.12), $V_{ji}(a)$, $a \in \mathrm{GF}(q)^-$ is given by

$$V_{ji}(a) = \log\left(\frac{q_{ji}(0)}{q_{ji}(a)}\right), \tag{5.15}$$

where $\mathbf{q}_{ji} = [q_{ji}(a)]_{a \in \mathrm{GF}(2^s)}$ is the probability-domain message sent by v_j to c_i.

In the EXIT chart analysis, the exact interconnections between the VNs and the CNs are not taken into consideration. Therefore, a VN→CN message like $\mathbf{V}_{ji}$ is simply referred by the LLR-vector random variable $\mathbf{V}$.

(c) **CN Processing**

The LLR-domain message sent by c_i to v_j is represented as $\mathbf{U}_{ij} = [U_{ij}(1) \dots U_{ij}(q-1)]^T$. $U_{ij}(a)$, $a \in \mathrm{GF}(q)^-$ is given by

$$U_{ij}(a) = \log\left(\frac{r_{ij}(0)}{r_{ij}(a)}\right), \tag{5.16}$$

where $\mathbf{r}_{ij} = [r_{ij}(a)]_{a \in \mathrm{GF}(2^s)}$ is the probability-domain message sent by c_i to v_j.

Now consider again the VN processing step in (5.15). Using (2.25), $\mathbf{V}_{ji}$ can be expressed as

$$\mathbf{V}_{ji} = \mathbf{L}_{\mathrm{ch}_j} + \sum_{i' \in \mathbf{N}(j) \setminus \{i\}} \mathbf{U}_{i'j} \tag{5.17}$$

$\mathbf{U}_{ij}$ can be efficiently computed in the probability-domain via the FFT after carrying out the permutation and the depermutation steps as shown in Section 2.7.1. However, this FFT-based technique is not feasible in the LLR domain. To apply the FFT-based CN processing as shown in (2.22), any LLR vector message $\mathbf{V}_{ji}$ must be converted into the corresponding probability-domain message $\mathbf{q}_{ji}$. This conversion can be carried out according to (5.13). After computing all $\mathbf{r}_{ij}$ messages using the FFT-based technique, they are converted back to the corresponding LLR-domain $\mathbf{U}_{ij}$ messages using (5.16).

The LLR-vector random variable $\mathbf{U}$ is used to represent a CN→VN message in the EXIT chart analysis.

5.2.4.2 Formulation of the Mutual Information for an LLR-vector Random Variable

Let us explain the meanings of the $+g$ and $\times g$ operations which will be used in the analysis of an LLR-vector random variable $\mathbf{W}$.

Definition 5.1. *Suppose a realization* $\mathbf{w}$ *of an LLR-vector random variable* $\mathbf{W}$ *is given by* $\mathbf{w} = [w_1\ w_2\ \ldots w_{q-1}]^T$. *For an element* $g \in \mathrm{GF}(q)$, $\mathbf{w}^{+g}$ *is defined by*

$$\mathbf{w}^{+g} \triangleq [(w_{1+g} - w_g)\ \ (w_{2+g} - w_g) \ldots (w_{q-1+g} - w_g)]^T,$$

where the subtractions are performed over $\mathrm{GF}(q)$. *Similarly* $\mathbf{w}^{\times g}$ *is defined by*

$$\mathbf{w}^{\times g} \triangleq \left[w_g\ w_{2.g} \ldots w_{(q-1).g}\right]^T,$$

where multiplications in the subscripts are performed over $\mathrm{GF}(q)$.

Three important lemmas and one theorem in [10] established the important properties of $\mathbf{W}$. These are presented below.

Lemma 5.1. *An LLR-vector random variable* $\mathbf{W}$ *is symmetric if and only if*

$$\Pr[\mathbf{W} = \mathbf{w}] = e^{w_i} \Pr\left[\mathbf{W} = \mathbf{w}^{+i}\right].$$

Under the assumptions of the cycle-free Tanner graph and all-zero codeword transmission, all messages ($\mathbf{U}$ and $\mathbf{V}$) on the graph at any iteration, are symmetric. Recall from Section 2.7.1 that, the iterative decoding of the NB-LDPC codes involves the permutation and the depermutation steps. The following lemma describes the permutation-invariance property associated with an LLR-vector random variable.

Lemma 5.2. *An LLR-vector random variable* $\mathbf{W}$ *is permutation-invariant if and only if, for any fixed* $h \in \mathrm{GF}(q)^-$, *the random variable* $\mathbf{\Omega} \triangleq \mathbf{W}^{\times h}$ *is distributed identically with* $\mathbf{W}$.

If the edge labels are selected uniformly at random from $\mathrm{GF}(q)^-$, then the CN$\rightarrow$ VN messages ($\mathbf{U}$) are permutation-invariant.

Theorem 5.1. *Let* $\mathbf{W}$ *be an LLR-vector random variable, Gaussian distributed with a mean* $\mathbf{m}$ *and covariance matrix* $\mathbf{\Sigma}$. *Assume that the probability density function* $f(\mathbf{w})$ *of* $\mathbf{W}$ *exists and that* $\mathbf{\Sigma}$ *is*

nonsingular. Then $\mathbf{W}$ *is both symmetric and permutation-invariant if and only if there exists* $\sigma > 0$ *such that*

$$\mathbf{m} = \begin{bmatrix} \sigma^2/2 \\ \sigma^2/2 \\ \cdots \\ \sigma^2/2 \end{bmatrix} \; ; \quad \mathbf{\Sigma} = \begin{bmatrix} \sigma^2 & & & \sigma^2/2 \\ & \sigma^2 & & \\ & & \cdots & \\ \sigma^2/2 & & & \sigma^2 \end{bmatrix}. \tag{5.18}$$

Thus, $m_i = \sigma^2/2$, $i = 1, \ldots, q-1$ *and* $\Sigma_{i,j} = \sigma^2$ *if* $i = j$ *and* $\sigma^2/2$ *otherwise.*

The following lemma now gives an expression for the mutual information between any message $\mathbf{W}$ and the corresponding codeword symbol C.

Lemma 5.3. *Under the assumption of a cycle-free Tanner graph and an all-zero codeword transmission, the mutual information between any message* $\mathbf{W} = [W_1\, W_2 \cdots W_{q-1}]^T$ *and the corresponding codeword symbol* C *is given by,*

$$I(C; \mathbf{W}) = 1 - E\left[\log_q \left(1 + \sum_{i=1}^{q-1} e^{-W_i}\right) | C = 0\right]. \tag{5.19}$$

5.2.4.3 Modeling of the Message Vectors

For the NB-LDPC codes over a BI-AWGN channel, the distribution of a channel LLR vector is not jointly Gaussian. To establish this fact, rewrite (5.14) for any VN as

$$L_{\text{ch}}(a) = \sum_{b=1}^{s} a_b \frac{2y_b}{\sigma_n^2} \tag{5.20}$$

$$= \sum_{b=1}^{s} a_b l_b, \tag{5.21}$$

where the bitwise channel LLR $l_b = \frac{2y_b}{\sigma_n^2}$ is distributed as $\mathcal{N}\left(\frac{2}{\sigma_n^2}, \frac{4}{\sigma_n^2}\right), \forall b \in \{1, \ldots, s\}$ [15].

The channel LLR vector $\mathbf{L}_{\text{ch}} = [L_{\text{ch}}(1) \ldots L_{\text{ch}}(q-1)]^T$ can be expressed in a compact form as

$$\mathbf{L}_{\text{ch}} = \mathbf{A}_s.\mathbf{l}, \tag{5.22}$$

where $\mathbf{A}_s$ is a $(2^s - 1) \times s$ matrix in which the ith row is the bit-level representation of the symbol i

and $\mathbf{l} = [l_1\ l_2 \dots l_s]^T$. The covariance matrix of $\mathbf{L}_{\text{ch}}$ is now given by

$$\Sigma_{\mathbf{L}_{\text{ch}}} = \frac{4}{\sigma_n^2}\mathbf{A}_s.\mathbf{A}_s^T. \tag{5.23}$$

As the number of rows is greater than the number of columns for $\mathbf{A}_s$, $\left(\mathbf{A}_s.\mathbf{A}_s^T\right)$ is not invertible. Therefore, $\Sigma_{\mathbf{L}_{\text{ch}}}$ is not invertible. Consequently $\mathbf{L}_{\text{ch}}$ cannot be considered as a multi-variate Gaussian random variable. Instead, the distribution of $\mathbf{L}_{\text{ch}}$ can be considered as a mixture of several one-dimensional Gaussian random variables. The non-Gaussianity of the channel LLR makes the modeling of the EXIT chart difficult for the NB-LDPC codes.

Now consider the VN processing shown in (5.17). Denoting $\mathbf{V}_{ji}$, $\mathbf{L}_{\text{ch}_j}$, and $\mathbf{U}_{i'j}$ by the random variables $\mathbf{V}$, $\mathbf{L}_{\text{ch}}$ and $\mathbf{U}_k$, respectively, we can rewrite (5.17) for a degree-i VN as

$$\mathbf{V} = \mathbf{L}_{\text{ch}} + \mathbf{T}, \tag{5.24}$$

where

$$\mathbf{T} = \sum_{k=1}^{i-1} \mathbf{U}_k. \tag{5.25}$$

$\mathbf{T}$ is called the *intermediate* message coming from a super CN as illustrated in Figure 5.4.

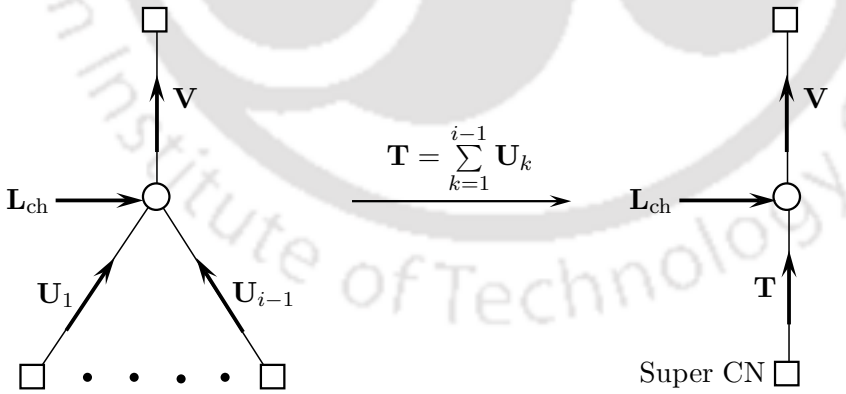


Figure 5.4: Diagrammatic illustration of the intermediate message

A CN→VN message $\mathbf{U}_k$ is assumed to be Gaussian distributed. It is symmetric and permutation-invariant. Therefore, using Theorem 5.1, the Gaussian distribution of any $\mathbf{U}_k$ has the mean vector and the covariance matrix shown in (5.18) specified by a single parameter σ. As $\mathbf{U}_k$ follows the Gaussian distribution shown in (5.18), the intermediate message $\mathbf{T}$ also follows the same distribution with a different σ parameter.

Observe that $\mathbf{V}$ involves two vectors: $\mathbf{L}_{\text{ch}}$ and $\mathbf{T}$. Out of them, $\mathbf{T}$ is modelled as Gaussian dis-

tributed and $\mathbf{L}_{\text{ch}}$ is non-Gaussian. Therefore, it becomes difficult to identify any accurate probabilistic distribution for $\mathbf{V}$. The authors in [10] have suggested to generate a sufficient number of samples of $\mathbf{V}$ to compute $I(C;\mathbf{V})$ empirically.

5.2.4.4 Steps of the EXIT Chart for the NB-LDPC Codes

The steps of the EXIT chart for the NB-LDPC codes can now be summarized as follows:

(a) **Approximation of the Mutual Information for any Symmetric and Permutation-invariant Gaussian Distributed Message**

The procedure of the EXIT chart model starts with an off-line step of polynomial fitting of the mutual information for a symmetric and permutation-invariant Gaussian distributed message $\mathbf{W}$. The mean vector and covariance matrix for $\mathbf{W}$ are given by (5.18) and depend only on σ. The corresponding mutual information $J_q(\sigma) = I(C;\mathbf{W})$ is a function of only one parameter σ. $I(C;\mathbf{W})$ can be calculated empirically using (5.19). For values of σ along a fine grid in $[0,7]$ (as $J_q(6.5) \approx 1$), $I(C;\mathbf{W})$ is computed and then the best polynomial fits for $J_q(\sigma)$ and $J_q^{-1}(\cdot)$ are found out. This step is done only once for a given field GF(q).

(b) **Computation of the VND Curve**

For an operating E_b/N_0 value, the VN→CN message $\mathbf{V}$ is obtained by adding the intermediate message $\mathbf{T}$ and the channel LLR $\mathbf{L}_{\text{ch}}$ as shown in (5.24). $\mathbf{T}$ follows the distribution shown in (5.18) specified by σ. $\mathbf{L}_{\text{ch}}$ is generated for the particular E_b/N_0 value. Therefore, the mutual information $J_V(\sigma) = I(C;\mathbf{V})$ depends only on σ for a given E_b/N_0 value. A sufficient number of samples of $\mathbf{V}$ are generated to compute $I(C;\mathbf{V})$ empirically using (5.19). The best polynomial fits for $J_V(\sigma)$ and $J_V^{-1}(\cdot)$ are found out by computing $J_V(\sigma)$ for the values of σ along a fine grid in $[0,7]$.

For a VN of degree i, the VND EXIT function is approximated as

$$I_{E,\text{VND}}(I_{A,\text{VND}}, i) \approx J_V\left(\sqrt{(i-1)\left[J_q^{-1}(I_{A,\text{VND}})\right]^2}\right). \tag{5.26}$$

The effective VND EXIT function $I_{E,\text{VND}}(I_{A,\text{VND}})$ can now be computed using (5.10) over a fine grid of $I_{A,\text{VND}}$ in $[0,1]$.

(c) **Computation of the CND Curve**

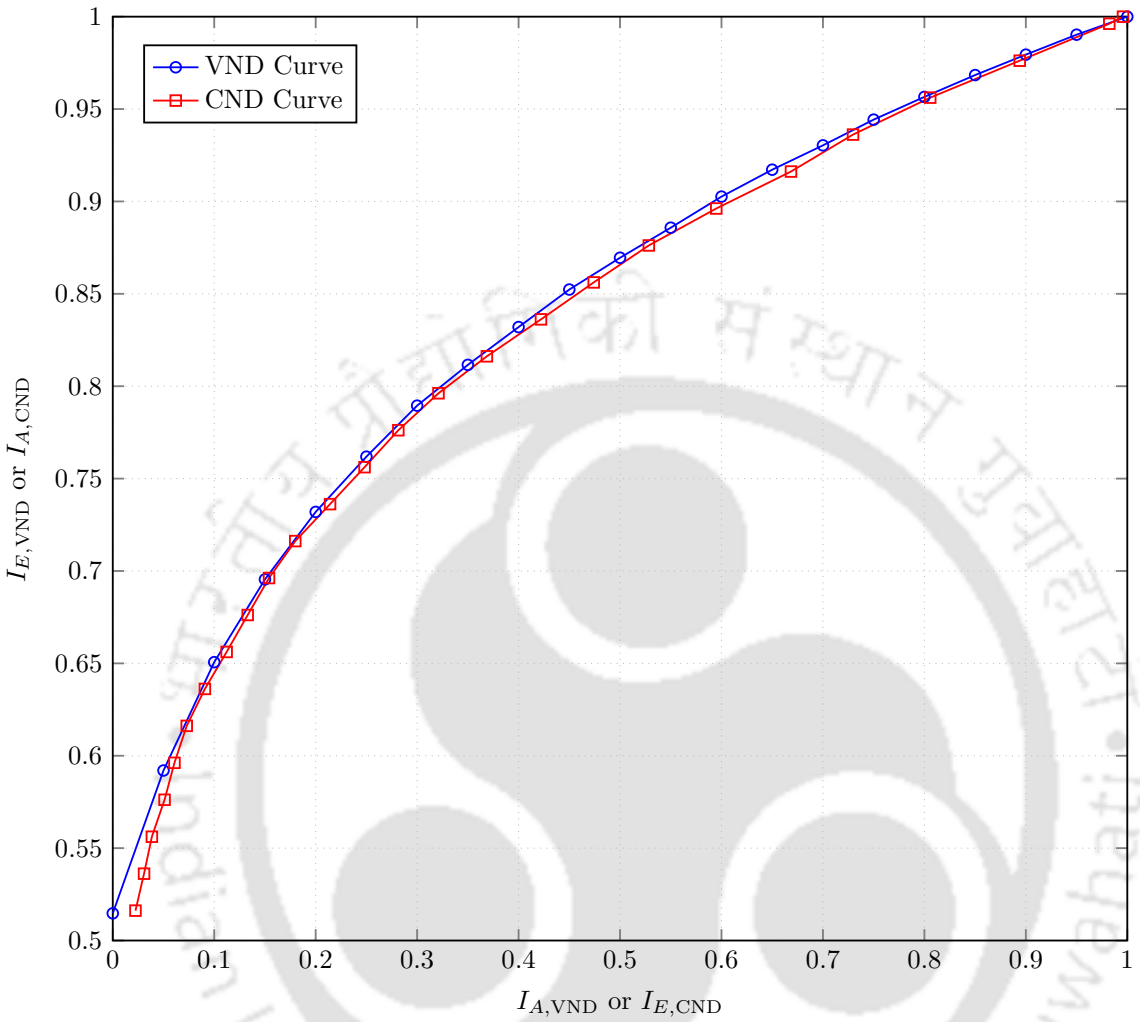


Figure 5.5: EXIT chart for the optimized rate-0.5 code over GF(2^4) [30] at $E_b/N_0 = 0.3638$ dB

The argument $I_{A,\mathrm{CND}}$ for the CND transfer curve represents the mutual information for the VN→CN messages. The computation of the mutual information for the CND output is somewhat tedious. It is because, an incoming message $\mathbf{V}$ (VN→ CN) does not follow any standard distribution. In order to generate the samples of $j-1$ number of $\mathbf{V}$ messages for a CN of degree j, one has to repeat the procedure for the VND curve. The samples of the channel LLRs $\mathbf{L}_{\mathrm{ch}}$ (specified by E_b/N_0) and the intermediate messages $\mathbf{T}$ (specified by σ) are generated to produce the samples for $j-1$ number of $\mathbf{V}$ messages. The value of σ is obtained by applying $J_V^{-1}(I_{A,\mathrm{CND}})$. The $j-1$ number of $\mathbf{V}$ messages are used to calculate the output of the CN, the CN→VN message $\mathbf{U}$. $\mathbf{U}$ is computed using the FFT-based technique after accomplishing the required transformation between the LLR domain and the probability domain as per (5.12) and (5.13). Sufficient samples

of $\mathbf{U}$ are generated to compute $I_{E,\mathrm{CND}}\left(I_{A,\mathrm{CND}}, j\right) = I\left(C;\mathbf{U}\right)$ using (5.19). The effective CND EXIT function $I_{E,\mathrm{CND}}\left(I_{A,\mathrm{CND}}\right)$ can be obtained from (5.11). Considering a fine grid of $I_{A,\mathrm{CND}}$ in $[I_{E,\mathrm{VND}}(0), 1]$, $I_{E,\mathrm{CND}}\left(I_{A,\mathrm{CND}}\right)$ values are calculated to obtain the CND EXIT curve.

Example 5.2. *Consider a rate-0.5 code over* GF (2^4) *having the following optimum degree distributions* [30]*:*

$$\lambda\left(x\right) = 0.5376x + 0.1678x^2 + 0.1360x^4 + 0.1586x^9$$

$$\textit{and} \quad \rho\left(x\right) = 0.5169x^4 + 0.4831x^5.$$

The EXIT chart for this code at $E_b/N_0 = 0.3638$ *dB is shown in Figure 5.5. It can be observed that the CND curve just touches the VND curve at* 0.3638 *dB. Therefore, the threshold for the code is* 0.3638 *dB.*

5.3 Proposed EXIT Chart for Punctured NB-LDPC Codes

We call puncturing to be *regular* if the same number of bits are punctured for all the VNs in a Tanner graph and *irregular* otherwise. The regular puncturing with a random selection of the VNs is the simplest case of puncturing for the NB-LDPC codes. In the *first* part, we develop the EXIT chart model in the above context to identify the main deciding factors for the performance of various puncturing patterns. Equipped with the findings of this part, we investigate the more complicated case of irregular puncturing with an intentional selection of the VNs in the *second* part.

5.3.1 Random Selection of the VNs and Regular Puncturing

Let $b_p \in \{1, \cdots, s\}$ denote the number of punctured bits per VN. Suppose the mother code is of rate r_0. Then for a target rate r_l, the number of punctured VNs is given by

$$N_l^p = \left\lfloor \frac{N \times s \times (r_l - r_0)}{r_l \times b_p} \right\rfloor, \tag{5.27}$$

where N is the total number of VNs in the Tanner graph.

Two extreme cases of N_l^p arise:

(a) When $b_p = s$ (symbolwise puncturing), N_l^p becomes minimum. We puncture the minimum possible number of VNs. But these VNs are punctured completely.

(b) When $b_p = 1$, N_l^p is the largest. We puncture the maximum number of VNs. Each of these VNs undergoes the minimal puncturing.

It is very difficult to straightway select the value of b_p for the optimum performance. The effects of the puncturing patterns with different values of b_p can be investigated with the help of the EXIT chart. The method to obtain the VND and the CND curves in the context of the regular puncturing of the randomly selected VNs is discussed below.

(a) **Computation of the VND Curve**

Let p_v denote the probability that a VN is punctured. p_v can be expressed as

$$p_v = \frac{N_l^p}{N}. \tag{5.28}$$

For the punctured NB-LDPC codes, the effect of puncturing should be incorporated in the samples of $\mathbf{L}_{\text{ch}}$. The expression for $\mathbf{L}_{\text{ch}} = [L_{\text{ch}}(a)]_{a\in \text{GF}(2^s)^-}$ from (5.20) is given by

$$L_{\text{ch}}(a) = \sum_{k=1}^{s} a_k \frac{2y_k}{\sigma_n^2} \tag{5.29}$$

Recall that $[a_1 \dots a_s]$ is the bit-level representation of the symbol a and y_k is the received value for kth bit of the symbol. Let $\mathcal{P}_{b_p}$ be the set of b_p puncturing bit indices selected uniformly at random from $\{1, 2, \dots, s\}$. For all $k \in \mathcal{P}_{b_p}$, we put $y_k = 0$ in (5.29). The resulting sample of $\mathbf{L}_{\text{ch}}$ is used in devising the EXIT chart for the punctured NB-LDPC codes. It is added to the intermediate message $\mathbf{T}$ (distributed as in (5.18) specified by σ) to obtain a sample of the VN→CN message $\mathbf{V}$. Let $J_V^0(\sigma)$ and $J_V^{b_p}(\sigma)$ denote respectively the mutual information for $\mathbf{V}$ flowing from an un-punctured and a punctured VN with b_p punctured bits. Then the effective $J_V(\sigma)$ can be found as

$$J_V(\sigma) = (1 - p_v) J_V^0(\sigma) + p_v J_V^{b_p}(\sigma), \tag{5.30}$$

The remaining steps for finding the VND curve are carried out as described in Section 5.2.4.4.

(b) **Computation of the CND Curve**

Consider a CN c of degree j connected to the neighboring VNs $\{v_1, v_2, \cdots, v_j\}$. Without loss of generality, we consider the message flowing along the edge $c \to v_1$ in order to find the CND curve.

We have to generate the $\mathbf{V}$ messages from the $j-1$ VNs in the set $\mathcal{N} = \{v_2, v_3, \cdots, v_j\}$. The probability that the set $\mathcal{N}$ contains τ number of punctured VNs is given by

$$p_{\mathcal{N}}(\tau) = \begin{pmatrix} j-1 \\ \tau \end{pmatrix} (p_v)^{\tau} (1-p_v)^{j-\tau-1}, \quad \tau = 0, 1, \ldots, j-1. \tag{5.31}$$

For each value of τ, we generate the samples of $\mathbf{L}_{\text{ch}}$ for the $(j-1)$ VNs in $\mathcal{N}$. τ VNs to be punctured are selected uniformly at random from $\mathcal{N}$. We generate the intermediate message $\mathbf{T}$ at each of the $(j-1)$ VNs and and add it to the respective $\mathbf{L}_{\text{ch}}$ to obtain the $j-1$ number of $\mathbf{V}$ messages at the input of the CN c. Using these $\mathbf{V}$ messages, we compute the output $\mathbf{U}$ at c. Suppose $I_{E,\text{CND}}(I_{A,\text{CND}}, j, \tau)$ is the EXIT function for the output $\mathbf{U}$ of the CN of degree j when the number of punctured VNs in $\mathcal{N}$ is τ. The effective EXIT function for a degree-j CN can be obtained as

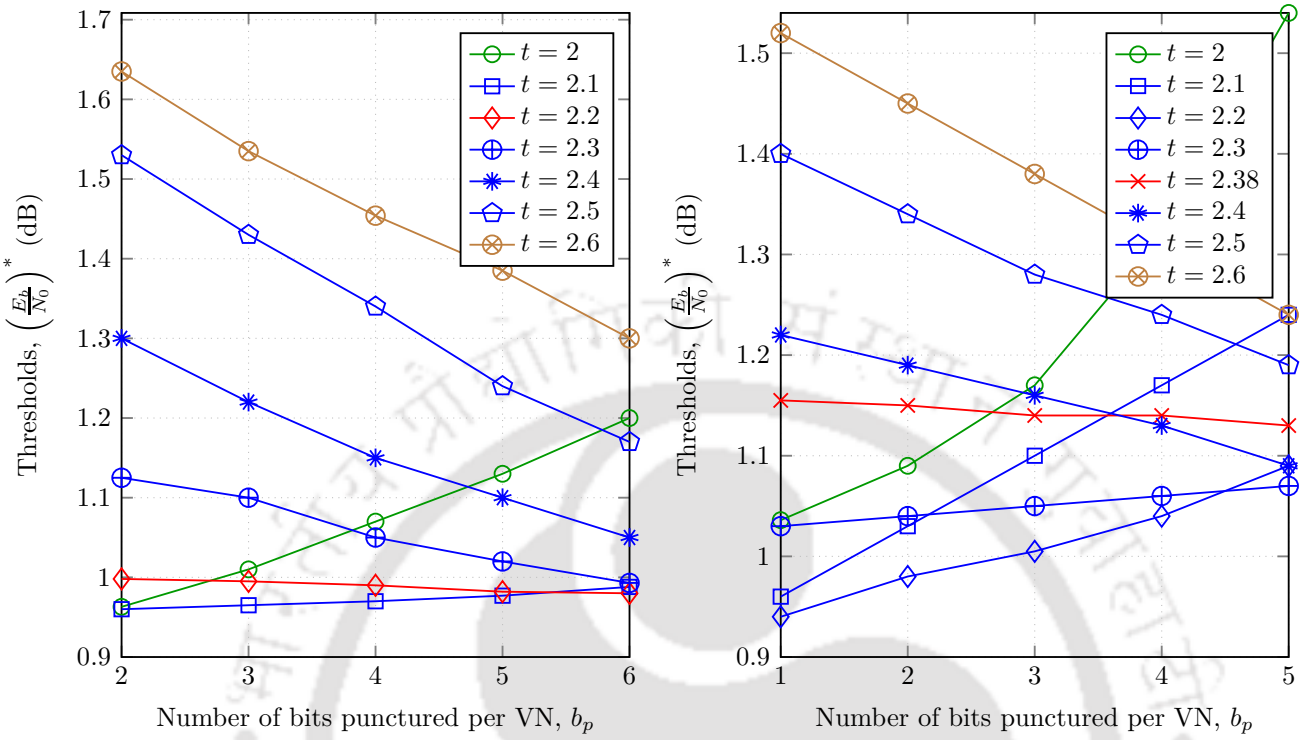
$$I_{E,\text{CND}}(I_{A,\text{CND}}, j) = \sum_{\tau=0}^{j-1} p_{\mathcal{N}}(\tau) I_{E,\text{CND}}(I_{A,\text{CND}}, j, \tau). \tag{5.32}$$

Finally, $I_{E,\text{CND}}(I_{A,\text{CND}})$ can be computed using (5.11).

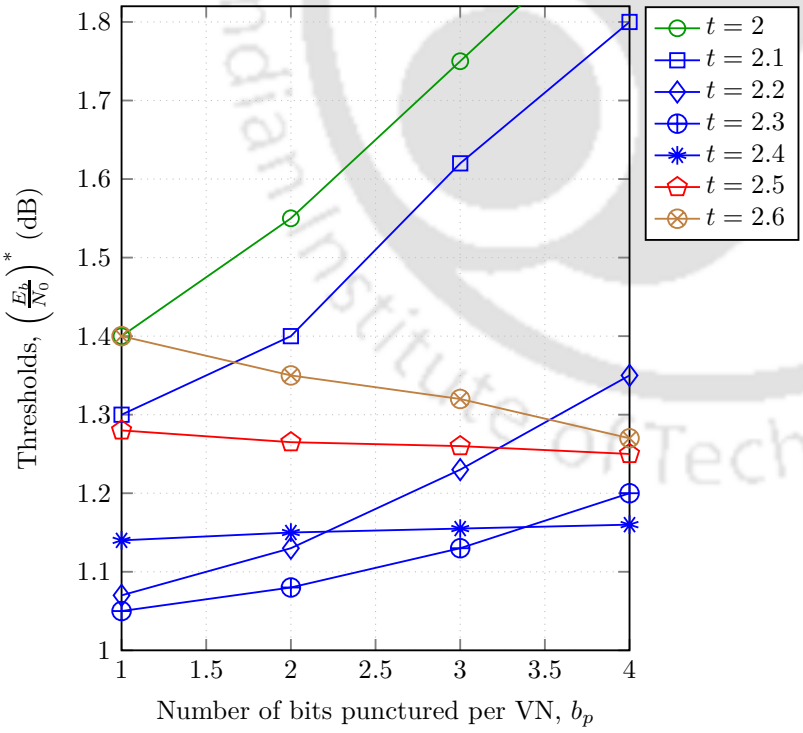
5.3.1.1 Study on Quasi-regular NB-LDPC Codes

We use the proposed EXIT chart model to investigate the effects of various puncturing patterns for quasi-regular NB-LDPC codes with different values of mean column weight (t). For a quasi-regular code of mean column weight t, the degrees of the VNs are restricted to $\{\lfloor t \rfloor, \lceil t \rceil\}$, where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the floor and the ceiling functions respectively. The degrees of the CNs are restricted to $\left\{\left\lfloor \frac{t}{1-r_0} \right\rfloor, \left\lceil \frac{t}{1-r_0} \right\rceil\right\}$, where r_0 is the rate. We calculate the thresholds for punctured quasi-regular codes over GF(2^6), GF(2^5) and GF(2^4) with $r_0 = 0.5$ and $r_l \in \{0.6, 0.7, 0.8\}$. We consider $N = 142$ to obtain the value of p_v.

Figure 5.6(a) shows the thresholds for punctured codes over GF(2^6) at $r_l = 0.6$. The threshold values are plotted against b_p for different values of t. Several interesting observations can be made from the figure. When $t = 2$, puncturing a smaller number of bits per VN gives better thresholds. At $t = 2.6$, the scenario is completely opposite. In between these two contrasting cases, there exists a point around $t = 2.2$, where the thresholds appear to be unaffected by b_p. We call this value of t as the *crossover* point (t_c). From Figure 5.6(b) and Figure 5.6(c), it can be observed that the values of



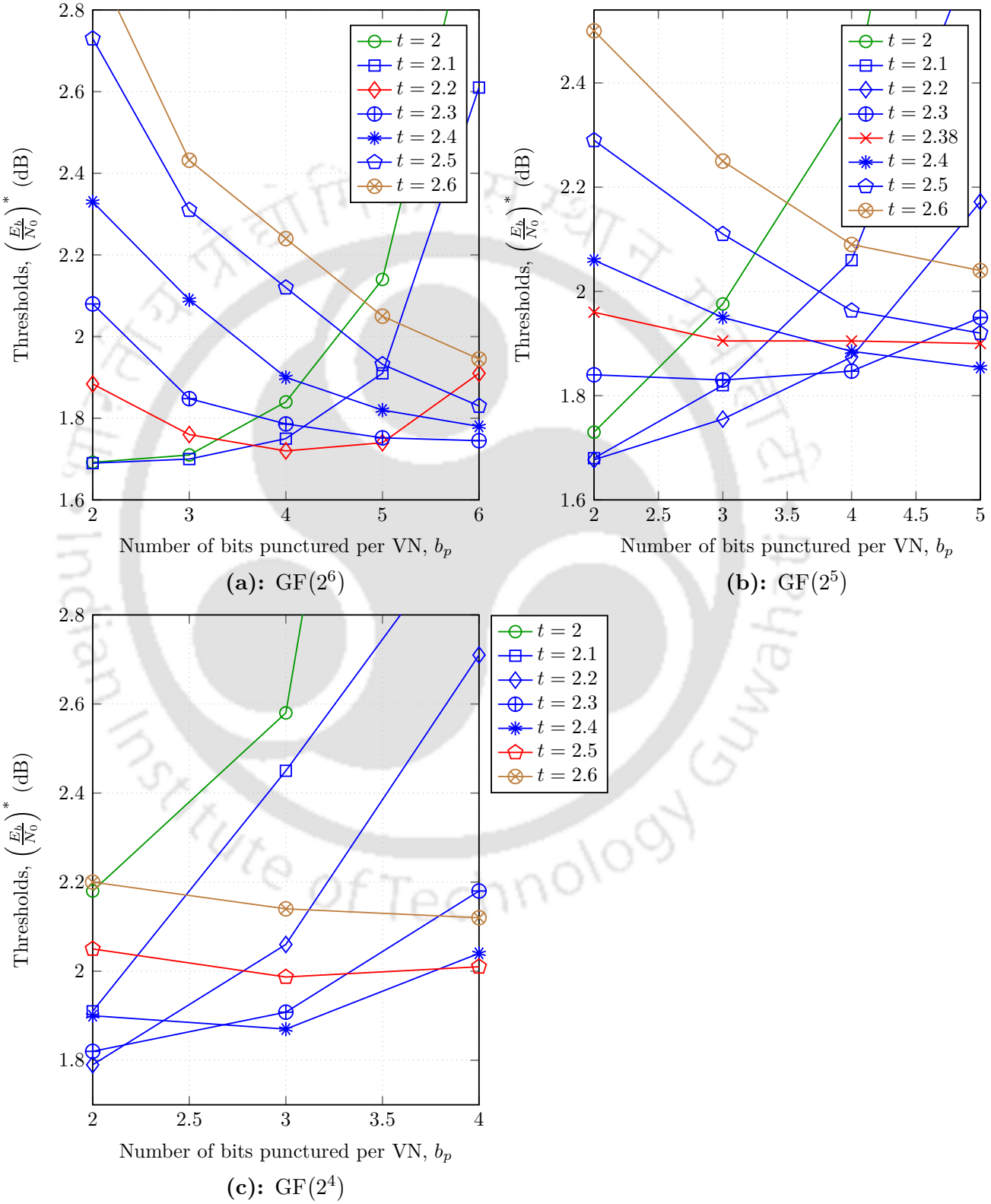
(a): GF(2^6) **(b):** GF(2^5)



(c): GF(2^4)

Figure 5.6: Thresholds for punctured quasi-regular codes with $r_0 = 0.5$, $N = 142$ at rate $r_l = 0.6$

t_c for the quasi-regular codes over GF(2^5) and GF(2^4) are 2.38 and 2.5 respectively.

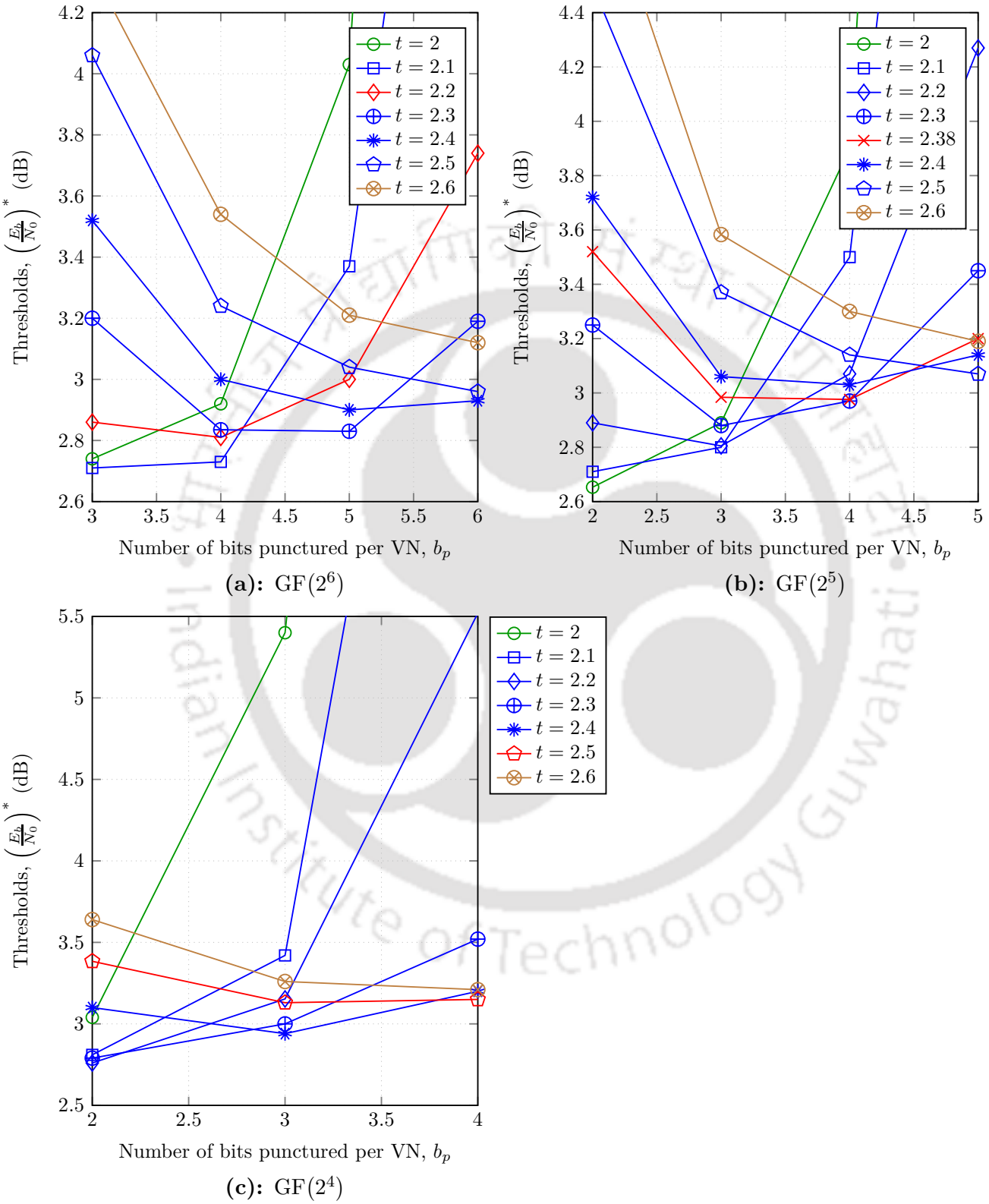


(a): GF(2^6)

(b): GF(2^5)

(c): GF(2^4)

Figure 5.7: Thresholds for punctured quasi-regular codes with $r_0 = 0.5$, $N = 142$ at rate $r_l = 0.7$



(a): GF(2^6) **(b):** GF(2^5)

(c): GF(2^4)

Figure 5.8: Thresholds for punctured quasi-regular codes with $r_0 = 0.5$, $N = 142$ at rate $r_l = 0.8$

Like any code ensemble having an optimum degree distribution, the quasi-regular codes also have

an optimum mean column weight (t_{opt}) for a given field. It is interesting to examine the relationship between t_{opt} and t_c. In [46], the authors have obtained the values of t_{opt} for quasi-regular NB-LDPC codes over a BI-AWGN channel at a rate of 0.5. For the codes over GF(2^4), GF(2^5) and GF(2^6), the values of t_{opt} are 2.3, 2.2 and 2.1 respectively. The values of t_c for the punctured codes at a rate $r_l = 0.6$ over GF(2^4), GF(2^5) and GF(2^6) are approximately 2.5, 2.38 and 2.2 respectively as found out from Figure 5.6. It can be seen that t_c is higher than t_{opt}. The gap between t_c and t_{opt} decreases with the increasing field order.

The thresholds for the quasi-regular codes at rates $r_l = 0.7$ and $r_l = 0.8$ are shown in Figure 5.7 and Figure 5.8 respectively. Observe that at these higher rates also, the mean column weight determines the performance of a puncturing pattern. If the mean column weight is very small, then the VNs should be punctured with the minimum b_p satisfying (5.27). On the other hand, if the mean column weight is on the higher side, then the VNs should undergo symbolwise puncturing. However, the threshold curves take a peculiar shape for the values of t around t_c. For the extreme puncturing patterns (i.e., with the minimum and the maximum possible values of b_p), the thresholds become high. The threshold curves experience a trough at an intermediate value of b_p. The puncturing

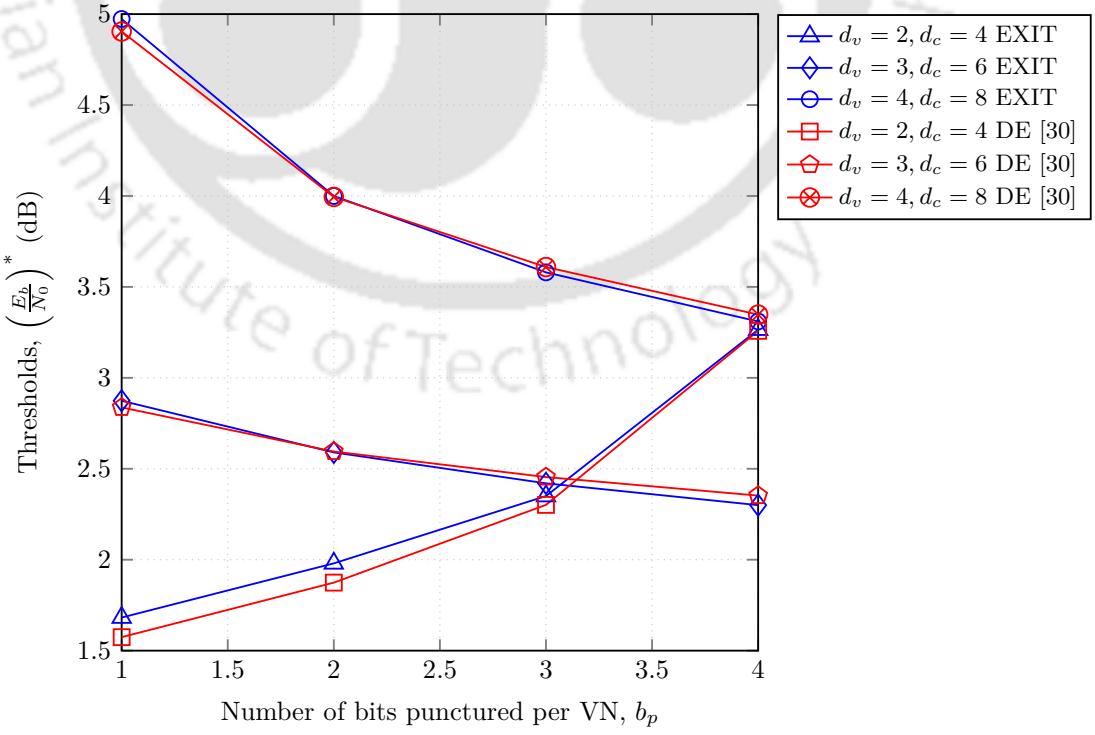


Figure 5.9: Comparison of thresholds for regular codes over GF (2^4) at $r_l = 2/3$

pattern should be judiciously determined in order to achieve the minimum possible threshold.

Note that the findings provided by the proposed EXIT chart model fully agree with those reported in [30] where the thresholds were computed via the DE. We compare these thresholds for (d_v, d_c) - regular codes over GF(2^4) for rate $r_l = 2/3$ in Figure 5.9.

Verification through Simulations

We verify the threshold-based analysis by performing simulations. We consider the threshold curves presented in Figure 5.6(a) over GF(2^6) at $r_l = 0.6$. The motive is to validate the existence of the cross-over points for the quasi-regular codes. We consider $N = 142$, $M = 71$ parity-check matrices over GF(2^6). A parity-check matrix with a particular t is generated by first forming a binary matrix according to the PEG algorithm [34] and then replacing the 1s by the randomly chosen elements from GF$(2^6)^-$. The equivalent binary streams for the codeword symbols are BPSK modulated and transmitted over a BI-AWGN channel. For decoding, we consider the FFT-QSPA with the maximum number of iterations set at 50.

For the puncturing patterns, we randomly select the required number of VNs for a given b_p. First, we show the symbol-error rate (SER) performances for a code with $t = 2$ in Figure 5.10(a). The SER performance gets worsened with increasing b_p. Next, we show the SER results for $t = 2.6$ in Figure 5.10(b). In this case, the SER performance gets improved with increasing b_p. Finally, we show the SER performances for a code with $t = 2.2$ in Figure 5.10(c). For this code, the performances of all the puncturing patterns are almost similar. These results are in full agreement with the conclusions derived from the threshold plot in Figure 5.6(a). We also performed the simulations for the quasi-regular codes over GF(2^5) and GF(2^4). The simulation results in these cases are also found to correlate strongly with the threshold analysis.

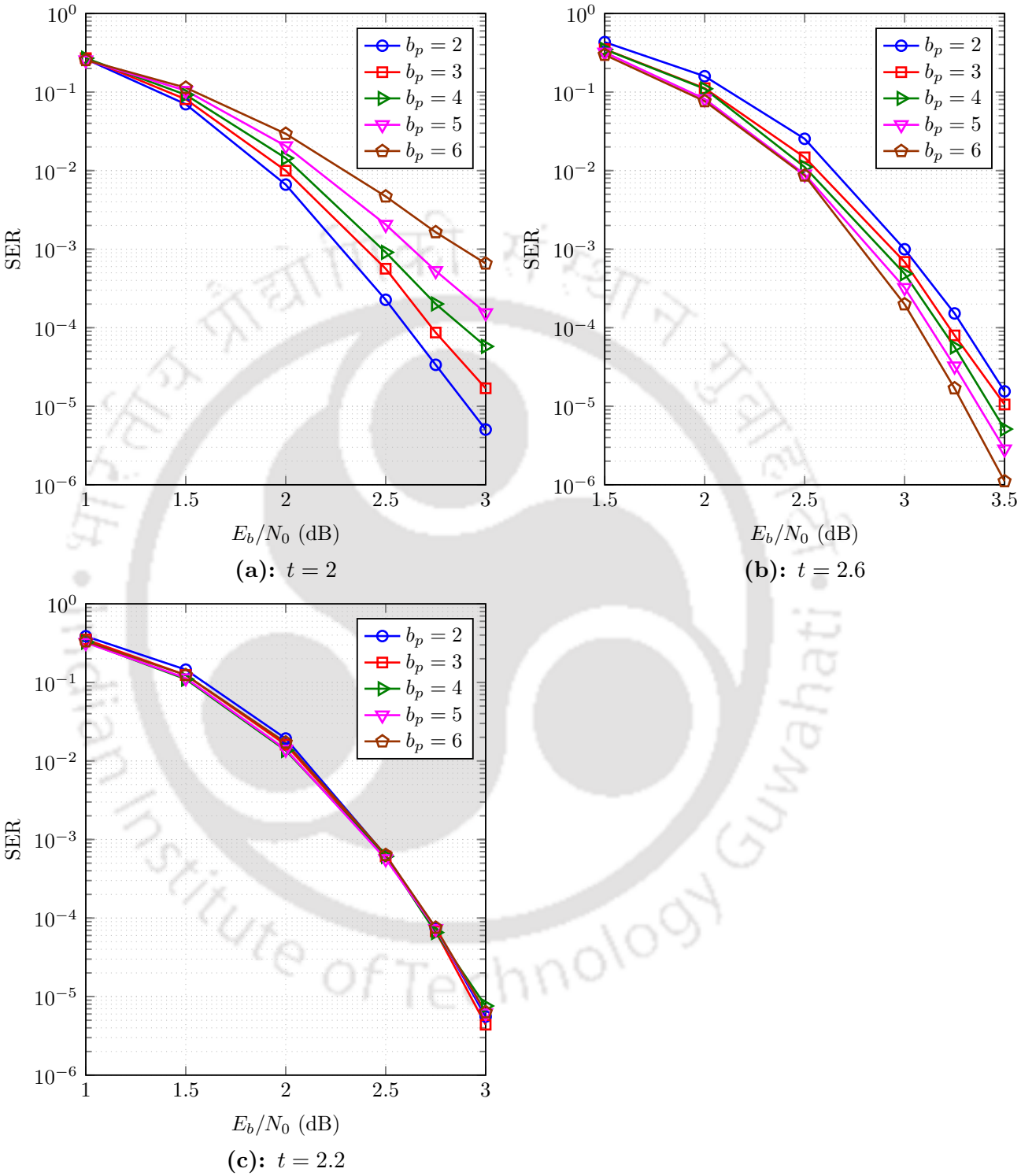


Figure 5.10: Simulation results of regular puncturing with random selection of the punctured VNs for quasi-regular codes over GF(2^6) having different mean column weights (t) with $r_0 = 0.5$ and $r_l = 0.6$

5.3.2 Selection of the VNs According to the Grouping Algorithm and Irregular Puncturing

In the above analysis, the same number of bits (b_p) are punctured per symbol for all the VNs. Now we formulate the EXIT chart model in the context of using different b_p for the VNs of different degrees in the case of irregular codes. We also take into account the grouping algorithm which imposes upper limits on the number of puncturable VNs of a particular degree.

Consider the set of recoverable VNs obtained from the grouping algorithm as given by $\mathbf{G}_R = \bigcup_{i=1}^{K} \mathbf{G}_i$, where $\mathbf{G}_i$ is the set of all i-SR VNs and K is the maximum number of iterations required for the recovery of any recoverable VN. Suppose the number of VNs of degree d_i in $\mathbf{G}_R$ is given by $N_{d_i}^g, i = 1, 2, \ldots, z$, where z is the total number of different degrees of the VNs in $\mathbf{G}_R$. Let N_{d_i} denote the total number of degree-d_i VNs and N_{d_i,b_p} denote the total number of degree-d_i VNs having b_p punctured bits. For a target rate r_l, the optimum values of N_{d_i,b_p}s can be obtained by solving the following optimization problem:

$$\left.\begin{aligned} \underset{\substack{N_{d_i,b_p} \\ i=1,\cdots,z,\, b_p=1,\cdots,s}}{\text{minimize}} \quad & \left(\frac{E_b}{N_0}\right)^* \\ \text{subject to} \quad & \sum_{i=1}^{z}\sum_{b_p=1}^{s} b_p N_{d_i,b_p} = \left\lfloor \frac{N \times s \times (r_l - r_0)}{r_l} \right\rfloor \\ \text{and} \quad & N_{d_i,b_p} \leqslant N_{d_i}^g,\ \forall i, \forall b_p \end{aligned}\right\}. \tag{5.33}$$

The aim of (5.33) is to minimize the threshold $\left(\frac{E_b}{N_0}\right)^*$ over the variables $N_{d_i,b_p}, i = 1, \ldots, z$ and $b_p = 1, \ldots, s$ satisfying two constraints. The *first* constraint requires that the total number of bits selected for puncturing must be equal to the prescribed number of bits needed to be punctured to reach r_l from r_0. The *second* constraint addresses the restrictions imposed by the grouping algorithm: one must select the bits only of the recoverable VNs. The number N_{d_i,b_p} of the VNs to be punctured of a particular degree d_i with b_p number of punctured bits must not exceed the maximum number $N_{d_i}^g$ of recoverable VNs of degree d_i.

The optimization is carried out as follows. First all possible combinations of N_{d_i,b_p} satisfying the constraints in (5.33) are found out exhaustively. The thresholds for these combinations are computed using the EXIT chart model presented below. The combination yielding the minimum threshold is considered as the optimum recoverable puncturing pattern. Then we select the punctured VNs from

$\mathbf{G}_R$ according to this optimum pattern.

The computation of the EXIT curves are presented below.

(a) **Computation of the VND Curve**

The probability of a degree-d_i VN containing b_p punctured bits is given by

$$p_{v,d_i}^{b_p} = \frac{N_{d_i,b_p}}{N_{d_i}}. \tag{5.34}$$

The mutual information for a $\mathbf{V}$ message emanating from a degree-d_i VN can be expressed as

$$J_{V,d_i}(\sigma) = \sum_{b_p=1}^{s} p_{v,d_i}^{b_p} J_V^{b_p}(\sigma) + \left(1 - \sum_{b_p=1}^{s} p_{v,d_i}^{b_p}\right) J_V^0(\sigma). \tag{5.35}$$

Note that, if $d_i \notin \{d_1, d_2, \ldots, d_z\}$, we have $N_{d_i,b_p} = 0$. Consequently from (5.34) and (5.35) we get

$$J_{V,d_i}(\sigma) = J_V^0(\sigma).$$

The effective mutual information is now given by

$$J_V(\sigma) = \sum_{d_i=1}^{d_{v\max}} \lambda_{d_i} J_{V,d_i}(\sigma). \tag{5.36}$$

After obtaining $J_V(\sigma)$, the same procedure is followed to get the VND curve as described in Section 5.2.4.4.

(b) **Computation of the CND Curve**

For the CND curve, we briefly revisit the situation described previously. The CN c of degree j is connected to $\{v_1, v_2, \cdots, v_j\}$. We consider the message flowing along the edge $c \to v_1$ in order to find the CND EXIT function. Out of the $(j-1)$ neighboring VNs in $\mathcal{N} = \{v_2, v_3, \cdots, v_j\}$, suppose τ VNs are punctured. These τ VNs may be punctured with different numbers of punctured bits per symbol.

Let Γ_{b_p}, $b_p \in \{1, \ldots, s\}$ be the random variable representing the number of VNs in $\mathcal{N}$ with b_p punctured bits per symbol. Clearly $\sum_{b_p=1}^{s} \Gamma_{b_p} = \tau$, where τ is the total number of punctured VNs

in $\mathcal{N}$. For a given τ, $\Gamma_1, \Gamma_2, \ldots, \Gamma_s$ are characterized by the joint probability mass function

$$p_{\Gamma_1,\ldots,\Gamma_s|\tau}(\gamma_1,\ldots,\gamma_s) = \frac{\tau!}{\prod_{b_p=1}^{s}\gamma_{b_p}!}\prod_{b_p=1}^{s}\left(p_v^{b_p}\right)^{\gamma_{b_p}}, \tag{5.37}$$

where $p_v^{b_p}$ is the probability that b_p bits of a VN are punctured given that the VN is selected for puncturing and given by

$$p_v^{b_p} = \frac{\sum_{i=1}^{z} N_{d_i,b_p}}{\sum_{i=1}^{z}\sum_{b'_p=1}^{s} N_{d_i,b'_p}}.$$

Recall from (5.31) that the probability that the set $\mathcal{N}$ contains τ number of punctured VNs is given by

$$p_{\mathcal{N}}(\tau) = \begin{pmatrix} j-1 \\ \tau \end{pmatrix}(p_v)^{\tau}(1-p_v)^{j-\tau-1}, \tag{5.38}$$

where p_v is the probability that a VN is punctured. In the case of irregular puncturing p_v can be obtained as

$$p_v = \sum_{i=1}^{z}\hat{\lambda}_{d_i}\frac{\left(\sum_{b=1}^{s} N_{d_i,b_p}\right)}{N_{d_i}},$$

where $\hat{\lambda}_{d_i}$ is the fraction of degree-d_i VNs.

Suppose $I_{E,\mathrm{CND}}(I_{A,\mathrm{CND}}, j, \tau, \gamma_1, \ldots, \gamma_s)$ is the EXIT function for a degree-j CN when the set $\mathcal{N}$ contains τ punctured VNs arranged as $(\Gamma_1 = \gamma_1, \Gamma_2 = \gamma_2, \ldots, \Gamma_s = \gamma_s)$. Let $I_{E,\mathrm{CND}}(I_{A,\mathrm{CND}}, j, \tau)$ be the EXIT function for a degree-j CN when the set $\mathcal{N}$ contains τ punctured VNs with any arrangement. $I_{E,\mathrm{CND}}(I_{A,\mathrm{CND}}, j, \tau)$ can be expressed as

$$I_{E,\mathrm{CND}}(I_{A,\mathrm{CND}}, j, \tau) = \sum_{\gamma_1,\ldots,\gamma_s} p_{\Gamma_1,\ldots,\Gamma_s|\tau}(\gamma_1,\ldots,\gamma_s)\, I_{E,\mathrm{CND}}(I_{A,\mathrm{CND}}, j, \tau, \gamma_1, \ldots, \gamma_s). \tag{5.39}$$

Now the effective EXIT function for a degree-j CN is given by

$$I_{E,\mathrm{CND}}(I_{A,\mathrm{CND}}, j) = \sum_{\tau=0}^{j-1} p_{\mathcal{N}}(\tau) I_{E,\mathrm{CND}}(I_{A,\mathrm{CND}}, j, \tau). \tag{5.40}$$

From (5.40) and (5.11), the CND curve can be obtained.

5.4 Simulation Results

We consider a code over GF(2^4) having the optimum degree distributions [30]:

$$\lambda(x) = 0.5376x + 0.1678x^2 + 0.1360x^4 + 0.1586x^9$$
$$\text{and} \quad \rho(x) = 0.5169x^4 + 0.4831x^5.$$

We construct a 71×142 parity-check matrix by first forming a binary matrix with the help of the PEG algorithm [34] and then replacing the 1s by the randomly chosen elements from GF$(2^4)^-$. The girth of the binary matrix is 8. The numbers of the VNs of degree 2, 3, 5 and 10 are 104, 22, 11 and 5, respectively. We consider this code as the mother code and puncture a certain number of carefully selected bits to achieve rate-adaptability. An all-zero codeword is considered for the performance evaluation. The codeword bitstream is BPSK modulated and transmitted over a BI-AWGN channel. Decoding is done by the FFT-QSPA with the maximum number of iterations set at 50. The SER for the mother code is shown in Figure 5.11.

We apply the grouping algorithm to partition the set of the VNs according to their recoverability levels. Table 5.1 shows the numbers of the recoverable VNs in different groups along with their degrees

Table 5.1: Numbers of recoverable VNs in different groups for the mother code over GF (2^4) with $N = 142, M = 71$ and the degree distributions as in [30, 31]

Degree (d_i)	$\mathbf{G}_1$	$\mathbf{G}_2$	$\mathbf{G}_3$	Total ($N_{d_i}^g$)
2	45	4	1	50
3	3	1	0	4
5	0	0	0	0
10	0	0	0	0
Total	48	5	1	54

as obtained from the grouping algorithm. The grouping algorithm did not select any VN of degree 5 and 10. The total number of recoverable VNs is 54. Thus the maximum rate attainable by the grouping algorithm is $\frac{71}{142-54} = 0.8068$.

By computing the thresholds by the proposed EXIT chart model, the optimum recoverable puncturing patterns are found out. Table 5.2 shows these patterns along with their thresholds. The thresholds of the optimum puncturing distributions without any constraints as reported in [30, 31] are also shown. It can be seen that for lower rates, the thresholds for the optimum recoverable patterns are close to the thresholds for the optimum puncturing distributions without any constraints. However,

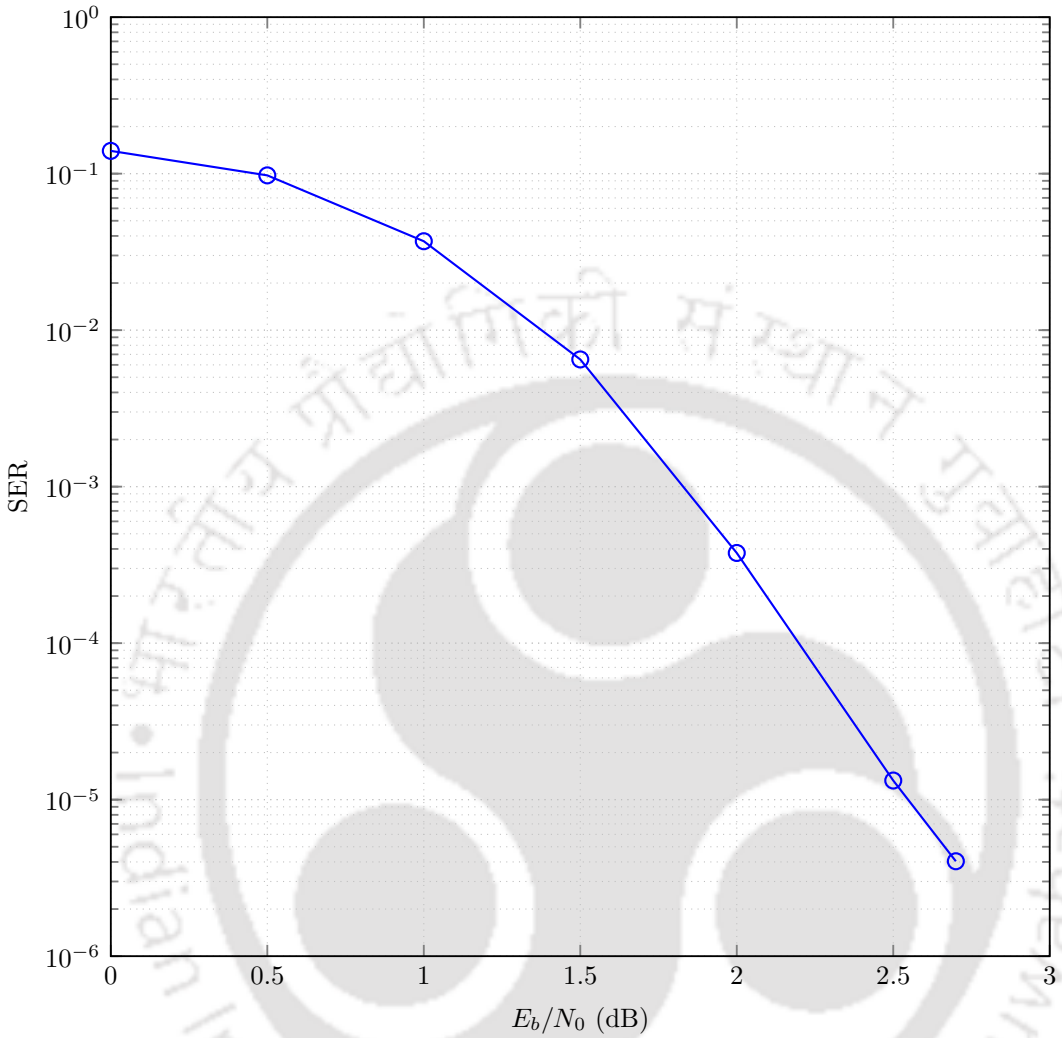


Figure 5.11: SER of the mother code over GF (2^4) with $N = 142, M = 71$ and the degree distributions as in [30, 31]

at higher rates, because of the small number of the puncturable VNs, the thresholds for the optimum recoverable patterns become high.

We consider four puncturing schemes for performance studies:

(1) $S1$ (proposed scheme) – the optimized recoverable pattern in Table 5.2 with the selection of the punctured VNs from $\mathbf{G}_R$.

(2) $S2$ – the optimized recoverable pattern in Table 5.2 with random selection of the punctured VNs.

(3) $S3$ – the optimized distribution without any constraints [30, 31] with the selection of as many punctured VNs as possible from $\mathbf{G}_R$. If the distribution requires more number of punctured VNs

Table 5.2: Optimized recoverable puncturing patterns in the context of the grouping algorithm and the corresponding thresholds

Rate r_l	Thresholds [30, 31] $\left(\frac{E_b}{N_0}\right)^*$, (dB)	Optimized patterns					
		d_i	$N_{d_i,1}$	$N_{d_i,2}$	$N_{d_i,3}$	$N_{d_i,4}$	Thresholds $\left(\frac{E_b}{N_0}\right)^*$, (dB)
0.6	0.9	2	17	33	0	0	0.9
		3	0	0	4	0	
0.65	1.22	2	0	31	19	0	1.22
		3	0	0	4	0	
0.7	1.46	2	0	4	46	0	1.64
		3	0	0	0	4	
0.75	1.97	2	0	0	27	23	2.17
		3	0	0	0	4	
0.8	2.37	2	0	0	3	47	3.29
		3	0	0	0	4	

of a particular degree than $\mathbf{G}_R$ can provide, then we select the remaining VNs of the same degree randomly from $\mathbf{G}_0$.

(4) $S4$ – the optimized distribution without any constraints with random selection of the punctured VNs [30, 31].

Table 5.3: Pattern for the scheme $S3$ at $r_l = 0.6$

d_i	$b_p = 1$			$b_p = 2$			$b_p = 3$			$b_p = 4$		
	$\mathbf{G}_R$	$\mathbf{G}_0$	$N_{d_i,b}$	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}
2	13	0	13	12	0	12	1	0	1	4	0	4
3	0	3	3	0	0	0	0	2	2	4	0	4
5	0	0	0	0	2	2	0	1	1	0	0	0
10	0	1	1	0	1	1	0	1	1	0	1	1

Table 5.4: Pattern for the scheme $S3$ at $r_l = 0.7$

d_i	$b_p = 1$			$b_p = 2$			$b_p = 3$			$b_p = 4$		
	$\mathbf{G}_R$	$\mathbf{G}_0$	$N_{d_i,b}$	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}
2	10	15	25	27	0	27	10	0	10	3	0	3
3	0	0	0	3	4	7	0	0	0	1	0	1
5	0	3	3	0	0	0	0	0	0	0	0	0
10	0	0	0	0	2	2	0	1	1	0	1	1

The puncturing in all the four schemes are carried out in a rate-compatible manner. Table 5.3, Table 5.4 and Table 5.5 show the patterns of the punctured VNs for the scheme $S3$ at $r_l = 0.6, r_l = 0.7$ and $r_l = 0.8$, respectively. While fixing the patterns, we give preference to the VNs having a higher

Table 5.5: Pattern for the scheme $S3$ at $r_l = 0.8$

d_i	$b_p = 1$			$b_p = 2$			$b_p = 3$			$b_p = 4$		
	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}	$\mathbf{G}_R$	$\mathbf{G}_0$	N_{d_i,b_p}
2	7	33	40	11	0	11	29	0	29	3	0	3
3	0	8	8	0	2	2	2	0	2	2	0	2
5	0	0	0	0	0	0	0	4	4	0	0	0
10	0	0	0	0	2	2	0	0	0	0	2	2

value of b_p. We start distributing the recoverable VNs starting from $b_p = 4$, then $b_p = 3$ and so on. It will ensure the recoverability of the heavily punctured VNs.

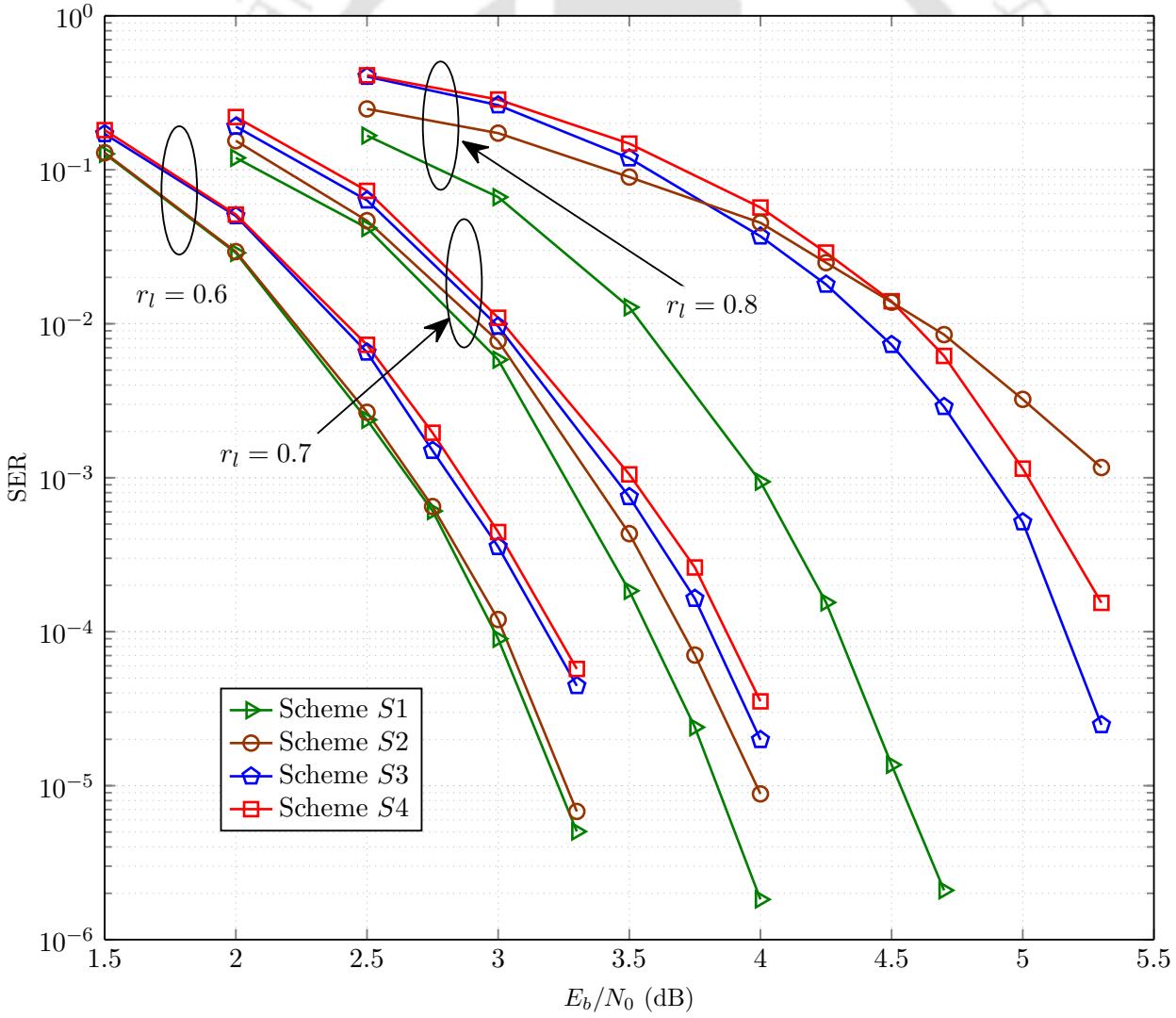


Figure 5.12: Performance comparisons for the schemes $S1$, $S2$, $S3$ and $S4$ at $r_l \in \{0.6, 0.7, 0.8\}$

Figure 5.12 shows the SER performances of all the schemes. It can be observed that $S1$ is better than $S2$ and $S3$ is better than $S4$ at all rates. This means if we select the VNs from $\mathbf{G}_R$ rather than selecting randomly, the performance is improved for both the optimized recoverable patterns in Table 5.2 and the distributions in [30, 31]. It can be further observed that at $r_l = 0.6$ and $r_l = 0.7$, both the schemes $S1$ and $S2$ perform better than the schemes $S3$ and $S4$. This observation can be explained as follows. The schemes $S1$ and $S2$ include only the degree-2 and degree-3 VNs whereas the schemes $S3$ and $S4$ include some degree-5 and degree-10 VNs in addition to degree-2 and degree-3 VNs. Puncturing a higher-degree VN increases the number of dead CNs [32]. Thus each of the schemes $S3$ and $S4$ results in a higher number of dead CNs compared to the schemes $S1$ and $S2$. However, at $r_l = 0.8$, the scheme $S2$ gives the worst result. This observation can be explained from the threshold results shown in Table 5.2. It can be observed that the threshold for the optimized recoverable pattern increases significantly as the rate increases. At $r_l = 0.8$, the threshold for the optimized recoverable pattern is 3.29 and that for the optimized distribution in [30, 31] is 2.37. Because of the significant difference in the thresholds, the performance of the scheme $S2$ is poorer than those of the schemes $S3$ and $S4$. Although the scheme $S1$ has the same threshold (3.29) as that for $S2$, it yields the best SER performance as all the punctured VNs are from $\mathbf{G}_R$.

5.5 Conclusions

For NB-LDPC codes defined over binary extension fields, the codeword symbols are transmitted in bitstreams through a binary input channel. Thus a puncturing scheme actually selects bits to be punctured. This fact offers one extra degree of freedom in designing the puncturing patterns which is the number of bits punctured per symbol b_p. In this chapter, we proposed an EXIT chart model to study the effects of b_p on various code ensembles by computing their thresholds. With the help of the EXIT chart model, we investigated the role of the degrees of the VNs in the puncturing performance. The grouping algorithm is an effective tool to find recoverable puncturing patterns specifically for short-length codes. We presented a method to obtain optimized puncturing patterns compatible with the grouping algorithm. Simulation results of various puncturing patterns are included. These result show that the optimum recoverable puncturing patterns obtained by the proposed EXIT chart analysis yield better performances at various rates.

6

A Cycle-based Rate-compatible Puncturing Technique for Non-binary LDPC Codes

Contents

6.1 Introduction

In the previous chapter, we have investigated various bitwise and symbolwise puncturing patterns for the NB-LDPC codes from an asymptotic performance point of view. An EXIT chart model was developed to compute the thresholds for different kinds of puncturing patterns. It was observed that the performances of bitwise and symbolwise puncturing schemes are essentially determined by the mean column weight of the code ensemble. With the help of the EXIT chart, an optimum puncturing pattern can be obtained. Considering the constraints imposed by the grouping algorithm, we obtain an optimum puncturing pattern which can be realized by puncturing only the recoverable VNs. In this chapter we investigate the performance of the puncturing schemes for the NB-LDPC codes from a finite-length perspective. Along with the earlier findings, the structure of the Tanner graph of the mother code is further examined to devise an effective RC puncturing technique. In the following, various popular and relevant RC puncturing techniques available in the literature for the binary and the NB-LDPC codes are reviewed.

RC puncturing schemes have been studied extensively for the finite-length binary LDPC codes [6, 32, 54, 71]. As mentioned in the previous chapter, the authors in [32], proposed the grouping and the sorting algorithms to select the puncturing bits. The grouping algorithm partitions the set of the VNs into various groups according to the level of recoverability. The order for puncturing within each group is determined with the help of the sorting algorithm. In [71], a scheme was proposed with the objective of keeping the punctured bits far apart from each other in the Tanner graph. In [54], the order of the puncturing bits was fixed by evaluating a cost function. A *stopping set check algorithm* was also proposed to ensure that the punctured bits do not form stopping sets. In [6], the ACE values of the short cycles have been considered for selecting the punctured bits.

For the NB-LDPC codes, the investigation of RC puncturing schemes was initiated by Klinc *et al.* [43]. They have proposed a bitwise puncturing scheme for the random NB-LDPC codes. The scheme selects the recoverable VNs using the grouping algorithm. The puncturing is performed over these recoverable VNs by taking two points into consideration: (1) the number of bits punctured per VN should be decided based on the recoverability of the VN. A higher number of bits are punctured per symbol for the VNs of a lower recovery level (which are expected to recover in a small number of iterations) compared to the VNs of a higher recovery level. (2) the bits to be punctured are uniformly spread over the VNs of the same recovery level. In [36], a structured protograph-based RC

NB-LDPC code has been designed. A protograph is a small bipartite graph which can be lifted to obtain the Tanner graph by the *copy-and-permute* operation [61, 69]. The puncturing scheme of [36] is symbolwise and structured as the puncturing pattern is fixed at the protograph level itself. The authors have considered a code over GF(2^6) with the mean column weight of 2.8. The performance of the structured and symbolwise puncturing for this code has been found to be comparable to that of the scheme proposed by Klinc *et al.* [43]. This puncturing is done in a structured way which makes the coding scheme desirable for high speed implementation.

The works described in this chapter can be divided into two parts. The *first* part investigates the effects of short cycles in the iterative decoding of the NB-LDPC codes. In the case of NB-LDPC codes, not all the short cycles present in the Tanner graph are harmful for iterative decoding. The cycles which satisfy a particular condition [4, 56] based on the non-binary edge-labels, can create problems to the iterative decoding. These cycles are called *non-binary (NB) cycles*. We analyze the probability of a cycle resulting in an NB cycle for a given field size. Based on this analysis, the advantages of the NB-LDPC codes over the binary LDPC codes with respect to the error floor are substantiated. The *second* part proposes an RC puncturing technique for the NB-LDPC codes. We devise a criterion to select the VNs to be punctured. This criterion is based on the lengths and the connectivity of the NB cycles present in the Tanner graph of the NB-LDPC code. An NB cycle of a smaller length is more detrimental. As in the context of the binary LDPC codes, the connectivity of an NB cycle can also be quantified by using the EMD [70]. The lower the EMD, the more is the harmfulness of the NB cycle. The proposed criterion is to select those VNs which are involved in a lower number of short NB cycles with low EMD values. We apply this criterion in place of the sorting algorithm [32].

The rest of the chapter is organized as follows. In Section 6.2, we discuss the notion of the cycles in NB-LDPC codes. Section 6.3 presents the proposed technique of RC puncturing for the NB-LDPC codes. Simulation results are presented in Section 6.4 and Section 6.5 concludes this chapter.

6.2 Cycles in NB-LDPC Codes

The effect of the cycles in the NB-LDPC codes is not the same as that in the binary ones. Unlike in the binary case, an edge in the Tanner graph of an NB-LDPC code includes a label of a nonzero member from a finite field. The concept of cycles in NB-LDPC codes can be explained through the idea of mis-satisfied CNs [4, 5]. A mis-satisfied CN corresponds to a syndrome equation which is satisfied

in spite of having some erroneous variables. We assume that an all-zero codeword is transmitted. Suppose we have a binary cycle in the Tanner graph. Then, a CN in the cycle is mis-satisfied if it is connected to two erroneously decoded VNs and the contribution of these VNs add up to zero in the parity-check sum. If all the CNs in the cycle path are mis-satisfied then that cycle is harmful for the iterative decoding of the NB-LDPC codes.

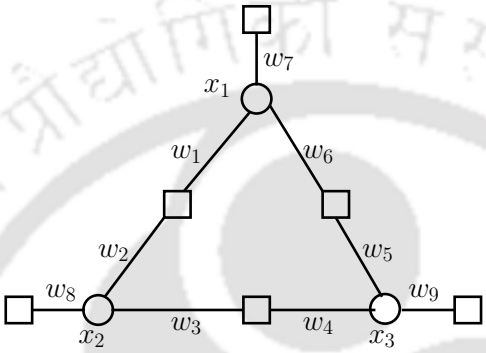


Figure 6.1: A cycle of length 6 along with the edge labels

The concept of a cycle in NB-LDPC codes defined over $\mathrm{GF}(2^s), s \in \mathbb{Z}^+ \setminus \{1\}$ can be explained with the help of Figure 6.1 which shows a cycle of length 6 along with the edge labels $w_1, \ldots, w_9$. There are three degree-2 CNs in the subgraph. Let us assume that the three VNs are in error and takes values x_1, x_2, and x_3 ($x_1 \neq 0, x_2 \neq 0,$ and $x_3 \neq 0$) as shown in the figure. In that case, the three degree-2 CNs will become mis-satisfied if the following conditions are satisfied over $\mathrm{GF}(2^s)$ [4, 5].

$$x_1 w_1 = x_2 w_2 \tag{6.1}$$

$$x_2 w_3 = x_3 w_4 \tag{6.2}$$

$$x_3 w_5 = x_1 w_6 \tag{6.3}$$

Multiplying (6.1), (6.2), and (6.3) gives the following condition:

$$w_1 w_3 w_5 = w_2 w_4 w_6. \tag{6.4}$$

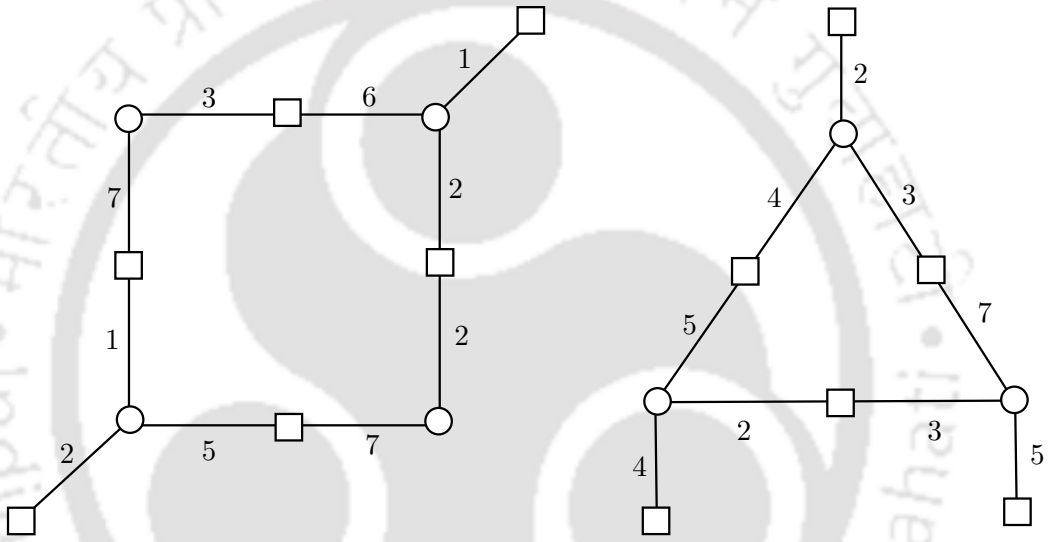
The above observation can be generalized as follows [4, 5, 56]:

Lemma 6.1. *Consider a cycle C_{2l} of length $2l$ in the binary sense. Let the labels of the edges be $(w_1, w_2, \ldots, w_{2l})$, $w_i \in \mathrm{GF}(2^s)^-, s \in \mathbb{Z}^+ \setminus \{1\}$. Then, C_{2l} is a legitimate cycle in the non-binary sense*

if and only if

$$\prod_{i=1}^{l} w_{2i-1} = \prod_{i=1}^{l} w_{2i} \quad \text{over GF}\,(2^s)\,. \tag{6.5}$$

The cycles satisfying the condition in (6.5) are called *NB cycles*. Observe from (6.5) that the condition for NB cycles depends only on the labels for the edges in the cycle. The values taken by the VNs in the cycle are irrelevant. It can be observed that all NB cycles are valid binary cycles and the converse is not always true.



(a) A legitimate NB cycle **(b)** Not an NB cycle

Figure 6.2: Examples for NB cycles

We consider the following examples to explain the idea of an NB cycle.

Example 6.1. *Consider an NB-LDPC code defined over* $\text{GF}(2^3)$ *with the primitive polynomial* $p\,(X) = X^3+$ *Suppose the Tanner graph of the code contains a cycle of length 8 as shown in Figure 6.2(a). The edge-labels for the edges of the cycle are also shown. For this cycle, we have*

$$3 \times 1 \times 7 \times 2 = 4 = 6 \times 7 \times 5 \times 2.$$

This cycle is considered to be harmful and therefore, a legitimate NB cycle.

Now consider the cycle of length 6 shown in Figure 6.2(b). For this cycle we can find that

$$4 \times 2 \times 7 = 2 \neq 7 = 3 \times 5 \times 3.$$

This cycle is not expected to be harmful for the iterative message-passing decoder and therefore, not

considered as an NB cycle.

A cycle can be harmful for the iterative decoding of the NB-LDPC codes and consequently an NB cycle only if it satisfies the condition in (6.5). The probability of a cycle becoming an NB cycle is an important measure as it can provide an assessment about the performance of the code. Amiri *et al.* [5] have determined the fraction of binary ASs which will result in the non-binary ASs, i.e., the ASs which are harmful in the context of the non-binary iterative decoding. Instead of the ASs, we consider the cycles in assessing the harmfulness to iterative NB decoding. We derive a theorem regarding the probability of a cycle becoming a legitimate NB cycle in the following.

Theorem 6.1. *Let C_{2l} be a cycle with the edge labels $(w_1, w_2, \ldots, w_{2l})$, $w_i \in \mathrm{GF}\,(q = 2^s)^-$, $s \in \mathbb{Z}^+\backslash\{1\}$. Suppose, the non-binary labels are selected uniformly at random from $\mathrm{GF}(q)^-$ during the construction of the code. Then, the probability $P^q_{\mathrm{nb},l}$ of C_{2l} being a legitimate NB cycle is given by*

$$P^q_{\mathrm{nb},l} = \frac{1}{q-1}. \tag{6.6}$$

Proof. We prove the theorem by the mathematical induction on l (half of the length of the cycle) with the field size being fixed at q. Note that the edge labels of an NB cycle satisfy the condition in Lemma 6.1.

First we prove the theorem for the case of 4-cycles, i.e., $l = 2$. Let the labels of the edges in C_4 be w_1, w_2, w_3 and w_4. The total number of different possible combinations for (w_1, w_2, w_3, w_4) is equal to $(q-1)^4$.

Next we find the total number of combinations of (w_1, w_2, w_3, w_4) such that $w_1 w_3 = w_2 w_4$ over $\mathrm{GF}(q)$. Consider $c \in \mathrm{GF}\,(q)^-$, such that $w_1 w_3 = w_2 w_4 = c$. Now the total number of different combinations of (w_1, w_3) such that $w_1 w_3 = c$ is $(q-1)$. Similarly for (w_2, w_4) also, the total number combinations given $w_2 w_4 = c$ is equal to $(q-1)$. Thus for each combination of (w_1, w_3), we have $(q-1)$ number of combinations of (w_2, w_4) such that $w_1 w_3 = w_2 w_4 = c$. Therefore, the total number of combinations of (w_1, w_2, w_3, w_4) such that $w_1 w_3 = w_2 w_4 = c$ is equal to $(q-1)^2$. Again c can take any of the $(q-1)$ symbols from $\mathrm{GF}\,(q)^-$. For all the choices of c, the total number of combinations of (w_1, w_2, w_3, w_4) such that $w_1 w_3 = w_2 w_4$ is equal to $(q-1)^3$. Thus, the probability that C_4 is a

legitimate NB cycle is given by

$$P^q_{\text{nb},2} = \frac{(q-1)^3}{(q-1)^4} \tag{6.7}$$

$$= \frac{1}{q-1}. \tag{6.8}$$

Hence the statement in the theorem is true for $l = 2$.

Now let us assume that the statement is true for $l = k$. That means we have

$$P^q_{\text{nb},k} = \frac{1}{q-1}.$$

The total number of combinations of $(w_1, w_2, \ldots, w_{2k})$ is $(q-1)^{2k}$. Therefore, under the induction hypothesis we can find that the total number of combinations of $(w_1, w_2, \ldots, w_{2k})$ such that $w_1 w_3 \cdots w_{2k-1} = w_2 w_4 \cdots w_{2k}$ is equal to $(q-1)^{2k-1}$. Given this condition, we try to prove the statement for $l = k+1$. We need to prove that the total number of combinations of $(w_1, w_2, \ldots, w_{2k}, w_{2k+1}, w_{2k+2})$ such that $w_1 w_3 \cdots w_{2k-1} w_{2k+1} = w_2 w_4 \cdots w_{2k} w_{2k+2}$ is equal to $(q-1)^{2k+1}$.

Let us consider the following equation.

$$w_1 w_3 \cdots w_{2k-1} w_{2k+1} = w_2 w_4 \cdots w_{2k} w_{2k+2} \tag{6.9}$$

Define w' and w'' as

$$\begin{aligned} w' &= w_1 w_3 \cdots w_{2k-1} \text{ and} \\ w'' &= w_2 w_4 \cdots w_{2k}. \end{aligned} \tag{6.10}$$

Substituting (6.10) in (6.9) we get

$$w' w_{2k+1} = w'' w_{2k+2}. \tag{6.11}$$

In (6.11), two cases may arise: (1) $w' = w''$ and (2) $w' \neq w''$. We consider these two cases separately in the following:

(1) $w' = w''$:

Observe from (6.11) that we have $w_{2k+1} = w_{2k+2}$ in this case. The number of ways w_{2k+1} can be equal to w_{2k+2} is $q-1$. Moreover, from the induction hypothesis, the number of ways w' can be equal to w'' is $(q-1)^{2k-1}$. Therefore, for this case, the total number of combinations for $(w_1, w_2, \ldots, w_{2k}, w_{2k+1}, w_{2k+2})$ such that $w_1 w_3 \cdots w_{2k-1} w_{2k+1} = w_2 w_4 \cdots w_{2k} w_{2k+2}$ is

$(q-1)^{2k-1}(q-1) = (q-1)^{2k}$.

(2) $w' \neq w''$:

The total number of combinations for $(w_1, w_2, \ldots, w_{2k})$ is $(q-1)^{2k}$. The number of combinations for $(w_1, w_2, \ldots, w_{2k})$ given $w' = w''$ is $(q-1)^{2k-1}$. Therefore, the number of combinations for $(w_1, w_2, \ldots, w_{2k})$ with $w' \neq w''$ is $(q-1)^{2k} - (q-1)^{2k-1}$.

In this case, $w_{2k+2} = (w'')^{-1} w' w_{2k+1}$. That means once w_{2k+1} is fixed, w_{2k+2} gets fixed automatically. w_{2k+1} can assume any of the $q-1$ values from GF$(q)^{-}$. Therefore, for this case, the total number of combinations for $(w_1, w_2, \ldots, w_{2k}, w_{2k+1}, w_{2k+2})$ such that $w_1 w_3 \cdots w_{2k-1} w_{2k+1} = w_2 w_4 \cdots w_{2k} w_{2k+2}$ is $\left((q-1)^{2k} - (q-1)^{2k-1}\right)(q-1) = (q-1)^{2k+1} - (q-1)^{2k}$.

Summing the number of combinations for both the cases, we find that the total number of combinations of $(w_1, w_2, \ldots, w_{2k+2})$ satisfying the condition $w_1 w_3 \cdots w_{2k-1} w_{2k+1} = w_2 w_4 \cdots w_{2k} w_{2k+2}$ is equal to $(q-1)^{2k+1}$.

The total number of combinations for $(w_1, w_2, \ldots, w_{2k}, w_{2k+1}, w_{2k+2})$ is $(q-1)^{2k+2}$. Therefore, the probability that $(w_1, w_2, \ldots, w_{2k}, w_{2k+1}, w_{2k+2})$ constitutes a legitimate NB cycle is given by

$$P^q_{\mathrm{nb},k+1} = \frac{(q-1)^{2k+1}}{(q-1)^{2k+2}} = \frac{1}{q-1}. \tag{6.12}$$

$$\therefore P^q_{\mathrm{nb},l} = \frac{1}{q-1}. \tag{6.13}$$

□

Observe from (6.13) that $P^q_{\mathrm{nb},l}$ is independent of the length $2l$. Therefore, P^q_{nb} is used instead of $P^q_{\mathrm{nb},l}$ in the remaining part of this chapter. We plot the probability values for different field sizes q in Figure 6.3. Observe that P^q_{nb} decreases as the field size q increases. Therefore, an NB-LDPC code over a higher order field is expected to contain a small number of NB cycles. In following, we present a corollary of Theorem 6.1 regarding the expected number of the NB cycles of a particular length.

Corollary 6.1. *Consider an NB-LDPC code defined over* GF(q). *Suppose the number of cycles of length* $2l$ *in a binary sense present in the Tanner graph of the code is* $N_{\mathrm{b},2l}$. *Then, on an average, the number* $N^q_{\mathrm{nb},2l}$ *of NB cycles of length* $2l$ *present in the Tanner graph is equal to*

$$N^q_{\mathrm{nb},2l} = N_{\mathrm{b},2l} \frac{1}{q-1}.$$

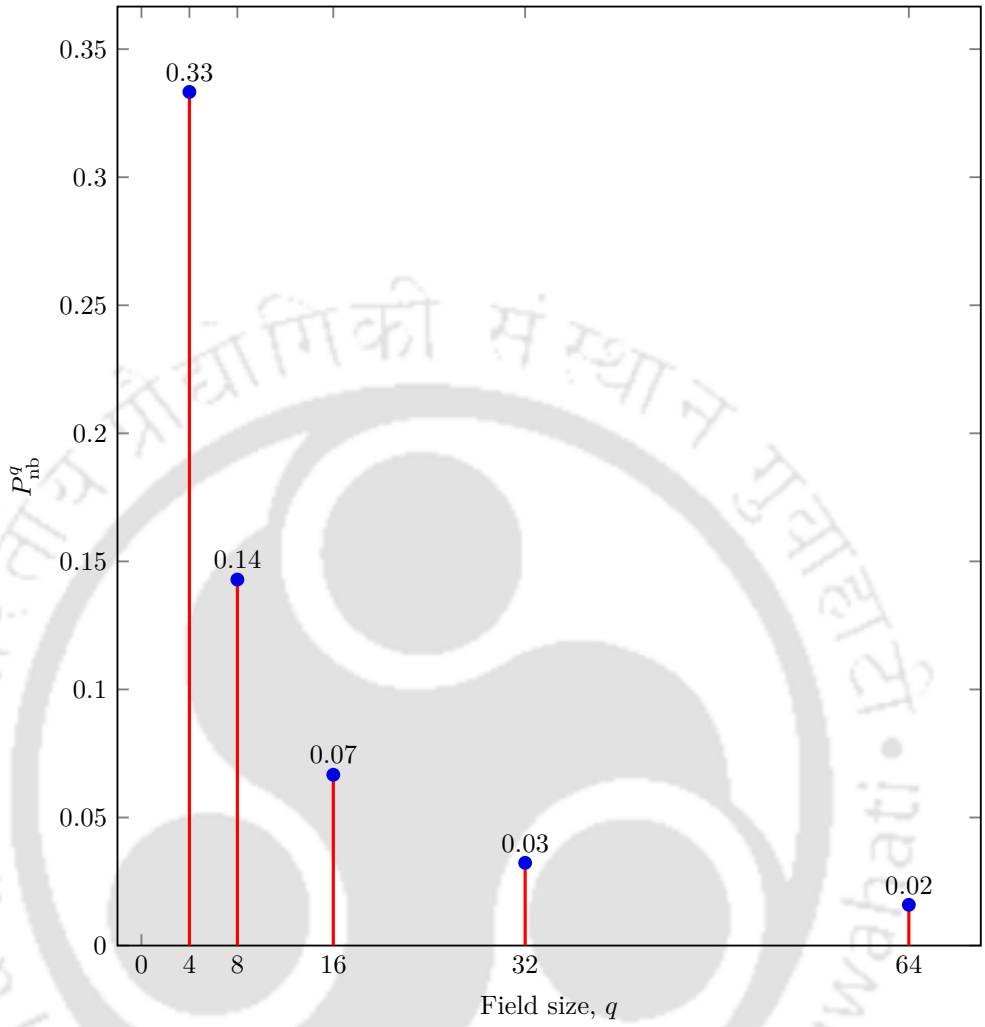


Figure 6.3: Probability of a cycle being an NB cycle for different field sizes

Observe from Corollary 6.1 that the average number of legitimate harmful cycles for a field of size q gets reduced by a factor $(q-1)$ of the binary case. We have demonstrated in Chapter 3 that a dominant TS contains one or multiple short cycles. As an NB-LDPC code contains a significantly small number of NB cycles, the number of TSs harmful to the non-binary iterative decoding is also small. This fact emphasizes that an NB-LDPC code can perform better than an equivalent binary LDPC code specifically in the short block-length regime. Consequently, the NB-LDPC codes are more robust to the error floor issue.

Remark 6.1. *Note that the above analysis is done under the hypothesis that the non-binary labels are selected uniformly at random from* $\mathrm{GF}(q)^-$. *The labels may be judiciously selected to minimize the chance of the formation of the NB cycles* [5, 56]. *However, the improvement in the performance*

diminishes as the field size q increases. Usually, for the moderate to high values of q ($q \geq 16$)*, the selection of the labels is carried out uniformly at random from* $\mathrm{GF}(q)^-$ [29, 35, 36, 43, 81].

Like the binary counterpart, the connectivity of an NB cycle can also be assessed with the help of its EMD value. Recall that the EMD of a cycle is the number of CNs singly-connected to the VNs involved in the cycle. For example, the EMD of the NB cycle shown in Figure 6.2(a) is 2.

6.3 Proposed Rate-compatible Puncturing Scheme

For codes with mean column weight $t \geq 3$, the Tanner graph contains a comparatively large number of short cycles. According to Corollary 6.1, if the field size is also relatively small, these codes are expected to contain a considerable number of short NB cycles. Such codes have been studied recently by Amiri *et al.* [5] in the context of the non-binary ASs. From the EXIT chart analysis in Chapter 5, we find that for the codes with comparatively higher mean column weights, the symbolwise puncturing delivers the better result. We apply the symbolwise puncturing technique of [32] in the case of the NB-LDPC codes and propose an improved version by considering the lengths and the EMD values of the short NB cycles. This work is inspired by the RC puncturing technique proposed by Asvadi and Banihashemi [6] in the context of the binary LDPC codes. Nevertheless the proposed work is different from that of [6] in two aspects: (1) The concept of a legitimate harmful cycle for the NB-LDPC codes is particularly different from that of the binary LDPC codes. It depends on the non-binary labels for the edges involved in the cycle as described in Section 6.2. (2) We consider the EMD values of the cycles unlike the ACE values considered in [6]. This step is motivated by the observation that the ACE may not be equal to the EMD as illustrated in Chapter 4. The ACE does not reflect the connectivity of the cycle correctly when it is not equal to the EMD. Specifically, recall that the ACE values of all cycles of the same length are the same for the regular codes. That means, apart from the length, the notion of the ACE of a cycle does not convey any additional information about the cycle in the case of the regular codes. In that case, only the EMD can express the connectivity of a cycle.

The improved symbolwise puncturing scheme comprises of two steps. In the *first* step, the set of the VNs in the Tanner graph of the code is partitioned into different groups, $\mathbf{G}_0, \mathbf{G}_1, \cdots, \mathbf{G}_K$ with the help of the grouping algorithm [32]. In the *second* step, the VNs in different groups are sorted. For that, we consider the lengths and the EMD values of the short NB cycles. The proposed cycle-based sorting procedure is described in the following.

Sorting via Cycle-based Criterion

The short NB cycles with low EMD are harmful and affects the performance of iterative decoders. If the VNs present in such a cycle are punctured, the reliability of the messages flowing in the cycle will be affected adversely. Consequently, it will make the cycle even more prone to error. The VNs which are involved in a higher number of harmful NB cycles possess smaller chances of correct recovery after being punctured. Thus the objective is to select those VNs which are involved in a lower number of short NB cycles with low EMD values. The details of the selection rule is explained below.

First, all the short NB cycles of length up to a specific length $l_{\max}$ are found out. The detrimental effect of an NB cycle is decided by two factors:

- *Length*: If the length of a cycle is small, the messages flowing in the cycle become correlated after a small number of iterations [61]. The smaller the length of the cycle, the more is the harmfulness.
- *EMD*: A cycle with low EMD has poor connectivity with the rest of the Tanner graph. Therefore, there is a lower chance of any error in the VNs being corrected.

By considering the above two factors, an NB cycle of length l can be denoted by a 2-tuple notation: (l, EMD). Using this notation, the NB cycles can be grouped into different classes. Let the class of (l_i, EMD_i) cycles be denoted by $\mathcal{C}_i$. Noting that a small l and a low EMD contribute to the deterioration of iterative decoding, different degrees of importance can be given to these two factors in sorting the VNs: more importance to l, more importance to EMD or equal importance to both. Our experimental observations suggest to give more importance to l.

Based on the above considerations, the relative harmfulness of two classes $\mathcal{C}_i$ and $\mathcal{C}_j$ $(i \neq j)$ is determined by the following two conditions:

(1) If $l_i < l_j$, then $\mathcal{C}_i$ is considered more harmful than $\mathcal{C}_j$.

(2) When $l_i = l_j$, then $\mathcal{C}_i$ is considered more harmful than $\mathcal{C}_j$ if $\text{EMD}_i < \text{EMD}_j$.

The short NB cycles can now be grouped into F classes $\mathcal{C}_1, \cdots, \mathcal{C}_F$ with the decreasing order of harmfulness. The number of classes F is a variable determined by $l_{\max}$ and the nature of the short NB cycles for the particular code.

Algorithm 6.1: Sorting via Cycle-based Criterion

input : The groups of the VNs obtained from the grouping algorithm: $\mathbf{G}_1, \ldots, \mathbf{G}_K$, the sets of the NB cycles of the ordered classes: $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_F$

output: The set P_l of the VNs selected for puncturing to achieve rate r_l, $1 \leq l \leq m$

Initialize the set P_0 to an empty set, $k = 1$;
for $l \leftarrow 1$ **to** m **do** // for all the rates r_l, $1 \leq l \leq m$
  Compute N_l^p ; // the number of VNs required to be punctured to attain r_l
  $P_l = P_{l-1}$; // rate compatible puncturing
  $\delta N_l^p = N_l^p - |P_l|$; // the remaining number of VNs to be selected for puncturing
  while $\delta N_l^p \neq 0$ **do** // select the required number of VNs
    Set $\Gamma = \mathbf{G}_k$; // the set of candidate VNs
    foreach *VN* v *in* Γ **do**
      Obtain the sets $\mathcal{C}_1^v, \mathcal{C}_2^v, \ldots, \mathcal{C}_F^v$,
      where $\mathcal{C}_i^v$ is the set of cycles of class $\mathcal{C}_i$ containing v.
    end
    Set $i = 1$; // index for the cycle classes
    while $i \leq F$ **do**
      Retain only those VNs v in Γ for which $|\mathcal{C}_i^v|$ is minimum over all $v \in \Gamma$,
      i.e. $\Gamma = \left\{ v \in \Gamma \text{ such that } |\mathcal{C}_i^v| = \min_{v' \in \Gamma} \left| \mathcal{C}_i^{v'} \right| \right\}$; // reduce the search space
      if $|\Gamma| = 1$ **then** // Γ contains only one VN
        $v^* = \Gamma$; // select the lone VN for puncturing
        break ; // the job is done
      else if $i = F$ **then** // all the cycle classes are examined
        Select a VN randomly from Γ and call it v^*;
      else // $|\Gamma| > 1$ and $i < F$
        $i \leftarrow i + 1$; // shrink Γ further by considering the next cycle class
      end
    end
    $P_l = P_l \cup \{v^*\}$; // include v^* in the set P_l
    $\delta N_l^p \leftarrow \delta N_l^p - 1$;
    $\mathbf{G}_k = \mathbf{G}_k \backslash \{v^*\}$;
    if $|\mathbf{G}_k| = 0$ **then** // if all the VNs in $\mathbf{G}_k$ have been selected
      $k \leftarrow k + 1$; // select the remaining VNs from the next group
    end
  end
end

Algorithmic Steps

The proposed sorting algorithm is shown in Algorithm 6.1. Let, $r_1, r_2, \cdots, r_m$ ($r_m \leq r_{\max}$) be the sequence of rates for which the puncturing patterns are to be fixed. Suppose, N_l^p is the number of VNs required to be punctured to attain a certain rate r_l.

The inputs to the algorithm are:

- The groups of the VNs of different recoverability levels as obtained from the grouping algorithm, $\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_K$.
- The sets of the NB cycles of different classes $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_F$.

The algorithm selects the VNs required to be punctured to attain the rates $r_1, r_2, \cdots, r_m$ from the mother code of rate r_0. The set P_l of the VNs to be punctured for rate r_l includes the set P_{l-1} of the VNs selected for rate r_{l-1} to ensure the rate-compatibility. Let the set of candidate VNs for puncturing be denoted by Γ. The VNs of the lowest recovery level, which are not yet selected, are included in Γ.

The search space Γ is gradually reduced by considering the involvement of the VNs of Γ in the formation of the NB cycles. Suppose, $\mathcal{C}_i^v$ denotes the set of all the cycles of the class $\mathcal{C}_i$ involving the VN v. For each VN v in Γ, the sets $\mathcal{C}_i^v$, $i = 1, \ldots, F$ are found out. The process of reduction of the search space Γ starts with the consideration of the first cycle class, i.e., $\mathcal{C}_i$ with $i = 1$. Only those VNs v are retained in Γ for which $|\mathcal{C}_i^v|$ is minimum over all $v \in \Gamma$. Thus the search space gets reduced. If Γ contains only one VN, then that VN is included for puncturing. Otherwise we check for the possibility of reducing the size of Γ further by considering the next cycle class. If all the cycle classes are examined and still $|\Gamma| > 1$, then a VN is selected randomly from Γ. This process of selecting a VN is repeated to obtain all the required number of VNs. The above steps are carried out till all the rates are considered. The algorithm outputs the sets P_l, $l = 1, \cdots, m$ containing the VNs selected for symbolwise puncturing to reach a rate r_l for $l = 1, \cdots, m$.

Remark 6.2. *The symbolwise puncturing scheme for the binary LDPC codes in* [32] *consists of a grouping algorithm followed by a sorting algorithm. The two selection criteria for the sorting algorithm in* [32] *are:*

(a) the number of survived neighboring CNs connected to a VN and

(b) the degree of a VN

For the proposed sorting scheme, the selection of the punctured VNs is accomplished only by examining the involvement of the VNs in the short NB cycles with low EMD.

6.4 Simulation Results

We have considered an $N = 250$, $M = 125$, $r_0 = 0.5$ random regular code over GF(2^4) as a mother code. The degrees of a VN and a CN are 4 and 8 respectively. The girth of the binary parity-check matrix is 6. An all-zero codeword is considered for the performance evaluation. The codeword bitstream is BPSK modulated and transmitted over a BI-AWGN channel. Decoding is done by the FFT-QSPA with the maximum number of iterations set at 50. The SER for the mother code is shown in Figure 6.4.

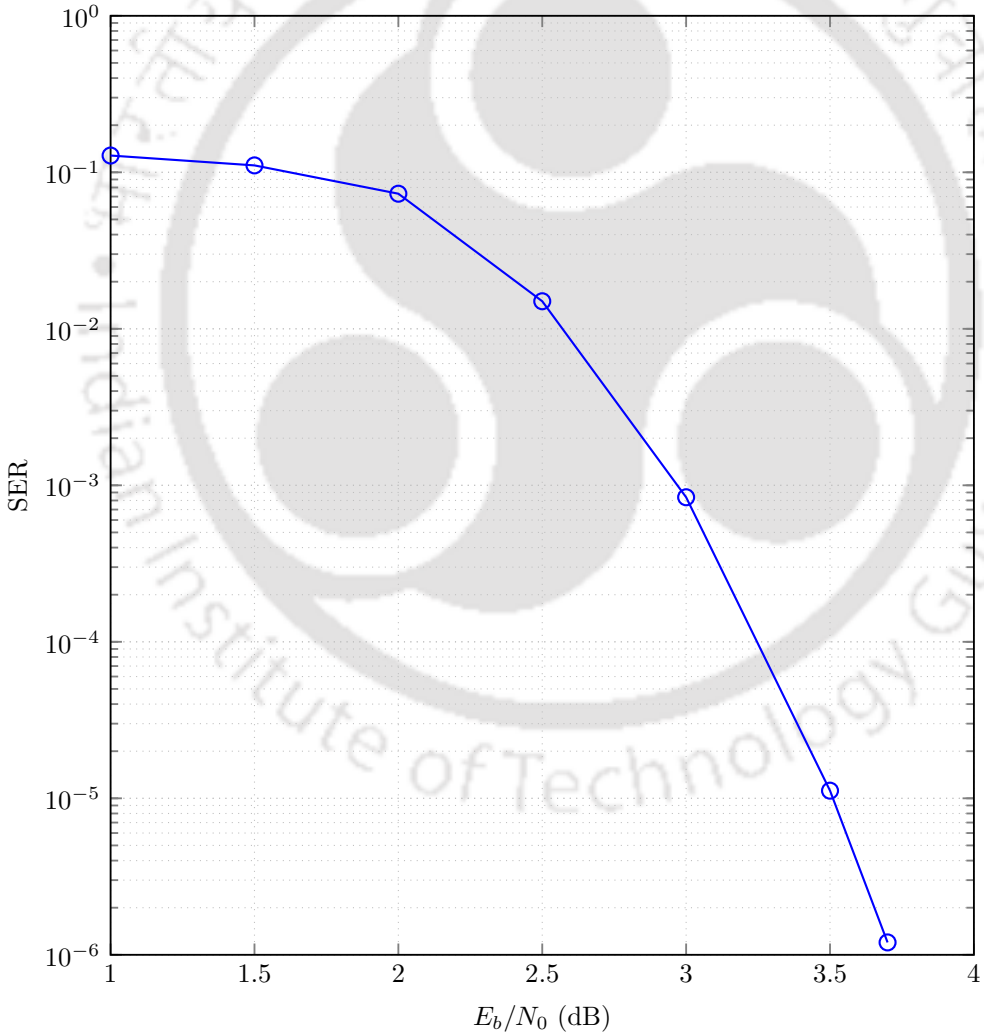


Figure 6.4: SER of the (4,8)-regular mother code over GF (2^4) with $N = 250, M = 125$

The grouping algorithm partitions the set of VNs into four groups. The number of VNs in these groups are shown in Table 6.1. Using (5.3), the maximum rate allowable in this code is found to be

Table 6.1: Numbers of VNs in different groups for the (4,8)-regular mother code over GF (2^4) with $N = 250, M = 125$

G_0	G_1	G_2	G_3
171	63	14	2

$r_{\max} = 0.731$. Therefore, we consider two rates: $r_1 = 0.6$ and $r_2 = 0.7$. For simulation studies, we consider one symbolwise and two bitwise puncturing patterns. The bitwise patterns are: B1 $\rightarrow$ uniform spreading over the VNs of the same recovery level as prescribed in [43] and B2 $\rightarrow$ nonuniform and full spreading over all the VNs of the same recovery level. The distribution of these puncturing patterns at $r_1 = 0.6$ and $r_2 = 0.7$ are shown in Table 6.2. The average number of bits punctured per VN for

Table 6.2: The distributions of different puncturing patterns

Level	Number of punctured bits per VN	$r_1 = 0.6$			$r_2 = 0.7$		
		Symbolwise	Bitwise (B1) [43]	Bitwise (B2)	Symbolwise	Bitwise (B1) [43]	Bitwise (B2)
1-SR	4	41			63	63	63
	3		55	40			
	2			23			
2-SR	4				8		
	3					11	3
	2						11
3-SR	1						2
Total		164	165	166	284	285	285
Avg. b_p		4	3	2.6349	4	3.8378	3.6076

each pattern is also shown in the table. For finding the NB cycles, we consider $l_{\max} = 12$. Table 6.3 shows the numbers of NB cycles of different classes. We get $F = 15$.

Table 6.3: Numbers of NB cycles of different classes for the (4,8)-regular mother code over GF (2^4) with $N = 250, M = 125$

Cycle Class	Number of **Cycles**
(6,6) ($\mathcal{C}_1$)	4
(8,6) ($\mathcal{C}_2$)	4
(8,8) ($\mathcal{C}_3$)	38
(10,6) ($\mathcal{C}_4$)	4
(10,7) ($\mathcal{C}_5$)	4
(10,8) ($\mathcal{C}_6$)	49
(10,9) ($\mathcal{C}_7$)	23
(10,10) ($\mathcal{C}_8$)	415
(12,6) ($\mathcal{C}_9$)	5
(12,7) ($\mathcal{C}_{10}$)	6
(12,8) ($\mathcal{C}_{11}$)	112
(12,9) ($\mathcal{C}_{12}$)	132
(12,10) ($\mathcal{C}_{13}$)	917
(12,11) ($\mathcal{C}_{14}$)	500
(12,12) ($\mathcal{C}_{15}$)	3153

In order to verify the efficiency of symbolwise puncturing for the particular mother code, we consider the following three schemes:

(1) *Symbolwise puncturing after [32]*: This scheme follows the symbolwise pattern. The selection of the VNs in a particular group is done according to the sorting algorithm [32].

(2) *Bitwise B1 random after [43]*: This bitwise scheme follows the pattern B1. The selection of the VNs in a particular group is done randomly.

(3) *Bitwise B1 cycle*: This bitwise scheme follows the pattern B1. The selection of the VNs in a particular group is done according to the proposed cycle-based sorting algorithm.

Figure 6.5 shows the SER performances of the three schemes at $r_1 = 0.6$ and $r_2 = 0.7$. Observe that the symbolwise scheme performs better than the two bitwise schemes at both the rates. Between the bitwise schemes, scheme (3) performs better than the scheme (2). This is due to the selection of the puncturing VNs according to the cycle-based sorting technique in the scheme (3). However, even after following the cycle-based selection process, the scheme (3) cannot surpass the symbolwise scheme. Thus the advantage of symbolwise puncturing for the particular mother code is established. Observe

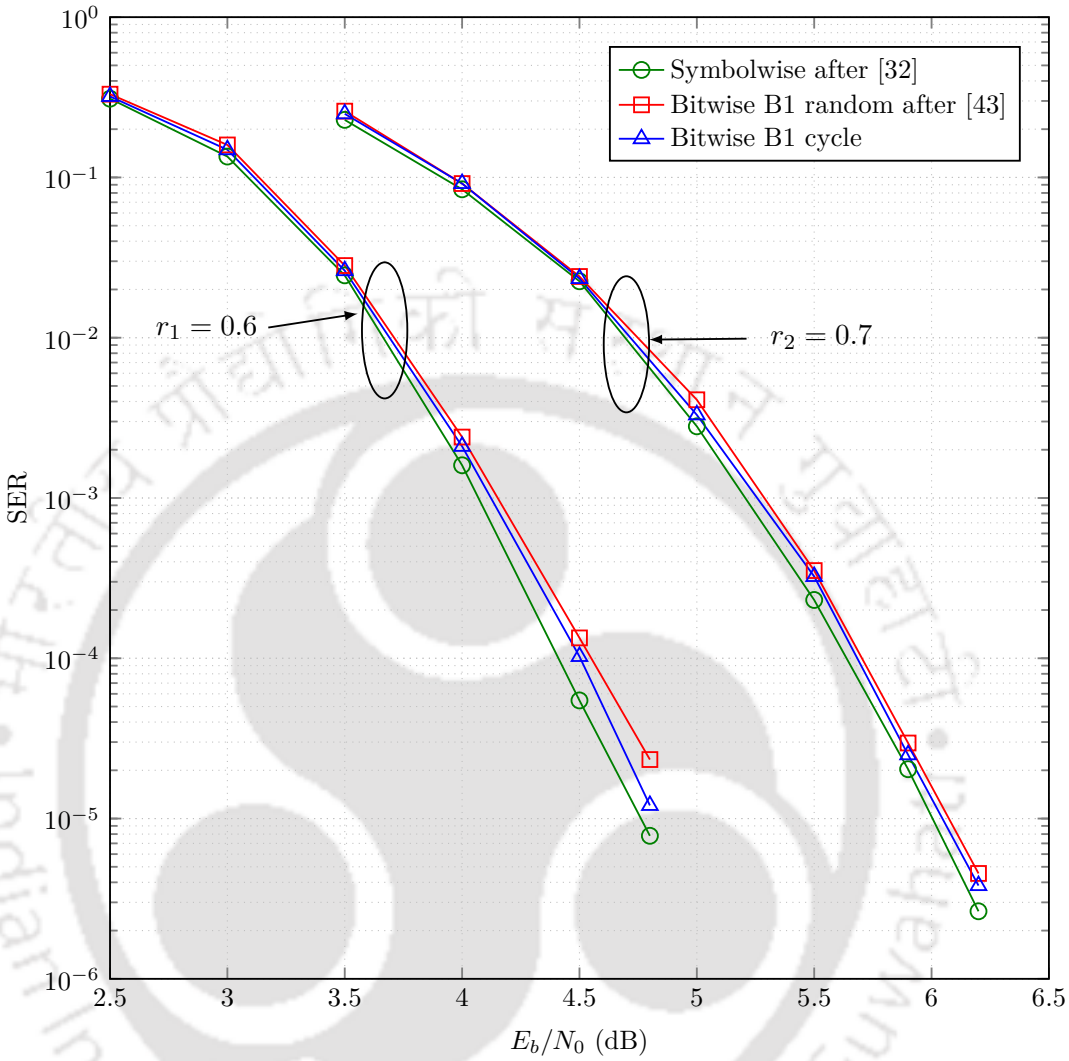


Figure 6.5: Comparison of symbolwise and bitwise puncturing schemes

from Figure 6.5 that the difference between the three schemes at $r_2 = 0.7$ is minimal. This is due to the fact that many VNs in the bitwise schemes must undergo symbolwsie puncturing because of the limited number of recoverable VNs. This fact can be verified from Table 6.2.

Since symbolwise puncturing produces better results in this code over GF(2^4), we turn to the proposed cycle-based symbolwise puncturing scheme to improve the performance further. The symbolwise puncturing scheme in [32] consists of the grouping algorithm followed by the sorting algorithm. Through simulations, we have found that even after the sorting algorithm, a large number of punctured VNs are selected randomly specifically at lower rates. We also include the cycle-based criterion after this sorting algorithm just to check the amount of improvements in the performance. Thus,

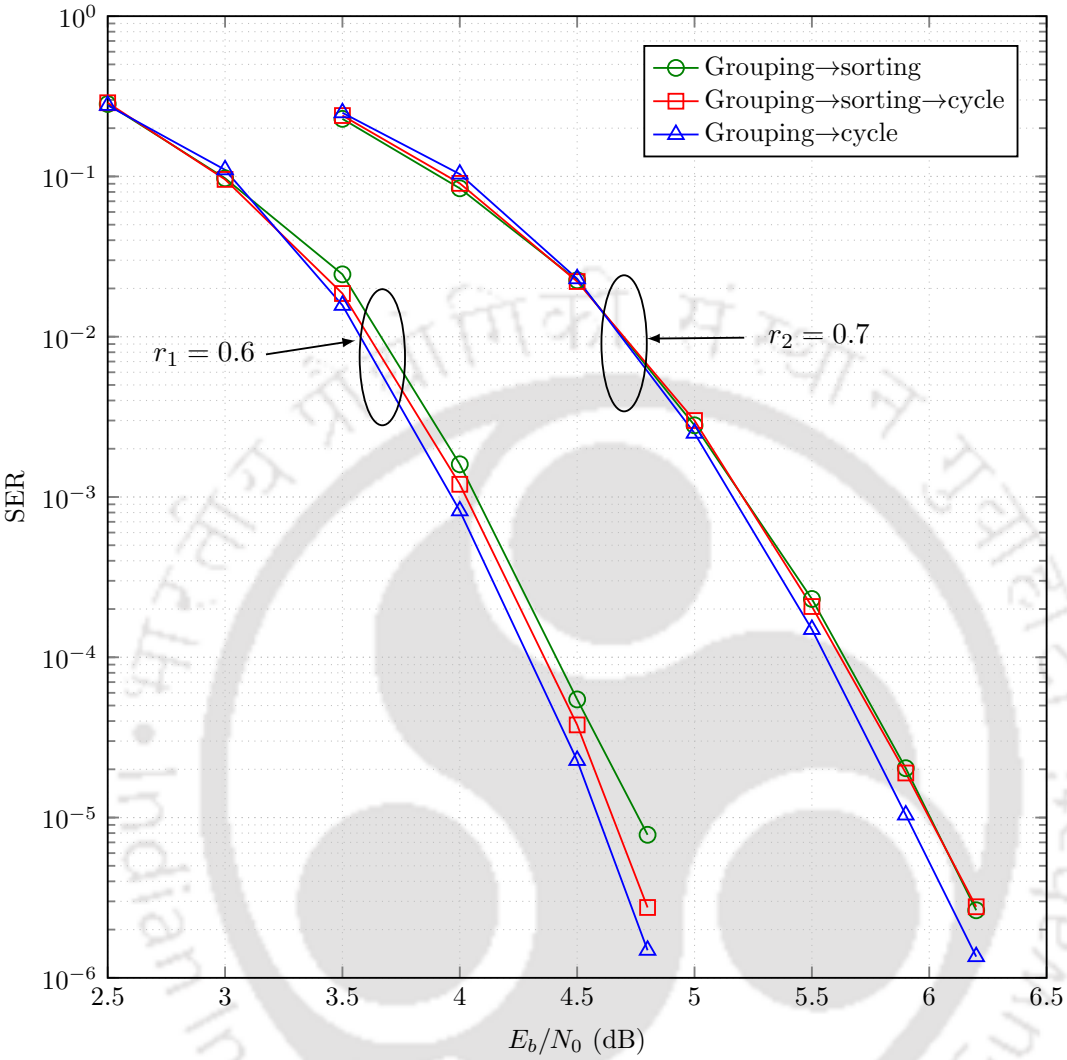


Figure 6.6: Comparison of different symbolwise puncturing schemes

altogether, we consider the following three symbolwise puncturing schemes:

(a) *Grouping* → *sorting* (standard approach [32])

(b) *Grouping* → *sorting* → *cycle* (to check the improvement)

(c) *Grouping* → *cycle* (proposed scheme)

Figure 6.6 shows the SER performances of these schemes at $r_1 = 0.6$ and $r_2 = 0.7$. It can be observed that at $r_1 = 0.6$, the scheme (c) performs better than the other two. The coding gain of the proposed scheme (scheme (c)) over the standard approach (scheme (a)) at SER=10^{-5} is about 0.2 dB. Scheme (b) performs better than the scheme (a) because of the removal of the randomness in

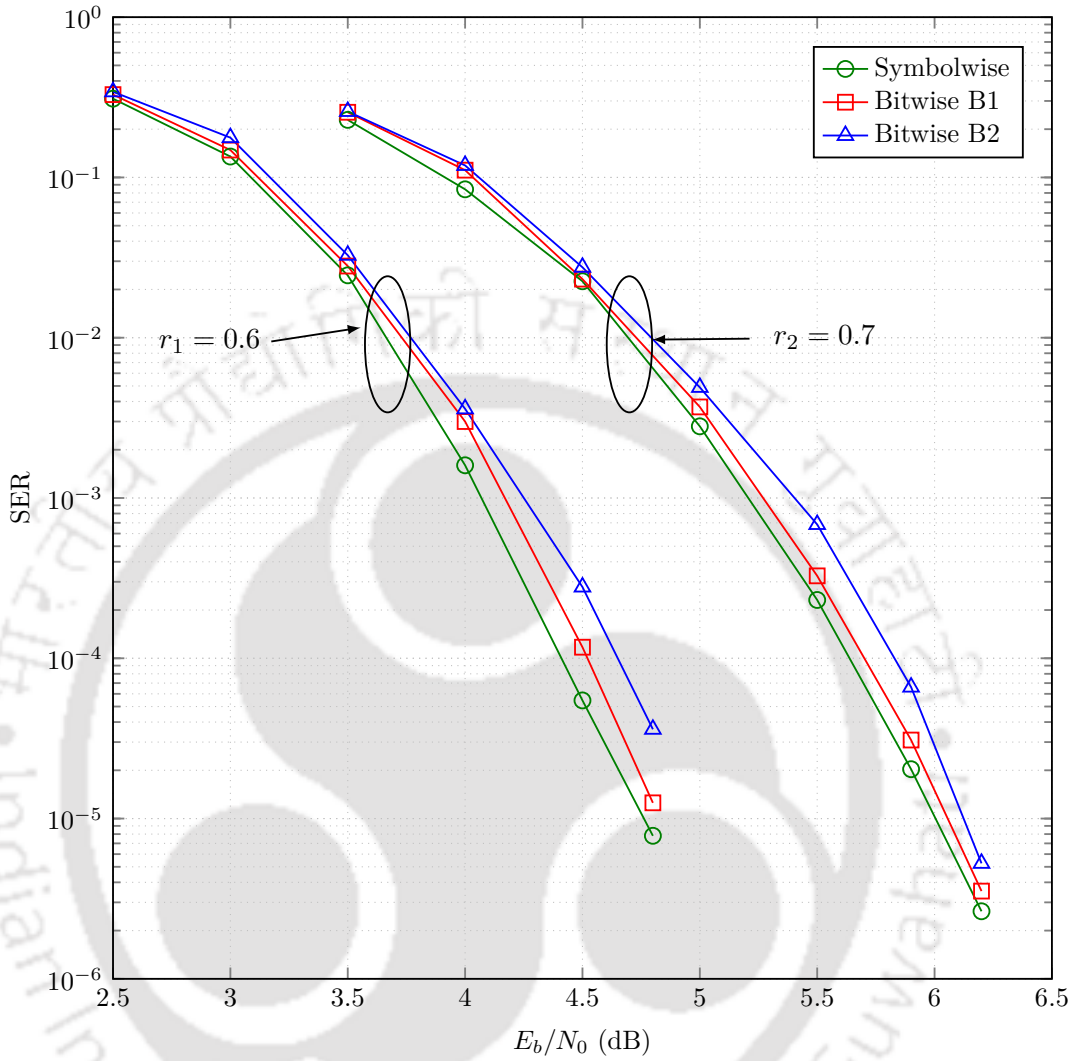


Figure 6.7: Comparison of symbolwise, bitwise B1 and bitwise B2 puncturing patterns for **Grouping → sorting**

the sorting algorithm through the concatenation of the cycle-based criterion. At $r_2 = 0.7$, we have to select 8 out of the 14 VNs in 2-SR level. Therefore, the difference between scheme (a) and scheme (b) is minimal. The scheme (c) performs better than the other two with a coding gain of about 0.12 dB at SER=10^{-5}.

We also evaluate the performance of the nonuniform and fully-spread bitwise puncturing pattern B2. These performances are compared with the symbolwise pattern and uniform bitwise puncturing pattern B1 in the context of the three selection rules considered previously, i.e.,: (a) Grouping → sorting, (b) Grouping → sorting → cycle and (c) Grouping → cycle . The performances of the puncturing patterns in these three contexts are shown in Figure 6.7, Figure 6.8 and Figure 6.9,

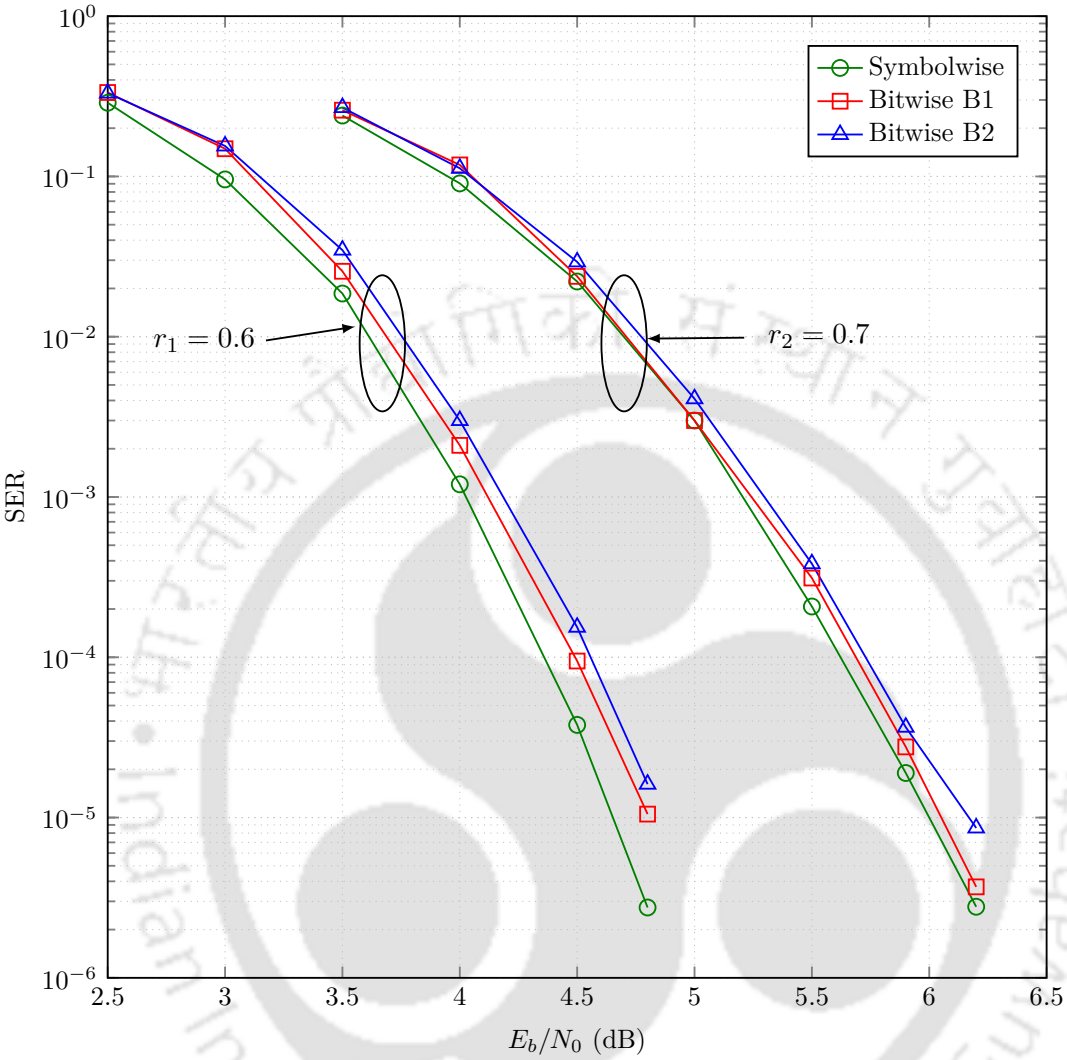


Figure 6.8: Comparison of symbolwise, bitwise B1 and bitwise B2 puncturing patterns for **Grouping → sorting → cycle**

respectively. It can be observed that the symbolwise puncturing ($b_p = 4$) delivers the best performance in all the situations, whereas, the puncturing pattern B2 yields the worst result. Observe from Table 6.2 that on an average the puncturing pattern B2 has $b_p = 2.6349$ at $r_1 = 0.6$ and $b_p = 3.6076$ at $r_2 = 0.7$. Recall from Section 5.3 that as the mean column weight of the mother code is on the higher side, a higher b_p will result in a lower threshold. Amongst the three puncturing patterns in Table 6.2, the symbolwise pattern has the largest b_p and the bitwise pattern B2 has the smallest b_p. Consequently, the threshold for the symbolwise pattern will be the lowest and for the bitwise pattern B2, the threshold will be the highest. Therefore, the simulation results are in full agreement with the threshold analysis done in Section 5.3.

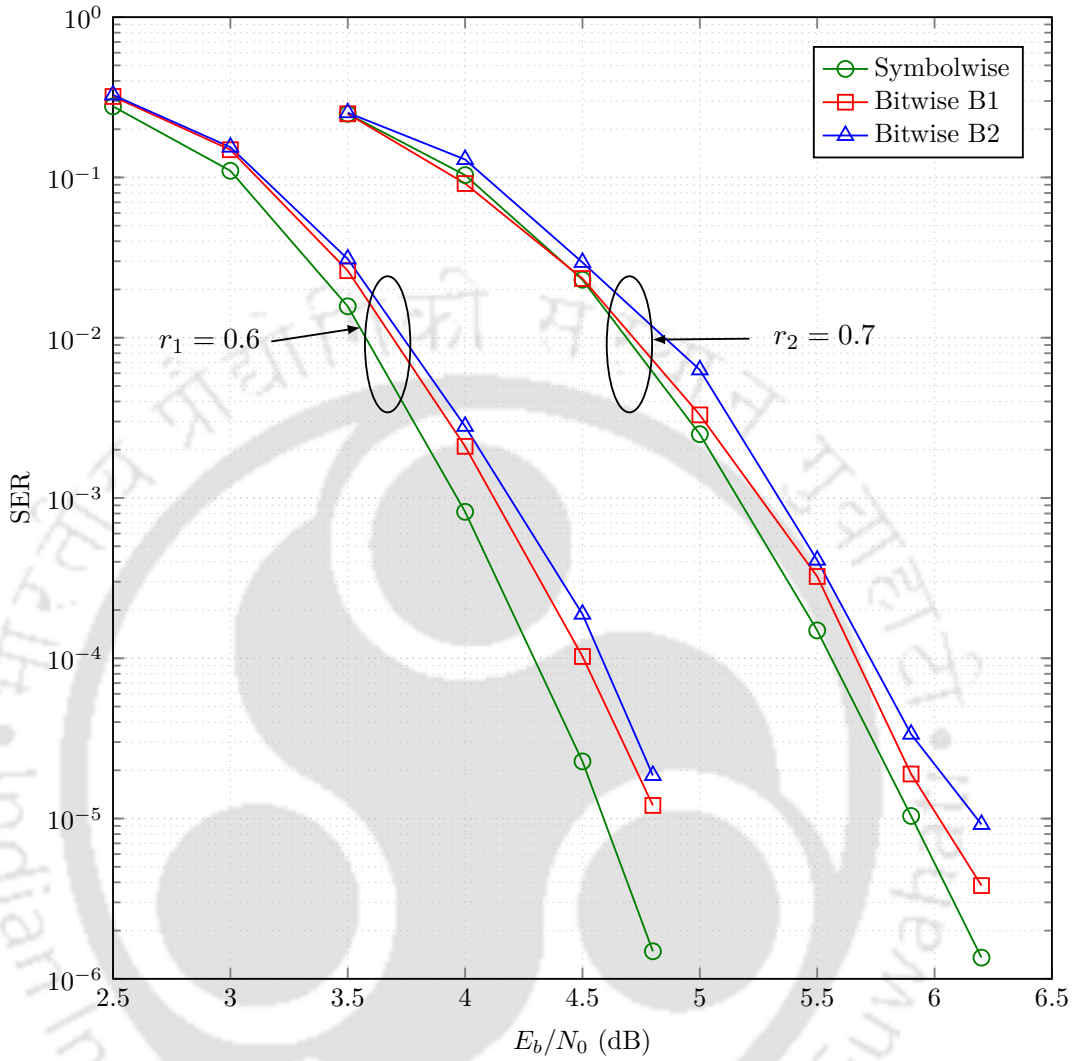


Figure 6.9: Comparison of symbolwise, bitwise B1 and bitwise B2 puncturing patterns for **Grouping → cycle**

Remark 6.3. *Note that some NB-LDPC codes of low mean column weight and specially defined over a higher order field may not contain any short NB cycle. For such codes, the proposed cycle-based selection rule will not be applicable.*

6.5 Conclusions

In this chapter, we have studied the effects of short cycles in the iterative decoding of the NB-LDPC codes. We discussed the concept of an NB cycle. A cycle in the binary sense can be treated as an NB cycle only if the edge-labels in the cycle satisfy a certain condition. Only the legitimate NB cycles present in the Tanner graph are harmful for the iterative decoders. The Tanner graph of

an NB-LDPC code usually contains a few short NB cycles. The advantages of the NB-LDPC codes over the equivalent binary counterparts in the context of the error floor are corroborated. For certain NB-LDPC codes defined over a smaller field and having higher mean column weights, the Tanner graph possesses a higher probability of containing short NB cycles. For such kind of codes we have proposed an improved version of RC puncturing algorithm by reckoning the lengths and the EMD values of the short NB cycles. The VNs which are involved in a small number harmful NB cycles are expected to recover fast and accurately after puncturing. Therefore, the proposed cycle-based selection criterion gives preference to those VNs which are involved in a less number of short cycles with low EMD. Simulation results are shown for this technique. Observations in different contexts established that the selection of the punctured VNs according to the cycle-based criterion can improve the performance significantly.

7

Conclusions

Contents

This thesis addressed some issues of the binary and the NB LDPC codes. This chapter summarizes the main contributions of the thesis and then proposes some directions of research that may be useful for future researchers.

7.1 Summary of Contributions

The contributions of the thesis can be divided into four parts. The first two parts deal with the TSs and the short cycles which are responsible for the error floor in the iterative decoding of the binary LDPC codes. The last two parts consider the design of optimum puncturing patterns for the NB LDPC codes in order to achieve rate-compatibility. The contributions in these four parts are summarized below:

- Finding of a list of dominant trapping sets of an LDPC code is an important task. The list aids in the construction and the decoder design of an LDPC code. The literature contains several approaches to find a list of dominant trapping sets. We considered the hierarchical approach where a smaller trapping set is gradually expanded by adding one or more VNs to obtain larger trapping sets. Based on the hierarchical approach, we proposed a technique to find the dominant trapping sets of the irregular LDPC codes. We considered only six types of sources to find the trapping sets of a particular class. The proposed technique has been able to find the dominant trapping sets of several commonly considered irregular LDPC codes. Numerical results have shown that it can find more trapping sets of a class than those by other methods available in the literature.

- A dominant trapping set of an LDPC code usually contains one or more harmful short cycles. In addition to the length, the connectivity of a cycle is also crucial parameter. A poorly connected cycle affects the performance of the iterative decoder significantly. The connectivity of a cycle is measured by the EMD. Because of simplicity, the ACE is usually used as a substitute for the EMD. However, in many cases, the ACE is not equal to the EMD. A cycle with unequal ACE and EMD contains some sub-cycles. For an ACE spectrum constrained code, these sub-cycles must satisfy the girth and the ACE constraints specified by the particular ACE spectrum. Considering these constraints, we derived three sufficient conditions for the equality of the ACE and the EMD. The first condition is based on the girth constraint and the remaining two are derived by taking

the ACE constraints into account. Investigations on two ACE spectrum constrained codes were carried out to examine the efficiency of these sufficient conditions.

- We proposed an EXIT chart model to analyze bitwise and symbolwise puncturing patterns for the NB LDPC codes. We observed that the performance of a puncturing pattern is determined by the degrees of the VNs. The grouping algorithm [32] is a standard tool to identify the VNs which can be recovered after puncturing. It normally selects the low-degree VNs for puncturing. By keeping the constraints from the grouping algorithm in mind, an EXIT chart based strategy is devised to obtain the optimum recoverable puncturing pattern for the NB LDPC codes. The novelty of this puncturing scheme arises from the joint use of both asymptotic (EXIT charts) and finite-length (grouping algorithm) techniques. The optimized recoverable puncturing patterns performed better than the other puncturing schemes available in the literature at different rates.

- We have studied the effects of the cycles in the context of the iterative decoding of the NB LDPC codes. The cycles which satisfy a certain condition based on the edge labels are harmful for the iterative decoder. These cycles are known as NB cycles. We find an expression for the probability of a cycle becoming an NB cycle for a code defined over GF(q). This probability depends only on q. The Tanner graph of an NB LDPC code defined over a very high order field, contains very few short NB cycles. However, some NB LDPC codes of higher mean column weights and defined over a smaller field may contain a sizable number of short NB cycles. We developed an effective RC puncturing technique for such NB LDPC codes. For that, we formulated a cycle-based rule to select the punctured VNs. The proposed rule selects those VNs which are involved in a lower number of short NB cycles with low EMD values. Simulation results in different contexts confirmed the usefulness and the efficiency of the cycle-based selection rule.

7.2 Possible Future Work

In continuation of the works presented in the thesis, there are several problems that can be the subject of future research. Some possible directions are outlined below.

- Using the list of dominant trapping sets, a decoder-based strategy may be devised to improve the error floor performance. The conventional decoding steps can be modified or some new steps can be appended in order to reduce the effects of the known dominant trapping sets.

- The ACE is an approximation for the EMD. An algorithm may be developed to construct a parity-check matrix based on the EMD spectrum instead of ACE spectrum.

- There is a scope of deriving more sufficient conditions for the equality of the ACE and the EMD of a cycle of the ACE spectrum constrained LDCPC codes. In addition to the sufficient conditions, the necessary conditions may be derived. The necessary and the sufficient conditions together will result in the complete characterization of the equality of the ACE and the EMD of a cycle.

- Shortening reduces the rate of a code. An EXIT chart model can be formulated in the context of shortening for the NB LDPC codes. With the help of this EXIT chart, the optimum shortening pattern can be identified.

- The improved RC puncturing scheme for the NB LDPC codes discussed in Chapter 6 takes the EMD values of the short NB cycles into account. The performance of the proposed cycle-based scheme is investigated for a regular code. An equivalent cycle-based scheme involving the ACE values of the short NB cycles can be investigated in the case of the irregular codes. By comparing this scheme with the proposed scheme, the benefit of considering the EMD instead of the ACE can be analyzed.

- The cycles influence the binary LDPC codes more significantly than the NB LDPC codes. The cycle-based puncturing scheme proposed in Chapter 6 for the NB LDPC codes can be examined in the case of the binary LDPC codes.

- For the RC coding schemes in a fast changing environment, the selection of the proper transmission rate is very crucial. There have to be some feedback mechanisms from the receiver to the transmitter so that the coding system can be adapted to the suitable rate depending on the current SNR value. This means that the receiver must send some information back to the transmitter regarding the current SNR value. The transmitter can thus judiciously select the convenient rate. The proposed RC puncturing schemes for the NB LDPC codes may be studied in the context of a more practical environment with the inclusion a feedback system.

Bibliography

[1] Available [Online] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5307322.

[2] "Short Blocklength LDPC Codes for TC Synchronization and Channel Coding, Recommendation for Space Data System Standards, CCSDS 231.0-O-y.y," [Available online] http://cwe.ccsds.org/sls/docs/SLS-CandS/MeetingOct. 2012.

[3] S. Abu-Surra, D. DeClercq, D. Divsalar, and W. Ryan, "Trapping set enumerators for specific LDPC codes," in *Information Theory and Applications Workshop (ITA), 2010*, 31 2010-Feb. 5 2010, pp. 1 –5.

[4] B. Amiri, J. Kliewer, and L. Dolecek, "Analysis and enumeration of absorbing sets for non-binary graph-based codes," in *IEEE International Symposium on Information Theory Proceedings (ISIT), 2013*, 2013, pp. 2815–2819.

[5] ——, "Analysis and Enumeration of Absorbing Sets for Non-Binary Graph-Based Codes," *IEEE Transactions on Communications,*, vol. 62, no. 2, pp. 398–409, February 2014.

[6] R. Asvadi and A. Banihashemi, "A Rate-Compatible Puncturing Scheme for Finite-Length LDPC Codes," *IEEE Communications Letters,*, vol. 17, no. 1, pp. 147–150, January 2013.

[7] R. Asvadi, A. Banihashemi, and M. Ahmadian-Attari, "Lowering the Error Floor of LDPC Codes Using Cyclic Liftings," *Information Theory, IEEE Transactions on*, vol. 57, no. 4, pp. 2213 –2224, April 2011.

[8] ——, "Design of Finite-Length Irregular Protograph Codes with Low Error Floors over the Binary-Input AWGN Channel Using Cyclic Liftings," *IEEE Transactions on Communications,*, vol. 60, no. 4, pp. 902 –907, April 2012.

[9] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2q)," in *IEEE Information Theory Workshop, 2003.*, March 2003, pp. 70–73.

[10] A. Bennatan and D. Burshtein, "Design and Analysis of Nonbinary LDPC codes for Arbitrary Discrete-memoryless Channels," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 549–583, Feb 2006.

[11] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *IEEE International Conference on Communications, 1993. ICC '93 Geneva.*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[12] N. Bonello, S. Chen, and L. Hanzo, "Low-Density Parity-Check Codes and Their Rateless Relatives," *IEEE Communications Surveys Tutorials*, vol. 13, no. 1, pp. 3–26.

[13] E. Cavus, C. Haymes, and B. Daneshrad, "An IS Simulation Technique for Very Low BER Performance Evaluation of LDPC Codes," *IEEE International Conference on Communications, 2006. ICC '06.*, vol. 3, pp. 1095 –1100, Jun. 2006.

[14] S.-Y. Chung, J. Forney, G.D., T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters,*, vol. 5, no. 2, pp. 58 –60, Feb 2001.

[15] S.-Y. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657 –670, Feb 2001.

[16] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "A General Method for Finding Low Error Rates of LDPC Codes," *CoRR*, vol. abs/cs/0605051, 2006.

[17] D. Declercq, M. Fossorier, and E. Biglieri, Ed., *Channel Coding: Theory, Algorithms, and Applications*. Academic Press, 2014.

[18] M. Davey and D. MacKay, "Low-density parity check codes over GF(q)," *IEEE Communications Letters,*, vol. 2, no. 6, pp. 165–167, 1998.

[19] David Mackay and Matthew Davey, "Evaluation of Gallager Codes for Short Block Length and High Rate Applications," Available online on David Mackay's wesite, "http://www.cs.toronto.edu/ mackay/seagate.ps.gz".

[20] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes Over GF(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, 2007.

[21] ——, "Improved impulse method to evaluate the low weight profile of sparse binary linear codes," in *IEEE International Symposium on Information Theory, 2008, ISIT 2008.*, July 2008, pp. 1963–1967.

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

[23] L. Dolecek, P. Lee, Z. Zhang, V. Anantharam, B. Nikolic, and M. Wainwright, "Predicting error floors of structured LDPC codes: deterministic bounds and estimates," *IEEE Journal on Selected Areas in Communications,*, vol. 27, no. 6, pp. 908 –917, August 2009.

[24] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of Absorbing Sets and Fully Absorbing Sets of Array-Based LDPC Codes," *IEEE Transactions on Information Theory,*, vol. 56, no. 1, pp. 181 –201, Jan. 2010.

[25] L. Dolecek, Z. Zhang, M. Wainwright, V. Anantharam, and B. Nikolic, "Evaluation of the Low Frame Error Rate Performance of LDPC Codes Using Importance Sampling," in *IEEE Information Theory Workshop, 2007. ITW '07.*, Sept. 2007, pp. 202 –207.

[26] Y. Fang, P. Chen, L. Wang, and F. Lau, "Design of Protograph LDPC Codes for Partial Response Channels," *IEEE Transactions on Communications,*, vol. 60, no. 10, pp. 2809–2819, October 2012.

[27] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. 8, no. 1, p. 21 Ű28, Jan 1962.

[28] ——, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[29] D. Goldin and D. Burshtein, "Iterative Linear Programming Decoding of Nonbinary LDPC Codes With Linear Complexity," *IEEE Transactions on Information Theory,*, vol. 59, no. 1, pp. 282–300, Jan 2013.

[30] M. Gorgoglione, V. Savin, and D. Declercq, "Optimized Puncturing Distributions for Irregular Non-binary LDPC Codes," in *International Symposium on Information Theory and its Applications*, 2010.

[31] M. Gorgoglione, [Online] http://tel.archives-ouvertes.fr/docs/00/81/94/15/PDF/PhD_Manuscript_MatteoGorgoglione.pdf.

[32] J. Ha, J. Kim, D. Klinc, and S. McLaughlin, "Rate-compatible Punctured Low-density Parity-check Codes with Short Block Lengths," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 728–738, 2006.

[33] Y. Han and W. Ryan, "Low-floor decoders for LDPC codes," *IEEE Transactions on Communications,*, vol. 57, no. 6, pp. 1663 –1673, June 2009.

[34] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Transactions on Information Theory,*, vol. 51, no. 1, pp. 386 –398, Jan. 2005.

[35] X.-Y. Hu and E. Eleftheriou, "Binary representation of cycle Tanner-graph GF(2b) codes," in *IEEE International Conference on Communications, 2004*, vol. 1, June 2004, pp. 528–532 Vol.1.

[36] J. Huang, W. Zhou, and S. Zhou, "Structured Nonbinary Rate-Compatible Low-Density Parity-Check Codes," *IEEE Communications Letters*, vol. 15, no. 9, pp. 998–1000, September 2011.

[37] M. Ivkovic, S. Chilappagari, and B. Vasic, "Eliminating Trapping Sets in Low-Density Parity-Check Codes by Using Tanner Graph Covers," *IEEE Transactions on Information Theory,*, vol. 54, no. 8, pp. 3763 –3768, Aug. 2008.

[38] M. Jiang, C. Wang, Y. Zhang, and C. Zhao, "An improved variable length coding scheme using structured LDPC codes," in *International Conference on Wireless Communications and Signal Processing (WCSP), 2010*, Oct. 2010, pp. 1 –5.

[39] J. Kang, Q. Huang, S. Lin, and K. Abdel-Ghaffar, "An Iterative Decoding Algorithm with Backtracking to Lower the Error-Floors of LDPC Codes," *IEEE Transactions on Communications*, vol. 59, no. 1, pp. 64–73, January 2011.

[40] J. Kang, L. Zhang, Z. Ding, and S. Lin, "A Two-Stage Iterative Decoding of LDPC Codes for Lowering Error Floors," in *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008.*, Nov 2008, pp. 1–4.

[41] M. Karimi and A. Banihashemi, "Efficient Algorithm for Finding Dominant Trapping Sets of LDPC Codes," *IEEE Transactions on Information Theory,*, vol. 58, no. 11, pp. 6942 –6958, Nov. 2012.

[42] S. Khazraie, R. Asvadi, and A. H. Banihashemi, "A PEG Construction of Finite-Length LDPC Codes with Low Error Floor," *IEEE Communications Letters,*, vol. 16, no. 8, pp. 1288 –1291, August 2012.

[43] D. Klinc, J. Ha, and S. McLaughlin, "On Rate-adaptability of Nonbinary LDPC Codes," in *5th International Symposium on Turbo Codes and Related Topics*, 2008, pp. 231–236.

[44] H. Kunz, "On the Equivalence Between One-Dimensional Discrete Walsh-Hadamard and Multidimensional Discrete Fourier Transforms," *IEEE Transactions on Computers,*, vol. C-28, no. 3, pp. 267–268, March 1979.

[45] G. B. Kyung and C.-C. Wang, "Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder," in *IEEE International Symposium on Information Theory (ISIT), 2010*, June 2010, pp. 739 –743.

[46] G. Li, I. Fair, and W. Krzymien, "Density Evolution for Nonbinary LDPC Codes Under Gaussian Approximation," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 997–1015, March 2009.

[47] M. Luby, M. Amin Shokrolloahi, M. Mizenmacher, and D. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation," *IEEE International Symposium on Information Theory, 1998. Proceedings. 1998*, p. 117, Aug 1998.

[48] D. Mackay, Encyclopedia of Sparse Graph Codes, [Online] http://www.inference.phy.cam.ac.uk/mackay/codes/data.

[49] D. MacKay, "Good Error-correcting Codes Based on Very Sparse Matrices," *IEEE Transactions on Information Theory,*, vol. 45, no. 2, pp. 399 –431, Mar 1999.

[50] D. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEE Electron. Lett.*, vol. 32, no. 18, pp. 1645 –1646, Aug 1996.

[51] Y. Mao and A. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," in *IEEE International Conference on Communications, 2001. ICC 2001.*, vol. 1, Jun 2001, pp. 41–44 vol.1.

[52] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic Spectra of Trapping Sets in Regular and Irregular LDPC Code Ensembles," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 39 –55, Jan. 2007.

[53] D. V. Nguyen, S. Chilappagari, M. Marcellin, and B. Vasic, "On the Construction of Structured LDPC Codes Free of Small Trapping Sets," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2280 –2302, April 2012.

[54] H. Y. Park, J. W. Kang, K. S. Kim, and K.-C. Whang, "Efficient Puncturing Method for Rate-Compatible Low-Density Parity-Check Codes," *IEEE Transactions on Wireless Communications,*, vol. 6, no. 11, pp. 3914–3919, November 2007.

[55] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[56] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2,d_c)-LDPC codes over GF(q) using their binary images," *IEEE Transactions on Communications,*, vol. 56, no. 10, pp. 1626–1635, 2008.

[57] K. Price and R. Storn, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *J . Global Optim.*, vol. 11, pp. 341–359, 1997.

[58] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. Communications, Control, and Computing*, 2003.

[59] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory,*, vol. 47, no. 2, pp. 619 –637, Feb 2001.

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

[61] W. Ryan and S. Lin, *Channel Codes: Classical and Modern.* Cambridge University Press, 2009.

[62] H. Saeedi and A. Banihashemi, "Design of irregular LDPC codes for BIAWGN channels with SNR mismatch," *IEEE Transactions on Communications,*, vol. 57, no. 1, pp. 6–11, January 2009.

[63] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, 1948.

[64] Shu Lin and Daniel J. Costello, *Error Control Coding.* Prentice Hall, 2004.

[65] B. S. Tan, K. H. Li, and K. C. Teh, "Analysis of MIMO Diversity With LDPC Codes Based on a Gaussian Approximation Approach Over Rayleigh Fading Channels," *IEEE Transactions on Vehicular Technology,*, vol. 60, no. 9, pp. 4650–4656, Nov 2011.

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

[67] S. Ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications,*, vol. 49, no. 10, pp. 1727–1737, Oct 2001.

[68] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of Low-density Parity-check Codes for Modulation and Detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, April 2004.

[69] J. Thorpe, "Low density parity check (LDPC) codes constructed from protographs," *JPL INP Progress Report*, pp. 42–154, 2003.

[70] T. Tian, C. Jones, J. Villasenor, and R. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Transactions on Communications,*, vol. 52, no. 8, pp. 1242 – 1247, Aug. 2004.

[71] B. Vellambi and F. Fekri, "Finite-length rate-compatible LDPC codes: a novel puncturing schem," *IEEE Transactions on Communications,*, vol. 57, no. 2, pp. 297–301, February 2009.

[72] D. Vukobratovic, A. Djurendic, and V. Senk, "ACE Spectrum of LDPC Codes and Generalized ACE Design," in *IEEE International Conference on Communications, 2007. ICC '07.* , June 2007, pp. 665 –670.

[73] D. Vukobratovic and V. Senk, "Evaluation and design of irregular LDPC codes using ACE spectrum," *IEEE Transactions on Communications*, vol. 57, no. 8, pp. 2272 –2279, Aug. 2009.

[74] C.-C. Wang, S. Kulkarni, and H. Poor, "Finding All Small Error-Prone Substructures in LDPC Codes," *IEEE Transactions on Information Theory,*, vol. 55, no. 5, pp. 1976 –1999, May 2009.

[75] Y. Wei, Y. Yang, M. Jiang, W. Chen, and L. Wei, "Joint Shortening and Puncturing Optimization for Structured LDPC Codes," *IEEE Communications Letters,*, vol. 16, no. 12, pp. 2060–2063, December 2012.

[76] N. v. D. S. S. L. S. Werner Henkel, Khaled Hassan and D. Declercq, "UEP Concepts in Modulation and Coding," *Hindawi Publishing Corporation, Advances in Multimedia Volume 2010, Article ID 416797.*

[77] N. Wiberg, "Codes and Decoding on General Graphs," Available online http://www.csie.ntu.edu.tw/ r92049/Files/LIU-TEK-THESIS-440.pdf, 1996, PhD thesis.

[78] M. Yazdani and A. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Communications Letters,*, vol. 8, no. 3, pp. 159–161, March 2004.

[79] Y. Zhang and W. Ryan, "Toward low LDPC-code floors: a case study," *IEEE Transactions on Communications,*, vol. 57, no. 6, pp. 1566 –1573, June 2009.

[80] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright, "Lowering LDPC Error Floors by Postprocessing," in *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008.*, 30 2008-Dec. 4 2008, pp. 1 –6.

[81] L. Zhou, B. Bai, and M. Xu, "Design of Nonbinary Rate-Compatible LDPC Codes Utilizing Bit-Wise Shortening Method," *IEEE Communications Letters,*, vol. 14, no. 10, pp. 963–965, October 2010.

List of Publications

In Journal

1. K. Deka, A. Rajesh, and P. K. Bora, "EXIT Chart Analysis of Puncturing for Non-Binary LDPC Codes", *IEEE Communications Letters*, vol.18, no.12, pp. 2089-2092, Dec. 2014.

In Conference Proceedings

1. K. Deka, A. Rajesh, and P. K. Bora, "On the Equivalence of the ACE and the EMD of a Cycle for the ACE Spectrum Constrained LDPC Codes", *8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), 2014,* pp. 67-71, 18-22 Aug. 2014, Bremen, Germany.

2. K. Deka, A. Rajesh, and P. K. Bora, "An Improved Puncturing Method for Rate-compatible LDPC Codes," *Twentieth National Conference on Communications (NCC), 2014*, Feb. 28 2014-March 2 2014, IIT Kanpur.

3. K. Deka, A. Rajesh, and P. K. Bora, "An Improved Technique to Find the Trapping Sets of the Irregular LDPC Codes", *National Conference on Communications (NCC), 2013,* 15-17 Feb. 2013, IIT Delhi.

4. K. Deka, A. Rajesh, and P. K. Bora, "Additional Check Node to Improve the Performance of LDPC Codes in the Error Floor Region", *National Conference on Communications (NCC), 2012,* 3-5 Feb. 2012, IIT Kharagpur.

5. K. Deka, A. Rajesh, and P. K. Bora, "Comparison of the Detrimental Effects of Trapping Sets in LDPC Codes", *IEEE International Conference on Communications (ICC), 2011,* 5-9 June 2011, Kyoto, Japan.