

https://dspace.org/

Symbol flipping decoding of non-binary low density parity check codes

Ullah, Waheed

2022
https://hdl.handle.net/10539/34397

Downloaded from a DSpace repository. https://dspace.org/

University of the Witwatersrand

Doctoral Thesis

Symbol Flipping Decoding of Non-binary Low Density Parity Check Codes

Author:
Waheed Ullah

A Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Center for Telecommunications Access and Services
School of Electrical and Information Engineering

April 2022



The financial assistance of the **Telkom** South Africa and **Sentech SOC LTD** towards this research is hereby acknowledged.

Declaration

I, **Waheed Ullah**, declare that this Thesis titled, "**Symbol Flipping Decoding of Non-binary Low Density Parity Check Codes**" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this Thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: Waheed Ullah

Date: October 2021

UNIVERSITY OF THE WITWATERSRAND

Abstract

Engineering and the Built Environment
School of Electrical and Information Engineering

Doctor of Philosophy

Symbol Flipping Decoding of Non-binary Low Density Parity Check Codes

by Waheed Ullah

Supervisors: Prof.Fambirai Takawira and Prof.Ling Cheng

NB-LDPC codes have shown better bit error performance than their binary counterpart but at the cost of an increased computational complexity. Furthermore, these decoding algorithms need to be implemented efficiently to fulfil the cost, time, power and bandwidth requirement. In addition, NB-LDPC codes are capable of correcting symbol wise error and are particularly suitable for burst noise channels where multiple bits are corrupted successively. The classical sum-product non-binary LDPC algorithm has a better performance than binary sum-product algorithm for short block length and low-rate codes. Using iterative belief propagation algorithms, NB-LDPC codes can be decoded in time linear to the block length. High decoding complexity is a main challenge in VLSI implementation of NB-LDPC codes. The research in this thesis is focused on proposing low complexity symbol flipping NB-LDPC decoding algorithms achieving the low complexity design of decoder to ensure optimum performance both at the waterfall and error floor regions. This thesis presents three new approaches to develop low complexity and improved performance symbol flipping NB-LDPC decoding algorithms. Firstly, a method of short-listing through voting has been introduced to update variable nodes which possibly have incorrect tentative decision and are most likely to be corrected. The proposed method plays key role in reducing complexity and enhances the error correction capability of the NB-LDPC decoder. Five improved algorithms are presented using the voting based short-listing of the variable nodes. These five algorithms are used with BPSK modulation. The proposed schemes give an appealing trade-off between complexity and performance in comparison to the

existing schemes. The method of short-listing by voting significantly lowers the decoding computational complexity. Secondly, to achieve the goal of high data rate using higher order QAM modulation, new joint iterative detection-decoding NB- LDPC algorithms are presented. Three new algorithms for improving the bit error rate performance of the non-binary LDPC decoder are presented. The first type is the symbol flipping decoding algorithm using a flipping function based on the channel reliability to identify the least reliable symbol position. The second and third type of joint algorithms are improvement to iterative joint detection-decoding algorithm by using the method of iterative hard decision based majority logic to select the new candidate symbol value. In these algorithms, if the predicted symbol value satisfies the check sum, then the value is declared as correct otherwise the value is adjusted and sent back to the QAM detector. The feedback value to the QAM detector is adjusted by using Euclidean distance between the current symbol and the newly selected symbol value. The complexity analysis shows that these proposed decoding algorithms help in reducing the overall latency. Thirdly, a layered based symbol flipping decoding algorithm is proposed for BPSK modulation. Layered decoding is the way to get efficient and high-throughput implementation of LDPC decoders. A new adaptive grouping of variable nodes in each iteration using bit reliability and majority voting of individual received symbol is presented for the class of symbol flipping decoding of NB-LDPC codes. Grouping of variable nodes and the subsequent decoding is based on individual symbol reliability and majority logic voting. Voting and bit reliability of symbols are used to form groups from high priority to low priority. A high priority group is considered to contain most un-reliable variable nodes and is decoded on priority while the second priority group contains variable nodes having second level of reliability. This novel hard decision adaptive group based decoding algorithm is different from the scheme used for soft iterative message passing algorithms. Though in this method, apparently the cumulative density function to forms groups increases complexity but it is computed once for each value of SNR but the overall decoding computational complexity is reduced as only the selected group is proceed. Therefore, this proposed scheme can be considered suitable for many applications.

I dedicate this to my mother

She was a wonderful lady: A simple, humble, kind and loving.

I dedicate this to my father

He always appreciated the value of education.

I dedicate this to my wife and kids

Nageen Naimat, Elizah Waheed, and Muhammad Umer Waheed. Their patience during that busy time is much appreciable.

Acknowledgements

Thanks to God Almighty for giving me the strength to complete my thesis and the opportunity to work under the supervision of Prof. Fambirai Takawira and Prof. Ling Cheng. They have great impact on my research career. My heartily gratitude to all my friends, colleagues and family as well. Also thanks to Prof. Fengfan Yang, my MS degree supervisor, for introducing University of the Witwatersrand and always ready to help during my this journey.

This would not have been possible without the unwavering faith and support from all of you. I am very grateful to my family, friends and all those who have stood by me and been the source of strength through this struggle time of my PhD research. I am specially thankful to my wife for taking such good care of me and my kids.

Very special thanks to the administrative staff of our school (Mumtaz Adam, Jeanne Do O Faustino, Moeniera Ismail and others) and faculty, for always being very cooperative and helpful.

Contents

Declaration i

Abstract ii

Dedication iv

Acknowledgements v

Contents vi

List of Figures ix

List of Tables x

Abbreviations xi

Publications xii

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 Research Questions and Contributions . . . 5

2 Non-binary LDPC Decoding Algorithms 7
- 2.1 Background . . . 7
  - 2.1.1 Construction of Non-binary LDPC Codes . . . 8
- 2.2 Decoding of NB-LDPC Codes . . . 10
  - 2.2.1 Preliminaries . . . 10
  - 2.2.2 Decoding NB-LDPC Codes with SPA . . . 13
  - 2.2.3 Decoding NB-LDPC Codes with FFT-QSPA . . . 14
  - 2.2.4 Log-Domain Sum-Product NB-LDPC Algorithm . . . 15
  - 2.2.5 Mixed Domain (FFT-LogSPA) . . . 16
  - 2.2.6 Min-Sum NB-LDPC Decoding Algorithms . . . 17
  - 2.2.7 Iterative Reliability Based NB-LDPC Algorithms . . . 17
- 2.3 Symbol Flipping Decoding Algorithms . . . 19
  - 2.3.1 Layered and Shuffled LDPC Coding Techniques . . . 24

3 Low Complexity Symbol Value Selection Decoding Algorithms 28
- 3.1 Introduction . . . 29
- 3.2 Symbol Flipping Decoding Algorithms . . . 33

3.2.1 Preliminaries . . . 33
3.2.2 Symbol Flipping Non-Binary LDPC Decoding Algorithms . 34
3.2.3 Candidate Symbols Short-listing By Voting . . . 38
3.3 Proposed Algorithms . . . 39
3.3.1 Voting Based Symbol Value Prediction Algorithm . . . 39
3.3.2 Voting Based Simplified Symbol Value Prediction Algorithm 41
3.3.3 Multiple Symbol Flipping Bit Reliability Based Decoding Algorithm . . . 43
3.3.4 Multiple Symbol Flipping Prediction Based Algorithm . . . 47
3.3.5 Voting Based B-MSFD Algorithm . . . 49
3.4 Complexity Analysis . . . 50
3.5 Results and Discussion . . . 53
3.6 Conclusion . . . 59

4 Joint Iterative Detection- Decoding Algorithms 61
4.1 Introduction . . . 62
4.2 Non-binary LDPC Codes . . . 66
4.2.1 NB-LDPC Codes Using BPSK . . . 67
4.2.2 NB-LDPC Codes Using QAM . . . 68
4.2.3 Non-Binary LDPC Decoders . . . 70
4.3 Proposed NB-LDPC Decoding Algorithms . . . 73
4.3.1 System Model . . . 73
4.3.2 Proposed Algorithm 1 . . . 76
4.3.2.1 Variable Node Selection by Voting . . . 77
4.3.2.2 QAM Based Flipping Function . . . 78
4.3.2.3 Symbol Value Selection . . . 79
4.3.2.4 Feedback to Detector . . . 81
4.3.3 Proposed Algorithm 2 . . . 83
4.3.4 Proposed Algorithm 3 . . . 86
4.4 Numerical Results and Analysis . . . 88
4.5 Complexity Analysis . . . 95
4.6 Conclusion . . . 97

5 Layered Symbol Flipping Decoding Algorithms 99
5.1 Introduction . . . 100
5.2 Background . . . 104
5.2.1 System Model . . . 104
5.2.2 Group Formation . . . 105
5.3 Proposed Adaptive Group Shuffled SF Decoding . . . 107
5.3.1 Adaptive Group Shuffling . . . 107
5.3.2 Bit Reliability Based SF Decoding . . . 111
5.3.3 Fixed Group SF Decoding . . . 114
5.4 Complexity Analysis . . . 117
5.5 Results and Performance Analysis . . . 120

5.6 Conclusion 128

6 Conclusion and Future Work **130**
6.1 Conclusion 130
6.2 Recommendations for Future Work 131

List of Figures

1.1 Typical LDPC Codes BER Performance Curve. 2

2.1 Figure (a)Parity check matrix. (b) Standard decoding. (c) Shuffle decoding with two groups 25

3.1 BER performance of NB LDPC code (204, 102) over $GF(16)$ 55
3.2 BER performance of NB LDPC code (120, 60) over $GF(128)$ 56
3.3 SER performance of NB LDPC code (120, 60) over $GF(128)$ 57
3.4 BER versus average number of iterations for NB LDPC code (120, 60) over $GF(128)$ 58
3.5 SNR versus average number of iterations for NB LDPC code (120, 60) over $GF(128)$ 60

4.1 System Model: NB-LDPC coded QAM modulation 74
4.2 16-QAM Gray coded constellation and the received symbol 75
4.3 Proposed Algorithm 1 Execution Cycle 76
4.4 Symbol Value Prediction 80
4.5 Proposed Algorithm 2 Execution Cycle 83
4.6 Proposed Algorithm 3 Execution Cycle 87
4.7 Comparison of ML and Quadrant QAM Detection 90
4.8 BER Performance of NB-LDPC Code (45, 2, 3) over $GF(8)$ 91
4.9 Average number of iterations verus BER for the (45, 2, 3) code over $GF(8)$ 92
4.10 BER Performance of NB-LDPC Code (45, 2, 3) over $GF(16)$ . . . 93
4.11 BER Performance of NB-LDPC Code(105, 2, 3) over $GF(8)$ 94
4.12 BER Performance of NB-LDPC Code(120, 4, 8) over $GF(8)$ 95

5.1 Flow chart of group decoding 110
5.2 Vertical Group Decoding 115
5.3 BER performance of NB-LDPC code (120, 3, 6) over $GF(128)$ . . . 122
5.4 BER performance of NB-LDPC code (105, 42) over $GF(8)$ 123
5.5 SER performance of NB-LDPC code(120, 4, 8) over $GF(128)$ 125
5.6 SNR versus average number of iterations for NB-LDPC code (120, 3, 6) over $GF(128)$ 126

List of Tables

3.1 Computational complexity per iteration for various NB-LDPC decoding algorithms 52
3.2 Memory required for decoding of NB-LDPC algorithms 53
3.3 Computational complexity per iteration of various algorithms at BER 10^{-4} FOR CODE(120, 60) over $GF(128)$ 59

4.1 Computational complexity of different NB-LDPC decoding algorithm per iteration 96

5.1 Computational complexity for various decoding algorithms 120
5.2 Numerical complexity of various algorithms at 5.5dB for code(120, 3, 6) over $GF(128)$ 128

Abbreviations

LDPC	**L**ow **P**arity **D**ensity **C**heck
NB-LDPC	**N**on-binary **L**ow **P**arity **D**ensity **C**heck
BPSK	**B**inary **P**hase **S**hift **K**eying
PCM	**P**arity **C**heck **M**atrix
AWGN	**A**dditive **W**hite **G**aussian **N**oise
SNR	**S**ignal-to-**N**oise **R**atio
SER	**S**ymbol **E**rror **R**ate
BER	**B**it **E**rror **R**ate
FEC	**F**orward **E**rror **C**orrection
BP	**B**elief **P**ropagation
GS	**G**roup-**S**huffled
QAM	**Q**uadrature **A**Amplitude **M**odulation
WIMAX	**W**orldwide **I**nterpolation for **M**icrowave **A**ccess
GF	**G**alois **F**ield
SF	**S**ymbol **F**lipping
wBRB	**W**eighted **B**it **R**eliability **B**ased
SF	**S**ymbol **F**lipping
SPA	**S**um **P**roduct **A**lgorithm
IHRB	**I**terative **H**ard **R**eliability **B**ased
ISRB	**I**terative **S**oft **R**eliability **B**ased
MLgD	**M**ajority **L**ogic **D**ecoding
EXI	**E**xtrinsic **I**nformation **S**um
VN	**V**ariable **N**ode
CN	**C**heck **N**ode
MSFD	**M**ultiple **S**ymbol **F**lipping **D**ecoding

List of Publications

[1]. **Waheed Ullah**, Ling Cheng, Fambirai Takawira. "Layered Decoding of Multiple Symbol Flipping Non-binary LDPC Codes" (To be submitted to an IEEE Journal).

[2]. **Waheed Ullah**, Ling Cheng, Fambirai Takawira. "Predictive Syndrome Based Joint Iterative Detection-Decoding Algorithm for Non-Binary LDPC Codes". IEEE Access, 22 Feb, 2021. DOI: 10.1109/ACCESS.2021.3060806

[3]. **Waheed Ullah**, Ling Cheng, Fambirai Takawira. "Low Complexity Bit Reliability and Predication Based Symbol Value Selection Decoding Algorithms for Non-Binary LDPC Codes." IEEE Access, vol. 8, pp. 142691–142703, 2020.

[4]. **Waheed Ullah**, Ling Cheng, Fambirai Takawira. "Prediction and Voting Based Symbol Flipping Non-Binary LDPC Decoding Algorithms." IEEE 31st Annual International Symposium Personal, Indoor and Mobile Radio Communications (PIMRC 2020), London, UK, Virtual , Sep 2020.

[5]. Moein Sarvaghad-Moghaddam, **Waheed Ullah**, Nalin Dushantha K. Jayakody, Safien. "A New Construction of High Performance LDPC Matrices for Mobile Networks". Sensors, vol. 20, no. 8, p. 2300, April 2020.

CHAPTER 1

Introduction

1.1 Introduction

Channel coding is an important and integral part of transmission and storage devices of digital information. All communication systems depend on how precisely receiver can assess the transmitted signal. Perfect estimation is possible only in the absence of noise, but noise is always present in communication media and systems . The superimposed noise on transmitted signal limits the receiver capability to correctly determine the intended signal and therefore limits the rate of transmitted information. The noise arises from different sources and can be classified as human created or naturally present. To ensure a desired level of reliability of the information and communication systems, data is always encoded before transmission for wireless channel or wired channel and secondly for the storage systems like magnetic or optical recording devices. Claude Shannon laid the foundation for error correction coding by presenting his famous theorem of information [1] and revolutionaries the idea of reliable transmission of information over unreliable (noisy) channels. No major research work was done to achieve the limits set by Shannon for decades before the invention of Turbo [2] code and the rediscovery of the LDPC codes [3] which revolutionized the field of error correction coding. Davey and Mackay showed in [4] that near Shannon limit performance can be achieved for a very long sparse LDPC codes which set the path for future reliable communication. LDPC codes have now been included in many standards like second generation terrestrial digital video broadcasting (DVB-T2), second generation satellite digital video broadcasting (DVB-S2) and IEEE 802.16e mobile

WiMAX [5]. The Consultative Committee for Space Data Systems (CCSDS) [6] has also included LDPC codes in its recommendation for deep space and near earth communication. The important parameter to be considered for inclusion of the non-binary LDPC codes into the standards of communication systems are the coding gain, error floor, implementation complexity and the delay in encoding and decoding. The performance of LDPC codes are evaluated mainly at three regions; at low SNR, water fall [7, 8] and at high SNR [9, 10] as shown in Figure 1.1.

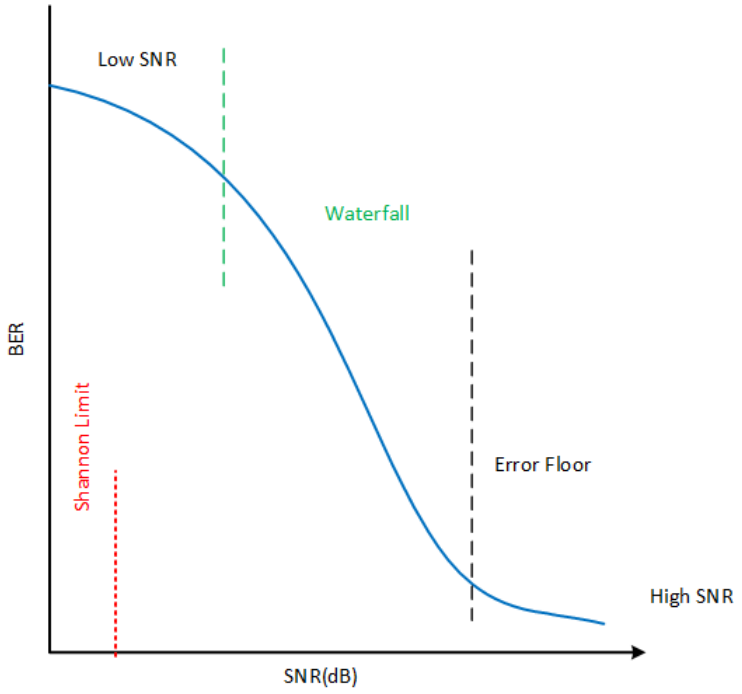


FIGURE 1.1: Typical LDPC Codes BER Performance Curve.

In the sum-product NB LDPC decoding [11, 12], each message passing through each edge of a Tanner graph [13] is a vector of length equal to the field size q over Galois field $GF(q)$. As the field size q increases, the complexity of the NB LDPC decoding also increases. Symbol flipping decoding algorithms [14–18] are best among the low complexity decoding algorithms but the bit error rate(BER) performance is still far from the belief propagation message passing algorithms. The recently developed symbol flipping algorithms [19, 20] perform well but with an increased computational complexity. To reduce the complexity and maintain

the BER performance, new symbol flipping decoding algorithms [21, 22] are developed. The two symbol flipping decoding algorithms [20] perform well but they have high complexity due to the prediction of q candidate symbol for each of the received symbol sequence and correct one symbol per iteration. The D-SFDP algorithm in [20] adds the hard reliability as well as the Galois field structure to the flipping function to correctly identify the position of the least reliable symbol and predicts the candidate symbol value while P-SFDP algorithm in [20] adds reliability information using plurality logic. These two algorithms predict the candidate symbol value by taking into accounts the information before flipping and as well as after flipping.

Reliability based majority logic decoding algorithms offer low complexity as they send only the most reliable field message and are more similar to message passing decoding algorithms. In the iterative majority logic decoding (MLgD) algorithm for non-binary LDPC codes, each symbol is iteratively updated by the extrinsic information-sums (EXIs) with the most reliable field element along each edge of the Tanner graph. The iterative reliability based hard (IHRB) and soft (ISRB) majority logic decoding (MLgD) algorithms [23] have lower complexity but suffer from performance degradation. Some performance improvements to IHRB-MLgD and ISRB-MLgD algorithms have been presented in [24–27]. To improve the performance of IHRB-MLgD algorithm, an enhanced (E-IHRB-MLgD) algorithm is presented in [25] by introducing soft reliability at the initialization and re-computing the extrinsic-information. The enhanced IHRB algorithm has the same complexity as that of IHRB except at the initialization but it has an improved performance which is close to ISRB algorithm. The majority logic decoding algorithms are particularly suitable for parity check matrices with high column weight, constructed based on finite fields [28, 29] and finite geometries [30]. These algorithms suffer from performance loss for codes with small column weight. An improved performance bit reliability based algorithms are presented in [31]. These algorithms introduce reliability updates in terms of the bit rather than symbols and are termed as bit reliability based (BRB) algorithms. The BRB algorithms require integer and Galois field operations only, which reduces the computational

complexity. The performance of the BRB algorithm is improved by multiplying the extrinsic information-sum by Hamming distance based weighted coefficients. This weighted BRB (wBRB) algorithm shows improved performance over ISRB and BRB and does not suffer from error floor for parity check matrices with small column weight. A full bit-reliability based decoder [32] is the binary form of weighted BRB as this algorithm uses the binary representation of the non-zero entries of the parity check matrix to facilitate hardware implementation. The c̈omplexity versus performance" tradeoff is the main criterion in the field of communication when it comes to the selection of suitable decoding algorithms. This research focuses on the development of a decoding techniques that iteratively utilizes the channel soft information to yield a better error correction performance at reduced computational complexity.

This PhD research is focused on the low complexity solutions for symbol flipping (SF) decoding algorithms. The SF decoding algorithms presented in this research work in two parts; the first part identifies the positions of unreliable symbols taking into account the reliability information before flipping and the second part choose the best candidate symbols taking into account the predicted reliability for those unreliable positions. The proposed scheme reduces complexity and enhances the convergence speed. Also, the soft reliability information are updated iteratively for improving further the BER performance.

High rate data transmission is another demand of present and future communication. To achieve reliability for high rate data transmission, new symbol flipping decoding algorithm is proposed. This SF algorithm is based on Euclidean distance based feedback and works on principle of joint iterative detection and decoding using higher order quadrature amplitude modulation (QAM). Improved performance iterative hard reliability based joint detection-decoding algorithms are proposed. This research is further extended to layered and shuffle symbol flipping decoding algorithms for further improvement in the tradeoff between complexity and performance.

This thesis is structured into chapters that individually address the research questions in Section 1.2 below. There is some overlap in the introductions to each

chapter to provide context for the work within. In Chapter 2, a broad background of NB-LDPC decoding algorithms is provided in order to supplement the readers understanding of subsequent chapters, which are self contained due to the fact that they are based on publications. Chapter 6 provides a summary and conclusions of the findings present in this thesis and suggests some future research direction.

1.2 Research Questions and Contributions

The main challenge for hardware implementation of non-binary LDPC decoding is the high computational complexity and large memory requirement. In the current literature, the mostly asked question is "How to reduce the complexity of non-binary LDPC decoder and make efficient implementation of the non-binary LDPC decoder without compromising performance?" In order to address this problem, the following research questions are posed in this PhD thesis.

1. How correctly the received symbol can be determined with most probable erroneous tentative decision and is likely to be corrected? Once the least reliable symbol position is identified, then how the most reliable decision can be made to replace the incorrect symbol value with correct value?

The symbol flipping decoding algorithms update, iteratively, the hard decision received vector to search for the valid codeword in the vector space of Galois field. The selection criterion for least reliable symbol positions is based on the information from the failed checks and the reliability information from the Galois field structure as well as from the received channel soft information. To choose the correct value for the candidate symbol, two methods are used. The first method is based on the prediction of the error symbol from the set of Galois field symbols which maximizes an objective function. In the second method, individual bits are flipped based on the reliability information obtained from the channel. To achieve the reduced complexity,a method of short-listing least reliable variable nodes which are most likely to be corrected, is used. The detail description of these methods are given in detail in Chapter 3.

2. How the symbol flipping concept can be exploited for the iterative joint detection-decoding using higher order modulation? Secondly how effectively the reliability information from the received sequence are processed for feedback to detector?

Symbol flipping decoding algorithm based on the channel reliability can be used to identify the least reliable symbol position. In this method, if the predicted symbol value satisfies the check sum, then the value is declared as correct otherwise the value is adjusted and sent back to the QAM detector. The feedback value to the QAM detector is adjusted by using Euclidean distance between the current symbol and the newly selected symbol value. Also, an improvement to iterative joint detection-decoding algorithm is proposed by using the method of iterative hard decision based majority logic decoding to select the new candidate symbol value. To reduce computational cost, majority voting scheme is used to short list the symbols with erroneous tentative decision and all the computation are carried out only for the short listed least reliable symbols which significantly lowers the processing complexity. Further detail is given in Chapter 4.

3. Layered technique has proven reduced complexity for message passing LDPC decoding. How effectively this technique can be implemented to reduce further the symbol flipping algorithms?

A layered scheme can be applied to symbol flipping non-binary LDPC decoding algorithms to achieve the goal of low latency and high reliability. The parity check matrix is divided column-wise into layers having equal number of variable nodes. Only those layers are considered for processing which contains erroneous variable nodes while layers having no erroneous variable node are considered reliable and are excluded from computation to find flipped value of the candidate symbol. This hard decision adaptive layered decoding algorithm is different from the scheme used for soft iterative message passing algorithms. An adaptive layer approach can be more effective for using by groups having variable nodes from least reliable to most reliable. Voting and bit reliability information can be utilised to form such groups. The layered schemes are explained in Chapter 5.

CHAPTER 2

Non-binary LDPC Decoding Algorithms

2.1 Background

Davey and Mackay [3] presented the non-binary LDPC codes by extending the order of Galois field $GF(q)$ with $q > 2$. Comparison of binary and non-binary LDPC codes shows that non-binary LDPC codes outperform their binary counterpart, particularly for short length codes. In particular, good LDPC codes over $GF(q)$ are defined by ultra-sparse matrices and thereby easier to decode, and they empirically show better performance around the waterfall region than their binary counterparts, especially for small code-word lengths. Dave and Mackay [3] used Monte Carlo methods to simulate the behaviour of the decoding algorithm applied to an infinite LDP code in an effort to investigate the effects of changes in field order, code construction and noise level on the decoding performance. These Monte Carlo results have been used to design better codes for practical decoding. Despite the bit error performance advantage of the NB-LDPC codes, but their decoding complexity over $GF(q > 2)$ is very high. For the received symbol sequence $\boldsymbol{y} \in GF(q)$, each element of the received sequence during the decoding need to be computed, stored in memory and passed between the variable (bit) nodes and check nodes. The check node processing is much more complicated than binary LDPC code. The major drawback of the NB-LDPC codes are the large memory requirement and check node processing complexity. At check node, the computation is considered as the convolution of the messages [11]. This convolution can be represented in frequency domain by applying fast Fourier transform (FFT) to reduce the computational cost. This modification of the belief propagation decoding

algorithm defined over high order Galois field has the computational complexity in the order of $qlog_2(q)$.

2.1.1 Construction of Non-binary LDPC Codes

The performance of LDPC codes specifically under iterative belief propagation algorithms is particularly dependent on the distribution of short cycles and generally on the structure of the code's Tanner graph [33],[34]. The short cycles play an important role particularly in the error floor of low density parity check codes, where they causes to create trapping sets [35]. Therefore, it is of interest to have an approximations for the number of cycles of particular length in a given graph of parity check matrix. Regarding this, it is also of interest to get the distribution of short cycles of a particular length in an ensemble of LDPC codes. The knowledge of cycles distribution in a given Tanner graph can help in the analysis and design of LDPC codes.

To get short cycles free parity check matrix (PCM), an important constraint on parity check matrix is that any two rows or columns of H can have at most one position in common and is called Row-Column constraint. This constraint on PCM H ensures that the Tanner graph of an LDPC code is at least free of four cycles. Based on the broad construction methods, LDPC codes are categorized into two types: random-like LDPC codes which are based on computer search such as progressive edge growth [33] and photograph based methods [36] while structured LDPC codes are based on algebraic techniques [37] such as finite geometries [30] combinatorial structures [38],[39] and finite fields [28],[40]. The QC-LDPC [41],[29] codes, more suitable for hardware implementation and also known as architecture-aware codes, are one of the well studied LDPC codes. QC-LDPC codes have their parity- check matrices with a special structure which facilitates the implementations of an LDPC encoder and decoder. In comparison to random like codes, the generator matrices of QC-LDPC codes are quasi cyclic and therefore, the encoder implementation have linear complexity.

Moreover, NB-LDPC codes can be categorized broadly in two classes: regular LDPC codes and irregular LDPC codes. $A((d_v, d_c)$ regular LDPC code is the one having parity check matrix H with constant column weight d_v and constant row weight d_c such that $md_c = nd_v$. However if the row and column weight of the parity check matrices are not constant then it is called irregular LDPC code. It has been shown that irregular LDPC codes [42, 43] perform better than regular LDPC code. In irregular LDPC codes both the variable node (VN) and check node (CN) degrees are allowed to vary and if properly designed then these codes can approach the Shannon Capacity limit. The randomly constructed regular and irregular LDPC codes perform well but in practice, it is very hard to implement on hardware as require more memory and computational complexity. To overcome this problem, well systematically designed LDPC codes called quasi cyclic (QC) are introduced which have performance close to channel capacity and have low memory requirement. QC LDPC codes are included in the CCSDS [6] recommendation for deep space and near earth Telemetry. They have included those codes which are represented by $b \times b$ circulant matrices as base matrix.

The bit error rate performance of the LDPC codes depends also on the well-designed parity check matrix (PCM) H which need not be full rank but in such case the rate of that code is bounded as $R \leq 1 - \frac{m}{n} = 1 - \frac{d_v}{d_c}$. In case of full rank, the inequality is replaced with equality. LDPC codes have three important parameters: Codeword length, message length k and its number of parity bits $m = n - k$. There are few methods to create NB-LDPC parity check matrices [44] but mostly these codes are constructed by taking known binary LDPC parity check matrix and the non-zero entries are replaced with the randomly generated Galois field elements. Similar to binary LDPC codes, NB-LDPC codes can be classified as random like codes constructed by computer under certain design rules and criteria, secondly structured NB-LDPC codes constructed on the bases of algebraic methods, finite field or finite geometry. An array dispersion method to construct good NB-LDPC code was presented by Shu Lin in [44]. All the methods and techniques developed for binary LDPC codes can be applied to construct NB-LDPC codes over GF(q) with some modifications. Keeping in view the performance and

implementation complexity, some specific construction techniques are devised for NB-LDPC codes. Some popular methods of construction are Euclidean Geometry (EG) and Projective Geometry (PG) based construction while the later method is mostly used to construct quasi cyclic (QC) codes.
The important drawback of initial construction of Gallagar and Mackay LDPC codes, is the lack of good structure to ensure low complexity encoding and decoding for hardware implementation. Encoding is performed via Gauss-Jordan elimination by converting H to the lower or upper triangular matrix of the form $[p^T\ I]$ from which we can get the systematic generator matrix $G = [P\ I]$. The issue performing encoding with G is that it is not sparse, hence these codes have high complexity. A relatively efficient method of encoding for H was proposed by Richardson and Urbanke where the H matrix is divided into sub matrices.

Based on the construction methods of LDPC codes, they are classified as random and structured codes. Random LDPC codes are designed by exhaustive computer search based on certain criteria, while structured codes follows certain design methods like quasi cyclic LDPC code [30],[28]. Quasi-cyclic LDPC codes are specially advantageous due to their simple encoders and decoder structure [45]. The algebraic construction of non binary quasi cyclic (NB-QC) LDPC codes are given in [40]. In [40] finite fields approach is used to generate a parity-check matrix with a girth of at least 6.

2.2 Decoding of NB-LDPC Codes

2.2.1 Preliminaries

An LDPC code which is fully represented by a Tanner graph, also aide in the description of decoding algorithm. A tanner graph has two types of nodes– variable (bit) node (VN) and check node (CN). Let $GF(q)$ be the Galois field with q elements. Consider a NB LDPC code $\mathcal{C}$ of length n with a regular parity check matrix H, defined over the *GF(q)*, with m as number of rows and n as number of columns such that each row of H has a constant weight of d_c and each column

of H has a constant weight of d_v. Each element $h_{i,j}(1 \leq i \leq m\ ,\ 1 \leq j \leq n)$ of H is an element over the Galois field $GF(q)$ where $q = 2^r$. The non-zero entries of $H(h_{i,j} \neq 0)$ in the Tanner graph show the i^{th} check node connected to the j^{th} variable node. For the column weight d_v and the row weight d_c, the Tanner graph edge connections are shown as $M(j) = \{i : 1 \leq i \leq m\ ,\ h_{i,j} \neq 0\}$ and $N(i) = \{j : 1 \leq j \leq n, h_{i,j} \neq 0\}$ and for regular matrix $\frac{m}{n} = \frac{d_v}{d_c}$.

Let $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_j, \ldots, \boldsymbol{c}_n)$ be a codeword in $\mathcal{C}$ and the binary representation of j^{th} symbol in $\boldsymbol{c}$ is $\boldsymbol{c}_j$= $(c_{j,1}, c_{j,2}, \ldots, c_{j,t}, \ldots, c_{j,r}), 1 \leq t \leq r$. This binary sequence is modulated with binary phase shift keying (BPSK), where 1 is modulated as +1 and 0 is modulated as -1. The BPSK modulated j^{th} symbol $\boldsymbol{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,t}, \ldots, x_{j,r})$ is transmitted over additive white Gaussian noise (AWGN) channel. The received binary sequence $\boldsymbol{y}_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,t}, \ldots, y_{j,n})$ is obtained from the transmitted sequence $\boldsymbol{x}_j$ after adding the additive white Gaussian channel noise $\boldsymbol{n} \rightarrow \mathcal{N}(0, \sigma^2)$ with zero mean and two sided power spectral density $N_0/2$ as $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j$. The hard decision binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is given by:

$$z_{j,t} = \begin{cases} 1 & \text{if } y_{j,t} \geq 0 \\ 0 & \text{else} \end{cases} \tag{2.1}$$

The hard decision binary sequence $z_{j,t}$ is mapped to an element in $GF(q)$ to obtain the symbol $\boldsymbol{z}_j$. Let $\boldsymbol{s} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_i, \ldots, \boldsymbol{s}_m) = H\boldsymbol{z}^T$ is the syndrome vector, then the i^{th} syndrome of the j^{th} hard decision symbol sequence can be defined for $h_{i,j} \neq 0$:

$$\boldsymbol{s}_i = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j \tag{2.2}$$

Here it is important to mention that the value of $\boldsymbol{s}_i$ is also an element in $GF(q)$. If $H\boldsymbol{z}^T \neq \boldsymbol{0}_{(1,n-k)}$, which means that some of the variable nodes connected to check node are in error.

Let $GF(q)$= $(\alpha_1, \alpha_2, \ldots, \alpha_l, \ldots, \alpha_q)$, then for the j^{th} received symbol $\boldsymbol{z}_j$, the priori probabilities $(\gamma_j^{\alpha_1}, \gamma_j^{\alpha_2}, \ldots, \gamma_j^{\alpha_l}, \ldots, \gamma_j^{\alpha_q})$ are equal to $\{\alpha_1, \alpha_2, \ldots, \alpha_l, \ldots, \alpha_q\}$ respectively such as $(\gamma_j^{\alpha_1}, \gamma_j^{\alpha_2}, \ldots, \gamma_j^{\alpha_l}, \ldots, \gamma_j^{\alpha_q})$=1 holds. The priori probability for

the symbol $\alpha_l \in GF(q)$ is given below.

For $1 \leq j \leq n, 1 \leq l \leq q, 1 \leq t \leq r$, the probability of the received binary sequence $z_{j,t}$ to be 1 for the channel out put $y_{j,t}$ is:

$$\gamma_{j,t}^{1} = \frac{1}{1 + \exp(\frac{4y_{j,t}}{N_0})} \tag{2.3}$$

and the probability for the received binary sequence $z_{j,t}$ to be 0 is $\gamma_{j,t}^{0} = 1 - \gamma_{j,t}^{1}$. Now the probability of $\gamma_{j,t}^{\alpha_l}$ being the element $\alpha_l \in GF(q)$ for the received symbol $\boldsymbol{z}_j$ corresponding to the channel output $\boldsymbol{y}_j$, is given by:

$$\gamma_{j}^{\alpha_l} = \prod_{t=1}^{r} \gamma_{j,t}^{\alpha_{l,t}}. \tag{2.4}$$

This probability of the j^{th} symbol is then used as priori information at the initialization of the QSPA. For an LDPC code defined over $GF(q = 2^r)$ where each received symbol α_l must be one of the symbols in $GF(q)$, the check node processing becomes more complicated as there are more possible values of the q to satisfy the parity check equations. For each element defined over $GF(q)$ of the parity check matrix $H(h_{i,j} \neq 0)$, there are q probabilities associated with it as compared to its binary counterpart which has only two probabilities for 0 and 1. Therefore, q messages need to be computed, stored in the memory, and exchanged between the check and variable nodes during this iterative process.

The major bottleneck of the sum-product NB-LDPC decoding algorithm is the check node computational complexity and the large memory required to store the q-tuple messages. The posteriori probability $\gamma_{i,j}^{\alpha_l}$ (message sent to check node from variable node) is given by:

$$\hat{\gamma}_{j}^{\alpha_l} = \sum_{\boldsymbol{z}_j \in GF(q)} Pr(\boldsymbol{s}_i = 0 | \boldsymbol{z}_j = \alpha_l) \prod_{j' \in N(i) \backslash j} q_{j',t}^{\alpha_l} \tag{2.5}$$

The computed values of $\gamma_{i,j}^{\alpha_l}$ for $\alpha_l \in GF(q)$ for $i \in M(j), j \in N(i)$ are then used to update the extrinsic information $q_{i,j}^{\alpha_l}$ to be used in the next iteration and is given as follows:

$$q_{i,j}^{\alpha_l} = \lambda \gamma_j^{\alpha_l} \prod_{i' \in M(j) \backslash i} \hat{\gamma}_{i',j}^{\alpha_l} \tag{2.6}$$

Where the scaling factor λ is chosen such that the sum of all probabilities equal to 1, i.e. $q_{i,j}^{\alpha_1} + q_{i,j}^{\alpha_2} + \ldots + q_{i,j}^{\alpha_q}$=1. The above mentioned equations serve as the check and variable node processing rule for message passing decoder.

2.2.2 Decoding NB-LDPC Codes with SPA

The NB-LDPC sum-product decoding algorithm [3, 46] is summarized as follows:

Sum Product NB-LDPC Algorithm

1. Initialization: For $1 \leq j \leq n, 1 \leq m \leq, 1 \leq l \leq q$
   Received symbol sequence: $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_j, \ldots, \boldsymbol{y}_n)$.
   Hard decision symbol sequence: $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$.
   Priori probabilities: $\gamma_j = Pr(\boldsymbol{z}_j = \alpha_l | \boldsymbol{y}_j)$.
   Extrinsic probabilities : $q_{i,j}^{\alpha_l} = \gamma_j^{\alpha_l}$.
2. Set $k = 1$ to I_{max}.
3. Compute $\boldsymbol{s}^k = \boldsymbol{z}^k H^T$, if $\boldsymbol{s}^k = \boldsymbol{0}$ or k=I_{max}, stop decoding.
4. Compute $\hat{\gamma}_{i,j}^{\alpha_l,k}$ using equation (2.5).
5. Compute $q_{i,j}^{\alpha_l,k}$ using equation (2.6).
6. Estimate the symbol sequence as $\boldsymbol{z}^k = \underset{\alpha_l}{\operatorname{argmax}} \hat{\gamma}_{i,j}^{\alpha_l,k}$.
7. $k = k + 1$, go to step 3.

Each j^{th} symbol of the received sequence $\boldsymbol{y}_j$ has associated likelihood in the order of $GF(q)$, as the order of $GF(q)$ increases the complexity increase in order of $O(q^2)$.

Writing the field elements of $GF(q)$ as power of primitive element α, $GF(q) = (\alpha^{-\infty}, \alpha^0, \alpha^1, \ldots, \alpha^{q-2})$ where α is the primitive element. When two field elements of $GF(q)$ are multiplied, it is equivalent to shift the powers of the primitive elements cyclically and then get the resultant field element. This cyclic shifts are known as permutation.

2.2.3 Decoding NB-LDPC Codes with FFT-QSPA

The main complexity of the q-ary sum-product algorithm (QSPA) lies in the check node processing. The check node computation is like convolution which can be implemented more easily in the frequency domain as term by term multiplication. In the frequency domain decoding [11], check node message at the k^{th} iteration is computed as:

$$\gamma_{i,j}^{k} = F^{-1}\Big(\prod_{j' \in N(i)\setminus j} F(q_{i,j}^{k})\Big) \tag{2.7}$$

here F and F^{-1} are the Fourier and the inverse Fourier transform and $\prod$ is the term by term multiplication. When the probability mass function is defined over the $GF(q)$, then the Fourier transform is the two point, r-dimensional message vector. Both the Fourier and inverse Fourier transform are 2×2 matrices given by:

$$F = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } \; F^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.8}$$

The FFT-QSPA decoding algorithm presented by Davey and Mackay, can be used over the Galois field $GF(q)$ where q is power of 2. Though it can be generalized to the prime power of q as presented in [11] but since q is commonly used as power of 2 for practical applications, therefore, the generalized FFT-QSPA is of not much consideration. Equation (2.8) is known as binary Fourier transform or the Walsh Hadamard transform. The Fast Fourier Transform based decoding reduces complexity by $O(q\ log\ q)$.

2.2.4 Log-Domain Sum-Product NB-LDPC Algorithm

A simplified version of the QSPA decoding called LogSPA is presented in [12] which uses additions, subtractions, and look-up tables only. The LogSPA decoder requires a fairly accurate knowledge of the signal-to-noise ratio (SNR) at the input of the decoder but less sensitive to quantization noise. The computational complexity for this LogSPA scales as $O(q^2)$. Denoting the $GF_0(q) = (\alpha_2, \alpha_3, \ldots, \alpha_l, \ldots, \alpha_q)$ without the zero element, then the log likelihood ration (LLR) of nonzero elements $\alpha \in GF(q)$ is written as:

$$L(\alpha_j) = L(\alpha_j = \alpha_2), L(\alpha_j = \alpha_3), \ldots, L(\alpha_j = \alpha_l), \ldots, L(\alpha_j = \alpha_q) \tag{2.9}$$

$$L(\alpha_j = \alpha_l) = \ln \frac{Pr(\alpha_j = \alpha_l)}{Pr(\alpha_j = 0)} \tag{2.10}$$

The priority probability of the j^{th} received symbol is given by:

$$\gamma_j = \frac{2}{\sigma^2} \sum_{j \in N(i)} \boldsymbol{y}_{j,t} \tag{2.11}$$

and the posteriori information is updated as follows:

$$\hat{\gamma}_j^k = \gamma_j + \sum_{j \in N(i) \backslash j'} q_{i,j'}^k \tag{2.12}$$

The check node message update for the next iteration is computed as follows:

$$q_{i,j}^{k+1} = \gamma_j + \sum_{i \in M(j) \backslash i'} q_{i',j}^k \tag{2.13}$$

NB-LDPC LogSPA Decoding Algorithm

1. Initialization: For $1 \leq j \leq n, 1 \leq m \leq, 1 \leq l \leq q$

Received symbol sequence: $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_j, \ldots, \boldsymbol{y}_n)$.

Hard decision symbol sequence: $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$.

Priori probabilities: $\gamma_j = \frac{2}{\sigma^2} \sum_{j \in N(i)} \boldsymbol{y}_{j,t}$.

Extrinsic probabilities : $q_{i,j} = \gamma_j$.

2. Set $k = 1$ to I_{max}.
3. Compute $\boldsymbol{s}^k = \boldsymbol{z}^k H^T$, if $\boldsymbol{s}^k = \boldsymbol{0}$ or k=I_{max}, stop decoding.
4. Compute $\hat{\gamma}_j^k$ using equation (2.12).
5. Update the check node message $q_{i,j}^{k+1}$ by using (2.13).
6. Estimate the symbol sequence as $\boldsymbol{z}^k = \underset{\alpha_l}{\operatorname{argmax}} \hat{\gamma}_{i,j}^{\alpha_l,k}$.
7. $k = k + 1$, go to step 3.

2.2.5 Mixed Domain (FFT-LogSPA)

A new mixed domain (FFT-LogSPA) was proposed which perform Fourier transform in probability domain and carries out the check node processing in log-domain. The main drawback is that it needs conversion between the two domains and also lookup tables. In the paper [47] a low complexity simplified soft distance (SSD) decoding algorithm for NB-LDPC codes is presented, which is an adaption for NB-LDPC codes of a previously reported SSD algorithm for binary LDPC codes [48, 49], but now operates in the transform domain, called the Fast Fourier Transform Simplified Soft Distance (FFT-SSD) decoding algorithm. This decoding algorithm uses squared Euclidean distance as the metric, which offers as a great advantage that it does not require knowledge of the signal-to-noise ratio of the received signal, and requires only additions, subtractions and look-up tables. Large lookup tables and conversion between the two domains are the main drawback of this algorithm.

2.2.6 Min-Sum NB-LDPC Decoding Algorithms

The QSPA is the optimal iterative decoding at the price of an increased computational complexity and noise variance dependence. The Min-Sum NB-LDPC decoding algorithm [50, 51] is a class of sub-optimal iterative decoding with less complexity than the Sum-Product decoding, and is independent of the noise estimation errors. The Min-Sum NB-LDPC decoding is considered sub-optimal due to the overestimation of extrinsic messages computed during check-node processing.

Other simplified approaches were developed to approximate the QSPA, such as the extended Min-Sum algorithm (EMSA), which applies message truncation and sorting to further reduce complexity and memory requirements [52][53]. Its main advantage in comparison to the Min-Sum NB-LDPC decoding is that it is computationally stable and does not need to normalize the exchanged messages. The second widely used decoding algorithm is the min-max (MMA) [54], which replaces the sum operations in the check node (CN) processing by max operations. The complexity of NB-LDPC decoders is still too high for practical applications, especially for the check node (CN) processing, which limits the maximum achievable throughput. Although a great effort has been made in the recent literature to overcome this disadvantage, the proposed decoders are still not ready for high-speed implementations over higher $GF(q)$. In the papers [55][56], a simplified trellis min-max algorithms are proposed, where the CN messages are computed in a parallel way using only the most reliable information. This decoder increases hardware efficiency in comparison to the existing solutions for the same code.

2.2.7 Iterative Reliability Based NB-LDPC Algorithms

Reliability based majority logic decoding(MLgD) algorithms are similar to message passing decoding but they send only the most reliable field message and that is why they have reduced computational and implementation complexity. Zhao et al [57] presented the iterative MLgD decoding algorithm for non-binary LDPC where each symbol is iteratively updated by the extrinsic information sums (EXIs) with the

most reliable field element along each edge of the Tanner graph. Using a plurality voting scheme, a symbol reliability based message passing decoding algorithm (SRBMP) over $GF(q)$ is presented in [58]. The SRBMP algorithm requires Galois field and integer operations only. This SRBMP algorithm is improved by MV-SF algorithm presented in [59]. The MV-SF algorithm performs well but an increased complexity due to test vector.

The iterative reliability based hard (IHRB) and soft (ISRB) MLgD algorithms [23] have lower complexity but at the cost of poor performance. The algorithms in [23] are further improved by different techniques to make a better tradeoff between complexity and bit error performance by proposing several modifications [24–27]. Both the iterative hard and soft reliability based MLgD algorithm and its improved versions achieve better bit error rate performance while they still have low complexity. These algorithms perform well for NB-LDPC codes with large column weight since large number of check nodes (CNs) messages are required to update the reliability of the variable nodes (VNs) reliability in each iteration.
A bit reliability based NB-LDPC decoding algorithms is presented in [31]. This algorithm updates the reliability information in terms of bit rather than symbols. The bit reliability based decoding method for non-binary LDPC is more efficient than others and is termed as weighted bit reliability based (wBRB) algorithm. The wBRB algorithm passes only one Galois field element as its reliability along each edge of a tanner graph in the same manner as traditional MLgD base decoding algorithms. The performance of wBRB algorithm is close to q-ary SPA (QSPA) for LDPC codes with small column weights. Also this algorithm requires the integer and Galois field operation only. To further enhance the wBRB decoding algorithms, a full bit-reliability base decoder is proposed in [32] based on the binary matrix representation of the non-zero entries of the parity check matrix and has achieved better performance.

2.3 Symbol Flipping Decoding Algorithms

Symbol flipping decoding algorithms have low complexity as compared to the bit reliability based decoding algorithms [31, 32] and the q-ary belief propagation [3, 11] message passing algorithms but at the cost of reduced bit error rate performance. In the symbol flipping decoding algorithm presented in [14], the least reliable position is determined by majority decision, while the flipped symbol value is computed from the reliability of the received bits. The weighted algorithm B (wt.Algo B) in [15] introduces the binary Hamming distance and plurality logic to improve performance. The parallel symbol flipping decoding (PSFD) algorithm [16] uses majority voting to get better flipping decision but this algorithm performs better only if the parity check matrix has a large column weight. The PSFD algorithm is improved further by using multiple votes in [18] and performs better even for the LDPC parity check matrix with small column weight. The non-binary LDPC decoder based on symbol flipping with multiple votes (MV-SF) [59] performs better than MV-PSFD but at the cost of an increased complexity and memory size especially for the higher order Galois field.

The other recently developed symbol flipping algorithms (D-SFDP and P-SFDP) [20] offer better performance than most of the existing symbol flipping algorithms, but a major disadvantage is the exhaustive and computationally complex prediction mechanism. The complexity of these algorithms [20] depend on the size of the Galois field order q and length n of a codeword. A decision-symbol reliability-based SFD (DRB-SFD) algorithm [60] is aimed to show better performance in applications like data storage based on NAND flash memory. In [60] is focused on an algorithm for NAND flash memory and the BER performance has not been compared with existing algorithms like MV-SF, MV-PSFD, and D-SFDP.

The low complexity hard decision symbol flipping decoding algorithm based on the majority logic decision is known as generalized algorithm B [15]. This algorithm is based on plurality logic which counts the number of occurrences of each symbol. Let τ be a predefined threshold, then if the number of occurrence of one finite

field symbol over $GF(q)$ exceeds τ, the algorithm chooses that received symbol as reliable, otherwise it chooses the same symbol. This algorithm is further improved by weighting factor $\theta_{d(\alpha,\boldsymbol{z}_j^{(k)})}$ based on the Hamming distance between the finite field symbol and extrinsic information-sum. This improved algorithm is called weighted algorithm B [15]. In this algorithm, the counts of occurrence of a finite field symbol is multiplied by the weighting coefficients assigned to the Hamming distance. The decision is taken based on the largest product. The optimal set of weights was not determined in the weighted algorithm B and is left as future work.

Let $d(\boldsymbol{z}_j, \sigma_{i,j})$ be the binary Hamming distance between EXI $\sigma_{i,j}$ and the hard decision symbol $\boldsymbol{z}_j$, and $\theta_{d(\boldsymbol{z}_j,\sigma_{i,j})}$ be the corresponding weighting factor. Denoting η_α as the number of occurrence of an element $\alpha \in GF(q)$ based on the plurality logic of each element $\sigma_{i,j}$ at the j^{th} variable node, then the weighted algorithm B (wt.Algo B) decision for an estimated correct symbol $v_j^{(k)}$ at k^{th} iteration is obtained as follow:

$$v_j^{(k)} = \begin{cases} \alpha, & \text{if } \eta_\alpha \theta_{d(\alpha,\boldsymbol{z}_j^{(k)})} \geq \tau \\ \boldsymbol{z}_j^{(k)}, & \text{otherwise.} \end{cases} \tag{2.14}$$

Here α corresponds to the largest product of $\eta_\alpha \theta_{d(\alpha,\boldsymbol{z}_j^{(k)})}$ and the preset threshold τ is determined through simulations.

In the voting scheme [16, 18], each unsatisfied check node gives one vote to the relevant variable node. The j^{th} variable node then collects all the votes, say $V_j^{(k)}$, from the failed check nodes at k^{th} iterations.

$$V_j^{(k)} = \sum_{i \in M(j)} V_{i,j}^{(k)} \tag{2.15}$$

where $V_{i,j}^{(k)} = 1$ if $\boldsymbol{s}_i^{(k)} \neq \boldsymbol{0}$, otherwise $V_{i,j}^{(k)} = 0$. For the accumulated voting $V_j^{(k)}$ equal or greater than a predefined threshold V_{th}, those variable nodes fulfilling the condition $V_j \geq V_{th}$ will be passed to calculate the flipping function $E_j^{(k)}$ through equations (2.14) and (2.15). Suppose $\boldsymbol{\delta}$ stores the positions of all the variable nodes

for $V_j \geq V_{th}$, then the values of $\boldsymbol{\delta}$ at k^{th} iteration can be calculated as follows:

$$\boldsymbol{\delta}_{j'}^{(k)} = V_j^{(k)}, \quad if \quad V_j \geq V_{th} \tag{2.16}$$

for $1 \leq j' \leq p$ and p shows the total number of the short listed variable nodes. This method reduces the computational complexity from n number of variable nodes to just few variable nodes p. Here $\delta_{j'}^{(k)} \in \boldsymbol{\delta}^{(k)}$ shows the position of each short listed variable node and $\boldsymbol{\delta}^{(k)}$ contains all those positions of the variable nodes. In other words, $\boldsymbol{\delta}^{(k)}$ have all the positions of variable nodes having less reliable information and must be replaced with reliable symbols from $\boldsymbol{z}_{j'}^{*(k)} \in \Gamma_{j'}^{*(k)}$. The multiple voting method presented in [18] defines two voting levels $\zeta_0 > \zeta_1 > 0$, using the same voting function defined in (2.15). For $s_j^{(k)} \neq 0$, $V_{ij}^{(k)} = \zeta_0$ with the largest flipping function and $V_{ij}^{(k)} = \zeta_1$ for the VN with the second largest flipping function. In the proposed algorithms, a single level voting scheme is used for short listing of least reliable symbols which reduces the memory consumption and computational complexity.

The concept of hard reliability to improve the performance of the bit reliability based (BRB) decoding algorithm was introduced in [31]. The binary Hamming distance between hard decision symbol sequence $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$ and the extrinsic information-sums (EXIs) indicate the hard reliability of the EXIs. The author in [31] showed that the Hamming distance $d(\boldsymbol{z}_j, \sigma_{i,j})$ indicates the correct probability of extrinsic information-sum $\sigma_{i,j}$. If p_b is raw bit error probability of variable nodes (VNs) over the Galois field GF($q = 2^r$), then the symbol error probability of the individual variable node is $1-(1-p_b)^r$. The relationship between the EXI and Hamming distance [31] is written as:

$$Pr(\sigma_{i,j} | d(\boldsymbol{z}_j, \sigma_{i,j}) = u) =$$

$$\frac{p_b^u q}{p_b^u q - (1-p_b)^{u-r} + (1-p_b)^{-d_c r + r + u}} \tag{2.17}$$

for $1 \leq u \leq r$.

Different from the wt.Algo B which only considers the information before flipping, the recent algorithms (D-SFDP and P-SFDP)[20] use both the information before and after flipping. These algorithms also take into account the hard reliability to evaluate correctly the contribution of the unsatisfied checksums. For the received hard decision symbol sequence $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n), (q-1)$ values for each symbol of the sequence is predicted through an exhaustive search for the symbol value which maximizes an objective function. The flipping function derived in [20] predicts symbols for each position of the received hard decision sequence $\boldsymbol{z}_j^{(k)}$ such that $\boldsymbol{z}_j^{(k)} \neq \ddot{\boldsymbol{z}}_j^{(k)}$ and $\ddot{\boldsymbol{z}}_j^{(k)}$ has all possible symbol values of the $GF(q)$ except for the symbol value of $\boldsymbol{z}_j^{(k)} \in GF(q)$. Therefore, there are $q-1$ possible values for each position of $\ddot{\boldsymbol{z}}_j^{(k)}$ which can be written as:

$$\ddot{\Gamma}_j^{(k)} = \{\ddot{\boldsymbol{z}}_j^{(k)} : \ddot{\boldsymbol{z}}_j^{(k)} \epsilon GF(q), \ddot{\boldsymbol{z}}_j^{(k)} \neq \boldsymbol{z}_j^{(k)}\} \tag{2.18}$$

Define the binary operator for hard decision symbol $\boldsymbol{z}_j^{(k)}$ and its corresponding channel soft symbol information $\boldsymbol{y}_j$ as:

$$\boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j = \sum_{t=0}^{r-1} (2z_{j,t}^{(k)} - 1)\boldsymbol{y}_{j,t} \tag{2.19}$$

To include the reliability derived from the structure of the Galois field, a vector of extrinsic weighting coefficients are defined as $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_k, \ldots, \theta_r]$. Here θ_k shows the extrinsic weighting factor corresponding to $d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)}) = k$ for $0 \leq k \leq r$. According to equation (2.17), $d_{\theta(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \geq 2$ does not carry much useful information and it has very low probability to be corrected [31]. According to this, the weighting coefficients are distributed as $\theta = [\theta_0, \theta_1, \theta_2]$ for the extrinsic information-sum and are optimized through simulation.

The flipping function of the D-SFDP algorithm introduced the hard reliability to predict the most reliable candidate symbol and can be written as follow:

$$E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j + \sum_{i \in M_j} \theta_{d(\ddot{\boldsymbol{z}}_j^{(k)}, \sigma_{i,j}^{(k)})}$$

$$- \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j - \sum_{i \epsilon M_j} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{2.20}$$

For plurality logic P-SFDP algorithm, the extrinsic weighting coefficients are defined as $\boldsymbol{\eta} = [\eta_0, \eta_1, \ldots, \eta_l, \ldots \eta_{d_c}]$ where η_l shows the weighting coefficient corresponding to the number of occurrence of $\alpha \in GF(q)$ in the extrinsic information-sum $\sigma_{i,j}^k$ for $0 \leq l \leq d_c$. Now equation (2.20) can be re-written to include the hard reliability, the number of occurrence of the element $\alpha \in GF(q)$, of the j^{th} symbol as follow:

$$E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j + \boldsymbol{\eta}_l(\ddot{\boldsymbol{z}}^{(k)}) - \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j - \boldsymbol{\eta}_l(\boldsymbol{z}^{(k)}) \tag{2.21}$$

To predict all the $q-1$ values of $\ddot{\boldsymbol{z}}_j^{(k)}$ for each symbol in $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$, the algorithm becomes computationally very complex and also requires more memory to store the processed values. For large values of $GF(q)$ and length of the code n, the algorithm [20] is very slow to find the candidate symbol value for the position to be flipped. The flipping function $E_j^{(k)}$ of the j^{th} symbol and its relevant flipped value $v_j^{(k)}$ are calculated as follows:

$$E_j^{(k)} = \max_{\ddot{\boldsymbol{z}}_j^{(k)} \in \ddot{\Gamma}_j^{(k)}} E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) \tag{2.22}$$

$$v_j^{(k)} = \operatorname*{argmax}_{\ddot{\boldsymbol{z}}_j^{(k)} \in \ddot{\Gamma}_j^{(k)}} E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) \tag{2.23}$$

The flipping function $E_j^{(k)}$ is very complex in computation and secondly a large memory is required to store all the $q-1$ predicted values for n symbols.

Though symbol flipping decoding algorithms are simple in nature and the decoder takes less computation to process the received information but the bit error rate performances of symbol flipping decoding are still far from belief propagation decoding algorithms and further investigation on this topic is important. The flipping function to determine correctly the position of a symbol in doubt plays main role in such type of decoding algorithms. Once the position of a symbol

is determined correctly, then the next question arises is the selection of suitable candidate symbol for that position.

2.3.1 Layered and Shuffled LDPC Coding Techniques

To improve the convergence speed, layered decoding scheme is used both for binary and NB-LDPC codes. Two type of strategies are used for layer decoding- row layered and column layered. The main difference between them is the type of layer and theoretically they are equivalent and have better decoding performance. Due to data dependencies, next layer cannot start until that of the current layer is completed. Column-layered decoding scheme is termed as Shuffled decoding [61]. In the column-layered decoding, the columns of the parity check matrix H are divided into groups or layers, and at a time, the decoding is performed on one column layer. For quasi cyclic LDPC codes, each group normally consists of one block column.

The standard SPA decoding updates the check node from the information values calculated at the previous iteration $(k-1)^{th}$ while in the shuffle decoding technique, certain values could already be computed based on partial computation of values at variable node. These partial computed information can be used for the check node update at the same k^{th} iteration instead of previous values calculated at $(k-1)^{th}$ iteration. This method provides the shuffling of the check and variable node.

Group shuffled decoding [61] algorithm divides the VNs into groups and performs decoding in parallel with in the same group while from group to group, the decoding is performed sequentially. This method partitions decoding iteration into several sub-iterations effectively and as a result achieves faster convergence. Other advantage of the group shuffled decoding is to prioritize the updates of those erroneous variable nodes likely to be corrected. This early update of the VNs, enhances the probability of reliable message passing and avoid the tentative incorrect local decision to propagate.

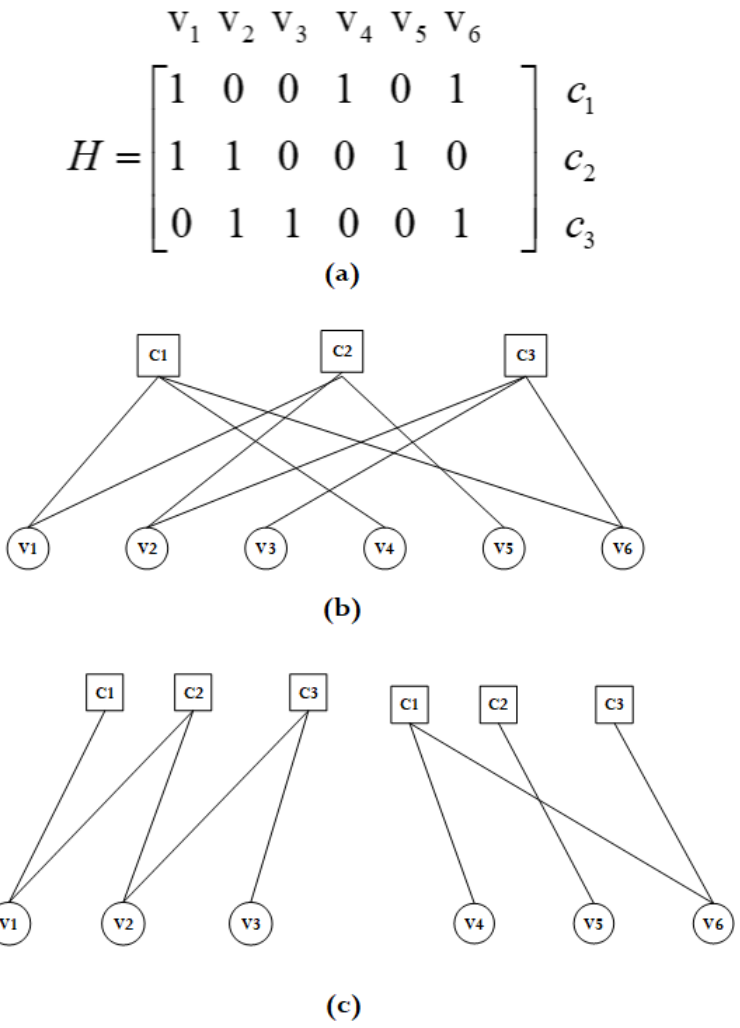


FIGURE 2.1: Figure (a)Parity check matrix. (b) Standard decoding. (c) Shuffle decoding with two groups

The Figure 2.1 depicts the decoding process for standard and group shuffle schemes for one iteration of SPA. Here the variable nodes are divided into two groups. If it is divided them into six, then it becomes the layered or just shuffle decoding. The shuffled SPA for LDPC codes have the computational advantage of forward-backward computation with the same complexity as standard SPA. This shuffling techniques can be generalized to the approximate version of SPA decoding.

In the shuffle SPA decoding [61],[62], initialization, stopping criterion, and the output stage are the same as in standard SPA. The only difference between the shuffle and the standard SPA lies in the updating procedure. The check node update procedure for $1 \leq j \leq n$ and $i \in M(j)$, the check node and variable node

are processed jointly. The n codeword is divided into G groups and each group is comprised of $n/G = \beta$ for $1 \leq g \leq \beta$. Equation (2.5) of the standard SPA is calculated for shuffle decoding as $g : \beta + 1 \leq j \leq (g + 1)\beta$.

For $G = 1$ the shuffle decoding becomes the standard SPA and with $G = n$ the group shuffle becomes the shuffle or layered SPA. It is observed that in one iteration the SPA algorithm can fully process in parallel while the shuffle SPA decoding process message passing in serial. This parallel shuffle scheme for LDPC code is called group shuffle decoding. In this algorithm, the code length is divided into specified groups where in each group the updating of messages is completed in parallel while inside the group, information is processed serially. The group shuffle decoding is summarised as:

• Within the same group, message passing is processed in parallel.

• Between the groups, message passing is processed in serial.

The variable node can be divided into groups of different sizes based on the local girth and edges of Tanner graph [63] which increases the bit error performance and convergence speed. The local girth of a variable node is used for the appropriate partition. A new dynamically re-grouping of VNs per iteration based on simple binary/integer decision has been presented in [64]. The following metric decides the selection of VNs from the index set v of the variable nodes with least reliable decision and high probability to be corrected during updates:

$$E_j = \Omega_j(\sum_{i \in M(j)} \boldsymbol{s}_i) \tag{2.24}$$

where $\Omega_j(x) = \frac{x}{d_j^v} d_{max}^v$ and $d_{max}^v = \max_j[d^v(j)]$. This equation (2.24) can be associated with the flipping function of the Gallagar algorithm B and is equal to the number of failed check-sum. The flipping function is defined as:

$$F_j = \sum_{i \in N(j)} q_{(i,j)} \boldsymbol{s}_i \tag{2.25}$$

where

$$q_{(i,j)} = \begin{cases} 1, & \text{if } \max_{j' \in N(i)} E_{j'} = E_j \text{ and } E_j \geq \hbar \\ 0, & \text{else} \end{cases} \tag{2.26}$$

Here F_j counts the number of unsatisfied check-sums and $\hbar$ is an integer optimized numerically. This method of identifying the most unreliable VNs can be explored for possible utilization in the NB-LDPC to improve performance as well as convergence speed. In literature, Group-Shuffled [61],[63] sum-product decoding algorithms divide either check nodes or variable nodes of the relevant bipartite graph into small sub-groups called layers. Also each main iteration is broken into multiple sub-iterations. The group might have one or more than one rows and columns. Also the quasi cyclic parity check matrix is divided horizontally into groups in such a manner that each group has exactly one non-zero entry in each column. This is the most common method used for sum-product LDPC decoder hardware implementation. For the variable nodes $V_1, V_2, \ldots, V_j, \ldots, V_n$, let G_τ be a group, then a set of groups G is given by:

$$G = \{G_1, G_2, \ldots, G_\tau, \ldots, G_\alpha\},\ 1 \leq \tau \leq \alpha \tag{2.27}$$

In layer decoder, extrinsic information are exchanged more sufficiently due to division of main iteration into sub iteration, therefore, this scheme results in more accurate decision and also lower the burden of processing. In the group decoding algorithm, groups are processed sequentially and VN belonging to the same group are updated in parallel. These grouping techniques are developed for the message passing sum-product algorithm and have not yet applied to the symbol flipping decoding technique.

CHAPTER 3

Low Complexity Symbol Value Selection Decoding Algorithms

This section is based on the following research publication.

Waheed Ullah, Ling Cheng, Fambirai Takawira. "Low Complexity Bit Reliability and Predication Based Symbol Value Selection Decoding Algorithms for Non-Binary LDPC Codes." IEEE Access, vol. 8, pp. 142691–142703, 2020.

Contribution Statement: The first author is responsible for the conceptualization of the work, experimental data collection, analysis and interpretation, as well as project execution and results collection. The work was done with assistance from the co-authors: Prof. Ling Cheng and Prof. Fambirai Takawira.

Abstract: The main challenge for hardware implementation of non-binary LDPC decoding is the high computational complexity and large memory requirement. To address this challenge, five new low complexity LDPC decoding algorithms are proposed in this paper. The proposed algorithms are developed specifically towards the low complexity, yet effective, decoding of the NB LDPC codes. The proposed decoding algorithms update, iteratively, the hard decision received vector to search for the valid codeword in the vector space of Galois field (GF). The selection criterion for least reliable symbol positions is based on the information from the failed checks and the reliability information from the Galois field structure as well as from the received channel soft information. To choose the correct value for the candidate symbol, two methods are used. The first method is based on the prediction of the error symbol from the set of Galois field symbols which maximize an objective function. In the second method, individual bits are flipped

based on the reliability information obtained from the channel. Algorithms 1 and 2 flip a single symbol per iteration whilst the other three algorithms 3,4 and 5 flip multiple symbols in each iteration. The proposed voting based Algorithms 1,2 and 5 first short list the unreliable positions using a majority voting scheme and then choose the candidate symbol value from the set of the symbols in $GF(q)$ while not violating the field order q. These methods simplify the decoding complexity in terms of computation and memory. Results and analysis of these algorithms show an appealing tradeoff between computational complexity and bit error rate performance for NB LDPC codes.

3.1 Introduction

Reliable and efficient communication depends on the performance of forward error correction (FEC) codes. Among error correction codes, non-binary (NB) LDPC codes have shown an excellent bit error rate (BER) performance, especially in comparison to binary LDPC codes [3, 65, 66]. An important feature of NB LDPC is that they have good performance for short and medium length codes [67]. The performance of LDPC codes depend on the type of decoders used. One of the good decoders is the $q-$ary sum-product algorithm (QSPA) decoder which passes a vector of q probability messages over each edge of the Tanner graph, representing the probability of all q elements of $GF(q)$. However, the high check node computational complexity of the QSPA decoder is a major obstacle to their finding a place in practical applications. After the invention of the NB LDPC codes [3], most of the research has focused on how to reduce the computational complexity of the check node processing. To lower the complexity of QSPA, a Fast Fourier transform based SPA (FFT-SPA) algorithm was proposed in [11] to reduce the check node computational complexity from order of $O(q^2)$ to $O(q\,log\,q)$ for each check node update. Other low complexity algorithms called extended min-sum (EMS) [68, 69], trellis based EMS [70], bubble check EMS [71] and min-max algorithms [54] were proposed but they have a performance degradation in comparison to QSPA. Further, the complexity of the computations of all these algorithms is

still too high for hardware implementation. On the other hand, reliability based message passing algorithms [24, 25],[26, 27] and the symbol flipping algorithms [15, 16] are simple and computationally very fast for NB LDPC codes but at a cost of reduced performance.

Reliability based majority logic decoding algorithms offer low complexity as they send only the most reliable field message and in this respect are similar to message passing decoding algorithms. In the iterative majority logic decoding (MLgD) algorithm for non-binary LDPC codes, each symbol is iteratively updated by the extrinsic information-sums (EXIs) with the most reliable field element along each edge of the Tanner graph. The iterative reliability based hard (IHRB) and soft (ISRB) majority logic decoding (MLgD) algorithms [23] have lower complexity but suffer from performance degradation. Some performance improvements to IHRB-MLgD and ISRB-MLgD algorithms have been presented in [24–27]. To improve the performance of IHRB-MLgD algorithm, an enhanced (E-IHRB-MLgD) algorithm is presented in [25] by introducing soft reliability at the initialization and re-computing the extrinsic-information. The enhanced IHRB algorithm has the same complexity as that of IHRB except at the initialization but it has an improved performance which is close to ISRB algorithm. The majority logic decoding algorithms are particularly suitable for parity check matrices with high column weight, constructed based on finite fields [28] and finite geometries [30]. These algorithms suffer from performance loss for codes with small column weight.

Better performing, low complexity message passing decoding algorithms were presented in [31]. These algorithms introduce reliability updates in terms of the bit rather than symbols and are termed as bit reliability based (BRB) algorithms. The BRB algorithm require integer and Galois field operations only, which reduces the computational complexity. The performance of the BRB algorithm is improved by multiplying the extrinsic information-sum by Hamming distance based weighted coefficients. This weighted BRB (wBRB) algorithm shows improved performance over ISRB and BRB and does not suffer from error floor for parity check matrices with small column weight. A full bit-reliability based decoder [32] is the binary

form of weighted BRB as this algorithm uses the binary representation of the non-zero entries of the parity check matrix to facilitate hardware implementation.

Symbol flipping (SF) decoding algorithms have low complexity as compared to the bit reliability based decoding algorithms [31, 32] and the q-ary belief propagation [3][11] message passing algorithms but at the cost of reduced bit error rate performance. In the symbol flipping decoding algorithm presented in [72], the least reliable position is determined by majority decision, while the flipped symbol value is computed from the reliability of the received bits. The weighted algorithm B (wt.Algo B) in [15] introduces the binary Hamming distance and plurality logic to improve performance. The parallel symbol flipping decoding (PSFD) algorithm [16] uses majority voting to get better flipping decision but this algorithm performs better only if the parity check matrix has a large column weight. The PSFD algorithm is improved further by using multiple votes in [18] and performs better even for the LDPC parity check matrix with small column weight. The non-binary LDPC decoder based on symbol flipping with multiple votes (MV-SF) [59] performs better than MV-PSFD but at the cost of an increased complexity and memory size especially for the higher order Galois field.

The other recently developed symbol flipping algorithms (D-SFDP and P-SFDP) [20] offer better performance than most of the existing symbol flipping algorithms, but a major disadvantage is the exhaustive and computationally complex prediction mechanism. The complexity of these algorithms [20] depend on the size of the Galois field order q and length n of a codeword. A decision-symbol reliability-based SFD (DRB-SFD) algorithm [60] is aimed to show better performance in applications like data storage based on NAND flash memory. In [60] is focused on an algorithm for NAND flash memory and the BER performance has not been compared with existing algorithms like MV-SF, MV-PSFD, and D-SFDP.

The objective of this chapter is to propose algorithms with good performance and low complexity to fulfill the requirement of low power and efficient memory usage. The proposed algorithms can be divided into three categories; 1) algorithms using voting and prediction [16][18][20]; 2) algorithms using voting with channel bit

reliability [16][18][17] and 3) multiple symbol flipping decoding algorithms based on prediction as well as bit reliability [20][17].

Algorithm 1 is based on the category 1. In [16] and its improved version [18], the flipping function is computed for all the received symbols and the majority voting scheme is used in parallel to the flipping function to select the flipping position while in the proposed Algorithm 1, the majority voting scheme is used only for short listing the least reliable variable nodes. The flipping function is then computed only for the short listed variable nodes. Algorithm 2 also belongs to category 1, but simplifies the flipping function by using the divide and conquer rule by using the flipping function for identifying first the symbol positions and then finding the candidate symbol value. Algorithm 3 is from category 3 but differs from the algorithm [17] as the proposed Algorithm 3 uses flipping function based on channel reliability information as well as information from Galois field structure while the flipping function in [17] is based on the syndromes and channel likelihood information. Algorithm 4 is also based on category 3 and flips multiple symbols per iteration but it uses the symbol value prediction instead of the bit reliability used in [17]. Algorithm 5 is based on categories 2 and 3. This algorithm uses majority voting scheme for short listing the least reliable symbols unlike in [16] and [18] where they use voting in parallel with the flipping function. The proposed Algorithm 5 flips the unreliable bit of the selected least reliable symbol instead of the prediction method adopted in [20]. Algorithm 5 also flips multiple symbols per iteration.

The set of failed parity checks and the infinite loop detection procedure are applied only when multiple symbols are flipped in each iteration. In the multiple symbol flipping decoding algorithms, the set of failed parity checks and infinite loop detection procedures restrict the number of symbols to be flipped in consecutive iterations. On the other hand, the proposed algorithms based on the voting scheme except for Algorithm 5, flip single symbol per iteration and do not require information from the failed checks and infinite loop detection procedure.

The rest of the chapter is summarized as follows. Section II summaries the relevant literature on NB LDPC decoding algorithms whilst Section III presents the proposed algorithms. Section IV compares the complexity and memory requirement of various algorithms. Section V shows the results and analysis of the existing and proposed algorithms in detail. Section VI gives a conclusion of the chapter.

3.2 Symbol Flipping Decoding Algorithms

3.2.1 Preliminaries

Let $GF(q)$ be the Galois field with q elements. Consider a NB LDPC code $\mathcal{C}$ of length n with a regular parity check matrix H, defined over the *GF(q)*, with m as number of rows and n as number of columns such that each row of H has a constant weight of d_c and each column of H has a constant weight of d_v. Each element $h_{i,j}(1 \leq i \leq m \ , \ 1 \leq j \leq n)$ of H is an element over the Galois field $GF(q)$ where $q = 2^r$. The non-zero entries of $H(h_{i,j} \neq 0)$ in the Tanner graph show the i^{th} check node (CN) connected to the j^{th} variable node (VN). For the column weight d_v and the row weight d_c, the Tanner graph edge connections are shown as $M(j) = \{i : 1 \leq i \leq m \ , \ h_{i,j} \neq 0\}$ and $N(i) = \{j : 1 \leq j \leq n, h_{i,j} \neq 0\}$ and for regular matrix $\frac{m}{n} = \frac{d_v}{d_c}$.

Let $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_j, \ldots, \boldsymbol{c}_n)$ be a codeword in $\mathcal{C}$ and the binary representation of j^{th} symbol in $\boldsymbol{c}$ is $\boldsymbol{c}_j$= $(c_{j,1}, c_{j,2}, \ldots, c_{j,t}, \ldots, c_{j,r}), 1 \leq t \leq r$. This binary sequence is modulated with binary phase shift keying (BPSK), where 1 is modulated as +1 and 0 is modulated as -1. The BPSK modulated j^{th} symbol $\boldsymbol{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,t}, \ldots, x_{j,r})$ is transmitted over additive white Gaussian noise (AWGN) channel. The received binary sequence $\boldsymbol{y}_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,t}, \ldots, y_{j,n})$ is obtained from the transmitted sequence $\boldsymbol{x}_j$ after adding the additive white Gaussian channel noise $\boldsymbol{n} \rightarrow \mathcal{N}(0, \sigma^2)$ with zero mean and two sided power spectral density $N_0/2$ as $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j$. The hard decision binary sequence $\boldsymbol{z}_j$ =

$(z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is given by:

$$z_{j,t} = \begin{cases} 1 & y_{j,t} \geq 0 \\ 0 & y_{j,t} < 0 \end{cases} \tag{3.1}$$

The hard decision binary sequence $z_{j,t}$ is mapped to an element in $GF(q)$ to obtain the symbol $\boldsymbol{z}_j$. The i^{th} syndrome of the j^{th} hard decision symbol sequence can be defined for $h_{i,j} \neq 0$:

$$\boldsymbol{s}_i = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j \tag{3.2}$$

Here it is important to mention that the value of $\boldsymbol{s}_i$ is also an element in $GF(q)$. If $\boldsymbol{s}_i \neq 0$, it means that some of the variable nodes contributing to this check node are incorrect.

For the given hard decision symbol $\boldsymbol{z}_j$, the extrinsic information-sum(EXI) denoted as $\sigma_{i,j}$ that is passed from i^{th} check node (CN) to the j^{th} variable node (VN) is given by:

$$\sigma_{i,j}^{(k)} = h_{i,j}^{-1} \sum_{j' \in N(i) \setminus j} h_{i,j'} \boldsymbol{z}_{j'}^{(k)} \tag{3.3}$$

for $1 \leq i \leq m, j \in N(i)$.

3.2.2 Symbol Flipping Non-Binary LDPC Decoding Algorithms

The low complexity hard decision symbol flipping decoding algorithm based on the majority logic decision is known as generalized algorithm B [15]. This algorithm is based on plurality logic which counts the number of occurrences of each symbol. Let τ be a predefined threshold, then if the number of occurrence of one finite field symbol over $GF(q)$ exceeds τ, the algorithm chooses that received symbol as reliable, otherwise it chooses the same symbol. This algorithm is further improved by weighting factor $\theta_{d(\alpha, \boldsymbol{z}_j^{(k)})}$ based on the Hamming distance between the finite

field symbol and extrinsic information-sum. This improved algorithm is called weighted algorithm B [15]. In this algorithm, the counts of occurrence of a finite field symbol is multiplied by the weighting coefficients assigned to the Hamming distance. The decision is taken based on the largest product. The optimal set of weights was not determined in the weighted algorithm B and is left as future work.

Let $d(\boldsymbol{z}_j, \sigma_{ij,})$ be the binary Hamming distance between EXI $\sigma_{i,j}$ and the hard decision symbol $\boldsymbol{z}_j$, and $\theta_{d(\boldsymbol{z}_j,\sigma_{i,j})}$ be the corresponding weighting factor. Denoting η_α as the number of occurrence of an element $\alpha \in GF(q)$ based on the plurality logic of each element $\sigma_{i,j}$ at the j^{th} variable node, then the weighted algorithm B (wt.Algo B) decision for an estimated correct symbol $v_j^{(k)}$ at k^{th} iteration is obtained as follow:

$$v_j^{(k)} = \begin{cases} \alpha, & \text{if } \eta_\alpha \theta_{d(\alpha,\boldsymbol{z}_j^{(k)})} \geq \tau \\ \boldsymbol{z}_j^{(k)}, & \text{otherwise.} \end{cases} \tag{3.4}$$

Here α corresponds to the largest product of $\eta_\alpha \theta_{d(\alpha,\boldsymbol{z}_j^{(k)})}$ and the preset threshold τ is determined through simulations. Equation 3.4 belongs to the weighted algorithm B in [15] while the research in this chapter proposes modification and improvement to the D-SPDF algorithm [20].

The concept of hard reliability to improve the performance of the bit reliability based (BRB) decoding algorithm was introduced in [31]. The binary Hamming distance between hard decision symbol sequence $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$ and the extrinsic information-sums (EXIs) indicate the hard reliability of the EXIs. The author in [31] showed that the Hamming distance $d(\boldsymbol{z}_j, \sigma_{i,j})$ indicates the correct probability of extrinsic information-sum $\sigma_{i,j}$. If p_b is raw bit error probability of variable nodes (VNs) over the Galois field GF($q = 2^r$), then the symbol error probability of the individual variable node is $1-(1-p_b)^r$. The relationship between the EXI and Hamming distance [31] is written as:

$$Pr(\sigma_{i,j}|d(\boldsymbol{z}_j, \sigma_{i,j}) = u) =$$

$$\frac{p_b^u q}{p_b^u q - (1-p_b)^{u-r} + (1-p_b)^{-d_c r + r + u}} \tag{3.5}$$

for $1 \leq u \leq r$.

Different from the wt.Algo B which only considers the information before flipping, the recent algorithms (D-SFDP and P-SFDP)[20] use both the information before and after flipping. These algorithms also take into account the hard reliability to evaluate correctly the contribution of the unsatisfied checksums. For the received hard decision symbol sequence $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n), (q-1)$ values for each symbol of the sequence is predicted through an exhaustive search for the symbol value which maximizes an objective function. The flipping function derived in [20] predicts symbols for each position of the received hard decision sequence $\boldsymbol{z}_j^{(k)}$ such that $\boldsymbol{z}_j^{(k)} \neq \ddot{\boldsymbol{z}}_j^{(k)}$ and $\ddot{\boldsymbol{z}}_j^{(k)}$ has all possible symbol values of the $GF(q)$ except for the symbol value of $\boldsymbol{z}_j^{(k)} \in GF(q)$. Therefore, there are $q-1$ possible values for each position of $\ddot{\boldsymbol{z}}_j^{(k)}$ which can be written as:

$$\ddot{\Gamma}_j^{(k)} = \{\ddot{\boldsymbol{z}}_j^{(k)} : \ddot{\boldsymbol{z}}_j^{(k)} \epsilon GF(q), \ddot{\boldsymbol{z}}_j^{(k)} \neq \boldsymbol{z}_j^{(k)}\} \tag{3.6}$$

Define the binary operator for hard decision symbol $\boldsymbol{z}_j^{(k)}$ and its corresponding channel soft symbol information $\boldsymbol{y}_j$ as:

$$\boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j = \sum_{t=0}^{r-1} (2z_{j,t}^{(k)} - 1)\boldsymbol{y}_{j,t} \tag{3.7}$$

To include the reliability derived from the structure of the Galois field, a vector of extrinsic weighting coefficients are defined as $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_k, \ldots, \theta_r]$. Here θ_k shows the extrinsic weighting factor corresponding to $d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)}) = k$ for $0 \leq k \leq r$. According to equation (3.5), $d_{\theta(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \geq 2$ does not carry much useful information and it has very low probability to be corrected [31]. According to this, the weighting coefficients are distributed as $\theta = [\theta_0, \theta_1, \theta_2]$ for the extrinsic information-sum and are optimized through simulation.

The flipping function of the D-SFDP algorithm introduced the hard reliability to predict the most reliable candidate symbol and can be written as follow:

$$E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j + \sum_{i \in M_j} \theta_{d(\ddot{\boldsymbol{z}}_j^{(k)}, \sigma_{i,j}^{(k)})}$$

$$- \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j - \sum_{i \epsilon M_j} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{3.8}$$

For plurality logic P-SFDP algorithm, the extrinsic weighting coefficients are defined as $\boldsymbol{\eta} = [\eta_0, \eta_1, \ldots, \eta_l, \ldots \eta_{d_c}]$ where η_l shows the weighting coefficient corresponding to the number of occurrence of $\alpha \in GF(q)$ in the extrinsic information-sum $\sigma_{i,j}^k$ for $0 \leq l \leq d_c$. Now equation (3.8) can be re-written to include the hard reliability, the number of occurrence of the element $\alpha \in GF(q)$, of the j^{th} symbol as follow:

$$E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j + \boldsymbol{\eta}_l(\ddot{\boldsymbol{z}}^{(k)})$$

$$- \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j - \boldsymbol{\eta}_l(\boldsymbol{z}^{(k)}) \tag{3.9}$$

To predict all the $q-1$ values of $\ddot{\boldsymbol{z}}_j^{(k)}$ for each symbol in $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$, the algorithm becomes computationally very complex and also requires more memory to store the processed values. For large values of $GF(q)$ and length of the code n, the algorithm[20] is very slow to find the candidate symbol value for the position to be flipped. The flipping function $E_j^{(k)}$ of the j^{th} symbol and its relevant flipped value $v_j^{(k)}$ are calculated as follows:

$$E_j^{(k)} = \max_{\ddot{\boldsymbol{z}}_j^{(k)} \in \ddot{\Gamma}_j^{(k)}} E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) \tag{3.10}$$

$$v_j^{(k)} = \underset{\ddot{\boldsymbol{z}}_j^{(k)} \in \ddot{\Gamma}_j^{(k)}}{\operatorname{argmax}} E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) \tag{3.11}$$

The flipping function $E_j^{(k)}$ is very complex in computation and secondly a large memory is required to store all the $q-1$ predicted values for n symbols.

3.2.3 Candidate Symbols Short-listing By Voting

In the voting scheme [16, 18], each unsatisfied check node gives one vote to the relevant variable node. The j^{th} variable node then collects all the votes, say $V_j^{(k)}$, from the failed check nodes at k^{th} iterations.

$$V_j^{(k)} = \sum_{i \in M(j)} V_{i,j}^{(k)} \tag{3.12}$$

where $V_{i,j}^{(k)} = 1$ if $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, otherwise $V_{i,j}^{(k)} = 0$. For the accumulated voting $V_j^{(k)}$ equal or greater than a predefined threshold V_{th}, those variable nodes fulfilling the condition $V_j \geq V_{th}$ will be passed to calculate the flipping function $E_j^{(k)}$ through equations (5.7) and (3.8). Suppose $\boldsymbol{\delta}$ stores the positions of all the variable nodes for $V_j \geq V_{th}$, then the values of $\boldsymbol{\delta}$ at k^{th} iteration can be calculated as follows:

$$\boldsymbol{\delta}_{j'}^{(k)} = j, \quad if \quad V_j \geq V_{th} \tag{3.13}$$

for $1 \leq j' \leq p$ and p shows the total number of the short listed variable nodes. This method reduces the computational complexity from n number of variable nodes to p. Here $\delta_{j'}^{(k)} \in \boldsymbol{\delta}^{(k)}$ shows the position of each short listed variable node and $\boldsymbol{\delta}^{(k)}$ contains all those positions of the variable nodes. In other words, $\boldsymbol{\delta}^{(k)}$ have all the positions of variable nodes having less reliable information and must be replaced with reliable symbols from $\boldsymbol{z}_{j'}^{*(k)} \in \Gamma_{j'}^{*(k)}$. The multiple voting method presented in [18] defines two voting levels $\zeta_0 > \zeta_1 > 0$, using the same voting function defined in (3.12). For $s_j^{(k)} \neq 0$, $V_{ij}^{(k)} = \zeta_0$ with the largest flipping function and $V_{ij}^{(k)} = \zeta_1$ for the VN with the second largest flipping function. In the proposed algorithms, a single level voting scheme is used for short listing of least reliable symbols which reduces the memory consumption and computational complexity.

3.3 Proposed Algorithms

3.3.1 Voting Based Symbol Value Prediction Algorithm

In this proposed algorithm, the least reliable variable nodes are short listed by majority voting and the candidate symbol value is selected by prediction method. The prediction based symbol flipping algorithms (D-SFDP and P-SFDP) [20] offer reasonably better performance than most of the symbol flipping algorithms in literature but at the cost of exhaustive and computationally complex prediction mechanism. The other disadvantage of the algorithms in [20] is the high memory requirement to store the matrix of predicted values for all symbols in the codeword. A voting based approach is used in this proposed algorithm, to lower decoding latency and computational power by short listing the symbols with low reliability. The flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ in equation (3.8) is calculated $q-1$ times for the j^{th} variable node of the codeword n, including the information before and after the flipping to predict correctly the most reliable symbol during each iteration. After calculating $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ for all the variable nodes, a matrix of size $n(q-1)$ is formed and must be stored for further processing.

The complexity of the flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ can be lowered from n to p by first short-listing the candidate symbols through voting as given in equation (3.11) and (3.12) respectively. This method also reduces the memory requirement from $n(q-1)$ to $p(q-1)$. This newly improved algorithm is termed as voting based symbol flipping decoding(V-SFD) algorithm. The new rule to calculate the flipping function and the corresponding flipped value for $j' \in \delta^{(k)}$ is given by:

$$E_{j'}^{(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \boldsymbol{z}_{j'}^{(k)}) = \ddot{\boldsymbol{z}}_{j'}^{(k)} \odot \boldsymbol{y}_{j'} + \sum_{i \in M_{j'}} \theta_{d(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \sigma_{i,j'}^{(k)})}$$

$$- \boldsymbol{z}_{j'}^{(k)} \odot \boldsymbol{y}_{j'} - \sum_{i \epsilon M_{j'}} \theta_{d(\boldsymbol{z}_{j'}^{(k)}, \sigma_{i,j'}^{(k)})} \tag{3.14}$$

The above flipping metric is calculated for the short listed symbols only, resulting in low complexity and memory. The flipping function $E_j^{(k)}$ gives the reliability measure of the j^{th} symbol to be flipped to the value $v_j^{(k)}$

$$E_{j'}^{(k)} = \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \ddot{\Gamma}_{j'}^{(k)}} E_{j'}^{(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \boldsymbol{z}_{j'}^{(k)}). \tag{3.15}$$

This flipping function is similar to the maximum likelihood method and the maximum value of the flipping function for all the possible candidate symbols $\ddot{z}_j^{(k)}$ shows the most reliable variable node update.

$$v_{j'}^{(k)} = arg \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \ddot{\Gamma}_{j'}^{(k)}} E_{j'}^{(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \boldsymbol{z}_{j'}^{(k)}) \tag{3.16}$$

Here j' shows the short listed least reliable variable nodes and the flipping function is calculated only for those selected variable nodes, thus reducing tremendously the computational complexity and memory requirement. In this chapter, the Hamming distance based reliability information are used as in D-SFDP algorithm and the plurality logic based reliability information used in the P-SFDP algorithm is left intentionally as intuitively the later will show the same performance with the proposed schemes.

Proposed Algorithm 1 (V-SFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Determine the un-reliable symbol $\delta_{j'}^{(k)}$ using the voting scheme by (3.12) and (3.13).
5. Calculate $E_{j'}^{(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \boldsymbol{z}_{j'}^{(k)})$ for $\boldsymbol{\delta}$ symbols by (3.14) and then calculate $E_{j'}^{(k)}$ by (3.15).
6. Find its flipped value $v_{j'}^{(k)}$ by (3.16) and update $\boldsymbol{z}^{(k+1)}$ with $v_{j'}^{(k)}$.
7. $k = k + 1$ and go to step 3.

3.3.2 Voting Based Simplified Symbol Value Prediction Algorithm

This algorithm further reduces the complexity of the flipping function in Algorithm 1 by applying a divide and conquer strategy. Equation (3.14) is divided into two parts: first the algorithm finds the unreliable positions to be flipped and then predicts the candidate symbol for that position. The unreliable symbol position is determined by the following equation.

$$E_{j'}^{(k)}(\boldsymbol{z}_{j}^{(k)}) = \boldsymbol{z}_{j'}^{(k)} \odot \boldsymbol{y}_{j'} + \sum_{i \in M_{j'}} \theta_{d(\boldsymbol{z}_{j'}^{(k)}, \sigma_{i,j'}^{(k)})} \tag{3.17}$$

$$E_{j'}^{(k)} = \min_{1 \leq j' \leq \delta} (E_{j}^{(k)}(\boldsymbol{z}_{j'}^{(k)})) \tag{3.18}$$

Equation (3.18) will determine the most unreliable position to be flipped. From the set $\ddot{\Gamma}_{j}^{(k)}$ of $q-1$ predicted symbols, a candidate symbol value is selected as the correct symbol which maximize the flipping function in equation (3.19). After a symbol value is predicted for each unreliable position, then the hard decision sequence is updated with that symbol at every k^{th} iteration as follows:

$$E_{j'}^{*k}(\ddot{\boldsymbol{z}}_{j'}^{(k)}) = \ddot{\boldsymbol{z}}_{j'}^{(k)} \odot y_{j'} + \sum_{i \in M_{j'}} \theta_{d(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \sigma_{i,j'}^{(k)})}. \tag{3.19}$$

After calculating the flipping function $E_{j'}^{*k}(\ddot{\boldsymbol{z}}_{j'}^{(k)})$, the position for the candidate symbols which maximize $E_{j'}^{*k}$ is given by:

$$E_{j'}^{*(k)} = \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \Gamma_{j'}^{(k)}} \{E_{j'}^{*(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}\} \tag{3.20}$$

The predicted symbol value $\ddot{v}_{j'}^{(k)}$ from the set of the predicted values $\ddot{\Gamma}_{j'}^{(k)}$ is found through the following equation:

$$\ddot{v}_{j'}^{(k)} = arg \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \Gamma_{j'}^{(k)}} E_{j'}^{*(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}) \tag{3.21}$$

The function $E_{j'}^{*(k)}$ in (3.20), determines the position of the most unreliable j^{th} symbol and equation (3.21) selects the flipped value $\ddot{v}_{j'}^{(k)}$.

The voting scheme, used in this chapter, helps to further reduce the computational complexity of the decoder. The voting scheme also helps in saving the memory consumption as the reliability information related to the selected least reliable variable nodes, are stored for further processing.

Equations (3.17), and (3.18) are computed only for $\boldsymbol{\delta}$ positions of the received codeword sequence instead of all the symbols of codeword having length of n. After selecting the most unreliable symbol position, a reliable symbol value from the set of possible predicted $q-1$ values such as $\boldsymbol{z}_{j'}^{(k)} \neq \ddot{\boldsymbol{z}}_{j'}^{(k)}$, is computed using equations (3.19), (3.20) and (3.21) respectively. Since equations (3.19), (3.20) and (3.21) are computed for a single variable node with possible $q-1$ values, the proposed Algorithm 2 further reduces the decoder computational complexity.

Proposed Algorithm 2 (Simplified V-SFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Determine the unreliable symbol $\delta_{j'}^{(k)}$ using the voting scheme by (3.12) and (3.13).
5. For $j' \in \boldsymbol{\delta}$, find the unreliable positions $E_{j'}^{(k)}(\boldsymbol{z}_{j'}^{(k)})$ for $\boldsymbol{\delta}$ symbols by (3.17) and (3.18).
6. Calculate the predicted candidate symbol position $E_{j'}^{*(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)})$ by (3.19) and (3.20).
7. Find predicted flipped value $\ddot{v}_{j'}^{(k)}$ by (3.21) and update $\boldsymbol{z}^{(k+1)}$ with $\ddot{v}_{j'}^{(k)}$.
8. $k = k + 1$ and go to step 3.

3.3.3 Multiple Symbol Flipping Bit Reliability Based Decoding Algorithm

In this section, a multiple symbol flipping decoding algorithm for NB LDPC decoder is proposed which flips the unreliable bit of a symbol by using the channel bit reliability information. Variable node short listing by voting is not used in this algorithm. This algorithm finds positions of symbols with less reliability using the information before flipping and then selects the reliable symbol values for those positions based on the information after flipping. The computational cost of this proposed algorithm is smaller than most of the NB LDPC decoding algorithms in literature as this algorithm compare and flips individual bit of a symbol. This algorithm also gives the advantage of multiple symbol flipping in each decoding iteration. The unreliable multiple symbol positions are determined by the following equations:

$$E_j^{(k)}(\boldsymbol{z}_j^{(k)}) = \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j + \sum_{i \in M_j} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{3.22}$$

$$E_j^{(k)} = \min_{1 \leq j \leq n} \{E_j^{(k)}(\boldsymbol{z}_j^{(k)})\} \tag{3.23}$$

Let τ be the threshold, then the number of flipping positions can be determined based on the column weight d_v and the number of variable nodes contributing to the failed checks. If ρ is the maximum number of the symbols to be flipped per iteration then it is determined as follows:

$$\rho = \begin{cases} \rho_1 & \text{if} \quad \tau \geq \varepsilon_1 d_v \\ \rho_2 & \text{else} \end{cases} \tag{3.24}$$

where ε_1 is an integer value. Equation (3.24) can also be used directly with $\rho = \beta$ where β is an integer value. Based on the value of ρ, the unreliable multiple flipping positions are determined by (3.23).

The number of symbols to be flipped is further restricted in the loop detection procedure (3.34) and secondly if the flipping positions are not in the set of the failed checks. The i^{th} syndrome of the j^{th} hard decision symbol at k^{th} iteration

can be defined as:

$$\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j^{(k)} \tag{3.25}$$

A valid codeword is received if m-tuple $\boldsymbol{s}_i^{(k)} = \mathbf{0}$ for $1 \leq i \leq m$. The cause of the failure of check sum is due to the contribution of the variable nodes in error. All those variable nodes are given as follow:

$$\upsilon = \{i : \ \boldsymbol{s}_i \neq \mathbf{0}, 1 \leq i \leq m, j \in M(i)\} \tag{3.26}$$

The variable nodes contributing to the successful check-sum are given by:

$$\bar{\upsilon} = \{i' : \ \boldsymbol{s}_i = \mathbf{0}, 1 \leq i \leq m, j \in M(i')\} \tag{3.27}$$

Suppose $\boldsymbol{\psi}$ is the vector containing all the flipping positions found by (3.23) and $\bar{\upsilon}$ contains all the correct variable nodes contributing to $\boldsymbol{s}_i^{(k)} = \mathbf{0}$, then those variable node contributing to the failed checks to which $\boldsymbol{\psi}$ must belong as a subset, are defined as:

$$T = \{\upsilon(j) \backslash \upsilon \cap \bar{\upsilon}\} \tag{3.28}$$

Equation (3.28) identifies and removes those variable nodes which are common in equation (3.26) and (3.27). This helps to decide the number of flipping positions in $\boldsymbol{\psi}$ as subset to T. Once the flipping positions are identified, the symbol values are updated by flipping that individual bit of symbol with the smallest absolute value. The absolute channel soft values are termed as reliability information. The smaller the absolute value of the bit, the less is the reliability of that bit and vice versa. For j^{th} received symbols $\boldsymbol{y}_j^{(k)}$, the bit closer to zero or having the minimum absolute value is expressed for $1 \leq t \leq r$ as:

$$\acute{y}_{j,t}^{(k)} = \min_{1 \leq t \leq r} |\boldsymbol{y}_j^{(k)}| \tag{3.29}$$

The corresponding bit $z_{j,t}^{(k)}$ in the hard decision symbol $\boldsymbol{z}_j^{(k)}$ is then flipped. The expression for the flipped bit is written as:

$$\acute{z}_{j,t}^{(k)} = \begin{cases} 1, & \text{if } \; z_{j,t}^{(k)} = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.30}$$

Flipping the bit in the hard-decision symbol can cause oscillating behaviour. Therefore, the channel soft reliability information $\boldsymbol{y}_j^{(k+1)}$ is also updated using equation (3.31) or (3.32):

$$y_{j,t}^{(k+1)} = -(y_{j,t}^{(k)}) \tag{3.31}$$

$$y_{j,t}^{(k+1)} = \begin{cases} -1 - y_{j,t}^{(k)}, & \text{if } \; z_{j,t}^{(k)} = 0 \\ 1 + y_{j,t}^{(k)}, & \text{if } \; z_{j,t}^{(k)} = 1. \end{cases} \tag{3.32}$$

In this chapter equation (3.32) is used. The new hard decision symbol sequence will be updated for the next $(k+1)^{th}$ iteration as:

$$\boldsymbol{z}^{(k+1)} = (\boldsymbol{z}_1^{(k)}, \boldsymbol{z}_2^{(k)}, \ldots, \acute{\boldsymbol{z}}_j^{(k)}, \ldots, \boldsymbol{z}_n^{(k)}) \tag{3.33}$$

If $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, then decoding failure due to some of the variable nodes (VNs) contributing to check nodes (CNs) will have occured. The information of the successful and failed syndromes are stored for further processing. Normally if construction of the parity check matrix H is 4 cycles free, then the variable nodes contributing to the failed checks can easily be identified. The process for isolating the incorrect symbols (variable nodes) is done by identifying only those variable nodes contributing to the failure of check nodes. Unique variable node positions contributing to the failure of check nodes are stored and are used further for comparison with the least reliable symbol positions identified through the flipping function.

The proposed multiple symbol flipping algorithm significantly lowers the complexity by selecting multiple positions and then flips the less reliable bit of each of the selected symbols. This newly proposed algorithm is termed as B-MSFD algorithm. Since the prediction mechanism of the D-SFDP algorithm[20] is complex

in computation, this method makes an effective tradeoff between complexity and performance.

Proposed Algorithm 3 (B-MSFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$
2. For iteration $k = 1$ to I_{max}
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword
4. Calculate $E_j^{(k)}(\boldsymbol{z}_j^{(k)})$ by (3.22) and determine the number of symbol positions ρ to be flipped by (3.24) and calculate $E_\rho^{(k)}$ by (3.23).
5. Find $\boldsymbol{\psi}^{(k)}$ positions by (3.28) and then corresponding flipped values by (3.29) and (3.30).
6. Update $\boldsymbol{z}^{(k+1)}$ with $\acute{\boldsymbol{z}}_j^{(k)}$ and the soft channel information $\boldsymbol{y}^{(k+1)}$ with $\acute{\boldsymbol{y}}_j^{(k)}$ by (3.33) and (3.32) respectively.
7. $k = k + 1$ and go to step 3.

In symbol flipping decoding algorithms, the flipping position might be the same for successive iterations when the decoder is trapped in an infinite loop. To avoid the decoding being trapped in successive iterations, the symbol positions to be flipped are recorded and are compared with the next iteration to make sure that the same symbol is not flipped in consecutive iterations. Therefore, a symbol is flipped in iteration k only if

$$\psi_b^{(k)} \neq \psi_b^{(k-1)} \tag{3.34}$$

for $1 \leq b \leq \rho, k > 1$. At first iteration $k = 1$ all the selected symbols positions will be flipped based on the preset threshold but in successive iterations ($k > 1$), the number of symbol positions to be flipped depends on condition (3.34).

3.3.4 Multiple Symbol Flipping Prediction Based Algorithm

Different from Algorithm 2 which flips single symbol per iteration and uses the voting scheme to short list the variable nodes before computing the flipping function, this proposed Algorithm 4 flips multiple symbols per iteration and does not short list the least reliable variable nodes for computing the flipping function. This proposed algorithm for non-binary LDPC decoder is primarily divided into two steps. In the first step, the positions of the least reliable symbols are selected and in the second step, the most reliable symbol value for those position are predicted.

In this algorithm, the successful and failed checks are isolated as shown in equations (3.26),(3.27) and (3.28). The set of the variable nodes that contributed to the failed checks, helps to decide which symbol position should be flipped and how many positions needs to be flipped. The unreliable symbols' positions are determined by the following equations.

$$E_j^{(k)}(\boldsymbol{z}_j^{(k)}) = \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j + \sum_{i \in M_j} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{3.35}$$

$$E_j^{(k)} = \min_{1 \leq j \leq n} (E_j^{(k)}(\boldsymbol{z}_j^{(k)})) \tag{3.36}$$

This proposed algorithm is used to flip multiple symbols in each iteration and the number of flipping symbols depend on a pre-defined threshold ρ.

After finding all the unreliable symbol positions ρ, those candidate symbol values are chosen for which the flipping function is maximized. A symbol value is predicted for each of the unreliable positions and then the hard decision sequence is updated with those symbols at every k^{th} iteration.

$$E_j^{*(k)}(\ddot{\boldsymbol{z}}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot y_j + \sum_{i \in M_j} \theta_{d(\ddot{\boldsymbol{z}}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{3.37}$$

$$E_j^{*(k)} = \max_{\ddot{\boldsymbol{z}}_j^{(k)} \in \Gamma_j^{(k)}} (E_j^{*(k)}(\ddot{\boldsymbol{z}}_j^{(k)})) \tag{3.38}$$

$$\ddot{v}_j^{(k)} = arg \max_{\ddot{\boldsymbol{z}}_j^{(k)} \in \Gamma_j^{(k)}} E_j^{*(k)}(\ddot{\boldsymbol{z}}_j^{(k)}) \tag{3.39}$$

In (3.37), the function $E_j^{*(k)}$ measures the position of the most unreliable j^{th} symbol to be flipped to the value $\ddot{v}_j^{(k)}$. In this proposed prediction based multiple symbol flipping decoding(P-MSFD) algorithm, symbols are predicted for ρ positions only which makes the algorithm very efficient and less complex in computation at the order of $O(\rho q)$ and is independent of the length of the code n. For understanding, let $\rho = 1$, so only $q-1$ values for the position $\boldsymbol{\psi} = j, \boldsymbol{\psi} \in \upsilon$ at k^{th} iteration will be calculated by (3.38) and (3.39).

Equation (3.37) predicts the most reliable position for symbol which maximize the function and then (3.38) determines the candidate symbol value $\boldsymbol{z}_j^{*(k)} \neq \boldsymbol{z}_j^{(k)}$ for that position. The new hard decision symbol sequence will be updated for the next $(k+1)^{th}$ iteration as:

$$\boldsymbol{z}^{k+1} = (\boldsymbol{z}_1^{(k)}, \boldsymbol{z}_2^{(k)}, \ldots, \ddot{\boldsymbol{z}}_j^{(k)}, \ldots, \boldsymbol{z}_n^{(k)}) \tag{3.40}$$

Proposed Algorithm 4 (P-MSFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Calculate $E_j^{(k)}(\boldsymbol{z}_j^{(k)})$ by (3.37) and determine the number of symbol positions ρ to be flipped by (3.24) and calculate $E_\rho^{(k)}$ by (3.35) and (3.36).
5. Calculate $E_\rho^{*(k)}(\boldsymbol{z}_\rho^{*(k)})$ and $E_\rho^{*(k)}$ by (3.37) and (3.38) respectively for $\ddot{\boldsymbol{z}}_\rho^{(k)} \in \Gamma_\rho^{*(k)}$.
6. Determine $\boldsymbol{\psi}^{(k)}$ positions by (3.34) and then its flipped value $\ddot{v}_{\boldsymbol{\psi}}^{(k)}$ by (3.39). Update $\boldsymbol{z}^{(k+1)}$ with $\ddot{v}_{\boldsymbol{\psi}}^{(k)}$.
7. $k = k + 1$ and go to step 3.

3.3.5 Voting Based B-MSFD Algorithm

The bit error rate performance and complexity of B-MSFD algorithm is further improved and simplified by using the voting scheme to flip multiple symbols in each iteration. Once these positions are selected, the bit with less reliability in each symbol is chosen to be flipped based on the channel reliability information. This algorithm is termed as voting based multiple symbol flipping decoding (VB-MSFD). The least reliable symbol positions are short-listed by (3.12) and (3.13). The selected p number of unreliable positions $\boldsymbol{\delta}$, are used only to find the flipping values using the bit reliability information and the flipping function is not required to determines the least reliable symbols. This algorithm does not need the loop detection procedure and also does not need to isolate the failed and successful variable nodes as in B-MSFD, resulting in further simplification. Finding the flipped value from the channel reliability information is the same as for the B-MSFD algorithm. The proposed algorithm significantly lowers the decoder computational complexity as well as memory requirement to store the information for further

processing.

Proposed Algorithm 5 (VB-MSFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = 0$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Determine the un-reliable symbol $\delta_{j'}^{(k)}$ using the voting scheme by (3.20) and (3.21).
5. Find the flipped values by (3.29) and (3.30) for each candidate symbol.
6. Update $\boldsymbol{z}^{(k+1)}$ with $\acute{\boldsymbol{z}}_{j'}^{(k)}$ and the channel soft information $\boldsymbol{y}^{(k+1)}$ with $\acute{\boldsymbol{y}}_{j'}^{(k)}$ by (3.33) and (3.32) respectively.
7. $k = k + 1$ and go to step 3.

3.4 Complexity Analysis

In this section, we evaluate the decoding computational complexity and memory requirement of the proposed algorithms in comparison with various existing NB LDPC decoding algorithms. The complexity analysis is carried out in terms of numerical operations per decoder iteration. Consider a NB LDPC code $\mathcal{C}$ for $GF(q) = 2^r$ having a parity check matrix H of size $m \times n$. Focusing on the regular LDPC code with constant column weight d_v and constant row weight d_c, the total number of edges of the Tanner graph is $\gamma = d_v n = d_c m$. In a single iteration, computational complexity of the proposed algorithms are mainly involved in the equations (3.14),(3.17),(3.22) and (3.35).

To get the hard decision symbol sequence $\boldsymbol{z}$ at the decoder initialization, the proposed algorithms require nr real comparisons. To calculate the syndrome check-sum for valid codeword, md_c Galois field multiplication and $m(d_c - 1)$ Galois field

additions are required. To get the extrinsic information-sums(EXIs), the decoder requires md_c Galois field multiplications and md_c Galois field additions.

The major complexity of the algorithm in [20] involves computation of the binary Hamming distance $d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})$ for each edge, the binary function $\boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j$ for n number of symbols and the second binary function $\ddot{\boldsymbol{z}}_j^{(k)} \odot y_j$ for nq predicted values.

Therefore, the total complexity of the algorithms in [20] are in order of $O(nq)$ when all $q-1$ values are computed to predict the candidate symbol value while the proposed Algorithms 1 and 2 lower this complexity to order of qp where p shows the number of unreliable variable nodes and is much less than n ($p << n$). The other two proposed Algorithms 3 and 5 further reduces the complexity as they do not require any prediction but simply flip a bit within each selected symbol.

In Algorithm 1, to compute the flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$, the binary functions $\boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j$ for the symbol values before flipping and $\ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j$ for the predicted values of $\ddot{\boldsymbol{z}}_j^{(k)} \in \Gamma$ are calculated. Ignoring multiplication of the plus or minus one, r real additions are required for the binary function before flipping and $r(q-1)$ addition are required to compute the $\ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j$ for all the predicted symbols. In step 1 of the algorithms in [20], $n(3+d_v)r$ additions are required for all the symbols while $p(3+d_v)r$ additions for the proposed Algorithm 1 and only δr additions for proposed Algorithm 5 as it does not require information after flipping.

For the decoding Algorithm 1, let $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \mathcal{F}_1 - \mathcal{F}_1$ where $\mathcal{F}_1 = \boldsymbol{z}_j^{(k)} \odot y_j^{(k)} + \sum_{i \in M(j)} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})}$ and $\mathcal{F}_2 = \ddot{\boldsymbol{z}}_j^{(k)} \odot y_j^{(k)} + \sum_{i \in M(j)} \theta_{d(\ddot{z}_j^{(k)}, \sigma_{i,j}^{(k)})}$. To compute $\mathcal{F}_1$, pd_v real additions are required and to compute $\mathcal{F}_2$, $pd_v(d_v+q)$ real additions are required. Computation of all values of $\mathcal{F}_1 - \mathcal{F}_1$, requires $p(d_v+q)$ real additions. For each symbol in the received sequence, the function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ requires (d_v+q) comparisons. The function $E_j^{(k)}$, for received sequence of length n, $p(d_v+q-1)$ real comparisons are required. Since Algorithms $1, 3$ and 5 also do not need any loop detection procedure, the decoder computational complexity is further reduced.

The proposed algorithms for NB LDPC codes also significantly lower the memory consumption as shown in Table 3.2. Suppose r bits are required to store each of

TABLE 3.1: Computational complexity per iteration for various NB-LDPC decoding algorithms

Algorithms	Number of operations				
	GM	GA	IA/RA	RC/IC	IM/RM
Algorithm1 (V-SFD)	$d_v(2d_c-1)$	$pd_v(d_v+r)+2d_v(d_c-1)$	$p(d_v^2+2d_vr+4r+2d_v)$	$nr+p(d_v-1)+2(p-1)$	
Algorithm5 (VB-MSFD)	md_v	$m(d_v-1)$	nd_v	$rn+p-1$	
D-SFDP	$d_v(2d_c-1)$	$nd_v(d_v+r)+2d_v(d_c-1)$	$n(d_v^2+2d_vr+4r+2d_v)$	$n(d_v+r-1)+2(n-1)$	
P-SFDP	$d_v(2d_c-1)$	$2d_v(d_c-1)$	$n(d_vr+5r+3d_v+1)$	$n(d_v+r-1)+2(n-1)$	
MV-SF	$nd_v(L+2)+2m(L-1)$	$nd_v(L+2)+2m(L-2)$	nd_vq+2nd_vL	$nd_vqlog_2q+n(q-1)+m(\sum_{i=d_c-\xi}^{d_c-1} i)$	
BRB	$2nd_v$	$2nd_v-m$	$2nd_vr$	$nd_v(2r+1)-3m$	
wBRB	$2nd_v$	$2nd_v-m$	$2nd_vr$	$nd_v(2r+1)-3m$	nd_v
PSFD	$2nd_v$	nd_v+2n-m	$2nd_v-2n+m$	$2nd_v-m$	
MV-PSFD	$2nd_v$	nd_v+2n-m	$3nd_v+m-1$	$2nd_v-m-n+nlog_2n$	$2nd_v+1$
EMS	$2nd_vn_m$	$9n_m(nd_v-2m)$	$n_m(20nd_v-18m-12n)$	$n_mlog_2n_m(9nd_v-4(3m-n))$	
ISRB	$2nd_v$	$2nd_v-m$	nd_v+n2^r	$2n2^r-2n$	$n2^r$
wt-Algo.B	$2nd_v$	$3nd_v-m$	nd_v	nd_v	nd_v

IM/IC/IA: Integer Multiplication/Comparision/Adittion; p is the total number of variable nodes stored in $\boldsymbol{\delta}$;
RA/RM/RC: Real Addition/Multiplication/Comparision; GA/GM: Galois Field Addtion/Multiplication;
Notations used: $\gamma = nd_v = md_c$, $L = v^{\xi}$ such that $\xi < d_c$, $n_m < q$

the $GF(q)$ element where $q = 2^r$ and b bits are required for storing floating-point values for each of the reliability metric. These algorithms require nrb bits to store the received sequence. The hard decision symbol sequence $\boldsymbol{z}_j^{(k)}$ and the extrinsic information-sums $\sigma_{i,j}^{(k)}$ require nr and nd_vr bits respectively.

For the weighting coefficients θ_d of the binary Hamming distance, d_vb bits are

TABLE 3.2: Memory required for decoding of NB-LDPC algorithms

Algorithms	Required Memory(bits)
Algorithm1 (V-SFD)	$nr(b+1)+b(d_v+q)+p(b+r)$
Algorithm5 (VB-MSFD)	$nr(b+1)+d_vb+r$
D-SFDP	$nrb+(2n+\gamma)r+(n+q+d_v)b+log_2n$
P-SFDP	$nrb+(2n+\gamma)r+(n+q)b+log_2n+q[log_2d_v]$
wBRB	$2nrb+\gamma(r+b-1)$
PSFD	$2nq+3n+\gamma)b+(n+\gamma)r+\gamma+n[log_2d_v]$
MV-PSFD	$2nq+4n+2\gamma)b+(n+\gamma)d+log_2m$
EMS	$2nqb+\gamma n_m(r+b)$
T-EMS	$2nqb+\gamma qb$
wt.Algo.B	$(n+\gamma)r+d_vb+log_2d_v$
MV-SF	$b(\gamma q+nq+d_cq+nr)+r(d_cq+d_cL+n)$

required to store the values. For the received sequence of length n, only p elements need to be stored. qb bits memory is required for each $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ and pqb bits for all the values. Similarly pb and pr bits of memory are required for storing $E_j^{(k)}$ and $v_j^{(k)}$ respectively. The proposed algorithms reduce the memory requirement from n to p only. The smaller are the number of variable nodes in error, the less is the value of the p in comparison to the length of the received sequence n. The proposed Algorithms 3 and 5 significantly lower the memory consumption as these algorithm does not need to store the set of the predicted values $\ddot{\boldsymbol{z}}_j^{(k)}$. The Algorithms 3 and 5 require only r bits to store the value for the flipped symbol $v_j^{(k)}$. Algorithm 5 offers significantly lower complexity as it does not calculate the flipping function for δ selected variable nodes. On the other hand, Algorithm 3 stores all the p number of selected variable nodes $\boldsymbol{\delta}$ to be used in finding the least reliable variable node during computation of the flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$.

3.5 Results and Discussion

In this section, the performance and complexity of the proposed algorithms are compared with various algorithms in the literature. An all zeros codeword is transmitted over additive white Gaussian channel(AWGN) using BPSK modulation ($1 \rightarrow +1$ and $0 \rightarrow -1$). In the proposed prediction based algorithms, all the

$q-1$ possible values such as $\boldsymbol{z}_j^{*(k)} \neq \boldsymbol{z}_j^{(k)}$ from $GF(q)$ are considered to predict the correct candidate symbol value. The other two Algorithms 3 and 5 do not use the symbol prediction method but only the less reliable bit of erroneous symbol is flipped. In the proposed algorithms, to predict the correct symbol, all possible $q-1$ values of $\boldsymbol{z}_j^{(k)}$ from $\ddot{\Gamma}_j^{(k)}$ are considered. However, only those values from the $GF(q)$ table can be considered whose binary representation is one bit different from the binary representation of $\boldsymbol{z}_j^{(k)}$ as there is higher chances of thier occurrence under AWGN. For example, in the hard decision symbol sequence, if the current value before prediction of the j^{th} symbol $\boldsymbol{z}_j^{(k)}$ is 101 for the code over $GF(8)$, then the predicted values look like $100, 001$, and 010. The maximum number of iterations in the following examples is determined through computer simulation and is set to 15. For code (120,60) over GF(128), the transmitted bits at 4.2dB are decoded using Algorithm 2 with 10, 15 and 20 iterations and the decoded bits have the error rate of 0.003117, 0.002011, and 0.001980 respectively. As the bit error rate difference of 15 and 20 is 0.000031 which is in the tolerance range. Therefore, the maximum iterations are kept as 15. The simulation is terminated either when 100 bits errors are received or the number of transmitted bits reaches to 10^8. For comparison, the low complexity bit reliability based majority logic decoding algorithm (wBRB) is included which shows performance close to q-ary SPA.

Example 1: In order to demonstrate the effective performance of the proposed algorithms, BER performance of NB LDPC decoding algorithms under additive white Gaussian noise as shown in Figure 3.1. The results are for a regular Gallagar code $\mathcal{C}1$ $(204, 102)$ [18] over $GF(16)$ with code rate 1/2. The code has a constant column weight $d_v = 3$ and constant row weight $d_c = 6$. The extrinsic weighting coefficients $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2]$ of the newly proposed algorithms are set as $\boldsymbol{\theta} = [2, .75, 0.5]$ for $d_{\theta(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \geq 2$.

Figure 3.1 illustrates the bit error rate performance of NB LDPC decoding algorithms. We see that the proposed Algorithms 2 and 4 are close in performance to the D-SFDP algorithm while Algorithms $1, 3$ and 5 perform better than BRB,

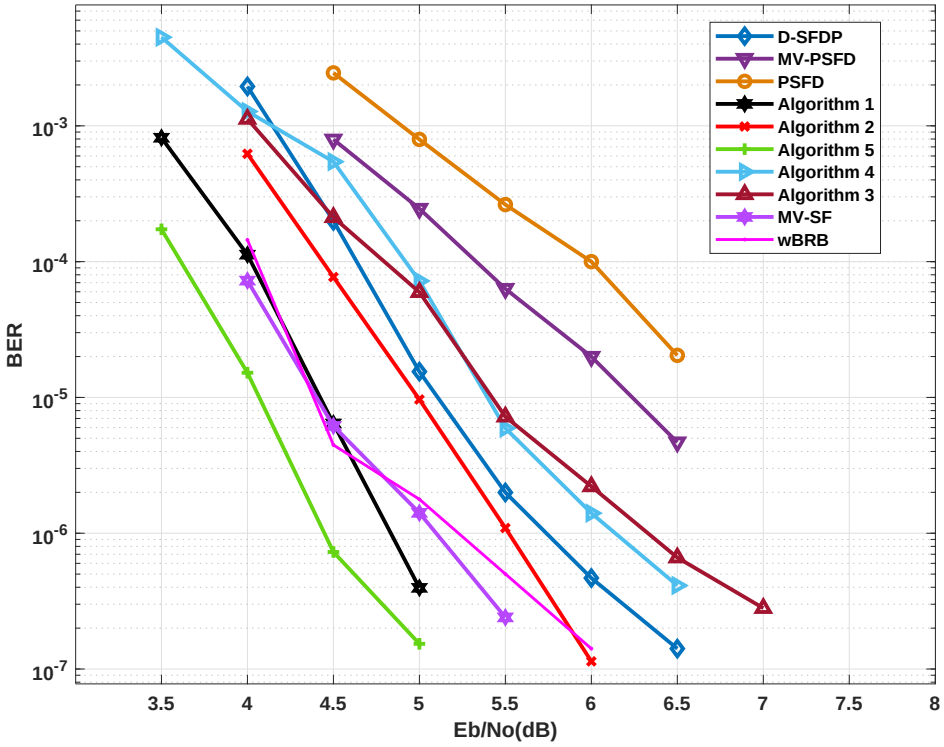


FIGURE 3.1: BER performance of NB LDPC code (204, 102) over $GF(16)$

MV-SFDP, PSFD, and D-SFDP. Algorithm 5 outperforms than all existing symbol flipping algorithms including MV-SF and wBRB. Similarly, the proposed Algorithm 1 performs close to MV-SF at the beginning and after 4.5 dB, it outperforms the MV-SF algorithm. The proposed Algorithms 3, 4 and 5 are multiple symbol flipping algorithms and perform better than BRB, ISRB, PSFD, MV-PSFD. At the BER $10^{-}6$, the proposed Algorithm 5 outperforms by about 1.5 dB and 0.65 dB in comparison to D-SFDP, MV-SF and wBRB algorithms respectively. The MV-PSFD algorithm performs better than PSFD and the performance gap between them is about 0.5 dB but less in performance to the proposed algorithms.

Example 2: In this example, the performance of the proposed algorithms is shown for a quasi cyclic (QC) NB LDPC code (120,60) over $GF(128)$. This is a regular parity check matrix with constant column weight $d_v = 4$ and constant row weight $d_c = 8$ with code rate 1/2. The performance of the proposed algorithms

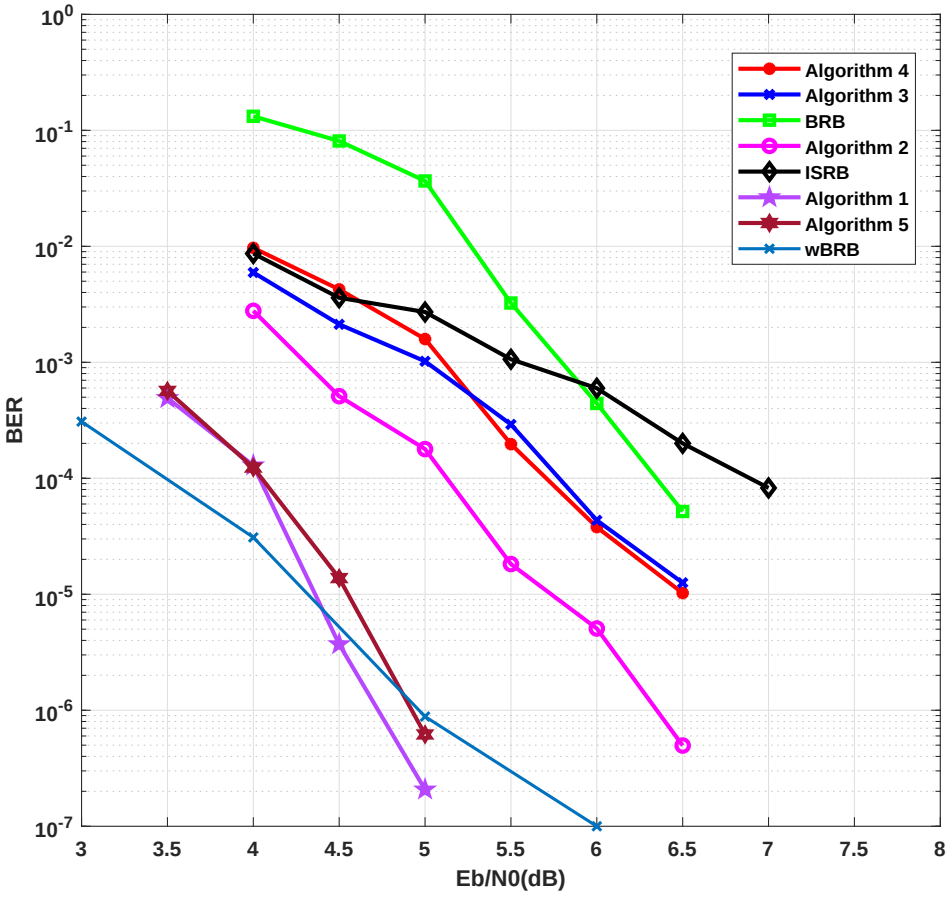


FIGURE 3.2: BER performance of NB LDPC code (120, 60) over $GF(128)$

is compared with reliability based algorithms like BRB and ISRB. The extrinsic weighting coefficient are set as $\theta = [2, .75, 0.5]$.

From the curves in the Figure 3.2, we see that all the proposed algorithms outperform than BRB and ISRB algorithms and are better than wBRB after 4.5 dB. At the bit error rate performance of 10^{-4}, the proposed Algorithms 1 and 5 achieves performance gain of about 2.2 and 3 dB over BRB and ISRB respectively. The performance gap between the proposed Algorithm 2 and BRB algorithm at BER 10^{-4} is around 1.3 dB and from the ISRB algorithm, the gap is around 1.8 dB. The Algorithms 3 and 4 have similar performance trend and show significant performance gain over all the algorithms in Figure 3.2. It is also observed that the decoding Algorithm 1 performs better than Algorithm 5. The reason for this is the higher order Galois field ($q = 2^r = 128, r = 7$) where each symbolS_j is

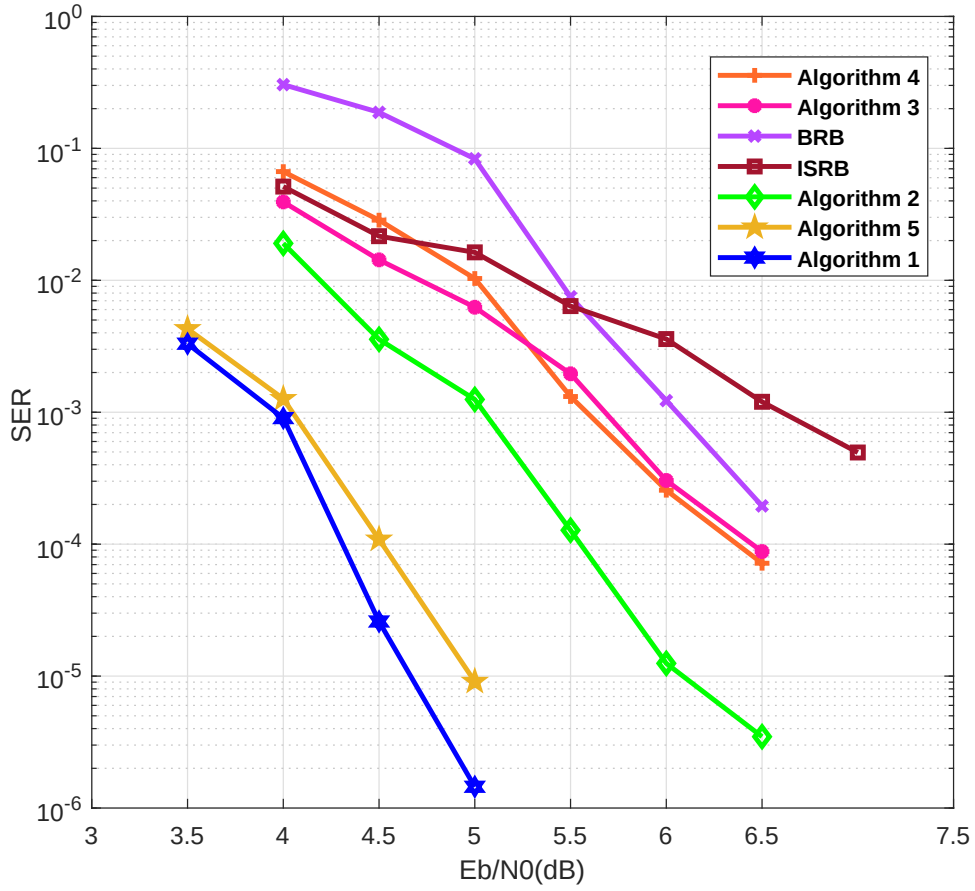


FIGURE 3.3: SER performance of NB LDPC code (120, 60) over $GF(128)$

composed of 7 bits ($e.g. S_j = b_0b_1b_2b_3b_4b_5b_6b_7$). The proposed Algorithm 5 correct one bit in each symbol per iteration and there are more chances to get more than one bit in error. The more are the number of bits in symbols, the more iterations are required to correct that symbol using Algorithm 5 in comparison to Algorithm 1 which is using the maximum likelihood technique. The Algorithm 1 try one by one each of the symbols $\ddot{\boldsymbol{z}}_j^{(k)} \neq \boldsymbol{z}_j^{(k)} \in GF(128)$ for the selected unreliable position and chooses the best candidate symbol value for the selected position. Due to this fact, Algorithm 1 performs better than Algorithm 5 for higher order Galois field $GF(q)$.

Similarly, Figure 3.3 shows the symbol error rate performance (SER) of the proposed algorithms in comparison to BRB and ISR algorithms. The performance curves show similar trends as the BER curves in Figure 3.2. However, the BER

curves in Figure 3.3 for BRB and ISRB are not smooth as the required number of iteration for these algorithms to converge properly are less in this example.

Example 3: In this example the average number of iterations versus BER and SNR are plotted to show the complexity and convergence of the proposed NB LDPC decoding algorithms in comparison to some other algorithms in the literature. The code for Example 2 with the extrinsic weighting coefficient set as $\boldsymbol{\theta} = [2, 0.75, 0.5]$ is used here as well. From the curves in the graphs shown in Figures 3.4 and 3.5, we see that the proposed Algorithm 5 converges slowly compared to the other proposed algorithms as well as to BRB and ISRB algorithms. BRB algorithm is the fast converging algorithm followed by ISRB while the proposed algorithms convergence speed is slower than BRB and ISRB but faster than D-SFDP.

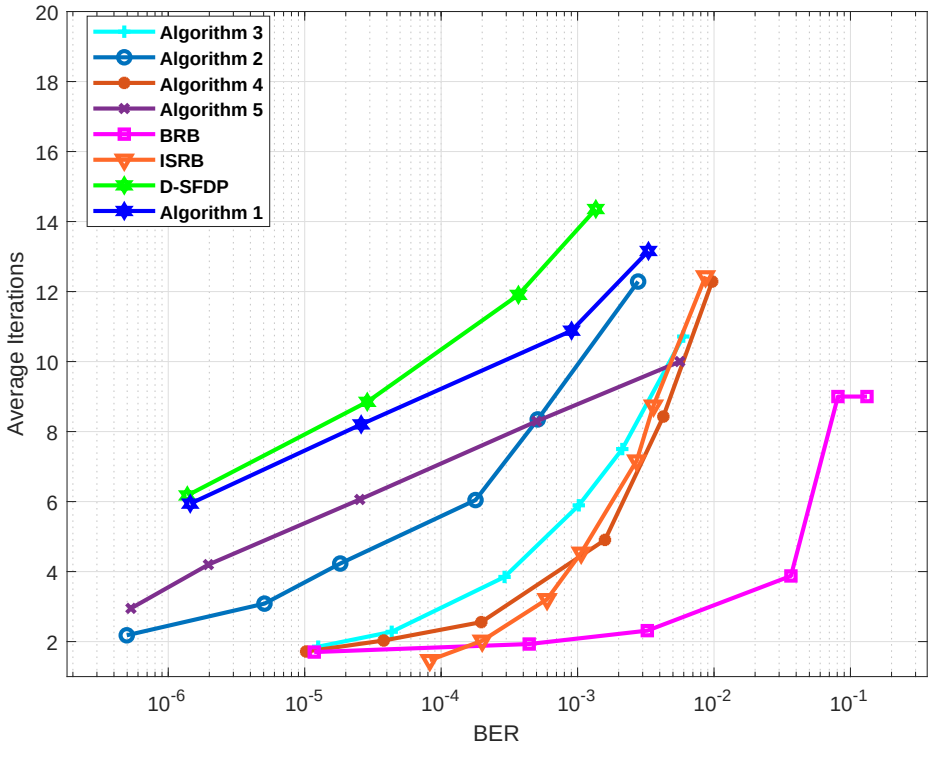


FIGURE 3.4: BER versus average number of iterations for NB LDPC code (120, 60) over $GF(128)$

Since the proposed Algorithms 1 and 2, flip a single symbol per iteration, the average number of iterations shown in Figures 3.4 and 3.5 are more than the proposed Algorithm 3 and 4 which flip multiple symbols in each iteration. Algorithm 5 flips

TABLE 3.3: Computational complexity per iteration of various algorithms at BER 10^{-4} FOR CODE(120, 60) over $GF(128)$

Algorithms	Number of operations				
	GM	GA	IA/RA	RC/IC	Avg. Itr
Algorithm5 (VB-MSFD)	60	216	136	8	8.4
Algorithm1 (V-SFD)	60	1112	216	20	10.89
D-SFDP	60	5336	12960	1438	13.91
BRB	960	900	6720	7020	2.1
ISRB	960	900	15840	30480	4.5

a single bit in each of the multiple selected symbols and that is why it takes more iterations for higher order Galois field. At the BER 10^{-4}, for the given average number of iterations, the computational complexity of BRB, ISRB and the proposed Algorithms 3 and 4 is lower than the D-SFDP and the Algorithms 2 and 5. As shown in Figure 3.5, we see that the Algorithm 5 performs better than other algorithms in terms of the average number of iterations, especially at the lower SNR. At higher SNR, the average number of iterations, except for Algorithm 5, exhibits almost the same trends.

Table 3.3 summarizes the complexity per iteration of the various algorithms in comparison to the proposed Algorithms 1 and 5. At BER 10^{-4}, the typical value for the p is around 6 to 15 at the first iteration and decreases after every iteration if the symbol flipped at the previous iteration is correct.

From the analysis of the BER performance curves, the proposed algorithms show an appealing trade-off between complexity and performance and therefore can be recommended as good candidates for applications like data storage and communication systems.

3.6 Conclusion

In this paper, low complexity symbol flipping decoding algorithms have been proposed. Incorporating the voting scheme to short list the least reliable symbols

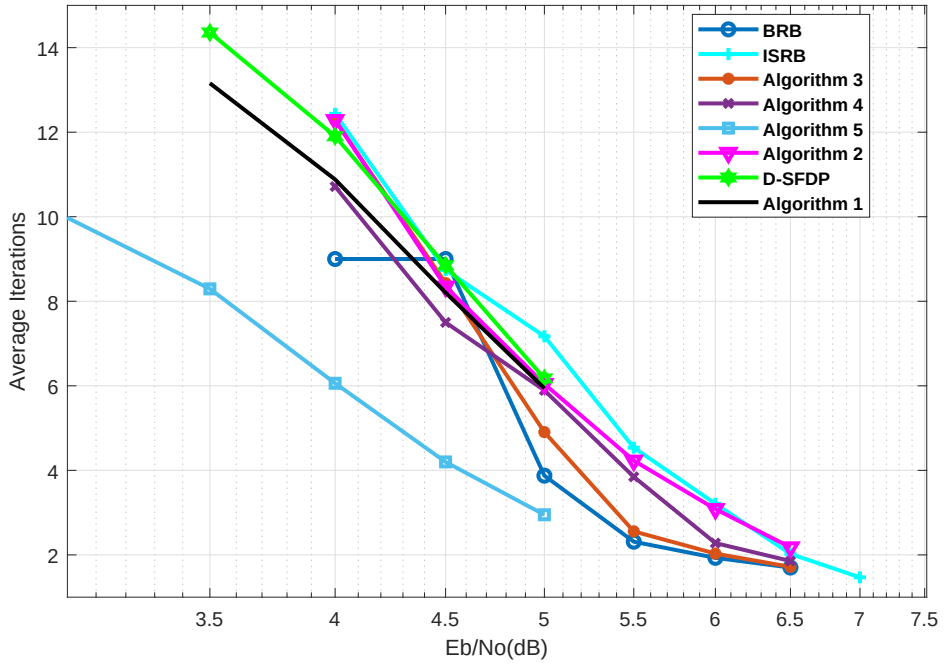


FIGURE 3.5: SNR versus average number of iterations for NB LDPC code (120, 60) over $GF(128)$

improves BER performance and reduces decoding latency. The voting scheme also helps in reducing memory requirements for storing the reliability values and extrinsic information for processing. Also, multiple symbol flipping decoding algorithms are proposed which helps in reducing the overall computational complexity of the NB LDPC decoder. Simulation results illustrate that the voting based proposed algorithms outperform all the existing symbol flipping decoding algorithms except Algorithm 2 whose performance is less than [59]. However, the Algorithm 2 is much superior when compared to the decoder complexity of [59]. The other proposed Algorithms 3 and 4, achieve the BER performance close to the algorithm [20] with significantly reduced complexity. From the results and complexity analysis, the proposed Algorithms 1 and 5 have shown an appealing trade-off between BER performance and computational complexity and are more suitable for practicable applications.

CHAPTER 4

Joint Iterative Detection- Decoding Algorithms

This chapter is based on the following research publication.

Waheed Ullah, Ling Cheng, Fambirai Takawira. "Predictive Syndrome Based Joint Iterative Detection-Decoding Algorithm for Non-Binary LDPC Codes". (Accepted 17 Jan 2021: IEEE Access)

Contribution Statement: The first author is responsible for the conceptualization of the work, experimental data collection, analysis and interpretation, as well as project execution and results collection. The work was done with assistance from the co-authors: Prof. Ling Cheng and Prof. Fambirai Takawira.

Abstract: This chapter addresses the problem of decoding non-binary low density parity check codes(LDPC) over finite field $GF(q)$ using symbol flipping approach. To achieve low complexity reliable communication, three new algorithms for improving the bit error rate performance of the non-binary LDPC decoder are presented. The first type is the symbol flipping decoding algorithm using a flipping function based on the channel reliability to identify the least reliable symbol position. In this algorithm, if the predicted symbol value satisfies the check sum, then the value is declared as correct otherwise the value is adjusted and sent back to the QAM detector. Algorithms 2 in this chapter is an improvement to iterative joint detection-decoding algorithm by using the method of iterative hard decision based majority logic to select the new candidate symbol value. The feedback value to the QAM detector is adjusted by using Euclidean distance between the current symbol and the newly selected symbol value. Algorithm 3 is a low complexity version of Algorithm 2 which is derived by applying a majority voting scheme. In the majority voting scheme, symbols are short listed first by voting and all the computation

are carried out only for the short listed least reliable symbols which significantly lowers the processing complexity. Numerical results and complexity analysis show that the proposed methods have good bit error rate versus complexity trade-off for various applications when compared with some existing algorithms.

4.1 Introduction

Non-binary low density parity check (NB-LDPC) codes defined over $GF(q > 2)$ are an extension of binary LDPC codes [3, 73]. NB-LDPC codes have shown better bit error rate (BER) performance than binary LDPC codes. The NB-LDPC codewords can be transmitted as binary sequence. The codeword symbols defined over $GF(q)$ are mapped to a binary sequence and stream of bit vectors is transmitted over a binary input channel. At the receiver, the NB-LDPC decoder performs de-mapping of the received bit vectors to symbols. The main challenge of LDPC codes defined over high order Galois fields $GF(q > 2)$ is the high computational complexity. Fast Fourier transform (FFT) based sum product algorithm (FFT-SPA) [11] for decoding of non-binary LDPC was proposed to reduce the check node complexity in terms of hardware computation.

In the bit reliability [32] and symbol flipping NB-LDPC decoding algorithms [21, 22] the codeword is transmitted as binary sequence and the received binary sequence is then used as reliability information in the decoder. Reliability based majority logic decoding algorithm is similar to message passing decoding but it sends only the most reliable field message and is considered as the low computational algorithm. In the iterative majority logic decoding (MLgD) algorithm for non-binary LDPC codes, each symbol is iteratively updated by the extrinsic information-sums (EXIs) with the most reliable field element along each edge of the Tanner graph. The iterative reliability based hard (IHRB) and soft (ISRB) MLgD algorithms [23] have lower complexity but perform well only for parity check matrices with high column weights. The two algorithms in [23] are further improved by introducing soft reliability information at the initialization to achieve

better trade-off between complexity and bit error performance by proposing several modifications [24, 27]. One of the drawback of IHRB and ISRB algorithms in literature, is the requirement of large column weights. To overcome this drawback, an improvement to these algorithms was presented in [31] by introducing reliability updates in terms of bit rather than symbols. The bit reliability based decoding algorithm for NB-LDPC is comparatively more efficient and is termed as weighted bit reliability based (wBRB) algorithm [32]. This bit reliability based algorithm requires integer and Galois field operations only which helps to reduce the hardware implementation complexity and processing time. To further enhance the wBRB decoding algorithms, a full bit-reliability based decoding scheme was proposed in [32]. To lower the processing complexity, this algorithm uses the binary representation of non-zero entries of the parity check matrix. Symbol flipping decoding algorithms have low computational complexity but at the cost of reduced performance. A majority logic decision based algorithm was presented in [72], where the symbol position to be flipped is determined by majority decision while the flipped value is calculated from channel output by flipping individual bits of the symbol with low reliability. Another algorithm termed as weighted algorithm B (wt.Algo B) presented in [15] introduces the binary Hamming distance and plurality logic performance improvement. The parallel symbol flipping decoding (PSFD) algorithm in [16], has good performance only for parity check matrix of large column weight. A multiple voting based PSFD (MV-PSFD) algorithm was proposed in [18] for the improvement of PSFD algorithm. The performance of the SRBMP decoding algorithm [58] has been improved by the multiple voting symbol flipping (MV-SF) decoding algorithm [59].

Non-binary LDPC codes can also be transmitted using direct mapping to the high order modulation. The advantage of NB-LDPC codes using high-order q-ary modulations is the direct encoding and decoding over the q-ary constellation as the binary to non-binary mapping and de-mapping operations are not required. Also the non-binary symbol likelihoods are computed directly without any conversion. The mapping and de-mapping operations are cost effective in terms of complexity

and introduce performance degradation that would have to be partially countered by a good choice of mapping and de-mapping at decoder. The NB-LDPC sum-product and its variants can be used to map to the higher order modulation to achieve high data rates. The binary Min-Sum decoding algorithm [74, 75] extended to the non-binary decoding, is known as Extended Min-Sum (EMS) algorithm [52, 69, 70], and it gives better trade-off between hardware complexity and bit error rate performance. The complexity in the Extended Min-Sum decoding largely arises from the computation of the check node (CN). To reduce the check node processing, a Forward-Backward scheme [54][76] was proposed where a serial computation is carried in the hardware and intermediate results are used during that serial computation. However this scheme adds high latency and offers less throughput which becomes more significant when the finite field size of $GF(q)$ is increased. To overcome these challenges, a new hardware aware algorithm called syndrome based EMS algorithm was presented in [77]. This algorithm achieves better bit error rate performance as well as throughput due to increased parallelism.

An M-QAM based hard decision NB-LDPC algorithm was proposed in [78, 79] and is called the iterative joint detection-decoding (IJDD) algorithm. This algorithm uses the M-QAM modulation to get the higher throughput but its performance is poor compared to the q-ary LDPC (QSPA). This algorithm uses the hard decision symbol sequence as feedback in each iteration to the M-QAM detector to move the received messages towards correct direction and reduce the noise effect.

To enhance the bit error rate (BER) performance of the IJDD, an Improved IJDD (IIJDD) algorithm has been proposed in [80, 81] and it shows better bit error rate performance. The Improved IJDD algorithm takes into account the probability of symbols at initialization from channel information and then combines with the plurality logic based reliability information during iterative updates. This iterative reliability update of information is termed as accumulated reliability of a symbol. The accumulated reliability of symbols make the IIJDD algorithm computationally very complex. The IJDD algorithms are used with the majority-logic decodable non-binary LDPC codes with high column weight but show performance loss for

low rate codes with low column weight [66]. As research in the literature is focused mostly on low column or ultra-low column weight LDPC codes, therefore, we have also chosen to use low/ultra-low column weight. In the literature, those symbol flipping NB-LDPC decoders, showing better BER performances for low column weight, are considered as good decoding algorithms LDPC codes using high column weight increase the decoding computational complexity and contribute towards error floor [7, 13, 25]. Therefore, ultra sparse LDPC codes are used to achieve low decoding latency and to overcome error as shown in [8, 9, 14, 28]. Other advantages of ultra-sparse LDPC codes like high girth and better BER performance are given in the references [19, 29, 34].

In this chapter, symbol flipping decoding algorithms are proposed for the QAM based NB-LDPC codes. The proposed algorithms are inspired by the IJDD algorithms [80, 81] and use the concept of iterative hard reliability based majority logic decoding algorithm in [23, 25]. Euclidean distance is used to define the step size of the feedback to the detector to move the symbol towards correct position unlike to [80, 81]. Precise step size of the feedback message plays an important role in improving the BER performance. The IJDD algorithm can show oscillating behaviour if the feedback step size is not controlled properly and results in performance degradation. The IJDD algorithms in literature use the step size as the difference of the first and second maximum numbers of occurrences of a symbol in the extrinsic information-sum as a scaling factor for the symbol feedback information to the QAM detector. This is not the efficient way of optimizing the received sequence to remove the noise by moving the channel output in the correct direction.

The followings are the main contributions in this chapter:

• A new symbol flipping decoding algorithm is presented for QAM modulated NB-LDPC codes. Currently all the symbol flipping algorithms in literature are for BPSK modulation but in this chapter, symbol flipping decoding algorithm for higher order modulation is introduced for the first time. Along with syndrome information, the flipping function of Algorithm 1 uses Euclidean distance with plurality logic to identify the positions of the short listed least reliable variable

nodes. A majority voting scheme is utilised to short list the unreliable variable nodes. The value of the candidate symbol is selected by prediction from the set of the finite field $GF(q)$.

• To achieve better BER performance, an iterative hard decision based majority logic decoding scheme is introduced in Algorithm 2. Through an iterative approach, the feedback to the QAM detector is improved by the Euclidean distance based reliability of the received sequence to remove noise by moving the message in the correct direction. This differs from the IJDD algorithms which use a difference in number of votes between first and second highest voted candidates in conjunction with column weight to move the symbol in correct direction.

• Algorithm 3 is the low complexity version of the Algorithm 2. The majority voting scheme is applied to short list the least reliable nodes and then the symbol values are computed only for those short listed candidate symbols. This algorithm uses only the first reliability values of the hard decision symbols to update the extrinsic information-sums unlike IIJDD algorithm which uses the first and second reliability values of the hard decision symbols.

The rest of the chapter is summarized as follows. Section II describes the relevant literature on non-binary LDPC decoding algorithms. The proposed algorithms on low complexity multiple symbol flipping decoding algorithm are presented in the section III. Section IV presents the results and discussion of the existing and proposed methods, while Section V gives complexity analysis. Section VI concludes the chapter.

4.2 Non-binary LDPC Codes

Consider a regular parity check matrix H with dimension $m \times n$ defined over the Galois field (GF) size $q > 2$ where each element $h_{i,j}(1 \leq i \leq m, 1 \leq j \leq n)$ of H is an element over the $GF(q)$ where $q = 2^r$ and r shows the number of bits in each symbol. Now consider a NB-LDPC code $\mathcal{C}$ of length n for a regular parity check matrix H, such that each row of H has weight of d_c and each column of H has weight of d_v. In the Tanner graph, for parity check matrix H, the non-zero

entries ($h_{i,j} \neq 0$) show the i^{th} check node (CN) connected to the j^{th} variable node (VN) and the Tanner graph edge connection is given by $M(j) = \{i : 1 \leq i \leq m, h_{i,j} \neq 0\}$ and $N(i) = \{j : 1 \leq j \leq n, h_{i,j} \neq 0\}$.

4.2.1 NB-LDPC Codes Using BPSK

Let $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_j, \ldots, \boldsymbol{c}_n)$ be a codeword in $\mathcal{C}$ and the binary form of j^{th} symbol in $\boldsymbol{c}$ is $\boldsymbol{c}_j$= $(c_{j,1}, c_{j,2}, \ldots, c_{j,t}, \ldots, c_{j,r}), 1 \leq t \leq r$. This binary sequence is modulated with binary phase shift keying (BPSK), where 1 is modulated as +1 and 0 is modulated as -1. The BPSK modulated j^{th} symbol $\boldsymbol{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,t}, \ldots, x_{j,r})$ is sent over additive white Gaussian noise (AWGN) channel. At receiver, the binary sequence $\boldsymbol{y}_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,t}, \ldots, y_{j,n})$ is obtained from the transmitted sequence $\boldsymbol{x}_j$ by adding the additive white Gaussian channel noise $\mathcal{N}(0, \sigma^2)$ with two sided power spectral density as $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j$. The hard decision (HD) binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is given by:

$$z_{j,t} = \begin{cases} 1, & \text{if } \; y_{j,t} > 0 \\ 0, & \text{if } \; y_{j,t} \leq 0 \end{cases} \tag{4.1}$$

The log likelihood ratio (LLR) for NB-LDPC codes corresponding to the j^{th} symbol, is computed as follows for the BPSK signalling in AWGN channel [12].

$$\begin{aligned} LLR(\boldsymbol{c}_j) &= ln\left(\frac{Pr(\boldsymbol{y}_j|\boldsymbol{c}_j=\alpha)}{Pr(\boldsymbol{y}_j|\boldsymbol{c}_j=0)}\right) \\ &= ln\left(\frac{Pr(\boldsymbol{y}_j|(c_{j,1}, c_{j,2}, \ldots, c_{j,t}) = \alpha)}{Pr(\boldsymbol{y}_j|(c_{j,1}, c_{j,2}, \ldots, c_{j,t}) = 0)}\right) \end{aligned} \tag{4.2}$$

The i^{th} syndrome of the j^{th} hard decision symbol sequence can be defined for the nonzero element of the parity check matrix $\boldsymbol{H} = [h_{i,j}]_{mn}$ with $h_{i,j} \in GF(q)$.

$$\boldsymbol{s}_i = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j \tag{4.3}$$

If $\boldsymbol{s}_i \neq \mathbf{0}$, it means that one or more of the variable nodes contributing to this check node are incorrect.

4.2.2 NB-LDPC Codes Using QAM

The codeword $\boldsymbol{c} \in GF(q)$ is mapped to the constellation χ of the quadrature amplitude modulation (QAM) before transmission. Defining $\Omega(\cdot)$ as the symbol mapping function for QAM constellation then the codeword $\boldsymbol{c}$ is mapped as $\boldsymbol{x}_j = \Omega(\boldsymbol{c}_j) \in \chi$ with complex signal vector $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_j, \ldots, \boldsymbol{x}_n)$ and is transmitted over the additive white Gaussian noise (AWGN) channel. The received symbol sequence $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_j, \ldots, \boldsymbol{y}_n)$ is obtained from the transmitted sequence $\boldsymbol{x}$ after adding the complex additive white Gaussian channel noise $\mathbf{n} \rightarrow \mathcal{CN}(0, \sigma^2)$ with zero mean and two sided power spectral density $N_0/2$ as $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j$.

At the receiver side, the detector estimate each symbol $\hat{\boldsymbol{x}_j}$ using the maximum likelihood decision rule.

$$\hat{\boldsymbol{x}}_j = \arg\min_{\boldsymbol{x} \in \chi} \|\boldsymbol{y}_j - \boldsymbol{x}\| \tag{4.4}$$

for $j = 1, 2, ...n$.

The above equation (4.4) becomes very complex for high order QAM. This complexity can be reduced by defining a radius D of a sphere and the received symbol is estimated for those symbols in the region of D.

$$\hat{\boldsymbol{x}}_j = \arg\min_{\boldsymbol{x} \in \chi} \|\boldsymbol{y}_j - \boldsymbol{x}\| \leq D \tag{4.5}$$

In this chapter we use the QAM constellation size χ equal to the Galois field size q [80]. For the given code rate $r_c = k/n$ where k is the information length, the spectral efficiency for this coded scheme is $\zeta = r_c log_2|\chi|$ bits per symbol.

The i^{th} syndrome of the j^{th} hard decision symbol sequence for $h_{i,j} \neq 0$ can be defined as:

$$\boldsymbol{s}_i = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j \tag{4.6}$$

If $\boldsymbol{s}_i \neq \mathbf{0}$, it means that one or more of the variable nodes contributing to this check node are incorrect.

At the initialization of the non-binary sum-product and min-sum decoding algorithms, the LLR value for each symbol of the received sequences [82] is computed with the assumption that all symbols have equal probability.

$$LLR(c) = ln\left(\frac{Pr(\boldsymbol{y}|\hat{\boldsymbol{x}})}{Pr(\boldsymbol{y}|\boldsymbol{x})}\right) \tag{4.7}$$

where $\hat{\boldsymbol{x}}$ is the symbol value which maximizes the probability function $Pr(\boldsymbol{y}|\hat{\boldsymbol{x}})$, i.e.

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in GF(q)}{\operatorname{argmax}}\{Pr(\boldsymbol{y}|\boldsymbol{x})\}. \tag{4.8}$$

If the value of $LLR(\boldsymbol{c}) = 0$ or close to 0, it means more reliability and vice versa.

When M-QAM coded modulation is used, the received symbol sequence contains noise which shifts the symbol from the position in the QAM constellation. Therefore, it is hard to detect correctly the symbol value. The Euclidean distance based likelihood method is used to calculate the approximate position of the received symbol using LLR which is given as:

$$LLR(\alpha) = \frac{2}{\sigma^2}(d(\boldsymbol{y}, \alpha)^2 - d(\boldsymbol{y}, \hat{\alpha}^2)) \tag{4.9}$$

where $\hat{\alpha}^2$ is the closest QAM point of the received symbol $\boldsymbol{y}$ and α is the primitive element of $GF(q)$ in the QAM constellation.

4.2.3 Non-Binary LDPC Decoders

A review of NB-LDPC codes and related decoding algorithms are presented in this section but particular emphasis is on the IJDD algorithms [78, 79] using QAM modulation.

For a given NB-LDPC code C over $GF(q)$, the j^{th} received symbol $\boldsymbol{y}_j$, can be any of the q element in the $GF(q)$. If α is the primitive element of $GF(q)$, then the q elements of the $GF(q)$ are expressed as $GF(q) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$. Therefore, there are q messages passed between the check and variable nodes in the classic belief propagation NB-LDPC decoding algorithm. There are q probable information for each of the channel received symbols. To satisfy the check sum in equation (4.6) for q symbols, the complexity is in order of $O(q^{d_c-2})$.

The low complexity hard decision symbol flipping decoding based on the majority logic decision is known as generalized algorithm B [15]. This algorithm is based on plurality logic which counts the number of occurrences of each symbol. For the preset threshold t, if the counts of more than one finite field symbol over $GF(q)$ exceed t, choose in favor of that received symbol, otherwise choose some common symbol. This algorithm is further improved by weighting coefficients based on the Hamming distance between finite field symbol and extrinsic information-sums. This improved algorithm is called weighted algorithm B [15]. In this algorithm, the counts of occurrences of a finite field symbol is multiplied by the weighting coefficients assigned to the Hamming distance. The decision is taken based on the largest product. The optimal set of the weights was not determined in the weighted algorithm B and was left as future work. For the given hard decision symbol $\boldsymbol{z}_j$, the normalized extrinsic information-sum (EXI) for the k^{th} iteration is calculated as:

$$\sigma_{i,j}^{(k)} = h_{ij}^{-1} \sum_{j' \in N(i) \setminus j} h_{i,j'} \boldsymbol{z}_{j'}^{(k)} \tag{4.10}$$

for $1 \leq i \leq m, j \in N(i)$.

The generalized algorithm B (Algo B) decision for estimated correct symbol is obtained using the plurality logic as:

$$\hat{v}_j^{(k)} = \begin{cases} \alpha & if \;\; \eta(\alpha) \geq T \\ \boldsymbol{z}_j^{(k)} & else \end{cases} \tag{4.11}$$

Here $\eta(\alpha)$ is the number of the occurrences of $\alpha \in GF(q)$ in the extrinsic information-sum $\sigma_{i,j}^{(k)}$ for α corresponding to the largest count of $\eta(\alpha)$. T is the preset threshold determined through simulations. In this chapter, it is set to the column weight of the parity check matrix.

In the IJDD algorithm [78, 79], the main idea is to correct the symbol vector $\boldsymbol{y}$ by removing noise in an iterative way and moving the received vector $\boldsymbol{y}$ closer towards the transmitted symbol point.

At the receiver side, the detector estimates each symbol $\hat{\boldsymbol{x}}_j^{(k)}$ using the maximum likelihood decision rule. The hard decision symbol $\boldsymbol{z}_j^{(k)}$ is made available to the NB-LDPC decoder after de-mapping the detector output $\boldsymbol{x}_j$ as follow:

$$\boldsymbol{z}_j^{(k)} = \Omega^{-1}(\hat{\boldsymbol{x}}_j^{(k)}) \tag{4.12}$$

From equation (4.11), let η_{max} be the largest reliability and η_{sub} as the second largest reliability of the symbol in $\sigma_{i,j}^{(k)} \in GF(q)$ at the k^{th} iteration, then $\Delta\eta_j^{(k)} = \eta_{max}^{(k)} - \eta_{sub}^{(k)}$ which represents the voting difference between two high reliable candidates. The pair of information $(v_j^{(k)}, \Delta\eta_j^{(k)})$ is provided to the QAM detector as feedback. The detector updates the received sequence $\boldsymbol{y}_j^{(k)}$ with the information provided by the LDPC decoder. Let $d(\boldsymbol{y}_j^{(k)}, r)$ be a search sphere with radius r, centered at $\boldsymbol{y}_j^{(k)}$. The following is the update rule at detector:

• If $\Omega(v_j^{(k)}) \in d(\boldsymbol{y}_j^{(k)}, r)$, then

$$\boldsymbol{y}_j^{(k+1)} = \boldsymbol{y}_j^{(k)} + \lambda^{(k)}\boldsymbol{\beta}^{(k)}, \tag{4.13}$$

• Otherwise $\boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)}$.

Here $\lambda^{(k)}$ and $\boldsymbol{\beta}^{(k)}$ are calculated as follow:

$$\lambda^{(k)} = \frac{\Delta\eta^{(k)}}{d_v} \tag{4.14}$$

and

$$\boldsymbol{\beta}^{(k)} = \begin{cases} \hat{\boldsymbol{x}}_j^{(k)} - \boldsymbol{y}_j^{(k)}, & \text{if } \Omega(v^{(k)}) = \hat{\boldsymbol{x}}_j^{(k)} \\ \Omega(v^{(k)}) - \hat{\boldsymbol{x}}_j^{(k)}, & \text{otherwise.} \end{cases} \tag{4.15}$$

The IJDD algorithm [79] is further improved by combining the plurality logic based scheme with the accumulated reliability [80, 81] from channel. The reliability vector $\boldsymbol{P}_j^{(k)}$ of the j^{th} variable node at k^{th} iteration for each element $\alpha \in GF(q)$ is given by:

$$\boldsymbol{P}_j^{(k)} = [p_j^{(k)}(0), p_j^{(k)}(1), \ldots, p_j^{(k)}(q-1)] \tag{4.16}$$

where $p_j^{(k)}(\alpha)$ is the probability that the estimated value of $\hat{v}_j^{(k)}$ is equal to the element $\alpha \in GF(q)$. The reliability vector $\boldsymbol{P}_j^{(k)}$ at $k = 1$ is initialized with likelihood values from the channel.

The second reliability metric $\bar{p}^{(k)}(\alpha)$ for the j^{th} variable node is calculated by the following equation:

$$\bar{p}^{(k)}(\alpha) = \frac{\eta_j^{(k)}(\alpha)}{\Phi} \tag{4.17}$$

where $\eta_j^{(k)}(\alpha)$ is the number of the votes of element α and Φ denotes the sum of the votes obtained by each candidate symbol in $GF(q)$ for the j^{th} variable node. If any of the element α has got zero vote, then it is assigned a value equal to half of the minimum votes of an element in (4.17) to avoid premature exclusion.

In order to combine the effect of reliability at current iteration k with the reliability at $(k-1)^{th}$, the normalised reliability for an element α of the j^{th} variable node is given by:

$$p^{(k)}(\alpha) = \frac{\bar{p}_j^{(k)}(\alpha) p_j^{(k-1)}(\alpha)}{\sum_{\dot{\alpha} \in GF(q)} \bar{p}_j^{(k)}(\dot{\alpha}) p_j^{(k-1)}(\dot{\alpha})} \tag{4.18}$$

such that $\sum_{\alpha \in GF(q)} p^{(k)}(\alpha) = 1$. The candidate symbol value $\hat{v}^{(k)_j}$ is selected based on the highest probability and $\delta\eta_j^{(k)}$ is calculated based on the first and second

highest reliability values of the two candidates.

The accumulated reliability based IJDD algorithm is further improved by using the first and second hard decision based symbols for each of the j^{th} variable node to improve the bit error rate performance and is termed as Improved IJDD(IIJDD) algorithm [80].

In the improved IJDD algorithm, for each check sum $\boldsymbol{s}_i$ connecting to d_c variable nodes, a variable node, say v_j with least reliability is chosen based on the Euclidean distance. Let $(\boldsymbol{z}_{j,1})$ and $(\boldsymbol{z}_{j,2})$ be the two most reliable symbols passed from the QAM detector to the variable node v_j in two configurations and $\boldsymbol{z}_l \in GF(q)$ be the least reliable symbol, then the two check-to-variable messages from the detector are $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{j,1}, \ldots \boldsymbol{z}_{d_c}\} \backslash \{\boldsymbol{z}_l, l \neq j\}$ and $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{j,2}, \ldots \boldsymbol{z}_{d_c}\} \backslash \{\boldsymbol{z}_l, l \neq j\}$. For the least variable node v_j, the most reliable symbol $\boldsymbol{z}_{j,1}^{(k)}$ and the next most reliable symbol $\boldsymbol{z}_{j,2}^{(k)}$ are delivered to the associated check node $\boldsymbol{s}_i$ while for other variable node connected to this check node $\boldsymbol{s}_i$, only the most reliable symbol $\boldsymbol{z}_{j,1}^{(k)}$ is delivered.

4.3 Proposed NB-LDPC Decoding Algorithms

4.3.1 System Model

This chapter uses the most common type of QAM modulation which is rectangular QAM, where the constellation points are arranged in a square grid. Figure 4.1 shows the receiver structure to reduce the complexity of the QAM detector where the received symbol quadrant is first identified and then the symbols in that quadrant are used to estimate the correct symbol.

The non-binary LDPC code defined over the $GF(q)$ is used in connection with the QAM constellation such that the size of the coded symbol is the same as the QAM constellation. The NB-LDPC encoder encodes information symbols to construct a codeword of length n. These n symbols are modulated by the QAM module and are transmitted over wireless channel. The complex AWGN channel $\mathcal{CN}(0, \sigma^2)$

with zero mean and two sided power spectral density $N_0/2$, affects each of the transmitted symbol as shown below:

$$\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j \tag{4.19}$$

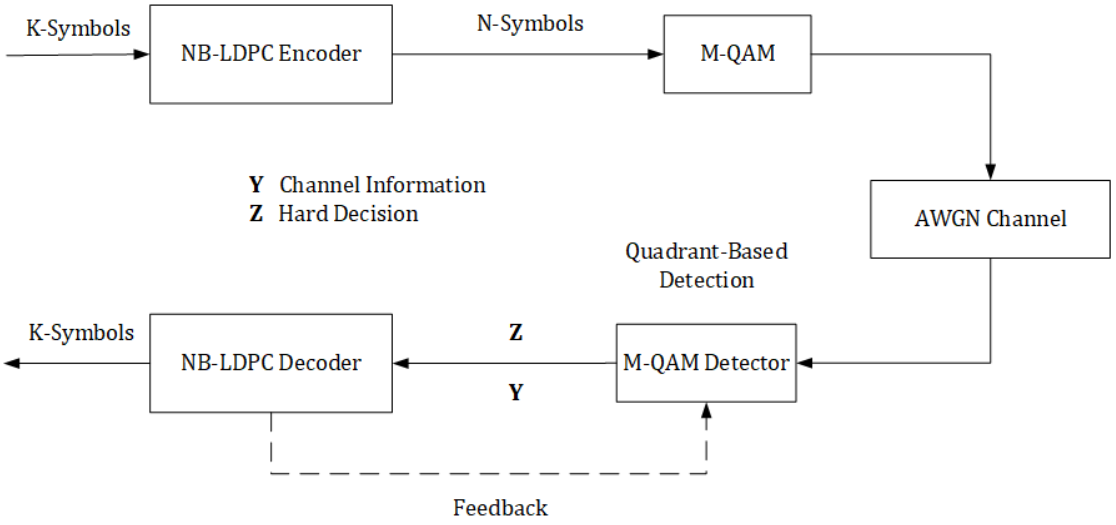


FIGURE 4.1: System Model: NB-LDPC coded QAM modulation

The quadrant based QAM detector estimates the transmitted signal as follows:

$$\hat{\boldsymbol{x}}_j^{(k)} = \underset{\boldsymbol{x}_\mathrm{q} \in \chi_\mathrm{q}}{\arg\min} \|\boldsymbol{y}_j^{(k)} - \boldsymbol{x}_\mathrm{q}\| \tag{4.20}$$

where χ_q contains all the constellation symbol in the q^th quadrant. The quadrant χ_q is determined from the sign of the real and imaginary parts of the symbol y_j. This quadrant based QAM (QQAM) detection reduces the complexity enormously as only four symbols are required to estimate the correct symbol. The QAM ML detection used in [80] as shown in equation (4.4) requires all the symbols in constellation which makes the estimation of correct symbols very complex in computation.

This proposed method is very different than the QAM detector presented in [80, 81] where maximum likelihood detection is used. The received symbol sequence estimated using reduced complexity quadrant based QAM detection is sent to the symbol flipping NB-LDPC decoder. The decoder predicts the unreliable symbol

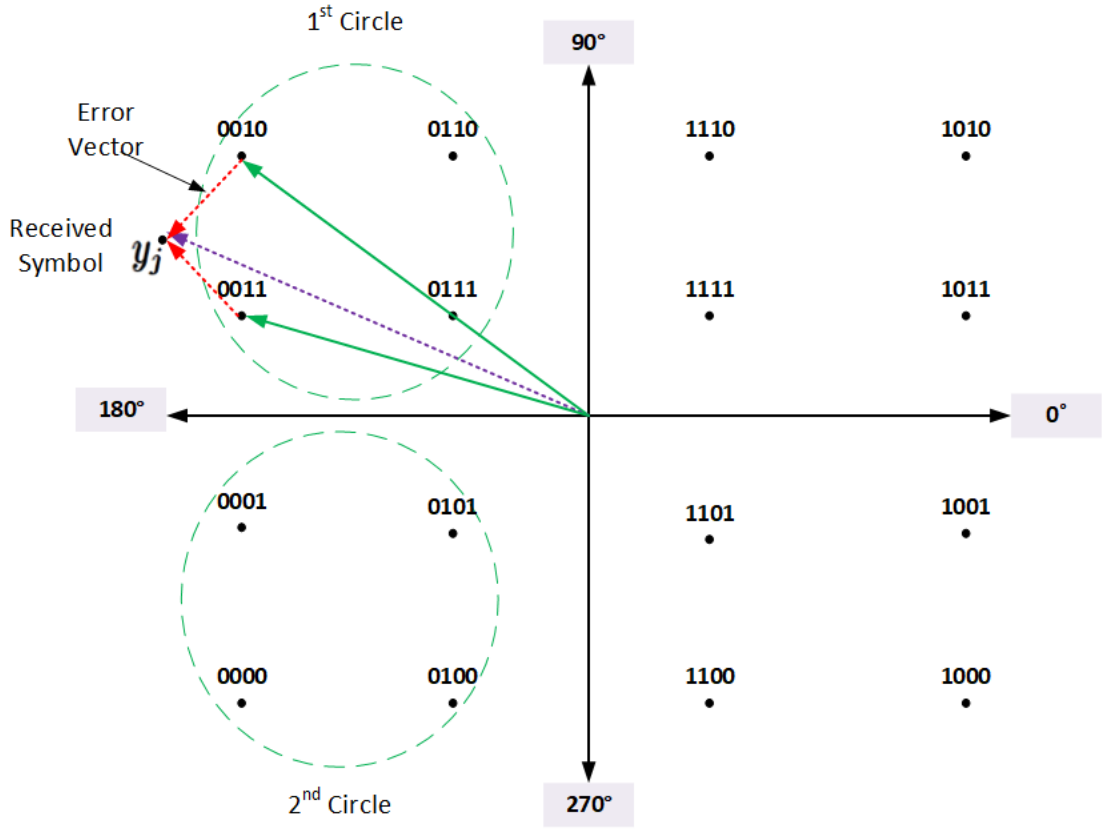


FIGURE 4.2: 16-QAM Gray coded constellation and the received symbol

value for each of the failed variable nodes contributing to failure of check-sum $\boldsymbol{s}_i$ at k^{th} iteration. Let the check node $\boldsymbol{s}_i$ have d_c number of connected variable nodes $v_1, v_2, \ldots, v_{d_c}$. For each unreliable node, a set of symbols from the $GF(q)$ table is predicted where each predicted symbol is different in one bit from the unreliable symbols with the assumption that AWGN channel has affected one bit in each symbol at the most. Also keeping in view that, in Gray coded QAM, the symbols in the same quadrant are different by one bit. Therefore, it is more likely to choose the correct symbol for the least reliable symbol of the connected failed check node.

In case the predicted symbol does not satisfy the check-sum, the NB-LDPC decoder updates the information and sends back the newly adjusted symbols to the$M$-QAM detector. The newly adjusted signal may move to another quadrant due to the iterative adjustment and then all the symbols in the new quadrant are used to estimate the newly adjusted symbol.

4.3.2 Proposed Algorithm 1

In this section, a new symbol flipping decoding algorithm for QAM modulated NB-LDPC codes is presented. Reliability information are added to the flipping function to decide the accurate symbol position in error as shown in Figure 4.3. For this selected position, a new candidate symbol value is predicted for the least reliable variable node connected to the failed check. In this proposed algorithm, only those variable nodes contributing to the failed checks are considered for flipping in order to maintain low complexity and fast convergence. The voting scheme adopted in parallel symbol flipping[16, 18] is used for selection of symbol positions to be flipped at the k^{th} iteration. In Figure 4.3, the short listing of least reliable symbols is completed first and the p number of variable nodes are sent to the next module for computing the flipping function. The flipping function decide the number of variable nodes to be flipped whereas the parity check module selects the flipping value for the positions selected by flipping function. The output of

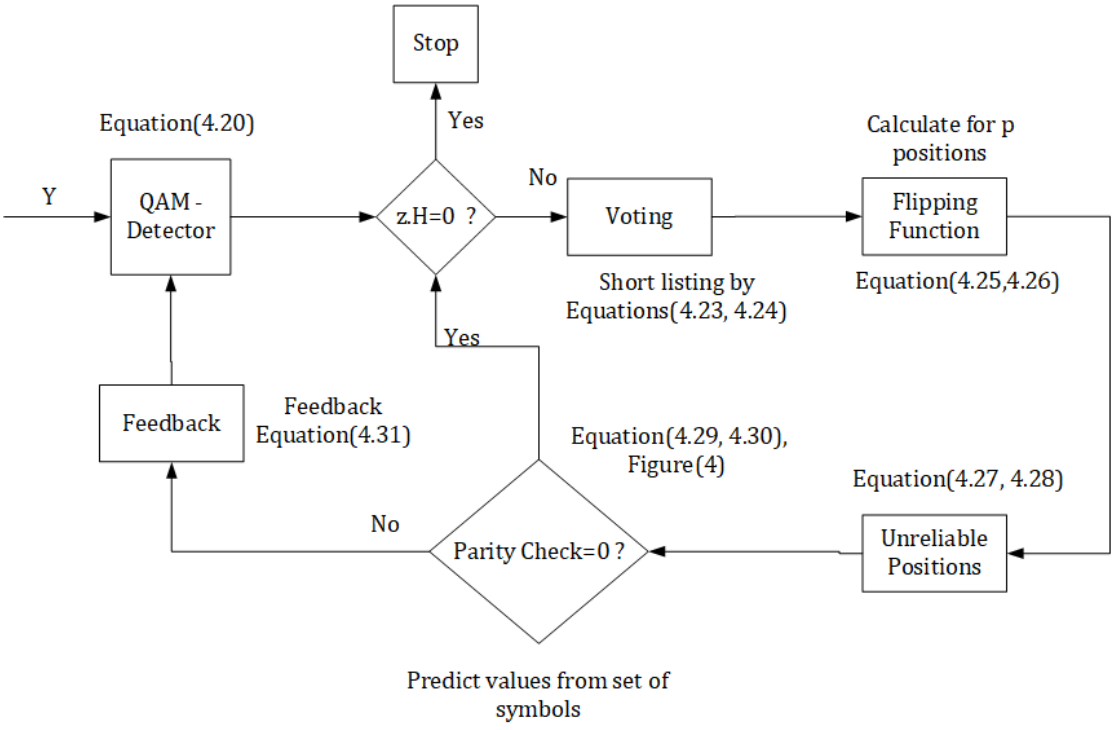


FIGURE 4.3: Proposed Algorithm 1 Execution Cycle

the QAM detector is fed into the check sum. The variable nodes contributing to

the successful check sum are given as:

$$\bar{\boldsymbol{v}} = \{i' : \ \boldsymbol{s}_i = \mathbf{0}, 1 \leq i \leq m, j \in M(i')\} \tag{4.21}$$

Those variable nodes contributing to the failed checks belong to ψ which is the vector containing the number of positions to be flipped, then it must be the subset of $\boldsymbol{\tau}$ and is defined as:

$$\boldsymbol{\psi} \in \boldsymbol{\tau} = \{\boldsymbol{v}(j) \backslash \boldsymbol{v} \cap \bar{\boldsymbol{v}}\}. \tag{4.22}$$

Here $\boldsymbol{\tau}$ is a set of erroneous variable nodes and equation (4.22) contains all variable nodes contributing to failed checks. If a single symbol is flipped in each iteration, then $\boldsymbol{\psi}$=1.

4.3.2.1 Variable Node Selection by Voting

In the voting scheme [16, 18], each unsatisfied check node gives one vote to the relevant variable node. The j^{th} variable node then collects all the votes, say $V_j^{(k)}$, from the failed check nodes at k^{th} iterations.

$$V_j^{(k)} = \sum_{i \in M(j)} V_{i,j}^{(k)} \tag{4.23}$$

where $V_{i,j}^{(k)} = 1$ if $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, otherwise $V_{i,j}^{(k)} = 0$. For the accumulated voting $V_j^{(k)}$ equal or greater than a predefined threshold V_{th}, those variable nodes fulfilling the condition $V_j \geq V_{th}$ will be passed to calculate the flipping function $E_j^{(k)}$ through equation (4.25). This reduces the computational complexity from n number of variable nodes to just few variable nodes, say $\boldsymbol{\delta}$. The values of $\boldsymbol{\delta}$ at k^{th}can be calculated simply as follows:

$$\delta_{j'}^{(k)} = j, \quad \text{if } V_j \geq V_{th} \tag{4.24}$$

for $1 \leq j' \leq p$.

Here p shows the total number of variable nodes having maximum votes and $\boldsymbol{\delta}^{(k)}$ contains all positions for those variable nodes (VNs). In other words, these are the number of variable nodes having less reliable information and need to be replaced with more reliable symbols. The positions of these variable nodes are stored as $\delta_{j'}^{(k)}$. The multiple voting method presented in [18] define two voting levels $\zeta_0 > \zeta_1 > 0$, using the same voting function defined in (4.24). For $\boldsymbol{s}_j^{(k)} \neq 0$, $V_{ij}^{(k)} = \zeta_0$ for the largest flipping function and $V_{ij}^{(k)} = \zeta_1$ for the VN with the second largest flipping function. In this chapter, a voting scheme is used for short listing of least reliable symbols which reduces the memory consumption and computational complexity.

4.3.2.2 QAM Based Flipping Function

Each unsatisfied syndrome(check-node) $\boldsymbol{s}_i^{(k)}$ provides one vote to the variable node and the variable node accumulates the votes. To find the position of variable nodes with less reliability, the following expression is used for $1 \leq j' \leq p$:

$$F_{j'}^{(k)} = \sum_{i \in M(j')} (2\boldsymbol{s}_i^{(k)} - 1)\xi_{j'}^{(k)} - \eta_{j'}^{(k)}(\alpha) \tag{4.25}$$

where $\eta_{j'}^{(k)}(\alpha)$ is number of occurrence of $\alpha \in GF(q)$ in the extrinsic information-sum $\sigma_{i,j}^{(k)}$ and $\xi_{j'}^{(k)}$ contains the reliability information derived from the channel received sequence by the following mathematical expression.

$$\xi_{j'}^{(k)} = ||\boldsymbol{y}_{j'}^{(k)} - \hat{\boldsymbol{x}}_{j'}^{(k)}||. \tag{4.26}$$

The flipping function in equation (4.25) is derived from BPSK based symbol flipping decoding algorithm in [14]. The proposed algorithm finds iteratively the valid codeword from vector space over $GF(q)$. At the k^{th} iteration, if $\boldsymbol{s}_i^{(k)} = \boldsymbol{y}_j^{(k)} h_{i,j} \neq 0$, the aim is to perturb $\boldsymbol{y}^{(k)}$ and create a new sequence of candidate symbols $\boldsymbol{y}^{(k+1)}$. Two steps are required to choose a new candidate symbol value: 1) compute the flipping function to identify the flipped position; 2) candidate symbol value selection by predictive syndrome method as shown in Figure 4.4.

The position to be flipped is determined for the variable nodes having less reliability information by the following equation.

$$j^{*(k)} = \underset{j'}{\text{argmax}}(F^{(k)}). \tag{4.27}$$

Importantly, $j^{*(k)}$ will belong to d_v number of check nodes. Therefore, to lower the complexity, d_v should be as small as possible. Let $\boldsymbol{\tau}$ be the threshold to determine the number of symbols to be flipped, then it can be determined in conjunction wih the column weight d_v. If ρ is the maximum number of symbols to be flipped per iteration then it can be determined as:

$$\rho = \begin{cases} \rho_1 & \text{if} \qquad \boldsymbol{\tau} \geq \varepsilon_1 d_v \\ \rho_2 & \text{else,} \end{cases} \tag{4.28}$$

where ε_1 is an integer value and $\rho_1 > \rho_2$. To find the position of the candidate symbol value, the equation for syndrome based symbol value prediction is written as:

$$\boldsymbol{s}_i^{(k)} = \sum_{j'=1}^{d_c} h_{i,j'} \boldsymbol{z}_{j'}^{(k)} = h_{i,1}\boldsymbol{z}_1^{(k)} + h_{i,2}\boldsymbol{z}_2^{(k)} + \ldots + h_{i,d_c}\boldsymbol{z}_{d_c}^{(k)}. \tag{4.29}$$

In equation (4.29), let $\boldsymbol{z}_2^{(k)}$ be in error and contributes to the failure of the check node. To choose a correct candidate symbol, the symbol value is predicted from the set of the symbols which are different in one bit. If $\boldsymbol{s}_i^{(k)}$ is satisfied, then prediction process is terminated otherwise we choose another check node connected to $\boldsymbol{z}_{j'}^{(k)}$ until d_v number of check nodes. If all these check nodes fail, then we choose the last predicted value of the symbol and adjust $\boldsymbol{z}_2^{(k)}$ as shown in the following section.

4.3.2.3 Symbol Value Selection

To replace an unreliable symbol, the candidate symbol value selection is important. To choose a correct symbol value, a symbol prediction method is used from set of values in $GF(q)$. A symbol in the same quadrant as the received symbol is selected as correct symbol which satisfies the parity check equation $\boldsymbol{s}_i^{(k)} = \mathbf{0}$.

Let $\boldsymbol{z}_{j'}^{(k)}$ be the variable node symbol considered to be the least reliable symbol connected to the failed check node $\boldsymbol{s}_i^{(k)}$. To satisfy this parity check equation, a symbol is predicted for the selected position of hard decision symbol $\boldsymbol{z}_{j'}^{(k)}$ such that $\boldsymbol{z}_{j'}^{(k)} \neq \ddot{\boldsymbol{z}}_{j'}^{(k)}$ and $\ddot{\boldsymbol{z}}_{j'}^{(k)}$ has all possible symbol values of the $GF(q)$ inside the quadrant. Therefore, the possible candidate symbol values from that quadrant for each position of $\ddot{\boldsymbol{z}}_{j'}^{(k)}$ are given below:

$$\ddot{\Gamma}_{j'}^{(k)} = \{\ddot{\boldsymbol{z}}_{j'}^{(k)} : \ddot{\boldsymbol{z}}_{j'}^{(k)} \in GF(q), \ddot{\boldsymbol{z}}_{j'}^{(k)} \neq \boldsymbol{z}_{j'}^{(k)}\} \tag{4.30}$$

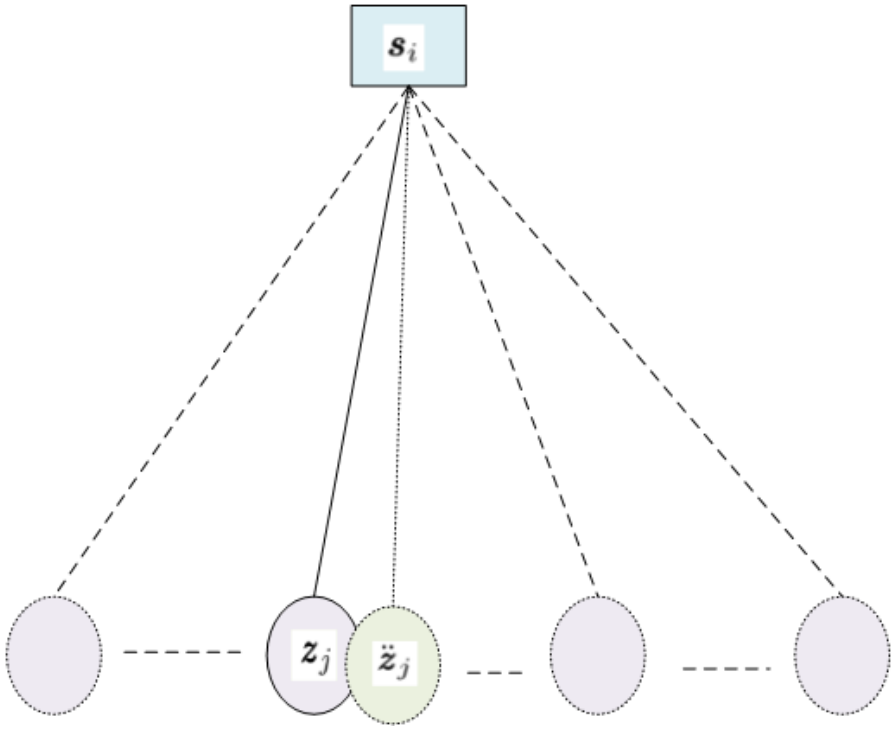


FIGURE 4.4: Symbol Value Prediction

The candidate symbol value prediction in equation (4.29) is illustrated in Figure 4.4. If $\boldsymbol{z}_{j'}$ is the least reliable symbol connected to check node $\boldsymbol{s}_i$ at k^{th} iteration, then symbol $\ddot{z}_{j'}$ is the predicted value such as $\boldsymbol{z}_{j'} \neq \ddot{\boldsymbol{z}}_{j'}$ and $\ddot{\boldsymbol{z}}_{j'} \in \ddot{\Gamma}_{j'}$ which satisfies the parity check equation i.e $\boldsymbol{s}_i = \mathbf{0}$.

4.3.2.4 Feedback to Detector

In this proposed algorithm, if the predicted value does not satisfy the check equation, then the value is adjusted and sent back to the QAM detector as follows:

$$\boldsymbol{y}_{j'}^{(k+1)} = \boldsymbol{y}_{j'}^{(k)} + f(\lambda^{(k)})\boldsymbol{\beta}^{(k)}. \tag{4.31}$$

Here f is a scaling factor and is determined through simulations. The other parameters $\lambda^{(k)}$ and $\boldsymbol{\beta}^{(k)}$ are calculated in the following equations. $\lambda^{(k)}$ is the $l2$-norm used to scale the second term of the equation (4.31).

For the selected candidate symbol value $v^{(k)}$, then $\lambda^{(k)}$ and $\boldsymbol{\beta}^{(k)}$ are calculated as follows:

$$\boldsymbol{\beta}^{(k)} = \begin{cases} \hat{\boldsymbol{x}}_{j'}^{(k)} - \boldsymbol{y}_{j'}^{(k)}, & \text{if } \Omega(v^{(k)}) = \hat{\boldsymbol{x}}_{j'}^{(k)} \\ \Omega(v^{(k)}) - \hat{\boldsymbol{x}}_{j'}^{(k)}, & \text{otherwise} \end{cases} \tag{4.32}$$

$$\lambda^{(k)} = \begin{cases} ||\hat{\boldsymbol{x}}_{j'}^{(k)} - \boldsymbol{y}_{j'}^{(k)}||, & \text{if } \Omega(v^{(k)}) = \hat{\boldsymbol{x}}_{j'}^{(k)} \\ ||\Omega(v^{(k)}) - \hat{\boldsymbol{x}}_{j'}^{(k)}||, & \text{otherwise} \end{cases} \tag{4.33}$$

Equations (4.32) and (4.33) give the step size to move the received symbol towards the correct position by removing the noise in a systematic way. The difference between the received symbol $\boldsymbol{y}_{j'}^{(k)}$ and the selected candidate symbol value determines the step size more accurately. The computed value of $\lambda^{(k)}$ can be large and needs to be truncated, otherwise this can lead to large step size and as a result the symbol will be moved to a far off location in the QAM constellation. The step size truncation is specially important in higher order QAM modulation as the number of symbols get congested and the distance between them decrease making it hard to estimate the correct symbol.

Continuous Loop Detection: A continuous loop detection procedure is adopted in this chapter and positions of all the flipped symbols are recorded. The symbol flipped in an iteration is not flipped again in the successive iterations. In case the new candidate symbol does not satisfy the check equation, then it is re-adjusted

and sent to the QAM detector. This loop detection procedure is applied to algorithm 1 only while the proposed algorithms 2 and 3 do not use any loop detection. For example, if a test flag (TF) is defined to store position of all flipping symbols, say $TF = [2, 4, 19]$ at k^{th} iteration and in next iterations these symbols are not flipped until the TF equal to total number of those variable nodes selected by the voting and then the TF is reset to zero for storing the symbol positions from next iteration.

Proposed Algorithm 1

1. Initialization: For $1 \leq j \leq n$, $1 \leq l \leq q$ and $1 \leq i \leq m$.
The hard decision symbols $\boldsymbol{z_j}$ is obtained from the received symbol $\boldsymbol{y_j}$ after de-mapping by using (4.20) and (4.12).
2. Set max iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Determine the variable nodes contributing to failed check-sum by (4.21) and (4.22).
5. Calculate the votes for each of the variable node by using (4.23) and short list the least reliable symbol positions by (4.24).
6. For $1 \leq j' \leq p$, compute the flipping function for the selected number of positions by using (4.26) and (4.29).
7. Predict the symbol values $\ddot{\boldsymbol{z}}_{j'}^{(k)}$ by (4.30) for the chosen number of the symbol positions. If $\boldsymbol{s}_i^{(k)} = \mathbf{0}$, update the symbol $\boldsymbol{z}_{j'}^{(k)}$ and go to step 2.
8. If $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, adjust the symbol by using (4.31).
9. The QAM detector updates the hard decision symbol $\boldsymbol{z}_{j'}^{(k)}$ based on the feedback.
10. $k = k + 1$ and go to step 3.

4.3.3 Proposed Algorithm 2

In this proposed scheme, iterative hard reliability based majority logic decoding algorithm in [25, 83] is used which offers very low complexity at the initialization as well as to compute iteratively the hard reliability of the symbols as shown in Figure 4.5. Secondly Euclidean distance based feedback is used to control the step size of the symbol in the QAM constellation and move the symbol towards correct position by removing noise. Precise step size of the feedback scheme play an important role in improving the BER performance. The IJDD algorithm can show oscillating behaviour if the feedback step size is not controlled properly which can result in performance degradation.

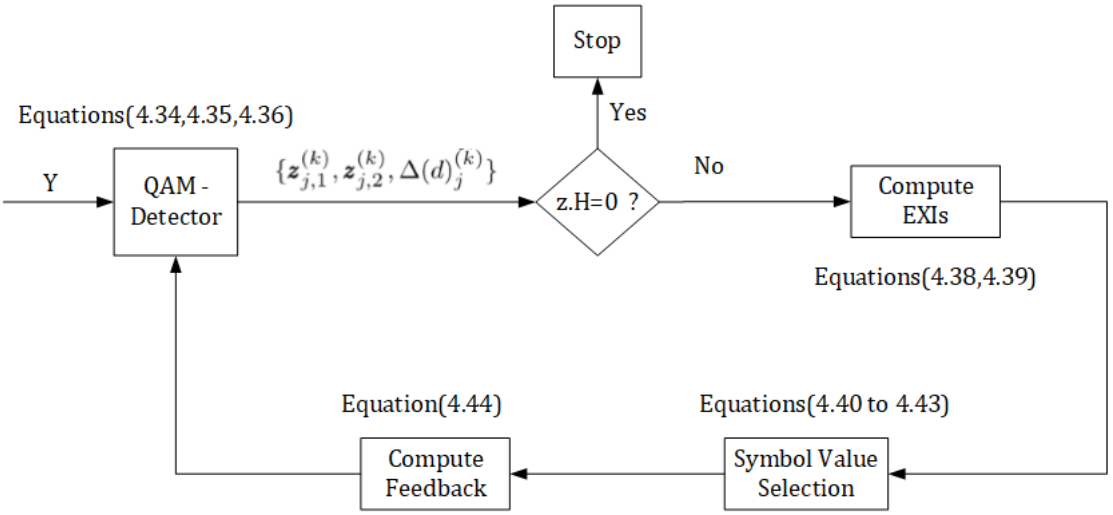


FIGURE 4.5: Proposed Algorithm 2 Execution Cycle

In the proposed Algorithm 2, for each check sum $\boldsymbol{s}_i$ connecting to d_c variable nodes, a variable node, say v_j with least reliability is chosen based on the Euclidean distance same as in [80]. Let $\boldsymbol{z}_{j,1}$ and $\boldsymbol{z}_{j,2}$ be in the quadrant q and are assumed as the two most reliable symbols passed from the QAM detector to the variable node v_j in two configurations. Suppose $\boldsymbol{z}_l \in GF(q)$ is the least reliable symbol, then the two check-to-variable messages from the detector are $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{j,1}, \ldots \boldsymbol{z}_{d_c}\} \backslash \{\boldsymbol{z}_l, l \neq j\}$ and $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{j,2}, \ldots \boldsymbol{z}_{d_c}\} \backslash \{\boldsymbol{z}_l, l \neq j\}$. For each least reliable variable node among the nodes connected by the same check node, the detector generates two most reliable symbols as follows:

$$\hat{\boldsymbol{x}}_{j,1}^{(k)} = \underset{\boldsymbol{x}_{\mathrm{q}} \in \chi_{\mathrm{q}}}{\arg\min} \|\boldsymbol{y}_j^{(k)} - \boldsymbol{x}_{\mathrm{q}}\| \tag{4.34}$$

and the next reliable symbol is generated by:

$$\hat{\boldsymbol{x}}_{j,2}^{(k)} = \underset{\boldsymbol{x}_{\mathrm{q}} \in \chi_{\mathrm{q}}, \boldsymbol{x}_{\mathrm{q}} \neq \hat{\boldsymbol{x}}_{j,1}^{(k)}}{\arg\min} \|\boldsymbol{y}_j^{(k)} - \boldsymbol{x}_{\mathrm{q}}\| \tag{4.35}$$

Let $d_{j,1}^{(k)} = \|\boldsymbol{y}_j^{(k)} - \hat{\boldsymbol{x}}_{j,1}^{(k)}\|$ and $d_{j,2}^{k} = \|\boldsymbol{y}_j^{(k)} - \hat{\boldsymbol{x}}_{j,2}^{(k)}\|$ be the radiabilities for the two symbols $\hat{\boldsymbol{x}}_{j,1}^{(k)}$ and $\hat{\boldsymbol{x}}_{j,2}^{(k)}$ and the reliability metric for each variable node v_j is expressed as:

$$\Delta d_j^{(k)} = |d_{j,1}^{(k)} - d_{j,2}^{(k)}| \tag{4.36}$$

$\Delta d_j^{(k)}$ represent the reliability of the symbol for corresponding VN. Smaller value of the $\Delta d_j^{(k)}$ shows less reliability of the variable node and vice versa. After demapping, $\boldsymbol{z}_{j,1}^{k} = \Omega^{-1}\boldsymbol{x}_{j,1}^{(k)}$ and $\boldsymbol{z}_{j,2}^{(k)} = \Omega^{-1}\boldsymbol{x}_{j,2}^{(k)}$, the pair of information $\{\boldsymbol{z}_{j,1}^{(k)}, \boldsymbol{z}_{j,2}^{(k)}, \Delta(d)_j^{(k)}\}$ are sent to the NB-LDPC decoder at $k+1^{th}$ iteration. The position for least reliable variable node v_j among all others VNs connecting to the same check node s_i, is selected as:

$$j = \underset{j \in N(i)}{\arg\min} \Delta d_l, \qquad l = 1, 2, \ldots, d_c \tag{4.37}$$

For this variable node v_j, most reliable symbol $\boldsymbol{z}_{j,1}^{(k)}$ and the next most reliable symbol $\boldsymbol{z}_{j,2}^{(k)}$ are delivered to the associated check node $\boldsymbol{s}_i$ while for other variable node connected to this check node $\boldsymbol{s}_i$, only most reliable symbol $\boldsymbol{z}_{j,1}^{(k)}$ is delivered. As there are two sets of message sequence from detector to NB-LDPC decoder, the extrinsic information-sum(EXI) in (4.10) results in the following configurations:

$$\sigma_{il}^{(k)} = \begin{cases} h_{ij}^{-1} \sum_{j' \in N(i) \setminus j} h_{ij'} \boldsymbol{z}_{j',1}^{(k)}, & \text{if } l = j \\ h_{il}^{-1} \sum_{l' \in N(i) \setminus \{j,l\}} h_{il'} \boldsymbol{z}_{l',1}^{(k)} + h_{i,j} \boldsymbol{z}_{j,2}^{(k)}, & \text{otherwise.} \end{cases} \tag{4.38}$$

and

$$\tilde{\sigma}_{il}^{(k)} = \begin{cases} h_{ij}^{-1} \sum_{j' \in N(i) \setminus j} h_{i,j'} \boldsymbol{z}_{j',1}^{(k)}, & \text{if } l = j \\ h_{il}^{-1} \sum_{l' \in N(i) \setminus \{jl\}} h_{il'} \boldsymbol{z}_{l',1}^{(k)} + h_{ij} \boldsymbol{z}_{j,2}^{(k)}, & \text{otherwise.} \end{cases} \tag{4.39}$$

Here (4.38) shows the extrinsic information-sums of the most reliable hard decision

messages and (4.39) denotes the configuration for next most reliable hard decision based messages.

The concept of iterative hard reliability [25, 83] is used to find the correct symbol for all those positions having less reliable information. Suppose $\boldsymbol{z}_j$, being the most reliable symbol, then it is used for the initialization of the reliability of a symbol $\alpha_l \in GF(q)$ to a value γ for $1 \leq l \leq q$:

$$R_{j,l} = \begin{cases} \gamma, & \text{if } \boldsymbol{z}_{j,l} = \alpha_{j,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.40}$$

Here γ is the predefined numerical value. Reliability information at the current iteration is used to find the most reliable symbol for the j^{th} position through the following expression:

If $\sigma_{i,j}^{(k)} = \alpha_l \in GF(q)$

$$R_{j,l}^{(k)} = R_{j,l}^{(k)} + 1 \tag{4.41}$$

and the reliability information are updated for the next $(k+1)^{th}$ iteration as follows:

$$R_{j,l}^{(k+1)} = R_{j,l}^{(k)}. \tag{4.42}$$

Based on these reliability information, the new candidate symbol is selected as follow:

$$\boldsymbol{z}_j^{(k+1)} = \underset{\alpha_l}{\operatorname{argmax}}\, R_{j,l}^{(k)}. \tag{4.43}$$

In the Algorithm 2, the update rule at detector is given by :

$$\boldsymbol{y}_j^{(k+1)} = \boldsymbol{y}_j^{(k)} + f(\lambda^{(k)})\boldsymbol{\beta}^{(k)}, \tag{4.44}$$

Here f is the scaling factor and $\lambda^{(k)}$ and $\boldsymbol{\beta}^{(k)}$ are calculated as given in equations (4.32) and (4.33) respectively.

Proposed Algorithm 2 (E-IJDD)

1. Initialization: For $1 \leq j \leq n$, $1 \leq l \leq q$ and $1 \leq i \leq m$.
Signal Detection: The hard decision symbols $\boldsymbol{z_j}$ is obtained from the received symbol $\boldsymbol{y_j}$ by (4.20) and (4.12). Also find $\hat{\boldsymbol{x}}_{j,1}^{(k)}, \hat{\boldsymbol{x}}_{j,2}^{(k)}$, and $\Delta d_j^{(k)}$ by using equations (4.34),(4.35) and (4.36).
Decoder: If $\boldsymbol{z}_j = \alpha_l \in GF(q)$, then $R_{j,l} = \gamma$ else $R_{j,l} = 0$.
2. Set max iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{ij} \boldsymbol{z}_j^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Calculate extrinsic information-sums $\sigma_{ij}^{(k)}$ by (4.38) and (4.39).
5. Calculate symbol reliability $R_{j,l}^{(k)}$ by (4.41) and update $R_{j,l}^{(k+1)}$ by (4.42).
6. Select the new candidate symbol value $\boldsymbol{z}_j^{(k+1)}$ by (4.43).
7. Adjust the symbol by using (4.44) and feedback to detector. Repeat equations (4.20),(4.12) (4.34),(4.35) and (4.36).
8. $k = k + 1$ for next iteration.

4.3.4 Proposed Algorithm 3

This proposed Algorithm 3 further simplifies Algorithm 2 by introducing the concept of short listing of the candidate symbols by voting scheme as shown in Figure 4.6. In this proposed algorithm, the variable nodes contributing to the failure of the check-sum are first short listed and then only those short listed variable nodes are processed to estimate the their correct values. This process results in very less computational complexity as well as require less memory to store the processed data.

At the decoder, for the hard decision symbol sequence $\boldsymbol{z}^{(k)}$, the candidate symbols are short listed. Suppose $\boldsymbol{\delta}$ stores the positions of all the variable nodes with

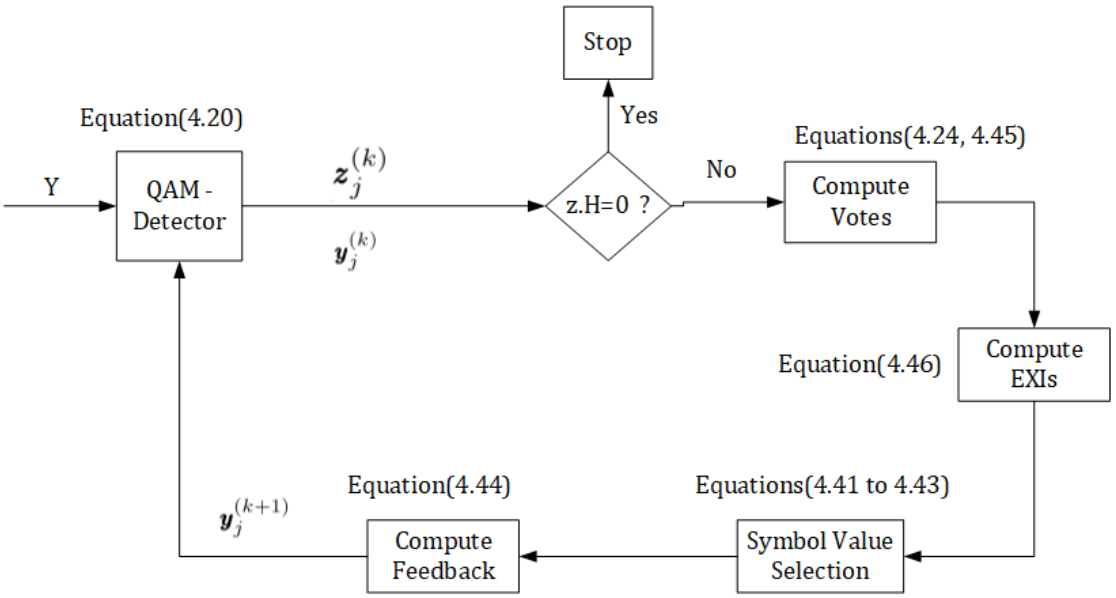


FIGURE 4.6: Proposed Algorithm 3 Execution Cycle

maximum voting, then the values of $\boldsymbol{\delta}$ at k^{th} iteration can be calculated as follows:

$$\boldsymbol{\delta}_{j'}^{(k)} = j, \quad if \quad V_j \geq V_{th} \tag{4.45}$$

for $1 \leq j' \leq p$ and p shows the total number of variable nodes with maximum votes. This reduces the computational complexity from n number of variable nodes to just few variable nodes p. Here $\delta_{j'}^{(k)} \in \boldsymbol{\delta}^{(k)}$ stores the position of each variable node with maximum votes and $\boldsymbol{\delta}^{(k)}$ contains all those positions of the variable nodes. In other words, $\boldsymbol{\delta}^{(k)}$ have all the positions of variable nodes having less reliable information and must be replaced with reliable symbols.

For the short listed candidate symbols $\boldsymbol{\delta}^{(k)}$, the normalized extrinsic information-sum (EXI) for the k^{th} iteration is calculated as:

$$\sigma_{i,j'}^{(k)} = h_{ij}^{-1} \sum_{\bar{j} \in N(i) \backslash j'} h_{i,\bar{j}} \boldsymbol{z}_{\bar{j}}^{(k)} \tag{4.46}$$

for $1 \leq i \leq m, 1 \leq j' \leq p$. The extrinsic information-sum computation in the IIJDD algorithm [80] takes into account the first and the second most reliable symbols $\{\boldsymbol{z}_{j,1}^{(k)}, \boldsymbol{z}_{j,2}^{(k)}\}$ while the proposed algorithm 3 does not require the second reliability value. The symbols value is selected based on the IHRB algorithm as

mentioned in algorithm 2. The feedback to the QAM detector is used as discussed in the algorithm 1 but only for the short listed variable nodes $\boldsymbol{\delta}$ at the k^{th} iteration.

Proposed Algorithm 3 (V-IJDD)

1. Initialization: For $1 \leq j \leq n$, $1 \leq l \leq q$ and $1 \leq i \leq m$.
The hard decision symbols $\boldsymbol{z_j}$ is obtained from the received symbol $\boldsymbol{y_j}$ after de-mapping by using (4.20) and (4.12).
if $\boldsymbol{z}_j = \alpha_l \in GF(q)$, then $R_{j,l} = \gamma$ else $R_{j,l} = 0$.
2. Set max iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{ij} \boldsymbol{z}_j^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Calculate the votes for each of the variable node by using (4.23) and (4.45). Now short list the least reliable symbol positions by (4.24).
5. For $1 \leq j' \leq p$, calculate extrinsic information-sum $\sigma_{i,j'}^{(k)}$ by (4.46).
6. Calculate symbol reliability $R_{j',l}^{(k)}$ by (4.41) and update $R_{j',l}^{(k+1)}$ by (4.42).
7. Select the new candidate symbol value $v_{j'}^{(k+1)}$ by (4.43).
8. Adjust the symbol by using (4.44) and feedback to detector as in (4.20) and (4.12).
9. $k = k + 1$ for next iteration.

4.4 Numerical Results and Analysis

The proposed decoding algorithms for the NB-LDPC code (n, d_v, d_c) with d_v and d_c as the column and row weights respectively, have been simulated using QAM as the modulation technique and the performance parameters are compared with IJDD decoding algorithms in literature. The parity check matrices with $d_v = 2$ in this chapter, is based on the construction method given in [66] and are available on the web site [84]. The ultra-sparse parity check matrices are designed for multiple possible applications and more specifically for building good NB-LDPC codes in higher order Galois field. The maximum number of iteration in all the

examples are kept as I_{max}=15 and the value of $\gamma = 1.75$. The scaling factor f is set to 0.5 for 8 QAM and 0.25 for 16 QAM. The simulation is terminated either when 100 bits errors are received or the number of transmitted bits reaches to 10^8. In the following examples, it is observed clearly from the BER performance curves that the proposed algorithm 2 performs better than the IJDD algorithms in literature. It should be noted that in all these examples, the QAM modulation order is the same as Galois field size.

Example 1: In this example, BER performance of quadrant based detection scheme is compared with the ML QAM detection. The simulation results are shown for 8,16 and 64 QAM. In the Figure 4.7, it is clear that the performance curves of the proposed system model are similar and almost perfectly overlap on the ML QAM detection. For higher order QAM modulator like 128 or 256, the BER performance curves might not overlap perfectly as the symbols are close enough to be detected exactly in one quadrant of the constellation.

Example 2: The construction of NB-LDPC code $(45, 2, 3)$ in this example over $GF(8)$ is based on the alist format given in [84]. In this example 8 QAM modulation has been used to simulate the proposed algorithms in comparison to the IJDD and Improved IJDD NB-LDPC decoding algorithms in literature [80]. The performance curves of various algorithms under AWGN are as shown in Figure 4.8. At BER of 10^{-6}, the algorithm 2 outperforms the Improved IJDD by about 3.5dB. The proposed Algorithm 3 and Improved IJDD have a performance gap of about 4.2dB. The proposed Algorithm 3 performance is better than Algorithm 1 as well as IJDD while the proposed Algorithm 1 which is based on the symbol flipping decoding, performs better than IJDD algorithm by approximately 2dB. The step size of the Algorithm 2 is based on the Euclidean distance between the received symbol and the newly selected candidate symbol which help in moving the received symbol in correct direction. Also the adaptation of the IHRB algorithm to select the candidate symbol value helps to improve the BER performance of Algorithm 2.

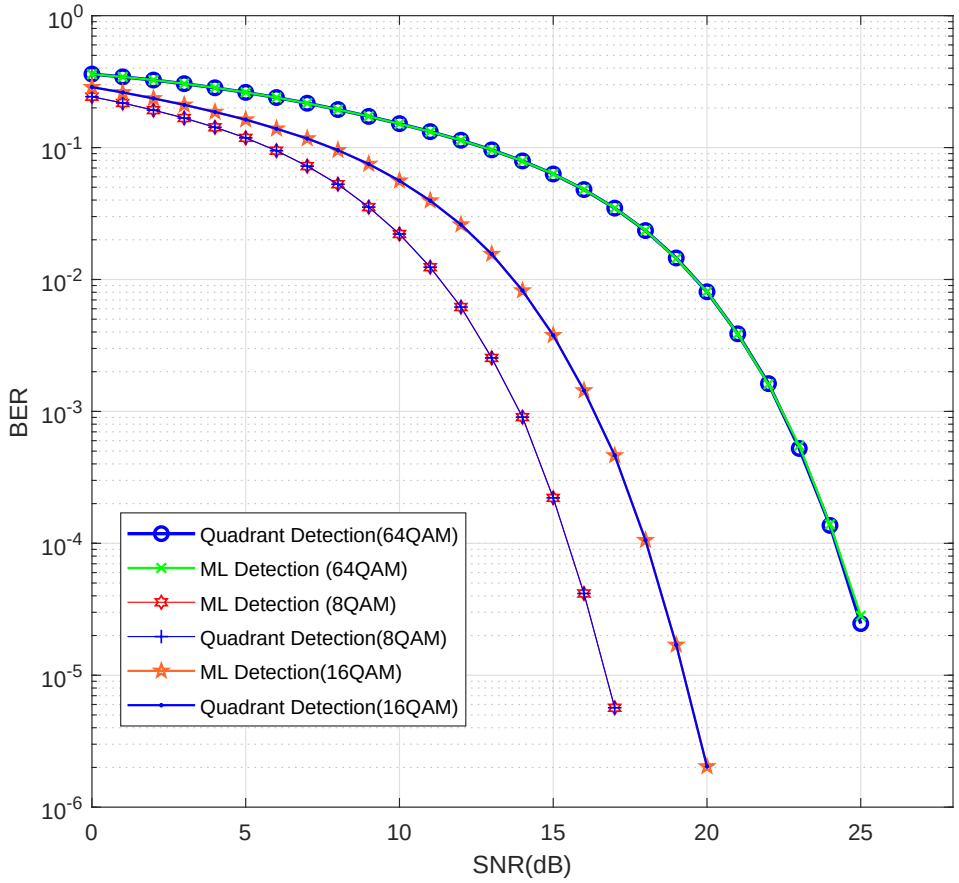


FIGURE 4.7: Comparison of ML and Quadrant QAM Detection

In this example, Figure 4.9 depicts the curves for BER and average number of iterations of various algorithms. It is visible from the graph that the algorithm 1 converges faster than the other algorithms. Around the BER 10^{-4}, the average number of iterations of the IJDD algorithm and algorithm 1 are equal while the proposed algorithm 3 and the Improved IJDD algorithm are almost equal with average number of iteration less than 3 while algorithm 2 has slightly more than 3 iteration.

Example 3: In this example, the LDPC code(45, 2, 3) over $GF(16)$ is used to illustrate the BER performance of the proposed decoding algorithms in comparison to Improved IJDD and IJDD algorithms. From the BER curves illustrated in Figure 4.10, it is observed that the proposed algorithm 1 and 3 perform better than IJDD algorithm and close to the Improved IJDD algorithm. The performance gap

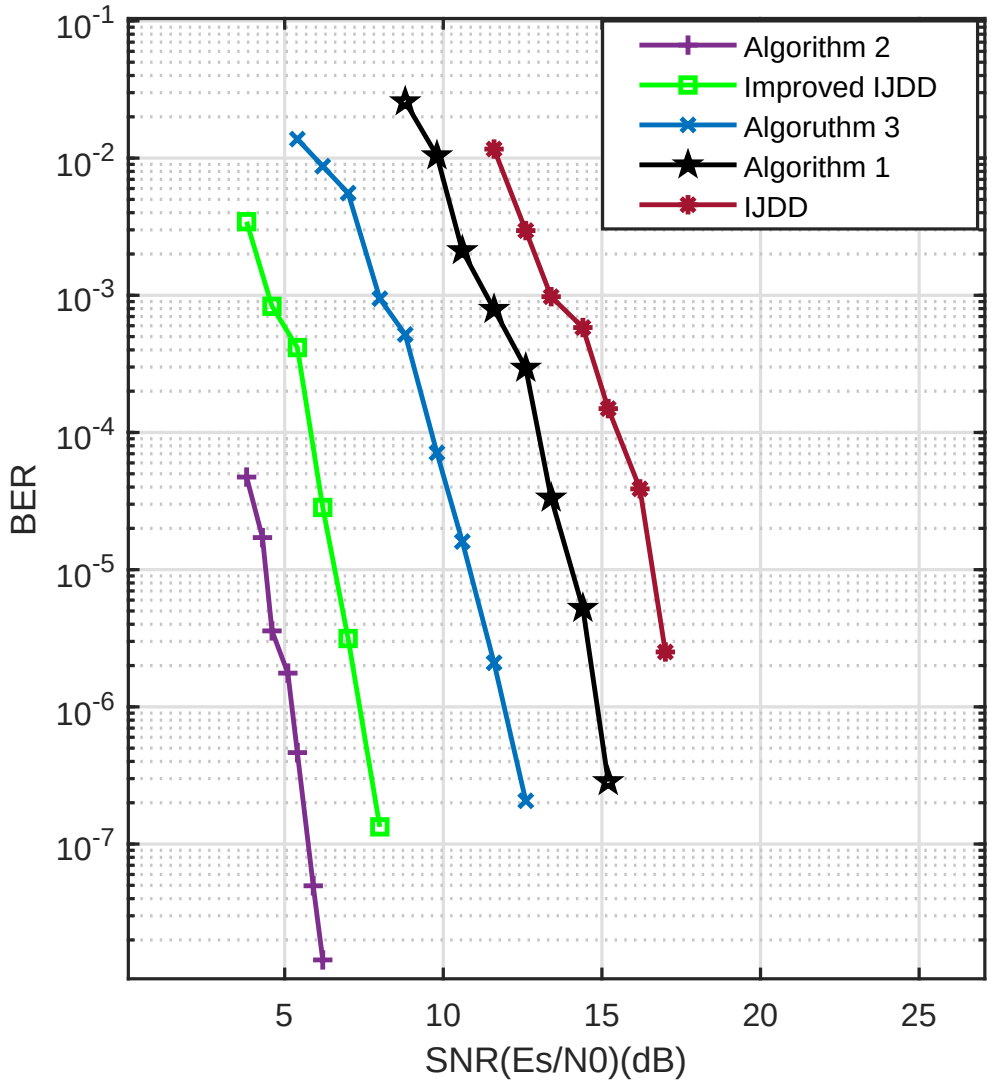


FIGURE 4.8: BER Performance of NB-LDPC Code (45, 2, 3) over $GF(8)$

at BER 10^{-6} between the proposed algorithm 2 is about 2.5dB while the algorithm 3 is around 2.5 less in performance than Improved IJDD and 2.5dB better than algorithm 1. Algorithm 1 shows an excellent performance in comparison to IJDD algorithm. It is worth mentioning that the algorithms in [21, 83] are using BPSK modulation while those presented in this chapter are using QAM. The NB-LDPC code (45, 2, 3) over $GF(16)$ has been simulated using BPSK modulation to get the BER performance curves for ISRB and V-SFD algorithms as shown in Figure 4.10.

Example 4: Here a NB-LDPC code$(105, 2, 5)$ over $GF(8)$ is simulated using the

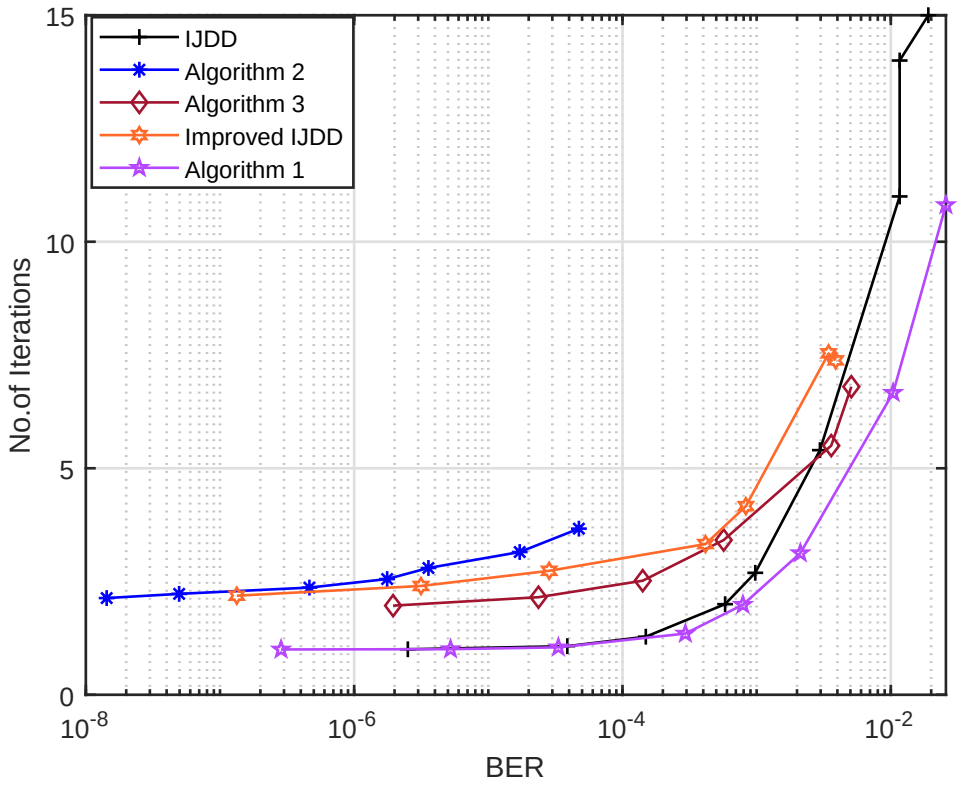


FIGURE 4.9: Average number of iterations verus BER for the (45, 2, 3) code over $GF(8)$

8 QAM modulation. The performance curves for the proposed algorithms and IJDD algorithms are illustrated in Figure 4.11. The BER performance curves depicted in Figure 4.11, shows that algorithm 2 outperforms by 3.0dB than Improved IJDD(IIJDD) at BER 10^{-6}. The performance gap between the algorithm 3 and IIJDD algorithm is around 4.0dB while algorithm 1 is around 3.8dB away from algorithm 2. Algorithm 1 is performing better than IJDD algorithm by 2dB. An individual decoder and QAM detector is also added to this example. Algorithm 3 is simulated without using the feedback to the 8 QAM detector. The BER performance curves shows that at 10.6db, the individual decoder-detector has achieved BER of 1.603^{-5} while the feedback based joint decoder-detector has achieved BER of 3.304^{-5}. Also at 10^{-3}, there is around 0.9dB difference which shows an obvious advantage of feedback based IJDD algorithm.

Example 5: In this example, a quasi cyclic high column NB-LDPC code $(120, 4, 8)$ over $GF(8)$ is simulated using the 8 QAM modulation. The performance curves for

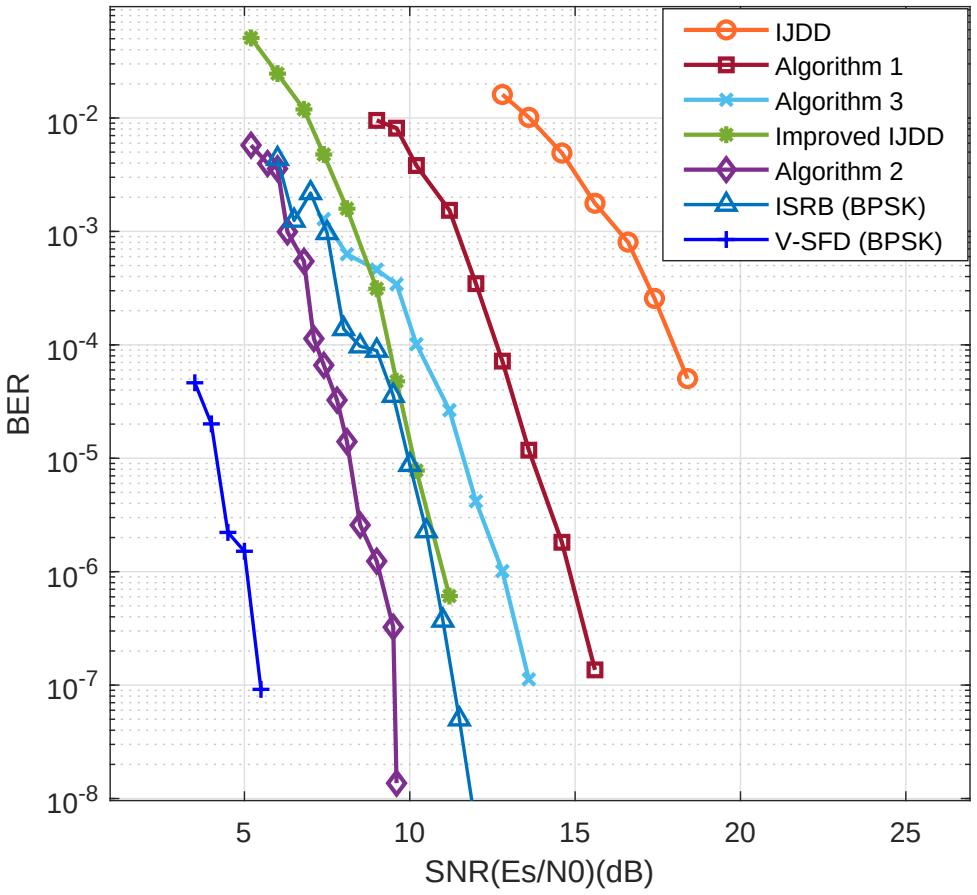


FIGURE 4.10: BER Performance of NB-LDPC Code (45, 2, 3) over $GF(16)$

the proposed algorithms and Improved IJDD algorithm are illustrated in Figure 4.12. The BER performance curves depicted in Figure 4.12, shows that algorithm 2 outperforms by 4.7dB as compared to IIJDD algorithm at BER $10^{-5.7}$. The performance gap between the algorithm 3 and Imrpoved IJDD algorithm is around 5.0dB while algorithm 1 is around 1dB away from algorithm 2 at the beginning and the performance gap is getting increased to more than 2.0dB.

From the above examples, it is observed that as the order of the QAM modulation is increased, the Euclidean distance based QAM detection gets more and more inefficient due to the symbols placed in more close vicinity. This property of the QAM modulation effects the iterative joint detection and decoding as well. The basic concept in IJDD algorithm is to move the received symbol towards the correct location by removing the noise in steps. This step size is very important and can

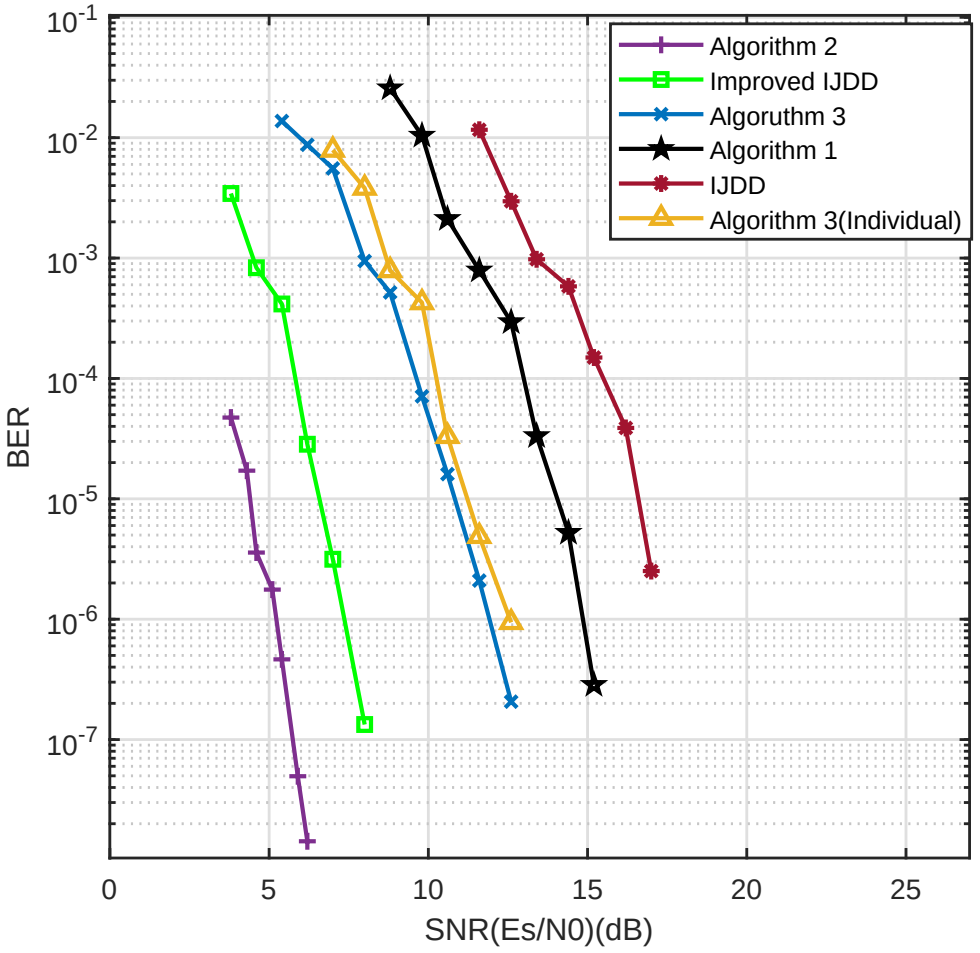


FIGURE 4.11: BER Performance of NB-LDPC Code(105, 2, 3) over $GF(8)$

lead to over or under estimation if it is not adjusted properly. It is important to note that the effect of the factor f becomes very sensitive for an increased size of the M-QAM where M is the modulation order. An adaptive step size can be investigated in further research to improve the BER performance.

Soft decision decoding algorithms have high complexity in comparison to the hard decision decoding [7, 9] which requires only integer's addition and multiplications and maintain hardware friendly architecture [85]. The soft decision iterative decoders achieve better BER performance but the main concerns is the high decoding complexity while on the other hand, hard decision decoding offer low complexity which is of interest for hardware implementation.

Since in this chapter, optimized LDPC codes from literature are used, therefore

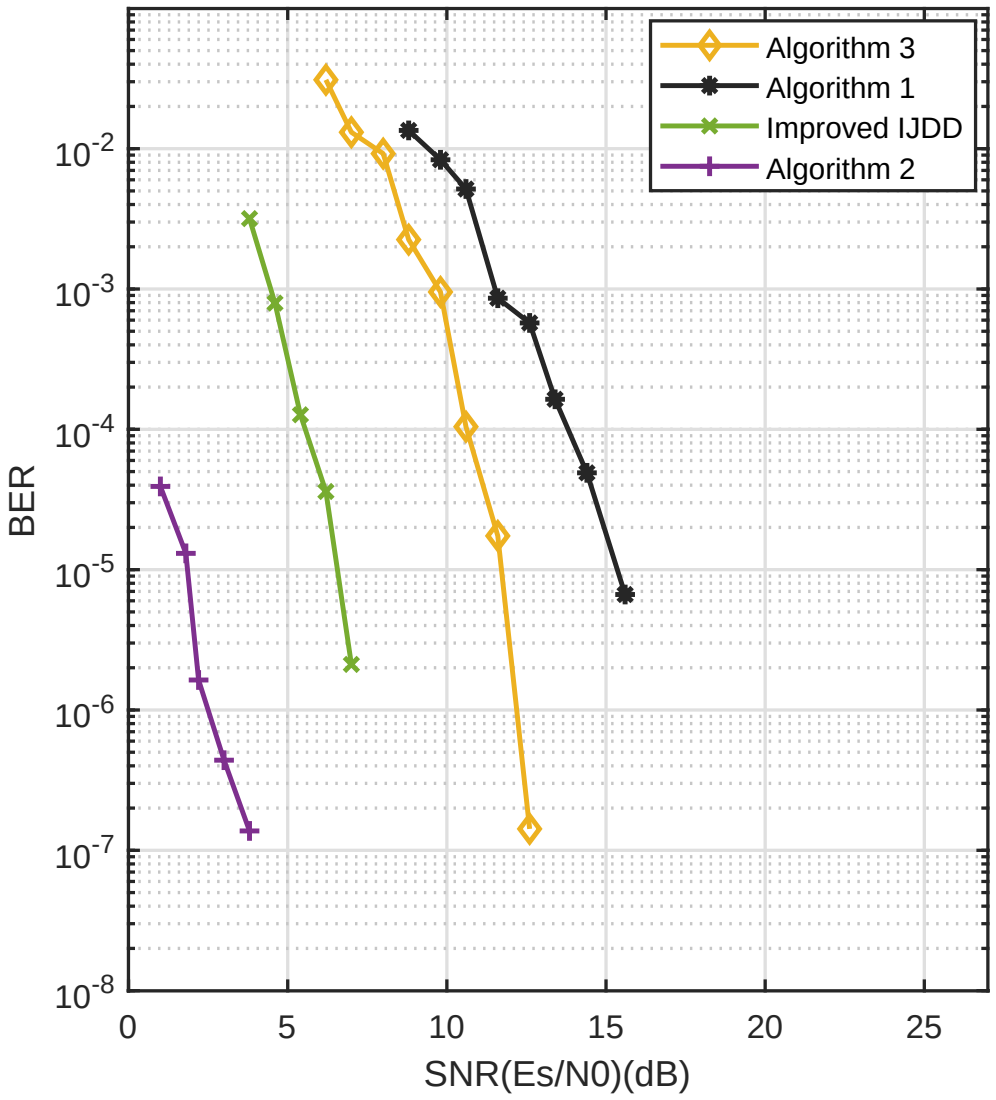


FIGURE 4.12: BER Performance of NB-LDPC Code(120, 4, 8) over $GF(8)$

EXIT chart [73, 86] and density evolution [87] are not applied to do the code analysis.

4.5 Complexity Analysis

In this section, the computation complexity of the proposed algorithms is analysed in detail and compared to that of the Improved IJDD algorithm. Consider an LDPC code (N, d_v, d_c) over GF(q) and let δ represents the nonzero elements in

TABLE 4.1: Computational complexity of different NB-LDPC decoding algorithm per iteration

Algorithms	Number of operations				
	GM	GA	IA/RA	RC/IC	IM/RM
Algorithm 1	$\gamma + \frac{1}{4}qd_v(d_c)$	$\gamma + \frac{1}{4}qd_v(d_c-1)$	$p(d_v+1)+\rho$	$\frac{1}{4}q(p+\rho)$	$2(p+\rho)$
Algorithm 2	$3nd_v$	$3nd_v$	$2n(q+1)$	$2n(q-1)$	nq
IIJDD	$3nd_v$	$3nd_v$	$2nd_v + n + 2n(q+1)$	$2n(q-1)$	$3nq+4n$
EMS	$2nd_vn_m$	$9n_m(nd_v -2m)$	$n_m(20nd_v - 18m - 12n)$	$n_m log_2 n_m(9nd_v -12m - 4n)$	
IJDD	$2nd_v$	$2nd_v - m$	$nd_v + n + 4n$	$n(2q-3)$	$4n$
wt-Algo.B	$2nd_v$	$3nd_v - m$	nd_v	nd_v	nd_v
ISRB	$2nd_v$	$2nd_v - m$	$nd_v + n2^r$	$2n2^r - 2n$	$n2^r$
V-SFD	$d_v(2d_c -1)$	$pd_v(d_v+r)+ 2d_v(d_c -1)$	$p(d_v^2 + 2d_vr + 4r + 2d_v)$	$nr + p(d_v - 1) + 2(p-1)$	
M-QAM FFT-QSPA	$(2q+1)nd_v$	$5nd_vq$	$6m + 2qnd_vlog_2q$	$7m + qnd_v(d_v + d_c - 2)$	$n(q-1)$

IM/IC/IA: Integer Multiplication/Comparision/Adittion; p is the total number of variable nodes stored in $\boldsymbol{\delta}$.
RA/RM/RC: Real Addition/Multiplication/Comparision; GA/GM: Galois Field Addtion/Multiplication;
Notations used: $\gamma = nd_v = md_c$, $L = \boldsymbol{\upsilon}^{\xi}$ such that $\xi < d_c$, $n_m < q$.

the parity check matrix $\boldsymbol{H}$ where $\gamma = nd_v = md_c$. At the ML QAM detector, q symbols of the QAM constellation are tested for each of the n received symbol sequence which results in nq real addition and multiplication. These operation are reduced to $(\frac{n}{4})q$ for the proposed quadrant based detection in this paper. The proposed Algorithms 1 and 3 require z_j only as opposite to IIJDD algorithm which require the pair of $\{(\boldsymbol{z}_{j,1}, \boldsymbol{z}_{j,2}), \Delta d_j\}$ information. This results in further complexity reduction by $2nq + n$ real additions.

At the NB-LDPC decoder initialization, md_c Galois field multiplication and $m(d_v-$

1) additions are required to compute the m check-sum and qn real comparisons are required to initialize the reliability information in case of Algorithm 2 and 3 only. For Algorithm 2, the update of the extrinsic information-sum $\sigma_{i,j}$ needs pair of $(\boldsymbol{z}_{j,1}, \boldsymbol{z}_{j,2})$, therefore, $2md_v$ Galois field multiplications and $2m(d_c - 1)$ Galois field additions are required. On the other hand Algorithms 1 and 3 require md_c multiplications and $m(d_c - 1)$ additions which means two times less computational complexity than IIJDD algorithm and Algorithm 2.

In the IIJDD algorithm, a pair of message $(v_j, \Delta\eta_j)$ is sent as feedback to the QAM detector while in the proposed algorithms, only the updated soft information $\boldsymbol{y}_j^{(k+1)}$ is used for the feedback. To compute the feedback $\boldsymbol{y}_j^{(k+1)}$, n real addition and multiplications are required for the proposed Algorithm 2 while for Algorithm 1, only ρ real addition and multiplications are required. To calculate the symbol flipping function in the Algorithm 1, pd_v real addition and p real multiplications are required. To compute the candidate symbol value for the position in error, d_v times the check-sum $\boldsymbol{s}_i$ is calculated. Therefore, to predict a single symbol value from the quadrant QAM constellation, $\frac{1}{4}qd_v(d_c - 1)$ Galois field addition and $\frac{1}{4}qd_v(d_c)$ multiplications are required for the symbols.

Table 1 shows the computational complexity of the proposed decoding algorithms in comparisons to other algorithms from the literature.

4.6 Conclusion

In this chapter, three algorithms are proposed for symbol flipping decoding of NB-LDPC over high order modulation. There is no symbol flipping decoding algorithm for the M-QAM in the literature. The results show significant performance improvement in terms of the bit error rate and complexity. The new concept of symbol flipping based IJDD is also showing better BER performance for the low column and row weights in comparison with the IJDD algorithms in literature. The proposed Algorithm 2 outperforms the Improved IJDD algorithm while Algorithm 1 and 3 perform better than IJDD. The step size of the Algorithm

2 is based on the Euclidean distance between the received symbol and the newly selected candidate symbol which help in moving the received symbol in correct direction. The numerical results and complexity analysis show that the proposed algorithms using the concept of iterative joint detection and decoding, offer an appealing trade-off between bit error rate and computational complexity.

CHAPTER 5

Layered Symbol Flipping Decoding Algorithms

This chapter is based on the following research publication.

Waheed Ullah, Ling Cheng, Fambirai Takawira. " Adaptive Group Shuffled Multiple Symbol Flipping Decoding Algorithm". (To be submitted to an IEEE journal)

Contribution Statement: The first author is responsible for the conceptualization of the work, experimental data collection, analysis and interpretation, as well as project execution and results collection. The work was done with assistance from the co-authors: Prof. Ling Cheng and Prof. Fambirai Takawira.

Abstract: In this chapter , two group-based decoding techniques are presented for the class of symbol flipping non-binary LDPC codes. Firstly, a new technique of adaptive grouping of variable nodes in each iteration using bit reliability and majority voting of each received symbol is presented. Grouping of variable nodes and the subsequent decoding are based on individual symbol reliability and majority voting. To form groups adaptively, the cumulative density of variable nodes is used where a high priority group is considered to contain the most unreliable variable nodes and is decoded while the second priority group contains variable nodes having a second level of reliability and so on. Secondly, a fixed grouping technique is applied to symbol flipping decoding of non-binary LDPC codes. In the fixed grouping decoding, each group contains an equal number of variable nodes, and selected symbols in each group are flipped according to pre-defined flipping criteria. Numerical results and analysis show that the proposed group-based hard decision symbol flipping decoding algorithms have the advantage of reduced computational complexity and show better performance.Therefore, the proposed algorithms can be considered for applications like data storage and mobile communication.

5.1 Introduction

The current research focus on non-binary LDPC (NB-LDPC) codes is to reduce the decoding computational complexity and to improve performance. NB-LDPC codes have higher computational complexity but they have the advantage over binary LDPC codes that they can be directly mapped to the order of modulation [3][88]. The NB-LDPC message passing algorithm in [3] over $GF(q)$ has a computational complexity in an order of $O(q^2)$ during check node processing and is the main hurdle in implementation. The check node complexity is reduced by using an efficient log-domain based fast Fourier transform (FFT) in [11]. The FFT based sum-product algorithm in [11] reduces the check node processing complexity from $O(q^2)$ to $O(qlogq)$. To further reduce the computational and implementation complexity, an Extended Min-Sum (EMS) algorithm in [52] is presented which has greatly reduced the check node computational complexity and has the implementation advantage over the other algorithms. For hardware implementation, authors in [76, 89] present a Bubble Check algorithm to reduce the number of comparisons during check node processing without any performance loss. Reliability based majority logic decoding algorithm is similar to message passing decoding but it sends only the most reliable field message and is considered as the low computational algorithm. In the iterative majority logic decoding (MLgD) algorithm for non-binary LDPC codes, each symbol is iteratively updated by extrinsic information-sums (EXIs) with the most reliable field element along each edge of the Tanner graph. The iterative reliability based hard (IHRB) and soft (ISRB) MLgD algorithms in [23] have low complexity but show better performance only for parity check matrices with high column weights. These two algorithms in [23] are further improved by introducing soft reliability information at the initialization to achieve better trade-off between complexity and bit error performance by proposing several modifications in [24, 27]. To overcome this drawback, an improvement to these algorithms is presented in [31] by introducing reliability updates in terms of bit rather than symbols. The bit reliability based decoding algorithm for NB-LDPC is comparatively more efficient and is termed as weighted bit reliability

based (wBRB) algorithm [32]. This bit reliability based algorithm requires integer and Galois field operations only which helps to reduce the hardware implementation complexity and processing time. To further enhance the wBRB decoding algorithm, a full bit-reliability based decoding scheme is proposed in [32]. To lower the processing complexity, this algorithm uses the binary representation of non-zero entries of the parity check matrix.

However, the hardware implementation of the NB-LDPC codes is still a challenge. To achieve hardware friendly architecture, a reduced complexity decoder architecture via layered decoding of LDPC codes is presented in [90]. A more hardware friendly high-throughput quasi-cyclic LDPC codes based on layered decoding scheme is presented in [91] for binary LDPC codes [73, 92]. Later this concept has also been adopted in [64, 93] for non-binary LDPC decoding algorithms.

In a layered scheme, the LDPC decoding algorithm processes information by dividing check nodes or variable nodes of the Tanner graph. Basically this scheme divides the Tanner graph into subgraphs which are termed as layers [61][93, 94]. Interestingly, both the vertical and horizontal processing of the layers achieve the same goal of fast convergence and better performance. Fully serialization of the LDPC codes can restrict the throughput and thus partial serial architecture in which each layer contains more than one column or row for VLSI implementation is presented in [61]. Standard parallel message passing LDPC decoder [95] normally takes much more iterations for decoding process to converge which causes high decoding delay. Secondly, desired LDPC code can have large codeword length which can be difficult for hardware implementation in a fully parallel manner. The aim of the layered LDPC decoder [61] is to increase the convergence speed and facilitate the hardware implementation.

The standard SPA decoding updates the check node at k^{th} iteration from the information values calculated at the previous iteration $(k-1)^{th}$ while in the shuffle decoding technique, certain values could already be computed based on a partial computation of values at variable nodes. These partial computed information can be used for the check node update at the same k^{th} iteration instead of previous

values calculated at $(k-1)^{th}$ iteration. This method provides the shuffling of the check and variable nodes and is called a group shuffle decoding [63, 64].

Other than the message-passing decoding, a simplified method to correct an error in the received sequence is the symbol flipping (SF) decoding algorithm. In the SF decoding, an unreliable symbol position is detected and a correct symbol value is chosen for that position. A majority logic decision based algorithm is presented in [14], where a symbol position to be flipped is determined by majority decision while the flipped value is obtained from channel output by flipping individual bits of a symbol with low reliability. Another algorithm termed as weighted algorithm B (wt.Algo B) presented in [15] introduces the binary Hamming distance and plurality logic for performance improvement. The parallel symbol flipping decoding (PSFD) algorithm in [16] flips multiple symbols per iteration and has introduced the concept of voting for each variable node. The maximum value of the flipping function is used in conjunction with the voting value of the corresponding variable nodes to identify a symbol to be flipped. The PSFD has shown good performance only for parity check matrix of large column weight. A multiple voting based PSFD (MV-PSFD) algorithm is proposed in [18] to improve bit error rate performance of the PSFD algorithm. The MV-PSFD algorithm has implied a method of multiple voting levels from each failed checksum to the corresponding variable nodes. The performance of a symbol-reliability based message-passing decoding (SRBMP) decoding algorithm [58] has been improved by multiple voting symbol flipping (MV-SF) decoding algorithm [59]. The MV-SF only passes the most reliable Galois field elements from each variable node to the corresponding check node. Therefore, a list of test vectors is required to compute the flipping function for each check node. MV-SF has shown an improved performance but due to the test vectors, its complexity increase with an increase in the size of the Galois field. Recently, low complexity voting-based symbol flipping decoding algorithms are presented in [21, 22] which can flip single and multiple symbols per iteration and have shown better bit error rate performance.

The method of adaptive layering [64] based on the most unreliable VNs, can be explored for possible utilization in a symbol flipping NB-LDPC decoding algorithm

to improve performance as well as convergence speed. To address the low complexity implementation of the SF decoding algorithms, an adaptive group shuffled and a fixed group-based symbol flipping decoding algorithms are proposed. The proposed methods have two main advantages over other NB-LDPC algorithms. One advantage is the hardware-friendly simple architecture and the second is the fast convergence due to multiple symbols flipping per iteration which reduces the overall decoding computation. The followings are the main contributions of this chapter:

1) A new method for adaptively grouping variable nodes is presented. In this method, a cumulative distributive function of the received bits sequence, which is being contaminated by Gaussian random noise, is utilized along with the majority voting based reliability of variable nodes. This proposed method is different from the existing method of layered decoding [61] and from shuffled decoding presented in [64]. In this proposed method, the variable nodes in each group are selected adaptively. A group having less reliability is decoded first and then three level of voting are defined to select either next group for processing or terminate and go for the next iteration. So far in the literature, voting and reliability of the variable nodes are not used to form and process groups. In [64], the non-binary sum-product algorithm decodes the least reliable variable nodes in a group and the variable nodes which satisfy certain condition are excluded from the group.

2) The major complexity of the conventional SF decoding algorithm lies in computation of the flipping function. The issue of computational complexity can be overcome by proposing adaptive group based symbol flipping algorithm. In the group based SF decoding, the flipping function is computed for the selected group only which helps in reducing processing latency and is different from traditional symbol flipping decoding presented in [21, 22]. A group is selected for decoding upon fulfillment of certain condition specially after the first iteration, therefore, this technique helps in reducing the overall computation as all the groups might not be selected to decode. This adaptive grouping scheme is also aimed to improve BER performance particularly at low SNR region.

3) A fix grouping technique is applied to the symbol flipping decoding in order to reduce the computational complexity. In this scheme variable nodes are divided equally among a predefined number of groups. In this proposed fixed grouping technique, a group is processed only if it contains erroneous variable nodes which makes it different from the fixed grouping technique presented in [61].

The rest of the chapter is divided into sections as follows. Section II briefly explains the existing layered and grouping schemes in literature. The proposed group based symbol flipping decoding algorithms is presented in section III. Section IV is based on the results and discussion of the existing and proposed methods. Finally Section V gives the conclusion of the chapter.

5.2 Background

5.2.1 System Model

Consider a regular parity check matrix H with dimension $m \times n$ defined over the Galois field (GF) size $q > 2$ where each element $h_{i,j}(1 \leq i \leq m, 1 \leq j \leq n)$ of H is an element over the $GF(q)$ such that $q = 2^r$ and r shows the number of bits in each symbol. Now consider a NB-LDPC code $\mathcal{C}(N, \gamma, \rho)$ of length N for a regular parity check matrix H, such that each column of H has constant weight of γ and each row of H has constant weight of ρ. In the Tanner graph, for parity check matrix H, the non-zero entries ($h_{i,j} \neq 0$) show the i^{th} check node (CN) connected to the j^{th} variable node (VN) and the Tanner graph edge connection is given by $M(j) = \{i : 1 \leq i \leq m \ , \ h_{i,j} \neq 0\}$ and $N(i) = \{j : 1 \leq j \leq n, h_{i,j} \neq 0\}$.

Let $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_j, \ldots, \boldsymbol{c}_n)$ be a codeword in $\mathcal{C}$ and the binary form of j^{th} symbol in $\boldsymbol{c}$ is $\boldsymbol{c}_j = (c_{j,1}, c_{j,2}, \ldots, c_{j,t}, \ldots, c_{j,r}), 1 \leq t \leq r$. This binary sequence is modulated with binary phase shift keying (BPSK), where 1 is modulated as +1 and 0 is modulated as -1. The BPSK modulated j^{th} symbol $\boldsymbol{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,t}, \ldots, x_{j,r})$ is sent over additive white Gaussian noise (AWGN) channel. At receiver, the binary sequence $\boldsymbol{y}_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,t}, \ldots, y_{j,r})$ is obtained from the transmitted

sequence $\boldsymbol{x}_j$ by adding the additive white Gaussian channel noise $\mathcal{N}(0, \sigma^2)$ with two sided power spectral density as $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j$. The hard decision (HD) binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is given by:

$$z_{j,t} = \begin{cases} 0, & \text{if } \; y_{j,t} < 0 \\ 1, & \text{if } \; y_{j,t} \geq 0 \end{cases} \tag{5.1}$$

5.2.2 Group Formation

In literature, Group-Shuffled [61],[63] sum-product decoding algorithms divide either check nodes or variable nodes of the relevant bipartite graph into small sub-groups called layers. Also each main iteration is broken into multiple sub-iterations. The group might have one or more than one rows and columns. In some methods, a quasi cyclic parity check matrix is divided horizontally into groups in such a manner that each group has exactly one non-zero entry in each column. This is the most common method used for sum-product LDPC decoder hardware implementation. For the variable nodes $V_1, V_2, \ldots, V_j, \ldots, V_n$, let G_τ be a group, then a set of groups G is given by:

$$G = \{G_1, G_2, \ldots, G_\tau, \ldots, G_\alpha\}, \; 1 \leq \tau \leq \alpha \tag{5.2}$$

The n number of variable nodes are divided into α groups and each group is comprised of $n/\alpha = \beta$ variable nodes for $1 \leq j' \leq \beta$. For $\alpha = 1$, the group based decoding becomes the standard flooded schedule SPA.

It is observed that in one iteration the SPA algorithm can fully process in parallel while the shuffle SPA decoding process message passing in serial. This parallel shuffle scheme for LDPC code is called group shuffle decoding. In this algorithm, the code length is divided into specified groups where in each group the updating of messages is completed in parallel while inside the group, information is processed serially. The group shuffle decoding is summarised as:

• Within the same group, message passing is processed in parallel.

• Between the groups, message passing is processed in serial.

The variable node can be divided into groups of different sizes based on the local girth and edges of Tanner graph [63] which increases the bit error performance and convergence speed. The local girth of a variable node is used for the appropriate partition.

A dynamically re-grouping of VNs in each iteration based on simple binary and integer decision is presented in [64]. The following metric decides the selection of VNs from the index set variable nodes with least reliable decision and high probability to be corrected during updates:

$$E_j = \Omega_j(\sum_{i \in M(j)} \boldsymbol{s}_i) \tag{5.3}$$

where $\Omega_j(x) = \frac{x}{\gamma_j}\gamma_{max}$ and $\gamma_{max} = \max_j[\gamma(j)]$. This equation (5.3) can be associated with the flipping function of the Gallagar algorithm B and is equal to the number of failed check-sum. The flipping metric is defined as:

$$F_j = \sum_{i \in N(j)} q_{(i,j)} \boldsymbol{s}_i \tag{5.4}$$

where

$$q_{(i,j)} = \begin{cases} 1, & \text{if } \max\limits_{j' \in N(i)} E_{j'} = E_j \text{ and } E_j \geq \hbar \\ 0, & \text{else} \end{cases} \tag{5.5}$$

Here F_j counts the number of unsatisfied check-sums and $\hbar$ is an integer optimized numerically. This method of identifying the most unreliable VNs can be explored for possible utilization in the NB-LDPC to improve performance as well as convergence speed. In literature, Group-Shuffled [61],[63] sum-product decoding algorithms divide either check nodes or variable nodes of the relevant bipartite graph into small sub-groups called layers. Also each main iteration is broken into multiple sub-iterations. The group might have one or more than one rows and

columns. Also the quasi cyclic parity check matrix is divided horizontally into groups in such a manner that each group has exactly one non-zero entry in each column

In layered decoder, extrinsic information are exchanged more sufficiently due to division of main iteration into sub iteration, therefore, this scheme results in more accurate decision and fast convergence as compared to non-layered decoding techniques. In the group decoding algorithm, groups are processed sequentially and VN belonging to the same group are updated in parallel. These grouping techniques are developed for the message passing sum-product algorithm and have not yet applied to the symbol flipping decoding technique.

5.3 Proposed Adaptive Group Shuffled SF Decoding

In this section, a new dynamically re-grouping of variable nodes in each iteration using bit reliability and majority voting of individual received symbol is presented for the class of symbol flipping decoding of NB-LDPC codes. The VNs achieving least reliability are grouped together and then variable nodes with second level of reliability are grouped and so on with the last group comparatively being more reliable. This grouping is accomplished by taking cumulative density function(CDF) of the received sequence which is being contaminated by Gaussian random noise. In this algorithm, the number of variable nodes in a group are selected according to the reliability value of each variable node. Secondly, a fixed grouping scheme is applied to the symbol flipping decoding algorithm in which the variable nodes are divided equally among predefined number of groups. During the sub-iterations, a group is selected for decoding if it contains a variable node connected to a failed check.

5.3.1 Adaptive Group Shuffling

For the given parity check matrix H defined over $GF(q)$, the received sequence is divided into groups such that each group contains variable nodes based on the bit

reliability of each symbol.

Variable nodes are grouped based on the reliability of the channel received soft and hard decision information. A symbol is considered to be less reliable if it has a minimum reliability value and vice versa. The following equation is used to compute reliability of individual symbol.

$$R_j^{(k)}(\boldsymbol{z}_j^{(k)}, \boldsymbol{y}_j^{(k)}) = \sum_{t=1}^{r} (2z_{j,t}^{(k)} - 1) y_{j,t}^{(k)} \tag{5.6}$$

here $R_j^{(k)}$ calculates the reliability information of a j^{th} symbol at k^{th} iteration for the hard decision symbol sequence $\boldsymbol{z}_j^{(k)}$ and the channel soft information $\boldsymbol{y}_j^{(k)}$. A symbol is considered to be less reliable if the numerical value of $R_j^{(k)}(\boldsymbol{z}_j^{(k)}, \boldsymbol{y}_j^{(k)})$ is minimum. As the received bit sequence is Gaussian random, therefore, the individual bit reliability is combined as a symbol reliability using equation (5.6).

Here it is important to mention that each group can have different number of variable nodes. The group containing the variable nodes with maximum votes is considered to be the most unreliable. After decoding of the first group, the next group is decoded under the defined criterion.

In this chapter, a group priority scheme is used to speed up the convergence. A priority is given to a group which contains variable nodes with most probable erroneous tentative decision and are more likely to be corrected. In (5.8), the reliability of a group increase from G_1 to G_α. Based on preset threshold, the variable nodes with minimum reliability are grouped as G_1, and variable nodes with the second reliability are grouped in G_2 and so on. A group can be chosen randomly in case there is a tie among groups.

To group variable nodes, assume that the cumulative density function (CDF) of R_j is known and let this be F_{R_j} and the probability density function (PDF) be f_j.

Definition: Use inverse of F_{R_j} for grouping and let α be the total number of groups, then the number of variable nodes are allocated to the group τ if and only

if the following condition holds:

$$F_{R_j}\big(1-\frac{\tau}{\alpha}\big) < R_j \leq F_{R_j}\big(1-\frac{\tau-1}{\alpha}\big) \tag{5.7}$$

In this section, majority voting scheme presented in [21, 22] is explained where each unsatisfied check node gives one vote to the relevant variable node. The j^{th} variable node then collects all the votes, say $V_j^{(k)}$, from the failed check nodes at k^{th} iteration.

$$V_j^{(k)} = \sum_{i \in M(j)} V_{i,j}^{(k)} \tag{5.8}$$

where $V_{i,j}^{(k)} = 1$ if $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, otherwise $V_{i,j}^{(k)} = 0$. Those variable nodes fulfilling the condition $V_j \geq V_{th}$ will be passed to calculate the flipping function. This reduces the computational complexity from n number of variable nodes to just few variable nodes, say $\boldsymbol{\delta}^{(k)}$. Here $\boldsymbol{\delta}^{(k)}$ stores all the position of those variable nodes having votes greater than the predefined threshold V_{th} and $\delta_{j'}^{(k)}$ is the positions of the j^{th}variable node. In other words, $\boldsymbol{\delta}^{(k)}$ stores the positions of all the variable nodes having less reliable information and must be replaced with reliable symbols from $GF(q)$.

After decoding of the first least reliable group, the decoder has to take decision whether to terminate the sub-iteration and go to main iteration or continue to decode next group. To make this decision, average voting of the variable nodes in that group is computed again. The average voting before and after flipping of the group are compared in order to decide whether to move to next group or terminate sub-iteration and shift to next main iteration. If the average voting of the that group after flipping is less than before flipping, then next group is selected for decoding based on the defined criterion. Therefore, three types of average voting of the variable nodes are required.

Average voting of each group before flipping are stored until the group processing is terminated for next iteration. However, average voting of each group is re-computed only after symbol flipping in that particular group. Average voting of a group remains the same if no symbol is flipped in a group. At each decoding iteration, three types of voting values are computed as follow:

$\bar{V}_\tau$: Average votes of each group. This voting is fixed after grouping the VNs till next iteration.

$\hat{V}_\tau$: Average votes for group G_τ after decoding.

$\ddot{V}_\tau$: Average votes for group G_τ after decoding of group $\tau - 1$.

Let $\beta^{(k,\tau)}$ be the number of variable nodes in a group τ, then the voting based reliability of each group τ at k^{th} iteration is determined by the following method:

$$\bar{V}_\tau^{(k)} = \frac{V_\tau^{(k)}}{\beta^{(k,\tau)}} \tag{5.9}$$

where $V_\tau^{(k)}$ is the sum of votes for a group τ.

Let τ be a group to be decoded, then a next group $G_{\tau+1}$ is selected for decoding based on voting using the following criterion:

$$\tau = \begin{cases} G_{\tau+1}, & \text{if } (\hat{V}_\tau > \bar{V}_\tau) \| (\hat{V}_\tau > \ddot{V}_\tau) \\ \text{Terminate sub-iteration,} & \text{otherwise.} \end{cases} \tag{5.10}$$

The flow chart for equation (5.10) is shown in Figure 5.1.

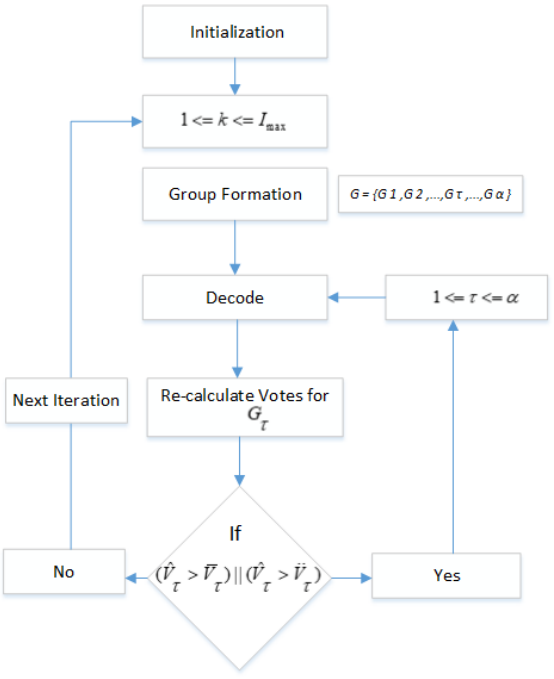


FIGURE 5.1: Flow chart of group decoding

Algorithm 1: Adaptive Group Shuffling

1. Set τ=1 to α.
2. Compute average voting $\bar{V}_\tau$ for each group using equation (5.9).
3. Decode group τ.
4. Compute votes $\hat{V}_\tau$ after flipping.
5. If the condition in equation (5.10) is false then terminate sub-iterations.
6. Set $\tau = \tau + 1$. If $\tau = \alpha$, terminate sub-iteration.

5.3.2 Bit Reliability Based SF Decoding

In this section, each group is decoded using bit reliability based symbol flipping decoding algorithm (B-MSFD) [21]. The flipping function is computed only for the variable nodes in the selected group τ which results in reduced computational complexity.

The symbol position to be flipped must belong to the set of failed checks. In case, the flipping position does not belong to the set of erroneous variable nodes $\mathbf{T}$, then a variable node is selected randomly from the set of erroneous VNs which will perturb the decoder flipping function values to avoid infinite looping [21, 22, 88]. Suppose ψ is the vector containing the number of positions to be flipped then it must be the subset of $\mathbf{T}$, as given in equation (5.14). The i^{th} syndrome of the j^{th} hard decision symbol at k^{th} iteration can be defined as:

$$\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j^{(k)}. \tag{5.11}$$

A valid codeword is received if $\boldsymbol{s}_i^{(k)} = \mathbf{0}$ for $1 \leq i \leq m$. The cause of the failure of check sum $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, is due to the contribution of the variable nodes which are in error. These failed check-sums are stored in a variable υ. For $1 \leq i \leq m$, this can be represented as:

$$\upsilon = \{i |\ \boldsymbol{s}_i \neq \mathbf{0}\}. \tag{5.12}$$

The variable nodes contributing to the successful check sum are stored in a variable $\bar{\upsilon}$. For $1 \leq i \leq m$, this can be represented as follows:

$$\bar{\upsilon} = \{i |\ \boldsymbol{s}_i = \mathbf{0}\}. \tag{5.13}$$

As $\bar{\upsilon}$ contains all the correct variable nodes contributing to $\boldsymbol{s}_i^{(k)} = \mathbf{0}$, therefore, the variable nodes contributing to the failed checks to which ψ must belong as a subset, are defined for $j \in N(i)$ as follows:

$$\psi \in \mathbf{T} = \{j \backslash \upsilon \cap \bar{\upsilon}\}. \tag{5.14}$$

Equation (5.14) contains all the variable nodes contributing to the failed checks.

Let j' is the symbol in a group τ, then the flipping function $E_{j'}^{(k,\tau)}(\boldsymbol{z}_{j'}^{(k,\tau)})$ to find unreliable variable node position $P_{j'}^{(k,\tau)}$ is computed for the group τ as follows:

$$E_{j'}^{(k,\tau)}(\boldsymbol{z}_{j'}^{(k,\tau)}) = \boldsymbol{z}_{j'}^{(k,\tau)} \odot \boldsymbol{y}_{j'}^{(k,\tau)} + \sum_{i \in M_{j'}} \theta_{d(\boldsymbol{z}_{j'}^{(k,\tau)}, \sigma_{i,j'}^{(k,\tau)})} \tag{5.15}$$

Let $\beta^{(k,\tau)}$ be the size of group τ which shows the total number of variable nodes in that group.

$$P_{j'}^{(k,\tau)} = \underset{1 \leq j' \leq \beta^{(k,\tau)}}{\arg\min} \{E_{j'}^{(k,\tau)}(\boldsymbol{z}_{j'}^{(k,\tau)})\} \tag{5.16}$$

The flipping position $P_{j'}^{(k)}$ must belong to $\mathbf{T}$ as given in equation (5.14) otherwise a variable node is chosen randomly from the set of erroneous variable nodes in (5.14).

After choosing the least reliable flipping position, the symbol value for that position is updated by flipping erroneous bit of the symbol. The absolute values of channel soft information are termed as reliability information and a bit with smallest absolute value in a symbol is considered erroneous [14]. The smaller the absolute value of the bit, the less is the reliability of that bit and vice versa. The

expression to find the absolute value of a bit in a j'^{th} received symbol is given as:

$$\acute{y}_{j',t}^{(k,\tau)} = \arg\min_{1\leq t\leq r} |y_{j',t}^{(k,\tau)}| \tag{5.17}$$

The corresponding bit $\acute{z}_{j',t}^{(k,\tau)}$ in the hard decision symbol $z_{j'}^{(k,\tau)}$ is then flipped. The expression for the flipped bit is written as:

$$\acute{z}_{j',t}^{(k,\tau)} = \begin{cases} 1, & \text{if } \; z_{j',t}^{(k,\tau)} = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5.18}$$

The corresponding to the hard decision bit flipping in a symbol, the channel soft reliability information $\boldsymbol{y}_{j'}^{(k,\tau)}$ is also updated using equation:

$$y_{j',t}^{(k,\tau+1)} = \begin{cases} -1 - y_{j',t}^{(k,\tau)}, & \text{if } \; z_{j',t}^{(k,\tau)} = 0 \\ 1 + y_{j',t}^{(k,\tau)}, & \text{if } \; z_{j',t}^{(k,\tau)} = 1. \end{cases} \tag{5.19}$$

The new hard decision symbol sequence is updated for the next $(\tau+1)^{th}$ sub-iteration as:

$$\boldsymbol{z}^{(k,\tau+1)} = (\boldsymbol{z}_1^{(k,\tau)}, \boldsymbol{z}_2^{(k,\tau)}, \ldots, \acute{\boldsymbol{z}}_{j'}^{(k,\tau)}, \ldots, \boldsymbol{z}_{\beta^{(k,\tau)}}^{(k,\tau)}) \tag{5.20}$$

Algorithm-2: Proposed Adaptive Group Shuffled SF (A-BMSFD)

1. Initialization: For $1 \leq j \leq n$, $1 \leq t \leq r$ and $1 \leq i \leq m$.
The hard decision binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is obtained by (5.1).
2. Create CDF values.
3. For $k = 1$ to I_{max}.
4. Check if $\boldsymbol{s}^{(k)} = \mathbf{0}$ or if $k = I_{max}$, declare $\boldsymbol{z}^{(k)}$ as codeword and stop decoding.
5. Map CDF values to the reliability values R as in (5.7) to form groups from low to high reliability.
6. Compute average voting $\bar{V}_\tau$ for each group using equation (5.9).
7. For $\tau = 1$ to α.
8. Find those VNs contributing to failed checks-sum in G_τ by using (5.12), (5.13) and (5.14).
9. Select candidate symbol value by using (5.15),(5.16) and (5.17) .
10. Update HD sequence and the soft information by using (5.18), (5.19) and (5.20) respectively.
11. Calculate votes $\hat{V}_\tau$ after flipping.
12. If the condition in equation (5.10) is false, then go to 14.
13. Set $\tau = \tau + 1$. If $\tau = \alpha$, terminate sub-iteration, else go to 8.
14. Set $k = k + 1$. If $k = I_{max}$ and go to 4.

5.3.3 Fixed Group SF Decoding

In this section, the concept used in [90], has applied to the symbol flipping decoding algorithm. For the given parity check matrix H defined over $GF(q)$, variable nodes are divided into groups such that each group contains more than one variable node and the size of all the groups are equal. For the variable nodes

$V_1, V_2, \ldots, V_j, \ldots, V_n$, let G_τ be a group of H as shown in Figure 5.2, then a set of the groups G is given by:

$$G = \{G_1, G_2, \ldots, G_\tau, \ldots, G_\alpha\} \tag{5.21}$$

for $1 \leq \tau \leq \alpha$ and α shows the total number of groups. It should be noted that in the proposed fixed group decoding, only those groups are chosen for decoding which have erroneous variable nodes and results in failure of check-sum.

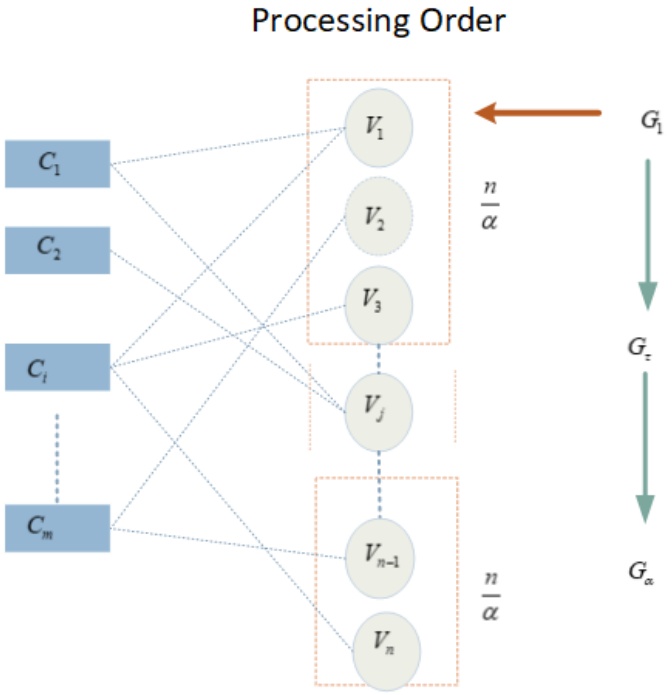


FIGURE 5.2: Vertical Group Decoding

This method helps to decrease overall decoding complexity as all the groups are not computed specially for iteration $k > 1$. The group containing all the reliable variable nodes are not included in the computation of flipping function and symbol value selection. The proposed grouping scheme results in reducing the decoder computational burden. Based on the following condition, a new set of groups is formed as follows;

$$\bar{G} = \{G_\tau : G_\tau \cap T \neq \emptyset, 1 \leq \tau \leq \alpha)\} \tag{5.22}$$

where $\emptyset$ is an empty set. The new set of groups $\bar{G}$ contains $c^{(k)}$ number of groups at k^{th} iteration which might be less than or equal to α and each group size is fixed as $\beta_F = \frac{n}{\alpha}$.

To decide the number of symbols to be flipped per group, let Υ be the preset threshold, then it can be determined in combination to the column weight γ and the number of the variable nodes contributing to the failed checks[21]. If ξ is the total number of symbols to be flipped per iteration then it can be given as follows:

$$\xi = \begin{cases} \xi_1 & \text{if} \quad \Upsilon \geq \varepsilon_1 \rho \\ \xi_2 & \text{else} \end{cases} \tag{5.23}$$

where ε_1 is an integer value. Based on the value of ξ, the number of flipping positions is determined. ξ_2 is normally kept as 1 to avoid decoder oscillation.

Algorithm-3: Fixed Group SF (F-BMSFD)

1. Initialization: For $1 \leq j \leq n$, $1 \leq t \leq r$ and $1 \leq i \leq m$.
The hard decision binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is obtained by (5.1).
2. Divide the parity check matrix H into groups using (5.8).
3. For $k = 1$ to I_{max}.
4. Check if $\boldsymbol{s}^{(k)} = \mathbf{0}$ or if $k = I_{max}$, declare $\boldsymbol{z}^{(k)}$ as codeword and stop decoding.
5. Find erroneous VNs by using (5.12), (5.13) and (5.14).
6. Select groups to decode by using (5.22).
7. For $\tau = 1$ to $c^{(k)}$ where $c^{(k)}$ is the number of groups having at least one erroneous VN.
8. Find the least reliable VN position in the selected group by using (5.15),(5.16) and (5.17).
9. Select candidate symbol value by using (5.15),(5.16) and (5.17).
10. Update HD sequence and the soft information by using (5.18), (5.19) and (5.20).
11. Set $\tau = \tau + 1$. If $\tau = c^{(k)}$, terminate sub-iteration, else go to 8.
12. Set $k = k + 1$. Go to 4.

5.4 Complexity Analysis

The decoding computational complexity and memory requirement of the proposed algorithms in comparison with existing B-MSFD decoding algorithm are evaluated in this section. The computational complexity of the algorithms are explained in terms of numerical computation per iteration. The flipping function for the proposed adaptive group decoding is computed for the selected group only. Let α be the total number of groups and $\beta^{(k,\tau)}$ is the size of group τ at k^{th} iteration where $\beta^{(k,\tau)}$ varies each time with k and τ. Let $\mathcal{C}$ be NB-LDPC code over $GF(q) = 2^r$,

for a given regular parity check matrix H of size $m \times n$ where each column of H has constant weight of γ and each row of H has constant weight of ρ. The total number of edges of the Tanner graph is $\delta = n\gamma$. The main complexity of the algorithm is due to the computation of the flipping function in the equations (5.15) and (5.16). To convert the received sequence into hard decision binary sequence and then to symbols $\boldsymbol{z}$ at the decoder initialization, the algorithms require nr real comparisons.

Let b and $c^{(k)}$ be the converging values such that $1 \leq k \leq b$ and $1 \leq \tau \leq c^{(k)}$, then to compute the syndrome check-sum for valid codeword, $n\gamma$ Galois field multiplication and $n\gamma - m$ Galois field additions are required. For each transmitted code-word, the numerical value of the variables k and τ changes. Therefore, b is the convergence value for a particular transmitted code-word of length n. Similarly, the numerical value of vairable $c^{(k)}$ changes during each iteration and might not obtain the maximum value α. To get the extrinsic information-sums (EXIs) for the n symbols, the decoder requires $n\gamma$ Galois field multiplications and $n\gamma$ Galois field additions but in case of layered decoder, the EXIs are computed for $\beta^{(k,\tau)}$ number of variable nodes in a group τ at the k^{th} iteration, then $\beta^{(k,\tau)}\gamma$ Galois field addition and multiplications are required. To compute the reliability of j^{th} symbol, r real comparisons are required. Voting for each group is obtained after syndrome check-sum computation.

To get votes for a particular symbol j, $\gamma - 1$ additions are required. For the code-word of symbols n, a total of $n\gamma - m$ additions are required. This is the additional computation in the proposed adaptive layered decoding (A-BMSFD) algorithm in comparisons to F-BMSFD and B-MSFD algorithms. The complexity of the proposed A-BMSFD algorithm involves computation of the binary Hamming distance $d(\boldsymbol{z}_j^{(k,\tau)}, \sigma_{i,j}^{(k,\tau)})$ for each edge, the binary function $\boldsymbol{z}_j^{(k,\tau)} \odot \boldsymbol{y}_j^{(k,\tau)}$ for $\beta^{(k,\tau)}$ number of symbols. To compute reliability $R_j^{(k)}$, r real additions are required. To form pre-defined number of groups α based on the reliability, n comparisons are required. Similarly, to compute the votes for each group, $\beta^{(k,\tau)}(\gamma - 1)$ integer additions are required. To compute the Hamming distance $d(\boldsymbol{z}_{j'}^{(k,\tau)}, \sigma_{i,j'}^{(k,\tau)})$ of a symbol j' in a group τ, $\beta^{(k,\tau)}\gamma$ integer comparisons are required and to compute summation of

$\theta^{(k,\tau)}_{d(\boldsymbol{z}^{(k,\tau)}_{j'},\sigma^{(k,\tau)}_{i,j'})}$, $\beta^{(k,\tau)}(\gamma-1)$ additions are required. To find the position of least reliable symbol $P^{(k,\tau)}_{j'}$, $\beta^{(k,\tau)}$ comparisons are required and $\xi^{(k,\tau)}r$ comparison are required to flip a bit in $\xi^{(k,\tau)}$ number of symbols. After symbol flipping, to move to next group or exit to main iteration $k+1$, $\alpha-1$ comparisons are required. Since the proposed A-BMSFD algorithm does not need any loop detection procedure as in B-MSFD algorithm, the decoder computational complexity is reduced. For the F-BMSFD algorithm, the size of each group β_F is fixed and also the grouping is not adaptive, therefore, it does not require voting and reliability information. In comparisons to B-MSFD algorithm, the group based decoding reduces the computation of the flipping function as it computed for the number of variable nodes in a group instead of whole code-word n.

The proposed algorithms for NB-LDPC codes also have an advantage of reduced memory consumption for storing the values of the flipping function as well as flipped values due to the fact that computation is done for the selected group only. However,the values for the CDF are required to store for the allocation of VNs in the next iteration which increases the memory size. Let r bits be required to store each of the symbol over $GF(q)$ and b bits be required for storing floating-point values for each of the reliability metric, then these algorithms require nrb bits to store the received sequence. Similarly, the hard decision symbol sequence $\boldsymbol{z}^{(k,\tau)}_{j'}$ requires nr bits and the extrinsic information-sums $\sigma^{(k,\tau)}_{i,j'}$ requires $\beta^{(k,\tau)}\rho r$ bits.

To store the weighting coefficients $\theta^{(k,\tau)}_{d(\boldsymbol{z}^{(k,\tau)}_{j'},\sigma^{(k,\tau)}_{i,j'})}$ of the binary Hamming distance, ρb bits are required to store the values. To keep each value of the flipping function in (5.15), b bits memory is required and $\beta^{(k,\tau)}rb$ bits for all the values. Similarly $\beta^{(k,\tau)}b$ and $\xi^{(k,\tau)}r$ bits of memory are required for storing $P^{(k,\tau)}_{j'}$ and $\acute{y}^{(k,\tau)}_{j',t}$ respectively. The proposed algorithms reduce the memory requirement from n to $\beta^{(k,\tau)}$. The A-BMSFD algorithm requires only r bits to store the value for the flipped symbol.

Table 5.1 shows the complexity of the proposed adaptive and fixed group based decoding algorithms in comparison to B-MSFD algorithm. From the table, it can be observed that algorithms A-BMSFD, F-BMSFD and B-MSFD algorithms have

TABLE 5.1: Computational complexity for various decoding algorithms

Algorithms	Number of operations			
	GA	GM	IA/RA	RC/IC
A-BMSFD	$\sum_{k=1}^{b}(n\gamma-m)+\sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta^{(k,\tau)}\gamma)$	$\sum_{k=1}^{b}(n\gamma)+\sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta^{(k,\tau)}\gamma)$	$\sum_{k=1}^{b}(n\gamma-m)+\sum_{k=1}^{b}\sum_{\tau=1}^{c(k)}(r+\beta^{(k,\tau)}(\gamma-1)+\xi^{(k,\tau)}r)$	$nr+\sum_{k=1}^{b}((n)+\sum_{\tau=1}^{c^{(k)}}(\beta^{(k,\tau)}r+(\alpha-1)+\beta^{(k,\tau)}+\xi^{(k,\tau)}r)$
F-BMSFD	$\sum_{k=1}^{b}(n\gamma-m)+\sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta_F\gamma)$	$\sum_{k=1}^{b}(n\gamma)+\sum_{k=1}^{b}\sum_{\tau=1}^{c(k)}(\beta_F\gamma)$	$\sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta_F(\gamma-1)+r+\xi_{\tau}^{(k)}r)$	$nr+\sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta_F(r+1)+\alpha\beta_F+\xi_{\tau}^{(k)}r)$
B-MSFD	$\sum_{k=1}^{b}(2nr-m)$	$\sum_{k=1}^{b}(2nr)$	$\sum_{k=1}^{b}(nr+r+\xi^{(k)}r)$	$nr+\sum_{k=1}^{b}(n(r+1)+\xi^{(k)}r)$

IM/IC/IA: Integer Multiplication/Comparision/Adittion;
RA/RM/RC: Real Addition/Multiplication/Comparision; GA/GM: Galois Field Addtion/Multiplication;
Notations used: $\delta = n\gamma = m\rho$.

similar complexity in case all the groups are decoded. Since the overall complexity of an algorithm depends on the converging value b, therefore effect of number of iterations k has been shown in computing numerical values of various decoding operations. The converging value of k varies for each code-word transmission and also for each value of SNR.

Also in case of adaptive group shuffled decoding, the values of $c^{(k)}$ and $\beta^{(k,\tau)}$ are not fixed and can be as small as 1 while in the fixed layered decoding algorithm, $c^{(k)}$ can be less than α as those groups having no erroneous variable nodes are not included in decoding.

5.5 Results and Performance Analysis

In this section, the proposed adaptive group shuffled SF algorithm is compared with the flooding schedule SF algorithms. The algorithms in the following examples for the NB-LDPC code (n, γ, ρ) with γ and ρ as the column and row weights respectively, have been simulated using BPSK as the modulation technique. In the following examples, the total number of iterations are kept as $I_{max} = 15$. The

simulation is terminated either when 100 bits errors are received or the number of transmitted bits reaches to 10^8. The performance and processing complexity of the proposed groups based algorithms are compared with flooding schedule SF algorithm in the literature[21, 22]. An all zeros codeword is transmitted over additive white Gaussian channel(AWGN) using binary phase shift keying modulation. 1 is modulated as +1 and 0 is modulated as -1. The CDF values are generated once for each value of SNR. Both $y - axis$ and $x - axis$ values are rounded to two digits for simplicity and for easy mapping. The weighting coefficients $\theta = [\theta_0, \theta_1 \ldots, \theta_r]$ of the proposed A-BMSFD as well as B-MSFD algorithm [21] are set as $\theta = [2, 0.75, 0.5]$ for $d_{\theta(\boldsymbol{z}_j^{(k,\tau)}, \sigma_{i,j}^{(k,\tau)})} \geq 2$ as it has very low probability to be corrected and does not carry much useful information [31]. Also the values of θ should be in a decreasing order of reliability information. The higher value means more reliability and vice versa. It should be noted that in the following examples, the proposed algorithms flip single symbol per group. The number of symbol per group for the B-BMSD algorithm is also set as 1. In the following examples, the maximum group size of the proposed decoding algorithms are kept as 6.

Example 1: In this example, the proposed decoding algorithms are simulated using a half rate quasi-cyclic (QC) NB-LDPC code (120,3,6) over $GF(128)$, whose column and row weights are 3 and 6 respectively. The performance of the proposed algorithms are also compared with B-MSFD algorithm in the literature. The BER performance of the proposed A-BMSFD and F-BMSFD algorithms show better BER performance than B-BMSFD algorithm as illustrated by performance curves in Figure 5.3.

The performance gap between non-layered B-MSFD and the proposed layered algorithms at 3.5dB and 4.5dB is around 0.5dB which is getting narrow at 5.5dB and 6dB. BER curves shows that the proposed adaptive group shuffled based NB-LDPC decoding algorithm has an improved performance than the proposed fixed group decoding algorithm.

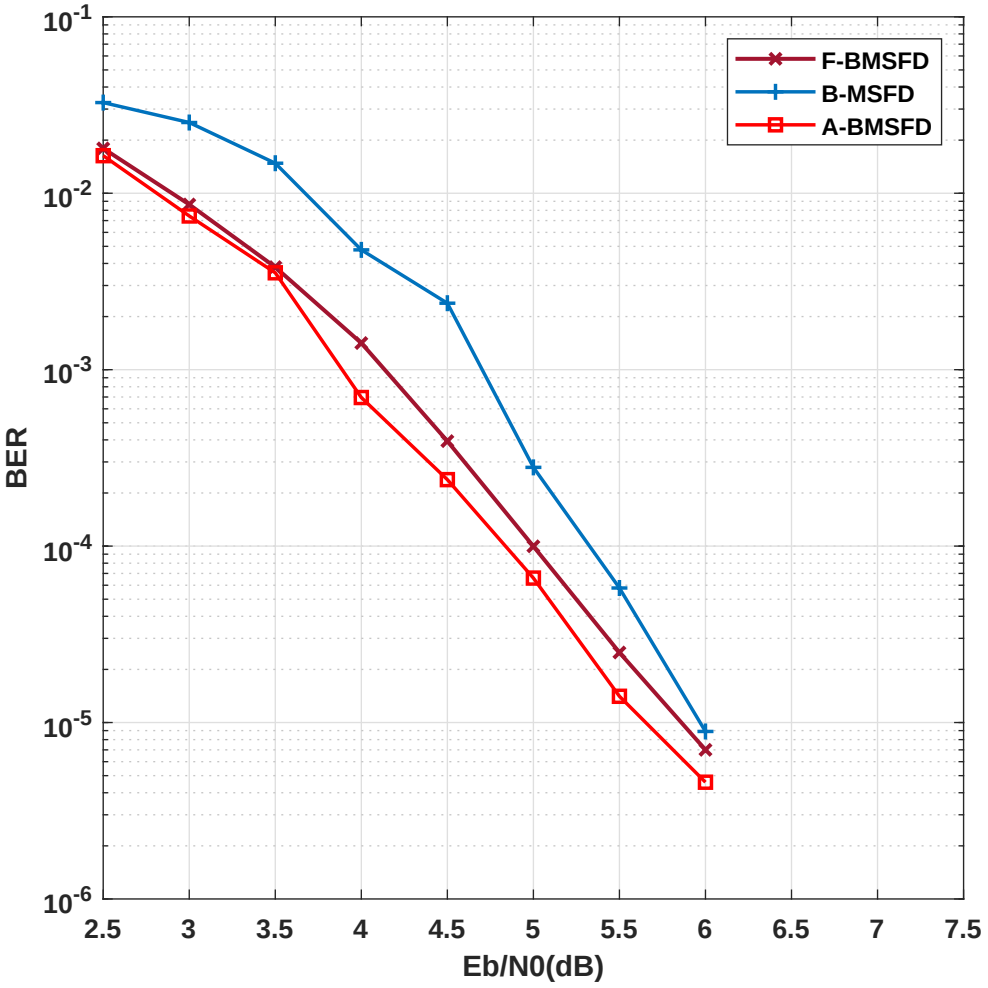


FIGURE 5.3: BER performance of NB-LDPC code (120, 3, 6) over $GF(128)$

Example 2: In this example, the NB-LDPC code (105,2,5) over $GF(8)$ is used to demonstrate the BER performance of the proposed decoding algorithms in comparison to B-MSFD. Here the parity check matrix H with constant row weight $\gamma = 2$ is based on the construction method presented in [66] and the alist format is available on the web site [84]. In this example, the code-word length is 105 and this can not be divided equally into groups. This is a special case of F-BMSFD algorithm and in this case, the even three groups have one more variable node and the group size is 18 while the odd three groups contain 17 variable nodes. Thus the first odd layer contains 17 variable nodes and the second even layer contains 18 variable nodes and so on.

From the BER curves shown in Figure 5.4, it is observed that the proposed adaptive group shuffled decoding algorithm performs better than flooded schedule B-MSFD algorithm at higher SNR. From the curves, it is also clear that these algorithms have less performance for ultra low column parity check matrices. The reason is that these algorithm use the technique to isolate the variable nodes contributing to failed checks and for the ultra low column weight matrices, it becomes difficult to isolate the variable nodes in error by comparing the set of VNs contributing to failed and successful check nodes. In this example as shown in Figure 5.4, the proposed A-BMFSD algorithm has shown better BER performance in comparison to F-MSFD and B-MSFD algorithms.

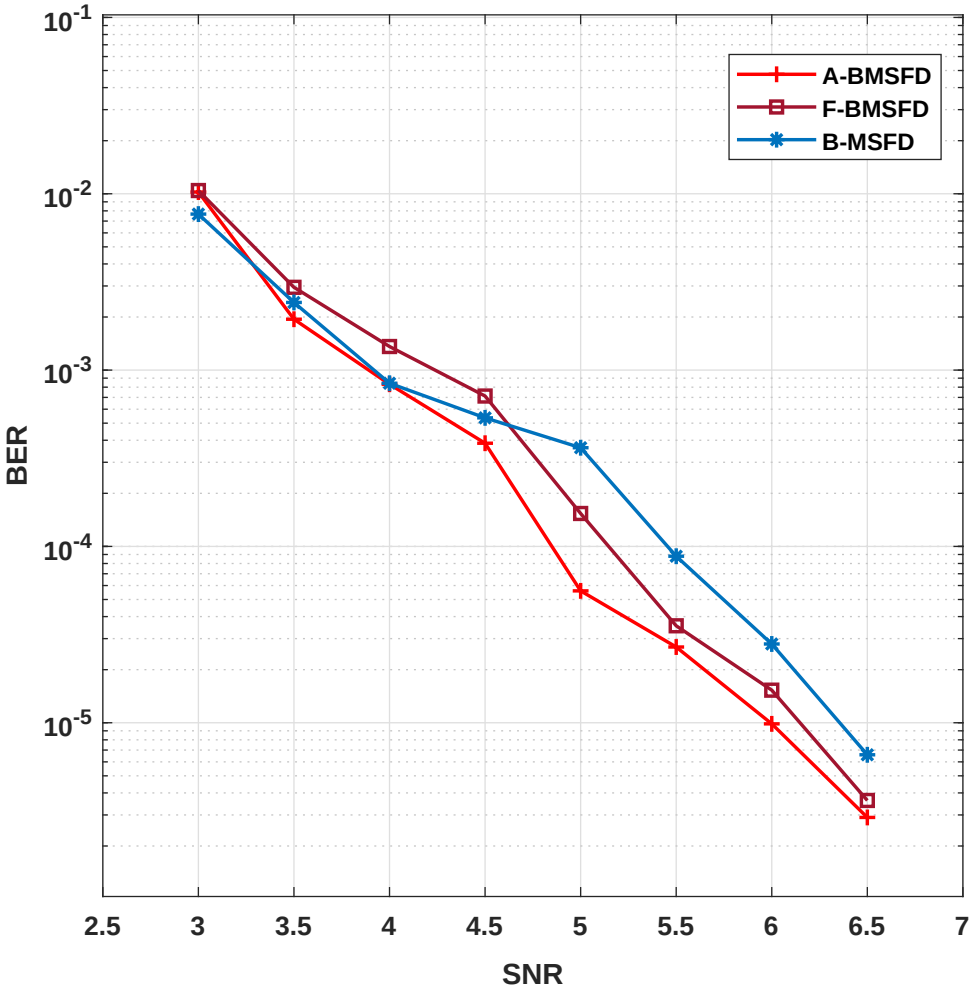


FIGURE 5.4: BER performance of NB-LDPC code (105, 42) over $GF(8)$

Example 3: In this example, Figure 5.5 shows the symbol error rate performance (SER) of the proposed algorithms in comparison to non-layered symbol flipping decoding algorithm. The non-binary LDPC decoding algorithms are simulated for a regular quasi cyclic (QC) NB-LDPC code (120,4,8) over $GF(128)$ to get the SER performance curves. This code has a parity check matrix with constant column weight $\gamma = 4$ and constant row weight $\rho = 8$. The SER performance curves show similar trends as the BER performance curves.

From the symbol error rate performance curves in the Figure 5.5, it is clear that the proposed decoding algorithms SER performance is also better than the B-MSFD. The proposed F-BMSFD algorithm SER performance at $\approx 10^{-3}$ is better than B-MSFD algorithm by approximately 0.5dB but less than A-BMSFD algorithm by around 0.1dB. The proposed A-BMSFD algorithm has shown better SER performance in comparison to F-BMSFD as well as B-MSFD algorithms. In this example, it can be noted that the proposed grouping based decoding performance is better for the relatively high column weigh of the NB-LDPC codes. In the symbol flipping decoding algorithm, computation of the flipping function is the main complexity. The proposed A-BMSFD algorithm takes advantage of the adaptive grouping and computes the flipping function for the variable nodes contained in the selected group τ only.

The reason for the better BER performance of the A-BMSFD algorithm is the reliability and voting based additional information to identify the position of least reliable variable node. In the adaptive scheme, the most unreliable variable nodes are grouped together. This benefits the flipping function to identify the flipping position more accurately as compared to the B-MSFD algorithm.

Example 4: A half rate quasi-cyclic (QC) NB-LDPC code (120, 3, 6) over $GF(128)$ is used whose column and row weights are 3 and 6 respectively. The average iteration (b) versus signal to noise ratio (SNR) is given in Figure 5.6 to illustrate the complexity and convergence of the proposed NB-LDPC decoding algorithm in comparison to B-MSFD algorithm in the literature.

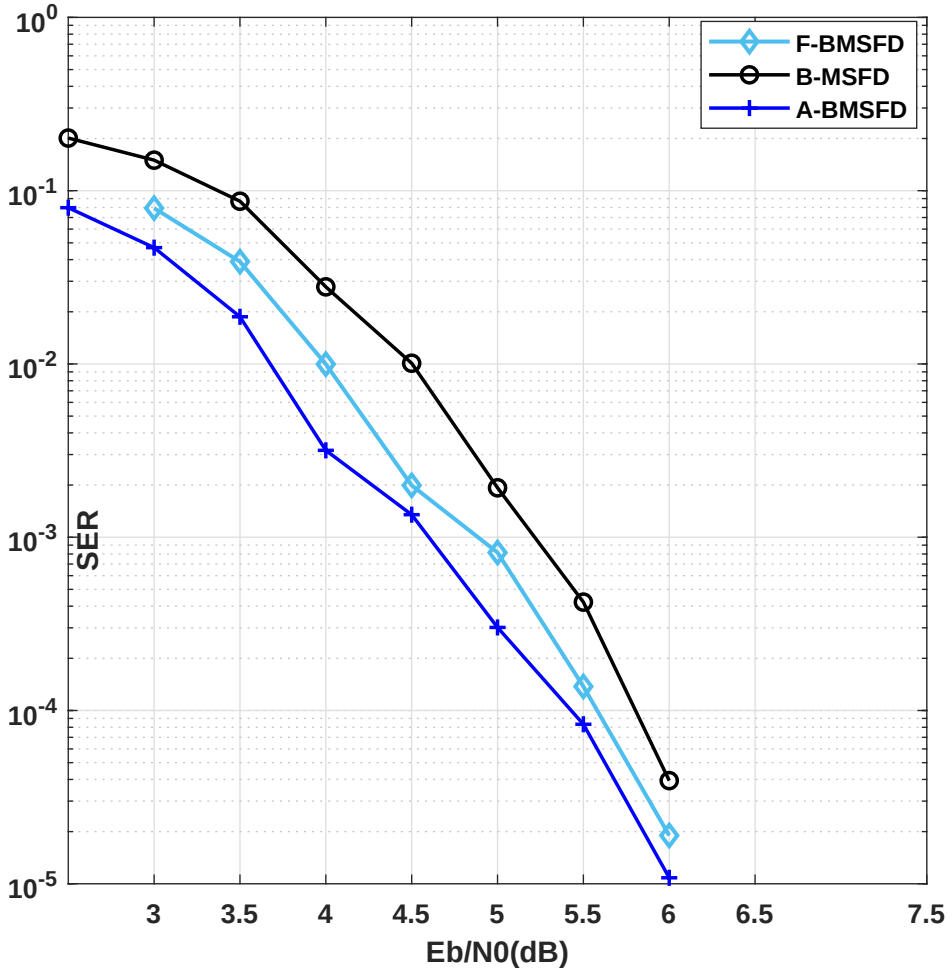


FIGURE 5.5: SER performance of NB-LDPC code(120, 4, 8) over $GF(128)$

For the average number of iteration at 5.5dB, the numerical computation complexity of the considered algorithms are listed in Table 5.2. In the B-MSFD and F-BMSFD algorithms, as the number of symbols to be flipped in each iteration and group, are not fixed and can vary each time, therefore, to compute numerical complexity, the number of symbol flipped per group is fixed as 1 for all the these algorithms. Remember that B-MSFD algorithm is using the code-word as single group. Looking at curves in Figure 5.6, the trend line of the convergence of B-MSFD algorithm is slower than adaptive layered and fixed layered algorithms. At 5.5dB, the average number of iteration of the adaptive layered decoder is getting close to B-MSFD algorithm. Since at higher SNR, there are less number of errors in the received sequence, therefore the convergence of the three algorithms

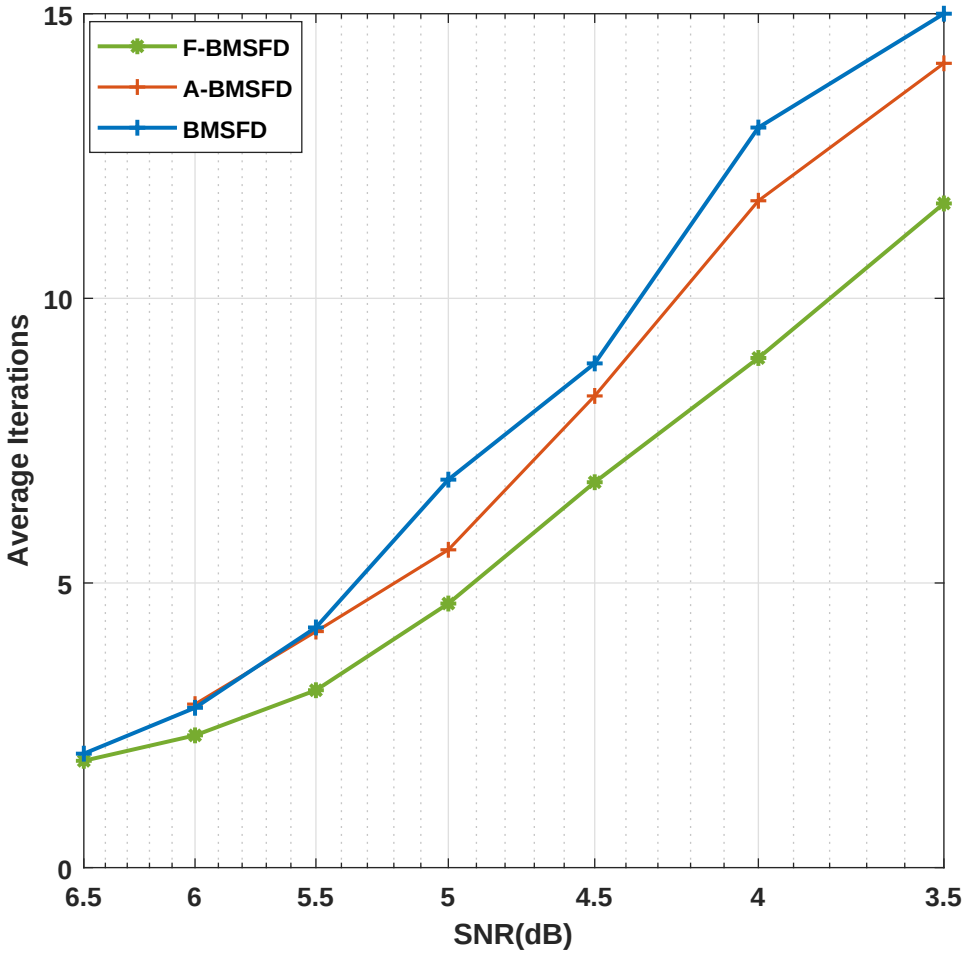


FIGURE 5.6: SNR versus average number of iterations for NB-LDPC code (120, 3, 6) over $GF(128)$

are getting closer at 6.5dB. The primary reason for fast convergence of the fixed layered decoding algorithm is due to the identification of erroneous variable nodes in each group. In a F-BMSFD algorithm, a group is processed only if it contains variable nodes which is useful in identifying a least reliable symbol but this in turn increase the integer comparison as it is obvious from the IC/RC section of Table 5.2. This feature of the F-BMFSD algorithm increases the complexity by 40% from A-BMSFD algorithm but 16% less than B-MSFD algorithm.

In Table 5.1, computational complexity of the algorithms are listed while in the Table 5.2, the computations are given as empirical values which are computed at 5.5dB. In the Table 5.2, the computation are carried out for NB-LDPC code

(120, 3, 6) over $GF(128)$ and the average computations are obtained by repeating the code-word transmission for 1000 times. From the Table 5.2, it is clear that complexity of the newly proposed algorithms is much lower than the existing symbol flipping B-MSFD algorithm in literature. The total number of operations for the proposed NB-LDPC algorithms as well as for the B-MSFD algorithm are given in Table 5.1.

The empirical data in Table 5.2 is obtained by computing the statistical average of the numerical values of $c^{(k)}$ and $\beta^{(k,\tau)}$. The numerical results in Table 5.2 are dependant on the variables listed in Table 5.1. In case of A-BMSFD algorithm, the numerical values depends on b which is the maximum value attained at which the algorithm converges, $c^{(k)}$, $\beta^{(k,\tau)}$ and $\xi^{(k,\tau)}$. The convergence of the proposed decoding algorithms depend on b which is the main iteration loop while $c^{(k)}$ varies at every k^{th} iteration and shows the complexity of the sub-iteration loop. Therefore Table 5.2 shows it clearly that the proposed A-BMSFD algorithm is numerically less complex in comparison to the other algorithms listed in Table 5.2. $\beta^{(k,\tau)}$ shows the size of the particular group τ which also effect the decoding latency and numerical complexity. This can also be observed from the curves in Figure 5.6.

From the complexity curves in Figure 5.6, the average value of b for B-MSFD algorithm is around 4.2. The total number of variable nodes for B-MSFD algorithm is 120. By replacing the variables in Table 5.1 by numerical values to compute GA, the number of operations are computed as 6804 which are approximately the same as the empirical values in Table 5.2. Similarly, for the F-BMSFD algorithm, the value of b is 3.2 as illustrated in Figure 5.6. The value of $c^{(k)}$ is obtained as 2.69 by comparing the values in Table 5.1 and Table 5.2 for $b = 3.2$ and $\beta_F = 20$. These values are in close approximation to get the empirical results in Table 5.2. The average number of iterations for A-BMFSD algorithm is $b = 4.12$ as shown in Figure 5.6, while the empirical value of group size is $\beta^{(k,\tau)} = 14.4$. By replacing these values in the Table 5.1 and comparing with Table 5.2, the value of $c^{(k)}$ is computed as 1.35. From the Table 5.2, it is observed that the proposed adaptive layered decoder has almost 64% less computational complexity than B-MSFD algorithm in terms of GF additions and multiplications but approximately 4%

more than fixed layered decoding algorithm. As the extrinsic information sum is computed for the selected number of variable nodes in a group i.e β_F and $\beta^{(k,\tau)}$ for F-BMSFD and A-BMSFD algorithms respectively, therefore, the number of computation for the proposed algorithms are much less than B-MSFD algorithm. Secondly, the proposed algorithms converge faster than B-MSFD algorithm and thus the numerical complexity is computed for less number of iterations. The proposed adaptive group A-BMSFD algorithm has extra integer additions to compute voting of each symbol in the received hard decision sequence. Therefore, it has an increased complexity in comparison to the fixed group F-BMSFD decoding algorithmic as shown in Table 5.2. From the Table 5.2, this can be observed clearly that the complexity due to IA/RA of the adaptive layered decoder is around 22% less than B-MSFD algorithm and almost similar to the F-BMSFD algorithm.

The average number of iterations of the layered decoding algorithms are less than non-layered symbol flipping decoding and it has an improved BER performance. Therefore, the proposed schemes provide an appealing trade-off between BER and complexity.

TABLE 5.2: Numerical complexity of various algorithms at 5.5dB for code(120, 3, 6) over $GF(128)$

Algorithms	Number of operations			
	GA	GM	RA/IA	IC/RC
A-BMSFD	1478	1704	2252	1460
F-BMSFD	1339	1534	2210	3488
B-MSFD	6807	7059	3588	4903

5.6 Conclusion

In this chapter, two types of layered decoding algorithms for symbol flipping NB-LDPC codes are proposed. The proposed grouping techniques reduce the computational complexity of the flipping function for the received symbols and reduce the decoder processing latency to select new candidate symbol value for the erroneous position. The CDF based adaptive grouping scheme applied to the SF

decoding algorithm helps to reduce numerical computation and also results in an improved BER performance. The proposed A-BMSFD algorithm uses the reliability of the received bit sequence and also takes into account the voting based reliability of each variable node. The voting based reliability information are used to process the next group or terminate the sub-iterations. In the fixed decoding algorithm, each group contains fixed number of variable nodes and the group is processed only if it has some erroneous variable nodes. The proposed two grouping methods can be applied to quasi-cyclic as well as non-quasi-cyclic NB-LDPC codes. The numerical results show that the proposed grouping schemes have efficiently reduced the decoder computation and also achieved better bit error rate performance. From the BER performance curves and the complexity analysis, the proposed group based symbol flipping decoding algorithms are more suitable for applications like storage systems and mobile communication.

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

Ultra reliable and low latency communication (URLLC) is the key concept of the 5^{th} generation communication systems. Other than URLLC, high data rate is another demand of next generation of communication which requires higher order modulation. To achieve ultra reliable data transmission in a noisy environment, NB-LDPC codes play an important role. Also, use of iterative joint detection-decoding of NB-LDPC codes using high order modulation shows improved bit error rate performance.

The main objective of this research is to propose NB-LDPC decoding algorithms with low complexity while maintaining the bit error rate performance. In this thesis, three types of low NB-LDPC decoding algorithms are presented. This goal is achieved by proposing voting based short listing of the least reliable candidate symbols. Also layered technique is applied to the symbol flipping algorithms to achieve better BER performance with reasonable computational complexity. Incorporating the voting scheme to short list the least reliable symbols improves BER performance and reduces decoding latency. The voting scheme also helps in reducing memory requirements for storing the reliability values and extrinsic information for processing.

Over high order modulation, three types of NB-LDPC decoding algorithms are proposed for symbol flipping decoding of NB-LDPC codes. The symbol flipping decoding algorithm for the M-QAM is proposed for the first time in this thesis.

The new concept of symbol flipping based IJDD algorithm has shown better BER performance for the low column and row weights in comparison with the IJDD algorithms in literature. Also, two types of modifications are proposed for the existing IJDD algorithms which have shown improved performance. Euclidean distance based feedback between the received symbol and the newly selected candidate symbol is introduced which help in moving the received symbol in correct direction. Least reliable candidate symbol short-listing by voting is also introduced to reduce the decoding computational complexity.

However, the hardware implementation has remained a challenge for NB-LDPC codes. To achieve hardware friendly architecture, a reduced complexity decoder architecture via layered decoding of LDPC codes is proposed as well. A new adaptive group shuffled algorithm for symbol flipping NB-LDPC codes is proposed. This adaptive group shuffling technique overcomes the complexity of computing the flipping function for all the received symbols and reduces the decoder processing latency for selecting new candidate symbol value for the erroneous position. The CDF based proposed grouping scheme applied to the SF decoding algorithm helps to improve convergence speed and results in an improved BER performance. The groups effectively support the reduced computational complexity as each group contains a fairly less number of variable nodes as compared to the whole parity check matrix. This method can be applied to quasi-cyclic as well as non-quasi-cyclic NB-LDPC codes. The numerical results and complexity analysis show that the proposed decoding algorithms in this thesis offer an appealing trade-off between bit error rate and computational complexity and are more suitable for practical applications like storage systems and mobile communication.

6.2 Recommendations for Future Work

The main focus of this thesis is to achieve high performance and low complexity NB-LDPC decoding. Although, the proposed decoding algorithms work well but there is always space for improvement. For the NB-LDPC decoding, a better hardware implementation is still lacking. The current methods by using voting and

layer techniques lower the decoding complexity and improve performance. However, further investigation can be carried out to improve the high order QAM based NB-LDPC decoding. Extraction of Reliability information from the received sequence of higher order modulation can be studied to improve the flipping function for better flipping decision. Future studies over the CDF values for layered decoding to further improve the performance and efficiency of the proposed algorithms without a significant increase in complexity will be of interest.

References

[1] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes." in *Proceedings of ICC - IEEE International Conference on Communications*. IEEE, 1993.

[3] M. C. Davey and D. MacKay, "Low-density parity check codes over GF(q)," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, 1998.

[4] D. MacKay, "Good error-correcting codes based on very sparse matrices," in *Proceedings of IEEE International Symposium on Information Theory.* IEEE, 1999.

[5] M. F. Declercq, David and E. Biglieri, *Channel Coding: Theory, Algorithms, and Applications*. Academic Press Library in Mobile and Wireless Communications. Academic Press., 2014.

[6] "https://public.ccsds.org/default.aspx."

[7] R. Yazdani and M. Ardakani, "Waterfall performance analysis of finite-length LDPC codes on symmetric channels," *IEEE Transactions on Communications*, vol. 57, no. 11, pp. 3183–3187, nov 2009.

[8] M. Noor-A-Rahim, K. D. Nguyen, and G. Lechner, "Finite length analysis of LDPC codes," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, apr 2014.

[9] X. Hu, Z. Li, B. V. K. V. Kumar, and R. Barndt, "Error floor estimation of long LDPC codes on magnetic recording channels," *IEEE Transactions on Magnetics*, vol. 46, no. 6, pp. 1836–1839, jun 2010.

[10] A. Hareedy, B. Amiri, R. Galbraith, and L. Dolecek, "Non-binary LDPC codes for magnetic recording channels: Error floor analysis and optimized code design," *IEEE Transactions on Communications*, vol. 64, no. 8, pp. 3194–3207, aug 2016.

[11] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(q)," in *Information Theory Workshop, 2003. Proceedings. 2003 IEEE.* IEEE, 2003, pp. 70–73.

[12] Wymeersch, Henk and Steendam, Heidi and Moeneclaey, Marc, "Log-domain decoding of LDPC codes over GF (q)," *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*, 2004.

[13] "An introduction to LDPC codes," in *Coding and Signal Processing for Magnetic Recording Systems.* CRC Press, nov 2004, pp. 649–668.

[14] B. Liu, J. Gao, G. Dou, and W. Tao, "Weighted symbol-flipping decoding for nonbinary LDPC codes," in *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing.* IEEE, 2010.

[15] K. Jagiello and W. E. Ryan, "Iterative plurality-logic and generalized algorithm B decoding of q-ary LDPC codes," in *Proc. IEEE Inf. Theory App. Workshop*, 2011, pp. 1–7.

[16] Huang, Chao Cheng and Wu, Chi Jen and Chen, Chao-Yu and Chao, Chi chao, "Parallel symbol-flipping decoding for non-binary LDPC codes," *IEEE communications letters*, vol. 17, no. 6, pp. 1228–1231, 2013.

[17] L. F. dos Santos, J. Portugheis, and C. de Almeida, "Symbol-flipping decoding of nonbinary LDPC codes over BI-AWGN channels," in *2014 IEEE Latin-America Conference on Communications (LATINCOM).* IEEE, nov 2014.

[18] N.-Q. Nhan, T. M. N. Ngatched, O. A. Dobre, P. Rostaing, K. Amis, and E. Radoi, "Multiple-Votes Parallel Symbol-Flipping Decoding Algorithm for Non-Binary LDPC Code." *IEEE Communications Letters*, vol. 19, no. 6, pp. 905–908, 2015.

[19] F. Garcia-Herrero, E. Li, D. Declercq, and J. Valls, "Multiple-vote symbol-flipping decoder for nonbinary LDPC codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 11, pp. 2256–2267, nov 2014.

[20] S. Wang, Q. Huang, and Z. Wang, "Symbol flipping decoding algorithms based on prediction for non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 65, no. 5, pp. 1913–1924, 2017.

[21] W. Ullah, L. Cheng, and F. Takawira, "Low Complexity Bit Reliability and Predication Based Symbol Value Selection Decoding Algorithms for Non-Binary LDPC Codes," *IEEE Access*, vol. 8, pp. 142 691–142 703, 2020.

[22] W. Ullah, L. Cheng, and F. Takawira, "Prediction and Voting Based Symbol Flipping Non-Binary LDPC Decoding Algorithms ," in *31th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, sep 2020.

[23] C.-Y. Chen, Q. Huang, C. chao Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 58, no. 11, pp. 3140–3147, nov 2010.

[24] Xiong, Chenrong and Yan, Zhiyuan, "Improved iterative soft-reliability-based majority-logic decoding algorithm for non-binary low-density parity-check codes," *IEEE 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2011.

[25] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 11, pp. 1938–1950, 2012.

[26] C. Xiong and Z. Yan, "Improved iterative hard-and soft-reliability based majority-logic decoding algorithms for non-binary low-density parity-check codes," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5449–5457, 2014.

[27] S. Yeo and I.-C. Park, "Improved Hard-Reliability Based Majority-Logic Decoding for Non-Binary LDPC Codes," *IEEE Communications Letters*, vol. 21, no. 2, pp. 230–233, 2017.

[28] Jiang, Xueqin and Lee, Moon Ho, "Large girth non-binary LDPC codes based on finite fields and Euclidean geometries," *IEEE Signal Processing Letters*, 2009.

[29] Li, Juane and Liu, Keke and Lin, Shu and Abdel-Ghaffar, Khaled, "A matrix-theoretic approach to the construction of non-binary quasi-cyclic LDPC codes," *IEEE Transactions on Communications*, 2015.

[30] Zeng, Lingqi and Lan, Lan and Tai, Ying Yu and Zhou, Bo and Lin, Shu and K.Abdel-Ghaffar, "Construction of nonbinary cyclic, quasi-cyclic and regular LDPC codes: A finite geometry approach," *IEEE transactions on Communications*, 2008.

[31] Q. Huang, M. Zhang, Z. Wang, and L. Wang, "Bit-reliability based low-complexity decoding algorithms for non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 62, no. 12, pp. 4230–4240, 2014.

[32] Q. Huang and S. Yuan, "Bit reliability-based decoders for non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 64, no. 1, pp. 38–48, 2016.

[33] 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.

[34] T. Halford and K. Chugg, "An algorithm for counting short cycles in bipartite graphs," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 287–292, jan 2006.

[35] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, "Lowering the error floor of LDPC codes using cyclic liftings," in *2010 IEEE International Symposium on Information Theory*. IEEE, jun 2010.

[36] D. Divsalar, S. Dolinar, and C. Jones, "Construction of protograph LDPC codes with linear minimum distance," in *2006 IEEE International Symposium on Information Theory*. IEEE, jul 2006.

[37] Y. Y. Tai, L. Lan, L. Zeng, S. Lin, and K. Abdel-Ghaffar, "Algebraic construction of quasi-cyclic LDPC codes for the AWGN and erasure channels," *IEEE Transactions on Communications*, vol. 54, no. 10, pp. 1765–1774, oct 2006.

[38] S. Johnson and S. Weller, "Regular low-density parity-check codes from combinatorial designs," in *Proceedings 2001 IEEE Information Theory Workshop (Cat. No.01EX494)*. IEEE, 2001.

[39] B. Vasic, "Combinatorial constructions of low-density parity check codes for iterative decoding," in *Proceedings IEEE International Symposium on Information Theory,*. IEEE, 2004.

[40] L. Lan, L. Zeng, Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach," *IEEE Transactions on Information Theory*, vol. 53, no. 7, pp. 2429–2458, jul 2007.

[41] S. Song, B. Zhou, S. Lin, and K. Abdel-Ghaffar, "A unified approach to the construction of binary and nonbinary quasi-cyclic LDPC codes based on finite

fields," *IEEE Transactions on Communications*, vol. 57, no. 1, pp. 84–93, jan 2009.

[42] S.-Y. Chung, G. Forney, 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, 2001.

[43] 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, 2001.

[44] W. E. Ryan and S. Lin, *Channel Codes*. Cambridge University Press, 2009.

[45] Z. Wang and Z. Cui, "Low-complexity high-speed decoder design for quasi-cyclic LDPC codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 1, pp. 104–114, jan 2007.

[46] Davey, Matthew C and MacKay, David JC, "Monte carlo simulations of infinite low density parity check codes over gf (q)," *Proc. of Int. Workshop on Optimal Codes and related Topics*, 1998.

[47] L. J. Arnone, C. A. Gayoso, C. M. Gonzalez, M. R. Rabini, J. C. Moreira, and P. G. Farrell, "Fast fourier transform simplified soft-distance decoding algorithm for decoding non-binary LDPC codes," in *2014 Argentine Conference on Micro-Nanoelectronics, Technology and Applications (EAMTA)*. IEEE, jul 2014.

[48] P. Farrell, L. Arnone, and J. C. Moreira, "Euclidean distance soft-input soft-output decoding algorithm for low-density parity-check codes," *IET Communications*, vol. 5, no. 16, pp. 2364–2370, nov 2011.

[49] L. Arnone, J. C. Moreira, and P. Farrell, "Field programmable gate arrays implementations of low complexity soft-input soft-output low-density parity-check decoders," *IET Communications*, vol. 6, no. 12, p. 1670, 2012.

[50] C.-L. Wang, X. Chen, Z. Li, and S. Yang, "A simplified min-sum decoding algorithm for non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 61, no. 1, pp. 24–32, jan 2013.

[51] X. Chen and C.-L. Wang, "High-throughput efficient non-binary LDPC decoder based on the simplified min-sum algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2784–2794, nov 2012.

[52] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF (q)," *IEEE Trans. Communications*, vol. 55, no. 4, pp. 633–643, 2007.

[53] A. Voicila, F. Verdier, D. Declercq, M. Fossorier, and P. Urard, "Architecture of a low-complexity non-binary LDPC decoder for high order fields," in *2007 International Symposium on Communications and Information Technologies*. IEEE, oct 2007.

[54] V. Savin, "Min-Max decoding for non binary LDPC codes," in *2008 IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 960–964.

[55] J. O. Lacruz, F. Garcia-Herrero, D. Declercq, and J. Valls, "Simplified trellis min–max decoder architecture for nonbinary low-density parity-check codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, pp. 1783–1792, sep 2015.

[56] S. Kim, "Improved trellis-based decoder for non-binary LDPC codes," in *2016 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, feb 2016.

[57] D. Zhao, X. Ma, C. Chen, and B. Bai, "A low complexity decoding algorithm for majority-logic decodable nonbinary LDPC codes," *IEEE Communications Letters*, vol. 14, no. 11, pp. 1062–1064, 2010.

[58] C. Chen, B. Bai, X. Ma, and X. Wang, "A symbol-reliability based message-passing decoding algorithm for nonbinary LDPC codes over finite fields," in

2010 6th International Symposium on Turbo Codes & Iterative Information Processing. IEEE, sep 2010.

[59] F. Garcia-Herrero, D. Declercq, and J. Valls, "Non-binary LDPC decoder based on symbol flipping with multiple votes," *IEEE Communications Letters*, vol. 18, no. 5, pp. 749–752, may 2014.

[60] Oh, Jieun and Han, Seokju and Ha, Jeongseok, "An Improved Symbol-Flipping Algorithm for Nonbinary LDPC Codes and its Application to NAND Flash Memory,," *IEEE Transactions on Magnetics*, 2019.

[61] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, feb 2005.

[62] Z. chuan gang, Y. jin sheng, L. xue hong, and L. jia ru, "Improvement of Shuffled Iterative Decoding," in *2006 IEEE Information Theory Workshop - ITW Chengdu*. IEEE, oct 2006.

[63] A. Manada, K. Yoshida, H. Morita, and R. Tatasukawa, "A grouping based on local girths for the group shuffled belief propagation decoding," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2133–2136, nov 2016.

[64] T. C.-Y. Chang and Y. T. Su, "Adaptive Group Shuffled Decoding for LDPC Codes," *IEEE Communications Letters*, vol. 21, no. 10, pp. 2118–2121, oct 2017.

[65] 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, apr 2007.

[66] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, dc)-LDPC codes over GF (q) using their binary images," *IEEE Transactions on Communications*, 2008.

[67] B.-Y. Chang, D. Divsalar, and L. Dolecek, "Non-binary protograph-based LDPC codes for short block-lengths," in *2012 IEEE Information Theory Workshop*. IEEE, 2012, pp. 282–286.

[68] D. Declercq and M. Fossorier, "Extended minsum algorithm for decoding LDPC codes over GF (q)," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.* IEEE, 2005, pp. 464–468.

[69] Voicila, Adrian and Declercq, David and Verdier, François and Fossorier, Marc and Urard, Pascal, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE transactions on communications*, 2010.

[70] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary ldpc codes and its hardware structure," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2600–2611, 2013.

[71] E. Boutillon and L. Conde-Canencia, "Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *Electronics Letters*, vol. 46, no. 9, p. 633, 2010.

[72] B. Liu, J. Gao, G. Dou, and W. Tao, "Majority decision based weighted symbol-flipping decoding for nonbinary LDPC codes," in *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, vol. 3. IEEE, 2010, pp. V3–6.

[73] M. Sarvaghad-Moghaddam, W. Ullah, D. N. K. Jayakody, and S. Affes, "A New Construction of High Performance LDPC Matrices for Mobile Networks," *Sensors*, vol. 20, no. 8, p. 2300, apr 2020.

[74] Ullah, Waheed and Yahya, Abid, "Comprehensive Algorithmic Review and Analysis of LDPC Codes," *Indonesian Journal of Electrical Engineering and Computer Science*, 2015.

[75] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 406–414, mar 2002.

[76] E. Boutillon and L. Conde-Canencia, "Bubble check: A simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *Electronics Letters*, vol. 46, no. 9, pp. 633–634, 2010.

[77] P. Schlafer, N. Wehn, M. Alles, T. Lehnigk-Emden, and E. Boutillon, "Syndrome based check node processing of high order NB-LDPC decoders," in *2015 22nd International Conference on Telecommunications (ICT)*. IEEE, apr 2015.

[78] X. Wang, B. Bai, and X. Ma, "A low-complexity joint detection-decoding algorithm for nonbinary LDPC-coded modulation systems," in *2010 IEEE International Symposium on Information Theory*. IEEE, jun 2010.

[79] S. Zhao, B. Bai, X. Wang, X. Ma, and T. Wang, "Joint detection–decoding of majority-logic decodable non-binary low-density parity-check coded modulation systems: an iterative noise reduction algorithm," *IET Communications*, vol. 8, no. 10, pp. 1810–1819, jul 2014.

[80] M. Zhu, Q. Guo, B. Bai, and X. Ma, "Reliability-based joint detection-decoding algorithm for nonbinary LDPC-coded modulation systems," *IEEE Transactions on Communications*, vol. 64, no. 1, pp. 2–14, 2016.

[81] M. Zhu, Q. Guo, H. Xu, B. Bai, and X. Ma, "A multiple-voting-based decoding algorithm for nonbinary LDPC-coded modulation systems," *IEEE Access*, vol. 5, pp. 9739–9747, 2017.

[82] O. Abassi, L. Conde-Canencia, A. Al Ghouwayel, and E. Boutillon, "A novel architecture for elementary-check-node processing in nonbinary ldpc decoders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 2, pp. 136–140, 2016.

[83] C. Chen, B. Bai, X. Wang, and M. Xu, "Nonbinary LDPC codes constructed based on a cyclic MDS code and a low-complexity nonbinary message-passing decoding algorithm," *IEEE Communications Letters*, vol. 14, no. 3, pp. 239–241, mar 2010.

[84] "https://perso-etis.ensea.fr/ declercq/graphs.php."

[85] Z. Xinmiao, *VLSI Architectures for Modern Error-Correcting Codes*. CRC Press, dec 2017.

[86] G. Byers and F. Takawira, "EXIT charts for non-binary LDPC codes," in *IEEE International Conference on Communications, . ICC 2005.* IEEE, 2005.

[87] G. Li, I. J. Fair, and W. A. Krzymien, "Density evolution for nonbinary LDPC codes under gaussian approximation," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 997–1015, mar 2009.

[88] W. Ullah, L. Cheng, and F. Takawira, "Predictive syndrome based low complexity joint iterative detection-decoding algorithm for non-binary LDPC codes," *IEEE Access*, vol. 9, pp. 33 464–33 477, 2021.

[89] C. Marchand, E. Boutillon, H. Harb, L. Conde-Canencia, and A. A. Ghouwayel, "Extended-forward architecture for simplified check node processing in NB-LDPC decoders," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS).* IEEE, oct 2017.

[90] D. Hocevar, "A Reduced Complexity Decoder Architecture via Layered Decoding of LDPC Codes," in *IEEE Workshop onSignal Processing Systems, 2004. SIPS.* IEEE, 2004.

[91] K. Zhang, X. Huang, and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic LDPC codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 985–994, aug 2009.

[92] W. Ullah, T. Jiang, F. Yang, and S. M. Aziz, "Two-Way Normalization of Min-Sum Decoding Algorithm for Medium and Short Length Low Density Parity Check Codes," in *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing.* IEEE, sep 2011.

[93] Y.-L. Ueng, C.-Y. Leong, C.-J. Yang, C.-C. Cheng, K.-H. Liao, and S.-W. Chen, "An Efficient Layered Decoding Architecture for Nonbinary QC-LDPC Codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 2, pp. 385–398, feb 2012.

[94] Y.-L. Ueng, K.-H. Liao, H.-C. Chou, and C.-J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using max-log-QSPA," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2940–2951, jun 2013.

[95] O. Abassi, L. Conde-Canencia, A. A. Ghouwayel, and E. Boutillon, "A novel architecture for elementary-check-node processing in nonbinary LDPC decoders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 2, pp. 136–140, feb 2017.