ISLAMIC UNIVERSITY OF TECHNOLOGY

EFFECIENT DESIGN OF SUM PRODUCT DECODING IN LOG DOMAIN FOR LDPC CODES IN WIRELESS NETWORK

SUPERVISED BY:

DR. MOHAMMAD RAKIBUL ISLAM

PROFESSOR

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

ISLAMIC UNIVERSITY OF TECHNOLOGY

SUBMITTED BY:

TAHMID RASHID

SAMI REFAYET

SHAHRIAR AREFIN OVEE

THIS PAPER IS PUBLISHED ON OCTOBER, 2012 FROM THE DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING, ISLAMIC UNIVERSITY OF TECHNOLOGY, SUBSIDIARY ORGAN OF THE ORGANIZATION OF ISLAMIC CONFERENCE (OIC)

EFFECIENT DESIGN OF SUM PRODUCT DECODING IN LOG DOMAIN FOR LDPC CODES IN WIRELESS NETWORK

A THESIS REPORT SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & ELECTRONIC ENGINEERING, ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT) IN PRACTICAL FULFILLMENT FOR THE DEGREE OF BACHELOR OF SCIENCE IN ELECTRICAL & ELECTRONIC ENGINEERING

SUPERVISED BY:

DR. MOHAMMAD RAKIBUL ISLAM
PROFESSOR
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
ISLAMIC UNIVERSITY OF TECHNOLOGY

SUBMITTED BY:

TAHMID RASHID (**082452**)
SAMI REFAYET (**082458**)
SHAHRIAR AREFIN OVEE (**082464**)
ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT
ISLAMIC UNIVERSITY OF TECHNOLOGY

DEPERTMENT OF ELECTRICAL & ELECTRONIC ENGINEERING,

ISLAMIC UNIVERSITY OF TECHNOLOGY

BOARD BAZAR, GAZIPUR

DECLARATION

This is to certify that the work presented in this thesis is the result of the implementation of ***"Efficient Design of Sum Product Decoding in Log Domain for Wireless Network"****which has been supervised by Prof. Dr Rakibul Islam, Professor, EEE Department of Islamic University Of Technology (IUT).*

This thesis was undertaken as a partial fulfillment of requirements for the Bachelor of Science degree in Electrical &Electronic Engineering. It is also declared that this thesis thereof has not been submitted anywhere else for the award of any degree or any publication.

AUTHORS:

TAHMID RASHID(082452) ----------

SAMI REFAET(082458) ----------

SHAHRIAR AREFIN(082464) ----------

COUNTERSIGNEDBY :

Prof. Dr Rakibul Islam, EEE Department

APPROVED BY:

Prof. Dr Shahid Ullah, Head, EEE Department

ACKNOWLEDGEMENT

At first we would like to start by praising the Almighty ALLAH that the thesis was able to reach the end.

We would like to thank our supervisor Prof. Dr Rakibul Islam, EEE department for his earnest help, guidance and motivation towards the completion of this thesis work.

We would like to thank our parents for their consistent inspiration and encouragement for us to consummate our goal.

We are grateful to all the teachers and staff members of the department of EEE for their consistent support and enthusiasm.

None but the less we thank our batch mates, our beloved friends, for their supports, assists and for being there no matter what.

Abstract:

A coding and modulation technique is studied where the coded bits of an irregular low-density parity-check (LDPC)code are passed directly to a modulator .At the receiver, the variable nodes of the LDPC decoder graph are connected to detector nodes, and iterative decoding is accomplished by viewing the variable and detector nodes as one decoder. The code is optimized by performing a curve fitting on extrinsic information transfer charts. Low Density Parity Check Codes (LDPC) gives groundbreaking performance which is known to approach Shannon's limits for sufficiently large block length. Historically and recently, LDPC have been known to give superior performance than concatenated coding. In this paper we propose a very simple but powerful method for the Sum Product decoding of LDPC codes. A simple, yet effective decoding algorithm is proposed for low-densitya parity-check (LDPC) codes, which significantly simplifies the check node update computation of the optimal sum-product algorithm. It achieves essentially optimal performance by applying scaling in the decoder's extrinsic information. If no such scaling is applied, then the proposed algorithm has small performance degradation. Our method improves the performance of the existing log-domain sum product coding. This is done by introduction of a difference to sum ratio factor, $\mathcal{K}$ in the check to bit node updating process. In this paper we compare the two methods and show that our proposed algorithm is better than the existing one. Our proposed algorithm gives us better signal to noise ratio for a reasonable complexity. Simulation results demonstrate that the proposed algorithms are effective in imparting a better performance in terms of a lower bit error rate (BER) at medium to high signal to noise ratio (SNR) when compared to the traditional sum product algorithm while adding fair amount of complexity

TABLE OF FIGURES' LIST

LIST OF FIGURES	PAGE NUMBER
Fig1.1: BER vs SNR waterfall like curves for sumproduct, minsum and simple log domain	13
Figure 2.1: Tanner graph corresponding to the parity check matrix in equation (1). The marked path $\mathcal{C}_2 \rightarrow \not{f}_1 \rightarrow \mathcal{C}_5 \rightarrow \not{f}_2 \rightarrow \mathcal{C}_2$ is an example for a short cycle. Those should usually be avoided since they are bad for decoding performance	16
Figure 2.2: Bipartite graph associated with the parity check matrix $\mathcal{A}$. (The dashed edges correspond to a cycle of length four, as discussed below.)	19
Figure 2.3: **A** parity check tree associated with the Tanner graph.	21
Figure 2.4: Illustration of the decoding performance of LPDC codes and the number of iterations to achieve decoding.	24
Figure 3.1: A parity check tree associated with the Tanner graph	27
Figure 3.2: A subset of the Tanner graph. (a) Viewed as a tree, with node for c_n as the root. (b) An actual portion of the Tanner graph, with node for c_n as root, showing a cycle.	29
Fig 3.3: The two-tier tree.	31
Fig.4.1: BER performance of proposed algorithm with ½ rate (972, 1944) LDPC codes with iteration number = 15 and frame=5	48
Fig.4.2: BER performance of proposed algorithm with ½ rate (972, 1944) LDPC codes with iteration number = 10 and frame=3	49
Fig 4.3:Comperative Comlexity Calculation of Different Algorithms.	.51
Fig 4.4: Plot of the number of summation and number of multiplication of the different algorithms.	52

Table of Contents

DECLARATION III
ACKNOWLEDGEMENT IV
ABSTRACT 7
TABLE OF FIGURES' LIST 8
CHAPTER 1: INTRODUCTION X
1.1 Objective X
1.2 Wireless Communication XI
1.3 LDPC (Low-Density Parity-Check) Code XI
1.4 History of LDPC Codes XII
1.5 Why LDPC XIV
CHAPTER 2: LOW DENSITY PARITY CHECK CODES XVI
2.1 Introduction XVI
2.2 LDPC properties XVII
2.3 Representation for LDPC codes XVII
2.4 LDPC Codes: Construction and Notation XVIII
2.5 Constructing LDPC codes XIX
2.6 Tanner Graphs XIX
2.7 Performance & Complexity XX
2.8 Transmission Through a Gaussian Channel XXI
2.9 Decoding LDPC Codes XXII
2.10 Why Using LDPC Codes: XXIV
2.11 Codeword and pseudo-codeword weights XXVI
2.12 Stopping sets XXVI
CHAPTER 3: DIFFERENT LDPC DECODING ALGORITHMS XXVII
3.1 Sumproduct Decoding XXVII
3.1.1 Hard Decoder XXVII
3.1.2 The Soft Decoder XXVIII

3.1.3 The Vertical Step: Updating **qmn**(X) XXIX
3.1.4 The Horizontal Step: Updating **rmn**(X) XXXIII
3.1.5 Terminating And Initializing The Decoding Algorithm..... XXXV
3.2 Bit Flipping Algorithm XXXVI
3.2.1 Bit Flip Decoding Algorithm And Its Variants XXXVIII
3.2.2 Bit Flip Decoding Algorithm..... XXXVIII
3.3 MIN-SUM Decoding Algorithm XXXIX
3.3.1 Improved Layered Min-Sum Decoding Algorithm XL
3.4 Log-Likelihood Algorithm..... XLI
CHAPTER 4: PROPOSED ALGORITHM..... XLIII
4.1 Introduction XLIII
4.2 Sum product Algorithm..... XLIV
4.3 Iterative Log Likelihood Decoding Algorithm For Binary LDPC Codes..... XLVII
4.4 Modified Sum Product Log Likelihood Decoding Algorithm For Binary LDPC Codes XLVIII
4.5 Simulation Result XLIX
4.6 Complexity Calculation..... LI
CHAPTER 6: CONCLUSION LV
REFERENCES LVI

CHAPTER 1

INTRODUCTION

1.1 Objective

Traditional communication systems are made up of three major components: the sender, the channel and the receiver. The sender transmits a signal across a noisy channel which introduces distortion to that signal. The receiver receives the now distorted signal and attempts to recover the original signal. In the design of any communication system the designers must consider the channel distortion as it will cause errors possibly rendering the received data unusable to the receiver. In general a certain level of signal distortion may be acceptable but it may be necessary to design a system in which the receiver is capable of correcting the errors in the received data in order to bring the distortion down to an acceptable level. This can be accomplished through the use of an error-correcting coding scheme. An error-correcting coding scheme adds two additional components to the communication system described above. A channel encoder, which adds redundancy to the transmitted data and a channel decoder which exploits this redundancy in order to find and correct errors caused by the channel noise. We have chosen LDPC decoding because of its low decoding complexity, parallel capability, high-speed decoding and flexibility. LDPC codes have already been applied to Digital Video Broadcasting (DVB-S2), Worldwide Interoperability for Microwave Access (WiMAX), Optical and satellite communication, Digital Subscriber Loop (DSL). LDPC codes can be obtained in different ways, but we are now only considering sum product decoding. There have been many works done for the sum product decoding, but we are trying to obtain a code for which the error is very less and close to Shannon limit. In this thesis we wanted to provide significant coding gains over conventional error correcting codes. We tried to design an efficient decoding algorithm for wireless communication using LDPC codes for wireless networking. We also wanted to evaluate some of the fundamental LDPC decoding algorithms for a comparative error performance analysis. Then we proposed a modification to the fundamental algorithms in order to achieve a better error performance with reasonable complexity.

1.2 Wireless Communication

Wireless communications refers to the transfer of information between two or more points that are not physically connected. Distances can be short, such as a few metres for television remote control, or as far as thousands or even millions of kilometres for deep-space radio communications. It encompasses various types of fixed, mobile, and portable applications, including two-way radios, cellular telephones, personal digital assistants (PDAs), and wireless networking. Wireless operations permit services, such as long-range communications, that are impossible or impractical to implement with the use of wires. The term is commonly used in the telecommunications industry to refer to telecommunications systems (e.g. radio transmitters and receivers, remote controls, computer networks, network terminals, etc.) which use some form of energy (e.g. radio frequency (RF),acoustic energy, etc.) to transfer information without the use of wires. Information is transferred in this manner over both short and long distances.Wireless networking is used to meet many needs. Another common use is for mobile networks that connect via satellite. The following situations justify the use of wireless technology: To span a distance beyond the capabilities of typical cabling wireless communication is needed. It is also useful to provide a backup communications link in case of normal network failure. To link portable or temporary workstations this communication is elementary. To overcome situations where normal cabling is difficult or financially impractical wireless communication is implemented. It is also used in remotely connected mobile users or networks. Wireless data communications are an essential component of mobile communication. We used LDPC codes to make an algorithm which would send signals with minimum error rate in the wireless sector. Wireless communications is one of the most active areas in technological department. Low-Density Parity-Check (LDPC) codes have recently drawn much attention due to their near-capacity error correction performance, and are currently in the focus of many standardization activities. In this contribution, we discuss several aspects related to the practical application of such codes to wireless communications systems. We consider flexibility, memory requirements, en- and decoding complexity and different variants of decoding algorithms for LDPC codes that enable to effectively trade-off error correction performance for implementation simplicity. We conclude that many of what have been considered significant disadvantages of LDPC codes can be overcome by appropriate use of different algorithms and strategies that have been recently developed – making LDPC codes a highly attractive option for forward error correction for wireless communications and systems.

1.3 LDPC (Low-Density Parity-Check) Code

(LDPC) code is a type of linear block error correction codes, defined by a sparse parity-check matrix using bipartite graphs. LDPC codes are a class of linear error correcting codes with very sparse parity-check matrices. These codes are usually decoded using the sum-product algorithm,

which is a message passing algorithm working on the Tanner graph of the code. The sparseness of the parity check matrix is essential for attaining good performance with sum-product decoding. The time complexity of the sum- product algorithm is linear in code length. This property makes it possible to implement a practical decoder for long lengths. It is a method of transmitting a message over a noisy transmission channel, and is constructed using a sparse bipartite graph. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close (or even arbitrarily close on the BEC) to the theoretical maximum (the Shannon limit) for a symmetric memory-less channel. The noise threshold defines an upper bound for the channel noise, up to which the probability of lost information can be made as small as desired. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length. LDPC codes are finding increasing use in applications requiring reliable and highly efficient information transfer over bandwidth or return channel–constrained links in the presence of data-corrupting noise. Although implementation of LDPC codes has lagged behind that of other codes, notably turbo codes, the absence of encumbering software patents has made LDPC attractive to some. There are several decoding algorithms are proposed to improve the performance of LDPC codes. They are Bit flipping algorithm ,Sum-product , Loglikelihood decoding, Minsum decoding**,** LDPC decoding based on belief propagation, Bootstrap decoding, Stochastic Decoding, LDPC decoding based on linear programming, Improvement of above algorithms is done by different researchers at different time. The improvements are done by improving the algorithms in order to establish minimum error wireless communication system.

1.4 History of LDPC Codes

Low-density parity-check codes were first proposed by Gallager in 1963 as an approach to linear block codes based on sparse parity-check matrices. Each column and row of the parity-check matrix was required to contain a relatively small and fixed number of 1's. The reasoning behind this design was to be able to decode these codes in linear time. Gallager proposed two different probabilistic decoding methods based on message passing between code-bits and the parity-checks they participate in. This type of decoder is called a message-passing decoder. Since each bit participates in a small fixed number of parity-checks, the complexity of this algorithm is linear. Unfortunately, the block lengths and the calculations required for the decoder were still too difficult for computers of the time and hence the performance of these codes was not fully explored. The discovery of convolutional codes [9] began a new area of study into so called `non algebraic codes'. Non algebraic because they were not based on linear transformations using generator and parity-check matrices, nor were they based on the algebraic constructions used for cyclic codes.

Much later, convolutional codes led to the discovery of a class of codes called Turbo codes [3], which are a type of concatenated convolutional codes that use an interleaver to randomize the

order of some of the bits. These so-called `random' codes have excellent performance characteristics and achieve performance well within 0.5dB of the Shannon limit. Turbo codes also utilize a message passing probabilistic decoder to achieve these results in linear time. Both the BCJR algorithm and the Turbo decoder can be shown to be instances of Pearl's belief propagation on a Bayesian network. This is a method of computing complex probability functions of multiple variables by organizing the relationships between the variables into a network or graph. Shortly after the discovery of Turbo codes, work on other classes of codes that could be decoded effectively using belief propagation led to the rediscovery of LDPC codes by MacKay and Neal where they demonstrate that LDPC codes are good codes with near Shannon limit performance approaching that of Turbo codes. In later work MacKay also proves that under the assumption of an optimal decoder there exist regular LDPC codes which are in fact capacity achieving. The decoding of LDPC codes was done through belief propagation on the factor graph representation of the parity-check matrix. The factor graph representation for LDPC codes was first analyzed by Tanner and it is often referred to as a Tanner graph. The message passing algorithm over a factor graph is known as the sum-product algorithm (SPA) .The authors demonstrate how Pearl's belief propagation, the BCJR algorithm and Turbo decoding can all be implemented as instances of the sum-product algorithm. These impressive results regarding the performance and low complexity of LDPC codes brought attention back to these long forgotten codes and a significant amount of new work has been done recently on the analysis of their performance. In particular, a technique called density evolution (DE) was developed as a means of determining the limit of performance for a class of LDPC codes under the assumption of the sub-optimal SPA decoding rather than the less practical assumption of optimal decoding. The DE technique computes the performance of SPA decoding numerically in the limit of long block length codes and assumes that a large number of iterations are used for decoding. This technique enables the comparison of the average performance of sub-classes of LDPC codes based on the code parameters (i.e., the weights of the rows and columns of the parity-check matrix). For regular LDPC codes each sub-class of LDPC codes is defined by the constant row and column weights for the parity-check matrix. It was shown that the best rate-1/2 regular LDPC code for the AWGN and BSC channels under SPA decoding is the (3,6) code (three parity-checks per bit and six bits per parity-check). In order to find LDPC codes with even better performance the definition of LDPC codes was loosened to allow for irregular LDPC codes where the number of 1's in the rows and columns of the parity-check matrix was allowed to vary from column to column and row to row. Irregular LDPC codes are determined by their degree distribution, which determines the proportion of rows or columns of a particular weight in the parity-check matrix. Using density evolution, Richardson, Shokrollahi and Urbanke were able to evaluate the performance of various degree distributions for the additive white Gaussian noise (AWGN) channel and BSC. They used DE to search for degree distributions which showed particularly good performance limits. The best degree distribution found for the AWGN using this technique has a performance limit under SPA decoding within 0.0045dB of the Shannon limit. While this result assumes impractically long block lengths, this LDPC code still outperforms even Turbo

codes over the AWGN when simulated with similar block lengths and decoder iterations achieving results within 0.04dB of the Shannon limit at a BER of 10^{-6}.The next logical step in the construction of irregular LDPC codes is to design codes which perform this well for channels with memory. Recently, there has been work done on extending the SPA decoding algorithm for LDPC codes to exploit the channel memory in models of channels with memory, such as the GEC. Finding good irregular LDPC codes using DE is much more complex for most channels with memory than for memory less channels.

1.5 Why LDPC

Low density codes have been known to give superior performance than concentrated codes. We have found that extensive research focused on binary LDPC codes have shown to achieve near Shannon capacity. Mackay presented that non-binary LDPC code can achieve increase in performance over their binary counterparts with increasing finite field size.

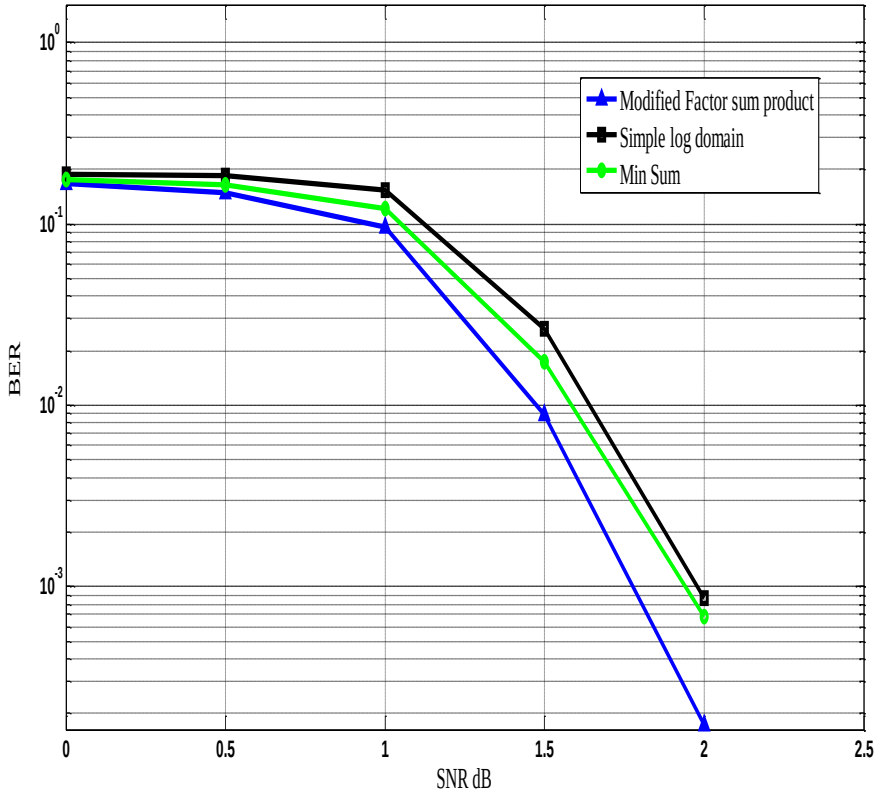


Fig: BER vs SNR waterfall like curves for sumproduct, minsum and simple log domain

LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close to the theoretical maximum (the Shannon limit) for a symmetric memory-less channel. The noise threshold defines an upper bound for the channel noise, up to which the probability of lost information can be made as small as desired. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length. So we chose LDPC codes to build an algorithm for the error correction coding.

CHAPTER 2

LOW DENSITY PARITY CHECK CODES

2.1 INTRODUCTION

In information theory, a low-density parity-check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel, and is constructed using a sparse bipartite graph. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close to the theoretical maximum (the Shannon limit) for a symmetric memory-less channel. The noise threshold defines an upper bound for the channel noise, up to which the probability of lost information can be made as small as desired. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length.

Low density parity check codes are codes specified by a matrix containing mostly 0's and relatively few 1's.In particular, an $(n, \mathcal{W}_c, \mathcal{W}_r)$ low density code is a code of block length n, where each column contains small fixed number $\mathcal{W}_c$ of 1's and each row contains small fixed number $\mathcal{W}_r$ of 1's.This type of matrix does not have the check digits appearing in diagonal form.However, for coding purposes, the equation represented by these matrices can always be solved to give check digits as explicit sums of information digits.

Low-density parity-check (LDPC) codes have performance exceeding, in some cases, that of turbo codes, with iterative decoding algorithms which are easy to implement (with the per-iteration complexity much lower than the per-iteration complexity of turbo decoders), and are also parallelizable in hardware. There are other potential advantages to LDPC codes as well. In a very natural way, the decoder declares a decoding failure when it is unable to correctly decode, whereas turbo decoders must perform extra computations for a stopping criterion (and even then, the stopping criterion depends upon a threshold that must be established, and the stopping criterion does not establish that a codeword has been found). Also, LDPC codes of almost any rate and block length can be created simply by specifying the shape of the parity check matrix, while the rate of turbo codes is governed largely by a puncturing schedule, so flexibility in rate is obtained only through considerable design effort. Also, because the validity of a codeword is validated if its parity checks, even when errors do occur, they are almost always detected errors (especially for long codes). As an additional boon on the commercial side, LDPC codes are not patent protected. On the negative side, LDPC codes have a significantly higher encode complexity than turbo codes, being generically quadratic in the code dimension, although this

can be reduced somewhat. Also, decoding may require many more iterations than turbo decoding, which has implications for latency.

2.2 LDPC properties

It is important to keep in mind that, when considering the properties of an LDPC code, often we are actually considering the properties of a particular choice of parity-check matrix for that code; a different choice of parity-check matrix for the same code might behave differently with sum–product decoding. This is in contrast with ML decoding, where the probability of incorrect decoding is determined solely by the weight distribution of the codewords, which in turn is independent of which generator or parity-check matrix is chosen to represent the code.

2.3 Representation for LDPC codes

Basically there are two different possibilities to represent LDPC codes. Like all linear block codes they can be described via matrices. The second possibility is a graphical representation.

Matrix Representation

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

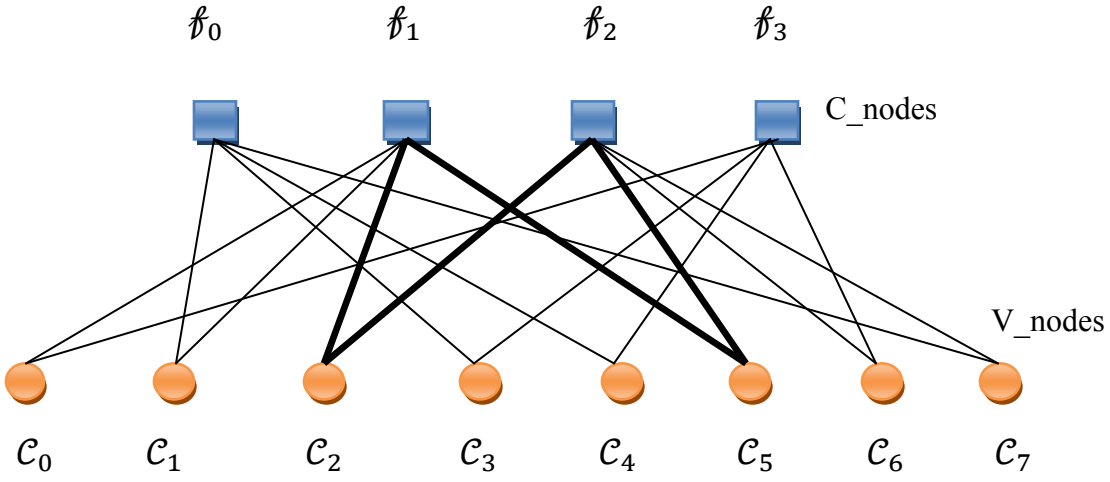


Figure 2.1: Tanner graph corresponding to the parity check matrix in equation (1). The marked path $c_2 \rightarrow f_1 \rightarrow c_5 \rightarrow f_2 \rightarrow c_2$ is an example for a short cycle. Those should usually be avoided since they are bad for decoding performance.

Let's look at an example for a low-density parity-check matrix first. The matrix defined in equation (1) is a parity check matrix with dimension $(n \times m)$ for a $(8, 4)$ code. We can now define two numbers describing this matrix. $\mathcal{W}_r$ for the number of 1's in each row and $\mathcal{W}_c$ for the columns. For a matrix to be called low-density the two conditions $\mathcal{W}_r \ll n$ and $\mathcal{W}_c \ll m$ must be satisfied. In order to do this, the parity check matrix should usually be very large, so the example matrix can't be really called low-density.

Graphical Representation

Tanner introduced an effective graphical representation for LDPC codes. Not only provide these graphs a complete representation of the code, they also help to describe the decoding algorithm as explained later on in this tutorial. Tanner graphs are bipartite graphs. That means that the nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The two types of nodes in a Tanner graph are called variable nodes (v-nodes) and check nodes (c-nodes). Figure 1 is an example for such a Tanner graph and represents the same code as the matrix in 1. The creation of such a graph is rather straight forward. It consists of m check nodes (the number of parity bits) and n variable nodes (the number of bits in a codeword). Check node f_i is connected to variable node c_j if the element h_{ij} of H is a 1.

2.4 LDPC Codes: Construction and Notation

We use $\mathcal{N}$ to denote the length of the code and $\mathcal{K}$ to denote its dimension and $\mathcal{M} = \mathcal{N} - \mathcal{K}$.Throughout this chapter, only binary LDPC codes are considered (although they can be constructed over other fields). Since the parity check matrices we consider are generally not in systemic form, we usually use the symbol $\mathcal{A}$ to represent parity check matrices, reserving the symbol $\mathcal{H}$ for parity check matrices in systematic form. Following the general convention in the literature for LDPC codes, we assume that vectors are column vectors. A message vector m is a $\mathcal{K} \times 1$ vector; a codeword is a $\mathcal{N} \times 1$ vector. The generator matrix $\mathcal{G}$ is $\mathcal{N} \times \mathcal{K}$ and the parity check matrix $\mathcal{A}$ is$(\mathcal{N} - \mathcal{K}) \times \mathcal{N}$, such that $\mathcal{H}\mathcal{G} = 0$. We denote the rows of a parity check matrix as

$$\mathcal{A} = \begin{bmatrix} a_1^T \\ a_2^T \\ . \\ . \\ . \\ a_{\mathcal{M}}^T \end{bmatrix}$$

The equation $a_i^T c = 0$ is said to be a linear parity-check constraint on the codeword c. We use the notation $z_m = a_m^T c$ and call z_m a parity check or, more simply, a check. For a code specified by a parity check matrix , it is expedient for encoding purposes to determine the corresponding generator matrix $\mathcal{G}$. A systematic generator matrix may be found as follows.

Using Gaussian elimination with column pivoting as necessary (with binary arithmetic) determine an $\mathcal{M} \times \mathcal{M}$ matrix $\mathcal{A}_{\mathcal{p}}^{-1}$ so that

$$\mathcal{H} = \mathcal{A}_{\mathcal{p}}^{-1} \times \mathcal{A} = [\mathrm{I} \quad \mathcal{A}_2]$$

(If such a matrix , does not exist, then $\mathcal{A}$ is rank deficient, $r = \mathrm{rank}(\mathcal{A}) < \mathcal{M}$. In this case, form $\mathcal{H}$ by truncating the linearly dependent rows from $\mathcal{A}_{\mathcal{p}}^{-1}\mathcal{A}$. The corresponding code has $\mathcal{R} = \mathcal{K} / \mathcal{N} > (\mathcal{N} - \mathcal{M})/\mathcal{N}$, so it is a higher rate code than the dimensions of $\mathcal{A}$ would suggest.) Having found $\mathcal{H}$, form

$$\mathcal{G} = \begin{bmatrix} \mathcal{A}_2 \\ \mathrm{I} \end{bmatrix}$$

Then $\mathcal{H}\mathcal{G} = 0$, so $\mathcal{A}_{\mathcal{p}}\mathcal{H}\mathcal{G} = \mathcal{A}\mathcal{G} = 0$, so $\mathcal{G}$ is a generator matrix for $\mathcal{A}$. While $\mathcal{A}$ may be sparse (as discussed below), neither the systematic generator $\mathcal{G}$ nor $\mathcal{H}$ is necessarily sparse. A matrix is said to be sparse if fewer than half of the elements are nonzero.

2.5 Constructing LDPC codes

Several different algorithms exist to construct suitable LDPC codes. Gallager himself introduced one. Furthermore MacKay proposed one MacKay to semi-randomly generate sparse parity check matrices. This is quite interesting since it indicates that constructing good performing LDPC codes is not a hard problem. In fact, completely randomly chosen codes are good with a high probability. The problem that will arise is that the encoding complexity of such codes is usually rather high.

2.6 Tanner Graphs

Associated with a parity check matrix $\mathcal{A}$ is a graph called the Tanner graph containing two sets of nodes. The first set consists of $\mathcal{N}$ nodes which represent the $\mathcal{N}$ bits of a codeword; nodes in this set are called "bit" nodes. The second set consists of $\mathcal{M}$ nodes, called "check" nodes, representing the parity constraints. The graph has an edge between the n^{th} bit node and the m^{th} check node if and only if n^{th} bit is involved in the m^{th} check, that is, if $\mathcal{A}_{mn} = 1$. Thus the Tanner graph is a graphical depiction of the parity check matrix. Figure 2.2 illustrates the graph for $\mathcal{A}$ .A graph such as this, consisting of two distinct sets of nodes and having edges only between the nodes in different sets, is called a bipartite graph. The Tanner graph is used below to develop insight into the decoding algorithm. For the Tanner graph representation of the parity check matrix of a regular code, each bit node is adjacent to $\mathcal{W}_c$ check nodes and each check node is adjacent to $\mathcal{W}_r$ b it nodes.

$$\mathcal{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

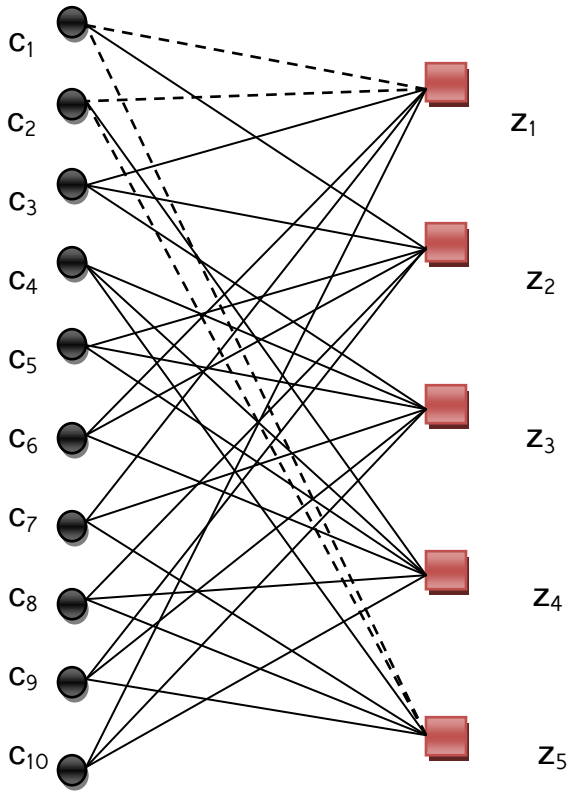


Figure 2.2: Bipartite graph associated with the parity check matrix $\mathcal{A}$. (The dashed edges correspond to a cycle of length four, as discussed below.)

2.7 Performance & Complexity

Before describing decoding algorithms in chapter 3, we would like to explain why all this effort is needed. The feature of LDPC codes to perform near the Shannon limit1 of a channel exists only for large block lengths. The large block length results also in large parity-check and generator matrices. The complexity of multiplying a codeword with a matrix depends on the amount of 1's in the matrix. If we put the sparse matrix H in the form [P^T I] via Gaussian

elimination the generator matrix G can be calculated as G = [I P]. The sub-matrix P is generally not sparse so that the encoding complexity will be quite high.

The term "local calculations" already indicates that a divide and conquer strategy, which separates a complex problem into manageable sub-problems, is realized. A sparse parity check matrix now helps these algorithms in several ways. First it helps to keep both the local calculations simple and also reduces the complexity of combining the sub-problems by reducing the number of needed messages to exchange all the information. Furthermore it was observed that iterative decoding algorithms of sparse codes perform very close to the optimal maximum likelihood decoder.

2.8 Transmission Through a Gaussian Channel

The decoding algorithm described below is a soft-decision decoder which makes use of channel information. We develop the decoder for code words transmitted through an additive white Gaussian noise (AWGN) channel. When a codeword is transmitted through an AWGN channel the binary vector c is first mapped into a transmitted signal vector t. For illustrative purposes, a binary phase-shift keyed (BPSK) signal constellation is employed, so that the signal $a = \sqrt{E_c}$ represents the bit 1 and the signal - a represents the bit 0. The energy per message bit E_b is related to the energy per transmitted coded bit E_c by $E_c = RE_b$, where $\mathcal{R} = k/n$ is the rate of the code. The transmitted signal vector t has elements $t_n = (2c_n - 1)a$. This signal vector passes through a channel which adds a Gaussian noise vector $\mathcal{V}$, where each element of $\mathcal{V}$ is independent, identically distributed with zero mean and variance $\sigma^2 = N_0/2$. The received signal is

$$r = t + \mathcal{V}$$

Given the received data, the posterior probability of detection can be computed as

$$\mathcal{P}(c_n = 1|r_n) = \frac{p(r_n \mid t_n = a)\mathcal{P}(t_n = a)}{p(r_n)}$$

$$= \frac{p(r_n \mid t_n = a)\mathcal{P}(t_n = a)}{p(r_n \mid t_n = a)\mathcal{P}(t_n = a) + p(r_n \mid t_n = -a)\mathcal{P}(t_n = -a)}$$

$$= \frac{p(r_n \mid t_n = a)}{p(r_n \mid t_n = a) + p(r_n \mid t_n = -a)}$$

where it assumed that $\mathcal{P}(c_n) = 1 = \mathcal{P}(c_n) = 0 = \frac{1}{2}$. The notation $\mathcal{P}(.)$ indicates a probability mass function and $p(.)$ indicates a probability density function. For an AWGN channel,

$$p(r_n \mid t_n = a) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{1}{2\sigma^2}(r_n - t_n)^2} = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{1}{2\sigma^2}(r_n - a)^2}$$

So we get,

$$\mathcal{P}(c_n = 1 | r_n) = \frac{1}{1 + e^{-2ar_n/\sigma^2}}$$

We shall refer to $\mathcal{P}(c_n = x | r_n)$ as the channel posterior probability and denote it by $p_n(x)$.

2.9 Decoding LDPC Codes

Some insight can be gained into the soft, iterative decoding algorithm we will eventually develop by considering a preliminary, iterative, hard decision decoder:

Hard Decoder: For each bit c_n compute the checks for those checks that are influenced by c_n. If the number of nonzero checks exceeds some threshold (say, the majority of the checks are nonzero), then the bit is determined to be incorrect. The erroneous bit is flipped and correction continues. This simple scheme is capable of correcting more than one error, as we now explain. Suppose that c_n is in error and that other bits influencing its checks are also in error. Arrange the Tanner graph with c_n as a root (neglecting for now the possibility of cycles in the graph). In Figure 2.3, suppose the bits in the shaded boxes are in error. The bits that connect to the checks connected to the root node are said to be in tier 1. The bits that connect to the checks from the first tier are said to be in tier 2. Many such tiers could be established. Then, decode by proceeding from the "leaves" of the tree (the top of the figure). By the time decoding on c_n is reached, other erroneous bits may have been corrected. Thus bits and checks which are not directly connected to c_n can still influence c_n.

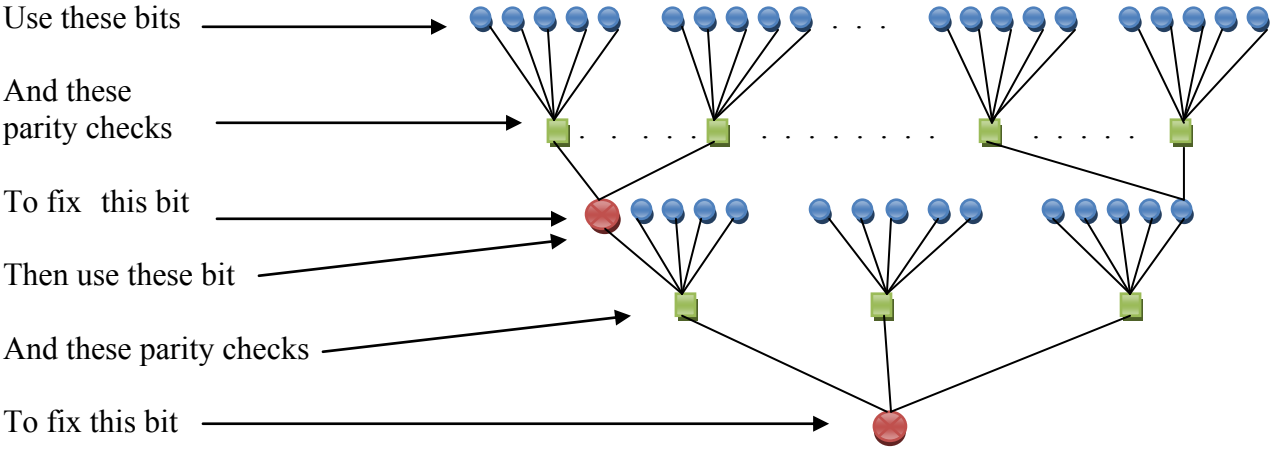


Figure 2.3: **A** parity check tree associated with the Tanner graph.

The soft decoder: In the soft decoder, rather than flipping bits (a hard operation), we propagate probabilities through the Tanner graph, thereby accumulating evidence that the checks provide about the bits. The optimal (minimum probability of decoding error) decoder seeks a codeword $\hat{c}_n$ which maximizes $\mathcal{P}(c \mid r, \mathcal{A}c = 0)$ that is, the most probable vector which satisfies the parity checks, given set of received data $r = [r_1 r_2 \ldots \ldots r_N]$.However, the decoding complexity for the true optimum decoding of an unstructured (i.e., random) code is exponential in $\mathcal{K}$, requiring an exhaustive search over all $2\mathcal{K}$ code words. Instead, the decoder attempts to find a codeword having bits c_n which maximize

$$\mathcal{P}(c_n \mid r, \text{all checks involving bit } c_n \text{ are satisfied})$$

that is, the posterior probability for a single bit given that only the checks on that bit are satisfied. As it turns out, even this easier, more computationally localized, task cannot be exactly accomplished due to approximations the practical algorithm must make. However, the decoding algorithm has excellent demonstrated performance and the complexity of the decoding is linear in the code length.

The decoding algorithm deals with two sets of probabilities. The first set of probabilities is related to the decoding criterion,

$$\mathcal{P}(c_n \mid r, \text{all checks involving bit } c_n \text{ are satisfied})$$

We denote this by,

$$q_n(x) = \mathcal{P}(c_n \mid r, \text{all checks involving bit } c_n \text{ are satisfied}) \ , \ \mathcal{X} \in \{0,1\}$$

or, using the notation defined in Section

$$q_n(x) = \mathcal{P}(c_n \mid r, \{z_m = 0, m \in \mathcal{M}_n\}), \quad \mathcal{X} \in \{0,1\}$$

This probability is referred to as the pseudoposterior probability and is ultimately used to make the decisions about the decoded bits. A variant of this probability, called $q_{mn}(x)$, is also used, which is

$$q_{mn}(x) = \mathcal{P}(c_n = \mathcal{X} \mid r, \text{all checks, except } z_m, \text{involving } c_n \text{are satisfied})$$

or, more briefly,

$$q_{mn}(x) = \mathcal{P}(c_n \mid r, \{z'_m = 0, m' \in \mathcal{M}_{n,m}\})$$

The second set of probabilities has to do with the probability of checks given the bits. These indicate the probability that a check is satisfied, given the value of a single bit involved with that check and the observations associated with that check. This probability is denoted by $r_{mn}(x)$, with $r_{mn}(x) = \mathcal{P}(z_m = 0 \mid c_n = x, r)$.

The quantities $q_{mn}(x)$ and $r_{mn}(x)$ are computed only for those elements $\mathcal{A}_{mn}$of $\mathcal{A}$ that are nonzero. The decoding algorithm incorporates information from the measured data to compute probabilities about the checks, as represented by $r_{mn}(x)$. The information about the checks is then used to find information about the bits, as represented by $q_{mn}(x)$This, in turn, is used to update the probabilities about the checks, and so forth. This amounts to propagating through the "tree" derived from the Tanner graph. Iteration between bit and check probabilities (qs and rs) proceeds until all the parity checks are satisfied simultaneously, or until a specified number of iterations is exceeded.

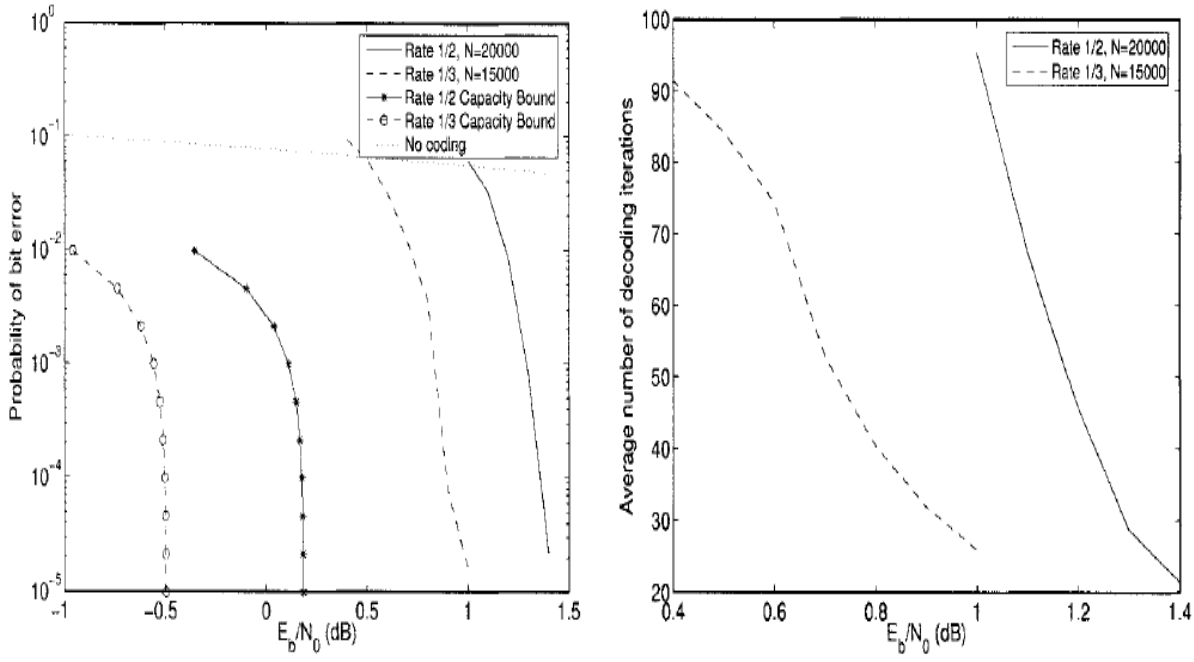
2.10 Why Using LDPC Codes:

LDPC codes have excellent distance properties. Gallager showed that for random LDPC codes, the minimum distance d_{min} between codewords increases with $\mathcal{N}$ when column and row weights are held fixed, that is, as they become increasingly sparse. Sequences of LDPC codes as $\mathcal{N} \to \infty$ have been proved to reach channel capacity. LDPC codes thus essentially act like the random codes used in Shannon's original proof of the channel coding theorem. Note, however, that for the nonrandom constructive techniques there may be an error floor.

The decoding algorithm is tractable. As observed, the decoding algorithm has complexity linearly proportional to the length of the code. Thus we get the benefit of a random code, but without the exponential decoding complexity usually associated with random codes. These codes thus fly in the face of the now outdated conventional coding wisdom, that there are "few known constructive codes that are good, fewer still that are practical, and none at all that are both practical and very good." It is the extreme sparseness of the parity check matrix for LDPC codes that makes the decoding particularly attractive. The low-density nature of the parity check matrix thus, fortuitously, contributes both to good distance properties and the relatively low complexity of the decoding algorithm.

For finite length (but still long) codes, excellent coding gains are achievable ,the BPSK probability of error performance for two LDPC codes, a rate 1/2 code with $(\mathcal{N}, \mathcal{K}) = (20000, 10000)$ and a rate 1/4 code with $(\mathcal{N}, \mathcal{K}) = (13298, 3296)$,compared with uncoded BPSK. These plots were made by adding simulated Gaussian noise to a codeword then iterating the algorithm up to 1000 times. As many as 100000 blocks of bits were simulated to get the performance points at the higher SNRs. In all cases, the errors counted in the probability of error are detected errors; in no case did the decoder declare a successful decoding that was erroneous! (This is not always the case. We have found that for very short toy codes, the decoder may terminate with the condition $\mathcal{A}\hat{c} = 0$, but $\hat{c}$ is erroneous. However, for long codes, decoding success meant correct decoding.)

Figure 2.4(b) shows the average number of iterations to complete the decoding. (The peak number of iterations, not shown, was in many instances much higher.) The high number of



(a) Performance for a rate 1/2 and a rate 1/4 code.

(b) Average number of decoding iterations.

Figure 2.4: Illustration of the decoding performance of LPDC codes and the number of iterations to achieve decoding.

iterations suggests a rather high potential decoding complexity, even though each iteration is readily computed. As the decoding threshold is approached, the number of iterations must increase. There are, of course, some potential disadvantages to LDPC codes. First, the best code performance is obtained for very long codes (as predicted by the channel coding theorem). This long block length, combined with the need for iterative decoding, introduces latency which is unacceptable in many applications. Second, since the $\boldsymbol{\mathcal{G}}$ matrix is not necessarily sparse, the encoding operation may have complexity $\mathcal{O}(\mathcal{N}^2)$**.** Some progress in reducing complexity is discussed in Section 15.12.

LDPC codes have an error floor, just as turbo codes do. Some efforts to lower the error floor have been made by increasing the girth of the Tanner graph, but this has met with only limited success. **A** tradeoff between decoding thresholds (from the density evolution analysis) and the error floor has been observed. Codes having very low error floors tend to perform around half a dB worse in terms of their decoding thresholds. However, this has not yet led to any design methodologies to reduce error floor.

2.11 Codeword and pseudo-codeword weights

We saw in Section 1.3 that the ML decoding performance of a given error correction code on a given channel depends entirely on the weight distribution of the code words. Furthermore, in channels with low noise the decoding performance can be predicted just by knowing the minimum distance of the code. Although LDPC codes are not decoded with ML decoding, the aim of an iterative decoder is to approach ML or MAP decoding performance and so the codeword weight distribution can be used to determine bounds on the performance of the iterative decoder for that code. We will consider the ML performance of LDPC codes in Chapter 8. Here we will discuss how the structure of a chosen LDPC code can affect its minimum distance. Regarding the performance of a code with sum–product decoding, errors can arise not just when the decoder chooses the wrong codeword; the decoder may fail to converge to any codeword. In such cases the decoder can be thought of as having converged not to a codeword but to a code structure called a pseudo codeword. When using iterative erasure decoding, pseudo-codewords have a straightforward definition and are called stopping sets.

2.12 Stopping sets

As we know, the message-passing decoding of LDPC codes on erasure channels is particularly straightforward since a transmitted bit is either received correctly or completely erased. Message-passing decoding is a process of finding parity-check equations that will check on only one erased bit. In a decode iteration all such parity-check equations are found and the erased bits corrected. After these bits have been corrected, any new parity-check equations checking on only one erased bit are then corrected in the subsequent iteration. The process is repeated until all the erasures are corrected or all the remaining uncorrected parity-check equations check on two or more erased bits. The question for coding theorists is when will this occur. For the binary erasure channel at least, the answer is known. The message passing decoder will fail to converge if the erased bits include a set of code bits $\mathcal{S}$ that is a stopping set. A stopping set $\mathcal{S}$ is a set of code bits with the property that every parity-check equation that checks on a bit in S in fact checks on at least two bits in S. The size of a stopping set is the number of bits it includes, and the minimum stopping set size of a parity-check matrix is denoted by $\mathcal{S}_{min}$. Recall that a set S containing the non-zero bits for a codeword of the code will have the property that every parity-check equation connected to a bit in S is also connected to an even number of bits in $\mathcal{S}$. So any set of non-zero codeword bits is also a stopping set. However, not all stopping sets are code words since the stopping sets may have parity-check nodes that connect to an odd number of bits in S.

CHAPTER 3

DIFFERENT LDPC DECODING ALGORITHMS

3.1 SUMPRODUCT DECODING

The main decoding for LDPC codes is the Sum product decoding in log domain algorithm but this requires complex computation of check nodes and hence it is difficult in hardware implementation. In the Sum product decoding in log domain algorithm, the decoding performance is sacrificed by trading off with computational complexity. In order to obtain better decoding performance, algorithms are devised that modifies Sum product decoding in log domain algorithm. It introduces a unique modified factor. Sum product decoding in log domain algorithm is a soft decision approach in decoding the codeword. This, in turn, reduced the probability of sign change in the decoded vector from which the codeword is obtained more efficiently.

Some insight can be gained into the soft, iterative decoding algorithm we will eventually develop by considering a preliminary, iterative, hard decision decoder.

3.1.1 HARD DECODER

For each bit c_n, compute the checks for those checks that are influenced by c_n. If the number of nonzero checks exceeds some threshold (say, the majority of the checks are nonzero), then the bit is determined to be incorrect. The erroneous bit is flipped and correction continues. This simple scheme is capable of correcting more than one error, as we now explain. Suppose that c_n is in error and that other bits influencing its checks are also in error. Arrange the Tanner graph with c_n as a root (neglecting for now the possibility of cycles in the graph). In Figure 3.1, suppose the bits in the shaded boxes are in error. The bits that connect to the checks connected to the root node are said to be in tier 1. The bits that connect to the checks from the first tier are said to be in tier 2. Many such tiers could be established. Then, decode by proceeding from the "leaves" of the tree (the top of the figure). By the time decoding on c_n is reached, other erroneous bits may have been corrected. Thus bits and checks which are not directly connected to c_n can still influence c_n .

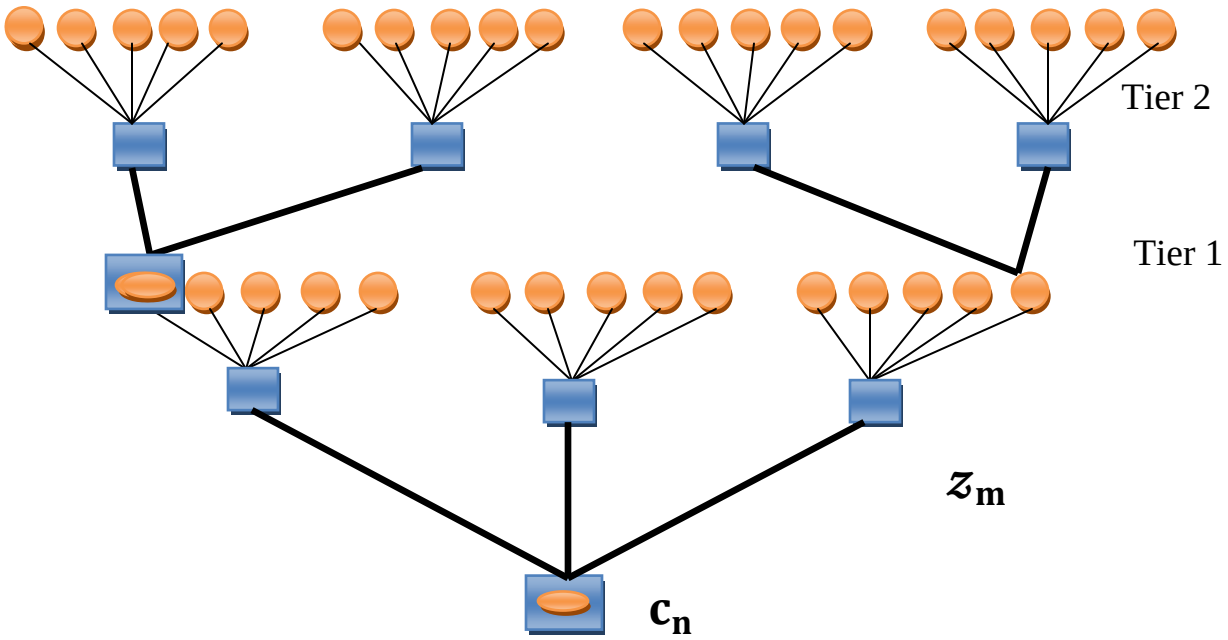


Figure 3.1: A parity check tree associated with the Tanner graph

3.1.2 THE SOFT DECODER

In the soft decoder, rather than flipping bits (a hard operation), we propagate probabilities through the Tanner graph, thereby accumulating evidence that the checks provide about the bits. The optimal (minimum probability of decoding error) decode seeks a codeword $\hat{c}$ which maximizes P(c|r, Ac = 0) , that is, the most probable vector which satisfies the parity checks, given set of received data r = [r 1 ,r 2, . . . , r_N] However, the decoding complexity for the true optimum decoding of an unstructured (i.e., random) code is exponential in K , requiring an exhaustive search over all 2^k codewords. Instead, the decoder attempts to find a codeword having bits c_n which maximize

$$P(c_n|r, \text{all checks involving bit } c_n \text{ are satisfied}),$$

that is, the posterior probability for a single bit given that only the checks on that bit are satisfied. However, the decoding algorithm has excellent demonstrated performance and the complexity of the decoding is linear in the code length.

The decoding algorithm deals with two sets of probabilities. The first set of probabilities is related to the decoding criterion, $P(c_n|r$, all checks involving bit c_n are satisfied). We denote this by

$$q_n(x) = P(c_n = x|r, \text{all checks involving } c_n \text{ are satisfied}),\ x \in \{0, 1\}$$

Or,

$$q_n(x) = P(c_n = x|r, \{z_m, = 0, m \in \mathcal{M}_n\}),\ x \in (0, 1)$$

This probability is referred to as the pseudo posterior probability and is ultimately used to make the decisions about the decoded bits. A variant of this probability, called $q_{mn}(x)$, is also used, which is

$$q_{mn}(x) = P(c_n = x|r, \text{all checks, except } z_m, \text{ involving } c_n \text{are satisfied})$$

Or more briefly,

$$q_{mn}(x) = P(c_n = x|r, \{z_{m'} = 0, m' \in \mathcal{M}_{n,m}\})$$

The second set of probabilities has to do with the probability of checks given the bits. These indicate the probability that a check is satisfied, given the value of a single bit involved with that check and the observations associated with that check. This probability is denoted by

$$r_{mn}(x), \text{ with } r_{mn}(x) = P(z_m = 0|c_n = x, r).$$

The quantities $q_{mn}(x)$ and $r_{mn}(x)$ are computed only for those elements A_{mn} of A that are nonzero. The decoding algorithm incorporates information from the measured data to compute probabilities about the checks, as represented by $r_{mn}(x)$. The information about the checks is then used to find information about the bits, as represented by $q_{mn}(x)$, This, in turn, is used to update the probabilities about the checks, and so forth. This amounts to propagating through the "tree" derived from the Tanner graph. Iteration between bit and check probabilities (qs and rs) proceeds until all the parity checks are satisfied simultaneously, or until a specified number of iterations is exceeded.

3.1.3 THE VERTICAL STEP: UPDATING $q_{mn}(X)$

Consider Figure 3.2(a), which is obtained by selecting an arbitrary bit node c_n from the tanner graph and using it as the root of a tree, with the subset of the Tanner graph connecting this bit to its checks and the other bits involved in these checks as nodes in the tree. The bits other than c_n which connect to the parity checks are referred to as bits in tier 1 of the tree. We shall assume that the bits represented in the first tier of the tree are distinct, and hence independent.
In reality, the portion of the redrawn Tanner graph may not be a tree, since the bits on the first tier may not be distinct. For example, Figure 3.2(b) shows a portion of the actual Tanner graph.

Such a cycle means that the bits in the first tier are not independent (as ideally assumed). However, for a sufficiently large code, the probability of such cycles is small (at least for trees represented out to the first tier). We therefore assume a tree structure as portrayed in Figure 3.2(a), with its corresponding independence assumption. Under the assumption of independence of bits in the first tier, the checks in the tree are statistically independent, given c_n. The decoding algorithm uses information that the checks provide about the bits, as indicated by the following

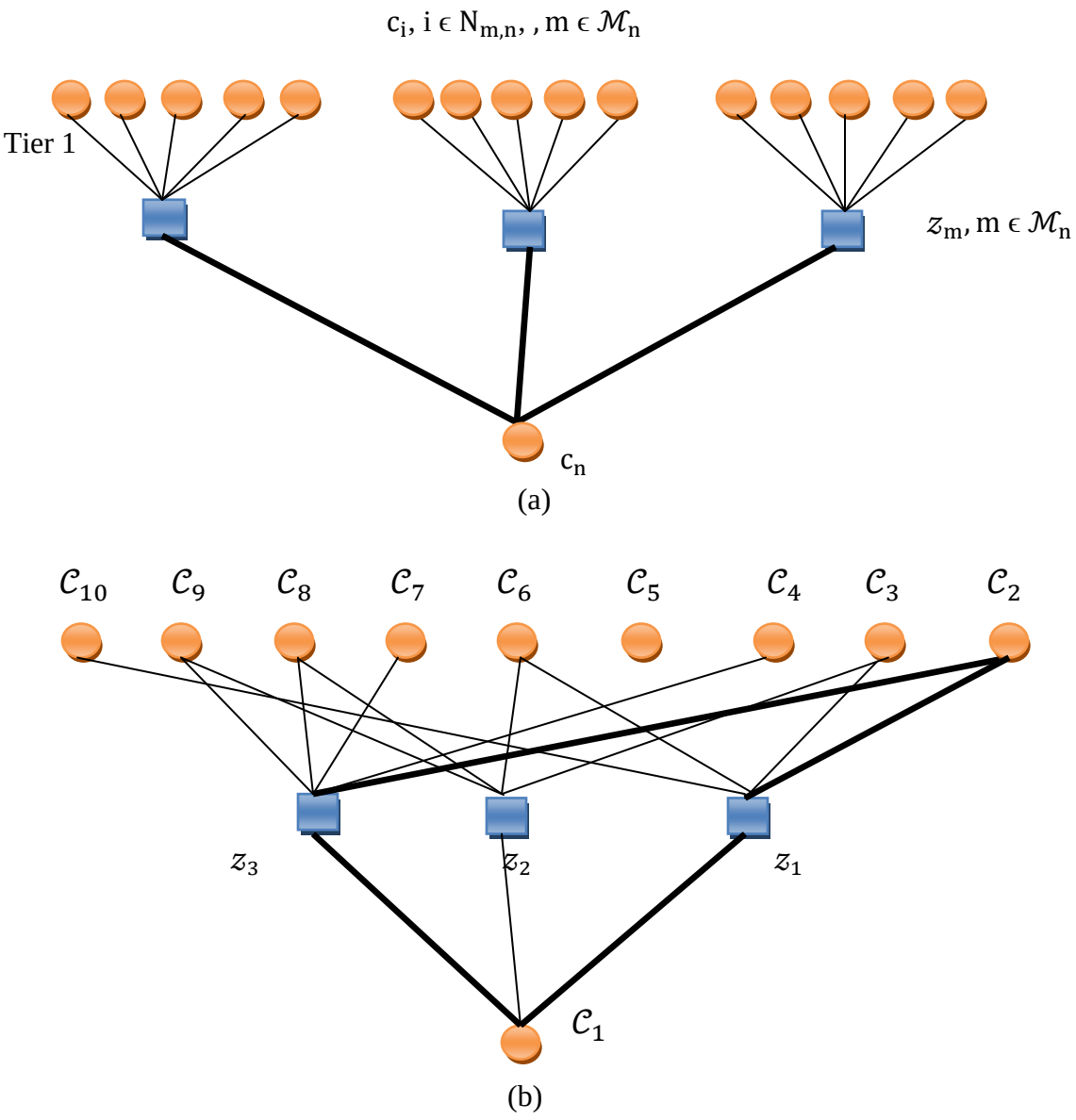


Figure 3.2: A subset of the Tanner graph. (a) Viewed as a tree, with node for c_n as the root. (b) An actual portion of the Tanner graph, with node for c_n as root, showing a cycle.

For a bit c_n involved in parity checks $\{ z_m, m \in \mathcal{M}_n \}$,if the checks are independent then

$$q_n(x) = \alpha P(c_n = x|r_n) \prod_{m \in \mathcal{M}_n} P(z_m = 0|c_n = x, r),$$

where α is a normalizing constant.

$$q_n(x) = P(c_n = 0|r, \{z_m=0, m \in \mathcal{M}_n\}) = \frac{P(c_n = 0, \{z_m=0, m \in \mathcal{M}_n\}|r)}{P\{z_m=0, m \in \mathcal{M}_n\}|r)}$$

$$= \frac{1}{P\{z_m=0, m \in \mathcal{M}_n\}|r)} P(c_n = x|r) P(\{z_m = 0, m \in \mathcal{M}_n\}|c_n = x, r)$$

Due to independence of bits and noise, the conditional probability $P(c_n = x|r)$ can be written as $P(c_n = x|r_n)$.Under the assumption that the checks are independent, the joint probability on the checks can be factored, so that

$$q_n(x) = \frac{1}{P\{z_m=0, m \in \mathcal{M}_n\}|r)} P(c_n = x|r) \prod_{m \in \mathcal{M}_n} P(z_m = 0|c_n = x, r)$$

The factor dividing this probability can be obtained by marginalizing:

$$q_n(x) = \frac{P(c_n = x|r_n) \prod_{m \in \mathcal{M}_n} P(z_m = 0|c_n = x, r)}{\sum_{x'} P(c_n = x'|r_n) \prod_{m \in \mathcal{M}_n} P(z_m = 0|c_n = x', r)}$$

That is, the factor is just a normalization constant, which we denote as α, which ensures that $q_n(x)$ is a probability mass function with respect to x. In $q_n(x)$ has two probability factors. The factor) $\prod_{m \in \mathcal{M}_n} P(z_m = 0|c_n = x, r)$ has been called the extrinsic probability. The other factor in (3.1), $P(c_n|r_n)$, expresses how much information there is about $c_n$based on the measured channel output r_n corresponding to c_n; it has been called the intrinsic probability. As before, let

$$r_{mn}(x) = P(z_m = 0|c_n = x, r)$$

Denote the probability that the mth check is satisfied, given bit c_n. We can write,

$$q_n(x) = \alpha P(c_n = x|r) \prod_{m \in \mathcal{M}_n} r_{mn}(x)$$

Each bit in tier 1 of the tree has its own set of checks, each with their own corresponding checked bits. This leads to a situation as in Figure 3.3. To do decoding on the tree, we again invoke an independence assumption: the set of bits connected to a check subtree rooted at a bit in the first tier are independent. (As before, cycles in the Tanner graph violate this assumption.)

The probability of a bit in the first tier of the tree is computed using bits from the second tier of the tree. Let n′ be the index of a bit in the first tier connected to the check z_m. Let

$$q_{mn'}(x) = P(c_{n'} = x \mid \text{all checks involving } c_{n'}, \text{ except for check } z_m, \text{ are satisfied})$$

Or, more briefly,

$$q_{mn'}(x) = P(c_{n'} = x \mid \{z_{n'} = 0, m' \epsilon \mathcal{M}_{n'm}\}, r)$$

Then, slightly modifying the results,

$$Q_{mn'}(x) = \alpha P(c_{n'} = x \mid r_{n'}) \prod_{m' \epsilon \mathcal{M}_{n'm}} r_{m'n'}(x) \qquad (3.1)$$

If there are w_c parity checks associated with each bit, then the computation (3.1) involves $w_c - 1$ checks. Using (3.1) the probabilities for bits in the first tier can be computed from the checks in the second tier, following which the probability at the root c_n can be computed.

Since the product in (3.1) is computed down the columns of the A matrix (across the checks), updating $q_{mn}(x)$ is called the vertical step of the decoding algorithm. The process can be described in words as follows: For each nonzero position (m,n) of A, compute the product of $r_{m'n'}(x)$ own the nth column of A, excluding the value at position (m,n) ,then multiply by the channel posterior probability. There are $w_c N$ values of q_{mn} to update, each requiring $O(w_c)$ operations, so this step has O (N) complexity. If the graph associated with the code were actually a tree with independence among the bits associated with the checks on each tier, this procedure could be recursively applied,

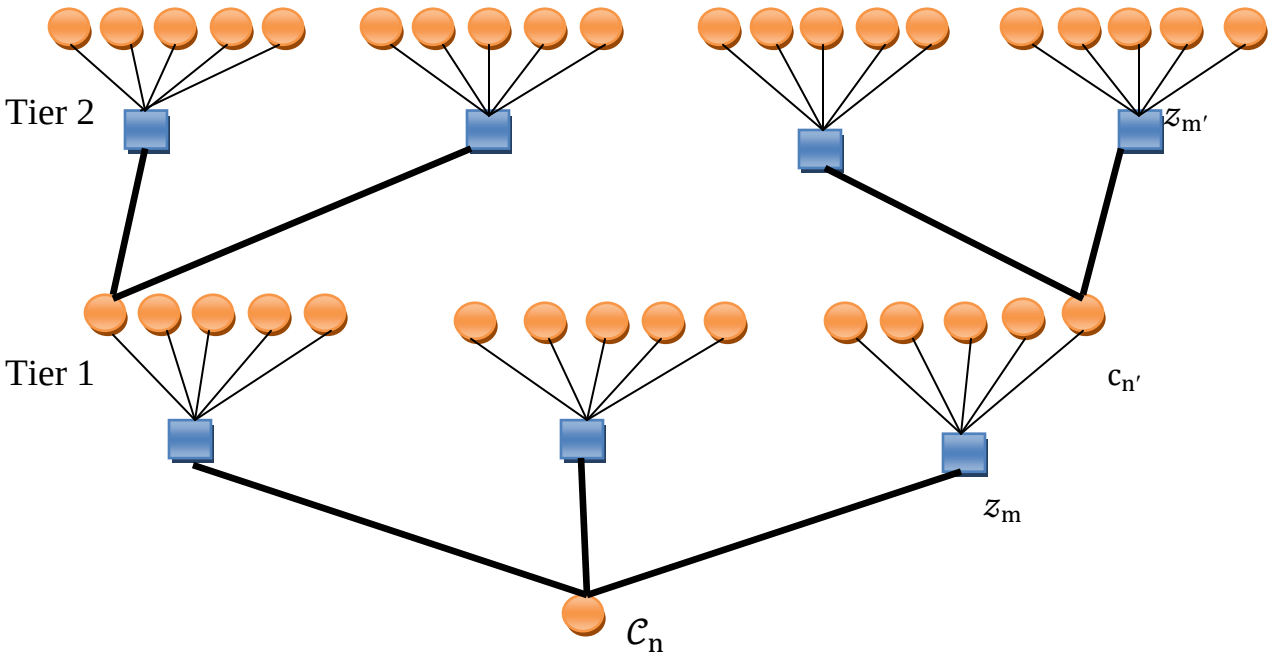


Fig 3.3: The two-tier tree.

starting at the leaf nodes of the tree -those not connected to further checks - and working toward the root. The probabilities at the leaf nodes could be computed using the channel posterior probabilities $p_n(x)$. Working from leaves toward the root, the probability $q_{mn'}(x)$ would be computed for each node c_n,on the second-to-last tier, using the leaf nodes (the last tier). Then $q_{mn'}$ would be computed for each node on the third-to-last tier, using the probabilities obtained from the second-to-last tier, and so forth until the root node is reached.

3.1.4 THE HORIZONTAL STEP: UPDATING $r_{mn}(X)$

The probability $r_{mn}(x) = P(z_m = 0|c_n = x , r)$, depends on all of the bits $\{c_{n'}, n' \in N_m\}$ that participate in z_m , so an expression involving all of the bits that influence check z_m necessary. The desired probability can be computed by marginalizing,

$$P(z_m = 0|c_n = x , r) = \sum_{\{x_{n'}, n' \in N_{m,n}\}} P(z_m = 0, \{c_{n'} = x_{n'}, n' \in N_{m,n}\}|c_n = x , r)$$

Where the sum is taken over all possible binary sequences $\{x_{n'}\}$ with $n' \in N_{m,n}$

The joint probability can be factored using conditioning as

$$r_{mn}(x) = \sum_{\{x_{n'}, n' \in N_{m,n}\}} [P(z_m = 0| c_n = x, \{c_{n'} = x_{n'}, n' \in N_{m,n}\}, r) \times P(\{c_{n'} = x_{n'}, n' \in N_{m,n}\}|r)]$$

Under the assumption that the bits $\{c_{n'} = x_{n'}, n' \in N_{m,n}\}$ are independent.

$r_{mn}(x)$is the total probability of sequences $\{x_{n'}, n' \in N_{m,n}\}$ of length $|N_{m,n}|$whose sum is equal to x , where each element $x_{n'}$ in each sequence occurs with probability $P(c_{n'}|r)$.

For a sequence of bit values $\{x_{n'}, n' \in N_{m,n}\}$ involved in check m, let

$$\zeta(k) = \sum_{i=1}^{k} xN_{m,n}(i)$$

be the sum of the first k bits of $\{x_{n'}\}$,Then,

$$\zeta(L) = \sum_{i=1}^{L} xN_{m,n}(i) = \sum_{l \in N_{m,n}} x_l{}^N$$

where $L= |N_{m,n}|$

The values of $\zeta(k)$ as a function of k may be represented using the trellis.There are two states in the trellis, 0 and 1, representing the possible values of $\zeta(k)$. Starting from the 0 state at k = 0, $\zeta(1)$ takes on two possible values, depending on the value of $xN_{m,n}(1)$. Then $\zeta(2)$ takes on two possible values, depending on the value of $\zeta(1)$ and $xN_{m,n}(2)$.The final state of the trellis represents $\zeta(L)$.

Let the values $x_{n'}$ occur with probability $P(x_{n'}) = P(c_n|r)$. Using this graph, it may be observed that r_{mn} is the probability that x= $\zeta(L)$, where the probability is computed over every possible

path through the trellis. The problem still looks quite complicated However, due to the structure of the trellis, the probabilities can be recursively computed.
Note that,

$$P(\zeta(1) = x) = P(xN_{m,n}(1) = x) = P(cN_{m,n}(1) = x|r)$$

Knowing $P(\zeta(1)=0)$ and $P(\zeta(1)=1)$, the event that $\zeta(2) = 0$ can occur in two ways: either $\zeta(1)=0$ and $N_{m,n}(2) = 0$, or $\zeta(1) = 1$ and $N_{m,n}(2)=1$. Similarly, the event that $\zeta(2)= 1$ can occur in two ways. Thus

$$P(\zeta(2)=0) = P(\zeta(1)= 0), xN_{m,n(2)}= 0) + P(\zeta(1)= 1, xN_{m,n(2)} = 1)$$

$$P(\zeta(2)=1) = P(\zeta(1)= 1), xN_{m,n(2)}= 0) + P(\zeta(1)= 0, xN_{m,n(2)} = 1)$$

By the (assumed) independence of the bits, the joint probabilities can be factored,
So it can be written

$$P(\zeta(2)=0) = P(\zeta(1)= 0), P(xN_{m,n(2)}= 0) + P(\zeta(1)= 1, P(xN_{m,n(2)} = 1)$$

$$P(\zeta(2)=1) = P(\zeta(1)= 1), P(xN_{m,n(2)}= 0) + P(\zeta(1)= 0, P(xN_{m,n(2)} = 1)$$

Let $\omega_k(x)$ be the probability

$$\omega_k(x) = P(\zeta(k) = x).$$

And in general

$$w_k(0)= w_{k-1}(0)\, P(xN_{m,n(k)}=0)+ w_{k-1}(1)\, P(xN_{m,n(k)} = 1)$$

$$w_k(1)= w_{k-1}(1)\, P(xN_{m,n(k)}=0)+ w_{k-1}(0)\, P(xN_{m,n(k)} = 1)$$

The recursion is initialized with $w_0(0)= 1$, $w_0(1)= 0$.

By the recursive computation the probabilities of all possible paths through the trellis are computed. By the definition of $w_k(x)$, $P(\zeta(L) = x) = w_L(x)$,

$$r_{mn}(0) = w_L(0) \qquad r_{mn}(1) = w_L(1)$$

Now consider computing $P(z_m = 0|c_n = x , r)$ for a check node z_m in the first tier of a multi-tier tree. The recursive update rule is written

$$w_k(0)= w_{k-1}(0)\, q_m N_{m,n(k)}=0)+ w_{k-1}(1) q_m N_{m,n(k)} = 1)$$

$$w_k(1)= w_{k-1}(1)\, q_m N_{m,n(k)}=0)+ w_{k-1}(0) q_m N_{m,n(k)} = 1)$$

This computation is used for each iteration of the decoding algorithm.

The probability update algorithm can be extended to codes with larger than binary alphabets by creating a trellis with more states. The LDPC decoding algorithm is thus applicable to linear codes over arbitrary fields. For binary codes, however, it can be re-expressed in an especially elegant way in terms of differences of probabilities. Let

$$\delta q_{ml} = q_{ml}(0) - q_{ml}(1) \text{ and } \delta r_{ml} = r_{ml}(0) - r_{ml}(1)$$

So inductively,

$$\omega_k(0) - \omega_k(1) = \prod_{i=1}^{k} (q_m N_{m,n(i)}(0) - q_m N_{m,n(i)}(1))$$

Using $\delta r_{mn} = r_{mn}(0) - r_{mn}(1)$, we have

$$\delta r_{mn} = \prod_{l' \epsilon N_{m,n}} \delta q_{m,l'}$$

In words what this update says is: For each nonzero element (m, n) of A , compute the product of the δq_{mn} across the mth row, except for the value at column n. This step is therefore called the horizontal step. The entire update has complexity O (N) . Having found the δr_{mn} and using the fact that $r_{mn}(0) + r_{mn}(1) = 1$, the probabilities can be computed as

$$r_{mn}(0) = (1 + \delta r_{mn})/2 \qquad r_{mn}(1) = (1 - \delta r_{mn})/2$$

3.1.5 TERMINATING AND INITIALIZING THE DECODING ALGORITHM

As before, let $q_n(x) = P(c_n = x|r, \{z_m, = 0, m \epsilon \mathcal{M}_n\})$.This can be computed as

$$q_n(x) = \alpha_n P_n (x) \prod_{m \epsilon \mathcal{M}_n}^{N} r_{mn}(x)$$

Where α_n is chosen so that $q_n (0) + q_n (1) = 1$. These pseudoposterior probabilities are used to make decisions on x : if $q_n (1) > 0.5$ a decision is made to set $\hat{c_n} = 1$. Since the decoding criterion computes $P(c_n = x|r,c$ hecks involving $c_n)$, with each bit probability computed separately, it is possible that the set of bit decisions obtained by the decoding algorithm do not initially simultaneously satisfy all of the checks. This observation can be used to formulate a stopping criterion. If $A\hat{c} = 0$, that is, all checks are simultaneously satisfied, then decoding is finished. Otherwise, the algorithm repeats from the horizontal step.

It may happen that $A\hat{c} = 0$ is not satisfied after the specified maximum number of iterations. In this case, a decoding failure is declared; this is indicative of an error event which exceeds the

ability of the code to correct within that number of iterations. Knowledge of a decoding failure is important, but not available with many codes, including turbo codes. In some systems, a decoding failure may invoke a retransmission of the faulty codeword. The iterative decoding algorithm is initialized by setting $q_{mn}(x) = p_n(x)$. That is, the probability conditional on the checks $q_{mn}(x)$ is set to the channel posterior probability, which is what would be used if the Tanner graph were actually a tree.

3.2 BIT FLIPPING ALGORITHM

Assume that we have a binary linear code defined using a specific Code graph (or Tanner graph). A Code graph is a bipartite graph with the variable nodes on one side and check nodes on the other side. As we saw in the previous lectures, each check node imposes a parity constraint on the set of variable nodes connected to it. In the previous lecture, we defined a decoding algorithm which we claimed has a guarantee of success for all Code graphs which are expander and have fixed variable node degrees. Before making these definitions more precise, we review the definition of an expander graph

Definition 1 A factor graph is a (ϵ, ∂)-expander if for every subset U of the set of variable nodes of size bounded by $|U| \leq \epsilon N$, it holds that $|\partial U| > \partial |U|$, where @U is the set of factor nodes adjacent to at least one node in U.

Assume that we have transmitted some codeword W = (W1, ...,Wm) and the decoder has received the output sequence Y = (Y1, ..., Ym) over a BSC channel with parameter p (meaning $P(Yi = Xi) = 1 - p$).

The bit flipping algorithm starts out with sequence Y = (Y1, ..., Ym). Each check node introduces a constraint. As we saw in the previous lecture, in each step, the algorithm tries to find a bit, flipping of which reduces the number of unsatisfied clauses. The algorithm stops when for every bit, the number of unsatisfied constraints is less or equal to the number of satisfied constraints. Since the number of unsatisfied constraints reduces in each step, the algorithm has to stop. We will prove the following theorem:

Theorem 1.Consider a Code graph where all variable nodes have degree l, and the graph is (ϵ, ∂)-expander.

Then the bit- flipping algorithm will correct any pattern of at most $N\epsilon/2$ errors.

Proof: Without loss of generality we can assume the all zero codeword has been sent (since the code is linear). Thus the hamming weight of the received codeword equals to the 15-1 number of bits we have received in error. Let

• X(t) denote the string at the tth iteration of the algorithm. Since we start out with the received message, we have X(0) = Y .

• W(t) denote the hamming weight of X(t).

• u(t) denote the number of unsatisfied check nodes.

• s(t) denote the number of satisfied checks that are adjacent to at least one Xi such that Xi = 1.

Since the number of errors is at most Nϵ/2, we have $w(0) \leq N\epsilon/2$.

It is clear from the bit-flipping algorithm that u(t) is a decreasing function of t. The following two propositions thus imply the result stated at Theorem 1.

Proposition 1. Let t_ be the iteration in which the algorithm stops, then w(t*) is either zero, or is greater than or equal to Nϵ

Proof: Assume that 0 < w(t*) < Nϵ. Consider the nodes which are one. Because of expansion, the number of check nodes adjacent to those nodes that are one is at least 3/4 lω(t*). Let set T denote these set of check nodes.
The check nodes adjacent to variable nodes that are ones are either counted in u(t) or s(t).Therefore $|T| = u(t*) + s(t*) > 3/4 l\omega(t*)$.

On the other hand, since every satisfied check is adjacent to at least two variable nodes with Xi = 1, the number of edges going out from the set T is at least u(t*) + 2s(t*). But since the degree of each variable node is l, the number of edges going out from set T is exactly equal to lω(t*), and therefore we get $u(t*) + 2s(t*) \leq l\omega(t*)$. This inequality together with $u(t*) + s(t*) > 3/4\ l\omega(t*)$, one can easily show, imply that u(t*) > 1/2 lω(t*).

But $u(t*) > 1/2\ l\omega(t*)$ implies that there exist at least one variable node with Xi = 1 that is adjacent to more than l/2 check nodes. This is a contradiction since this variable node is adjacent to more unsatisfied nodes than satisfied nodes and thus the algorithm couldn't have stopped at iteration t*

Proposition 2. The algorithm cannot terminate with a string of weight greater than Nϵ.

Proof: It is enough to prove that there is no t such that w(t) = Nϵ. Assume otherwise that w(t) = Nϵ. The above inequalities show that u(t) $> l/2w(t) = l/2N\epsilon$. On the other hand, u(t) ≤ u(0) = l/2Nϵ which is a contradiction.

3.2.1 BIT FLIP DECODING ALGORITHM AND ITS VARIANTS

The BF algorithm is very simple but has a poor performance. Therefore several variants of the algorithm were proposed to design decoding algorithm with moderate complexity and better performance than the BF algorithm.

Throughout this paper we assume that the signal is bipolar.Let H be a parity check matrix of an LDPC code of size M×N. In this paper only regular LDPC codes are used. Let j be the column weight. Let y = y1y2 ….vN be a received signal. Then a received word z = z1z2 ….zN is defined by

$$zi = \{{}^{1}_{0}\,; \text{if } yi > 0\} \qquad \text{for i} = 1;\ 2;\vdots;\ \text{N.}$$

3.2.2 BIT FLIP DECODING ALGORITHM

the following algorithm is called a maximum detection bit flipping algorithm is given as follows. It should be noticed that only integer operations are required and therefore the algorithm may be efficiently implemented by an electronic circuit.

The following variables are used: l : a count for iteration, l max : the maximum number of iterations, B : a threshold for the termination condition of the algorithm, v : a temporal code word, v0 : a code word for the next iteration. A symbol ‘v’ is used to mean a substitution from the right-hand side to the left-hand side.

Step0: l 0; v z.

Step1: Calculate syndrome s = s1; s2; : : : ; sM vHT ; If s = 0, then the algorithm terminates.

Step2: Calculate en by,

$$en \leftarrow \sum sm\ (m2M(n))$$

Step3: If maxj ej < B then the algorithm terminates where B is usually set to dj=2e.

Step4: Flip all bits vi with ei = maxj ej and l ← l + 1: Let v0 be a vector obtained by bit-flipping.

Step5: if l > lmax, then the algorithm terminates.

Step6: v ← v0 and go to Step 1.

The number en calculated in Step 2 is the number of unsatisfied check equations including n-th bit. Therefore if en is large, then the n-th bit may be incorrect with high Probability. At Step 4 we can employ a different criterion for finding bits to be flipped. For example, the following step can be possible.

Step4': Flip all bits vi with ei ≥ ∂ and l l + 1, where _ is a predefined integer The Step 3 is well defined only for regular LDPC codes. If H is not regular, B may depend on column. If the number of error bits in a received code word is small enough, then the errors bits may disappear as the algorithm proceeds.

This is the bit-flip algorithm for LDPC decoding. This decoding process is more efficient and less erroneous most other decoding algorithms. Later other decoding algorithms were introduced. Sum product decoding algorithm is one of them. Sum product decoding algorithm is even more efficient than BF algorithm. Now a days the sum-product decoding algorithm is mostly used.

3.3 MIN-SUM DECODING ALGORITHM

The Min-Sum algorithm is similar to the Sum-Product algorithm, with an approximation of check node process. It has some advantages in implementation against the Sum- Product algorithm, such as less computation and estimation of noise power is unnecessary for an AWGN channel. In Min-Sum algorithm can approach consists of the following main steps.

Step 0: The initial messages at variable nodes are set to:

$$L(q) = L(c) = y$$

Step 1: Check node update (CNU):

$$L(r) = (\prod \text{sign}(L(q)) \min |L(q)|$$

Step 2: Variable node update (VNU):

$$L(q) = L(c) + \sum L(r)$$

Step 3: Soft Decision:

$$L(Q) = L(c) + \sum L(r)$$

Step 4: Hard Decision:

$$C = \{{}_{1}^{0}L(Q < 0)\}\,;\ \text{otherwise}$$

Step 5: Parity Check:

$$H(c1, c2, c3 \ldots\ldots\ldots cn)\text{^}T = 0$$

MMS: a modified version of min-sum algorithm gives better performance compared to MS. The scaling factor (γ) is applied to eq. (3) above [4]. The MMS is then written as

$$L(q) = (L(c) + \sum L(r)) * \lambda$$

In most practical case, γ= 0.6-0.8 is used. For the sake of simplicity in hardware design we use γ= 0.75 in this report.

3.3.1 Improved Layered Min-Sum Decoding Algorithm

Similar to the standard layered decoding algorithm, the proposed decoding method can be summarized in three steps:

1. Initialization: A-posteriori information λ_n is initialized with the channel LLR λ_{ch}, and Each Λ_{nm} is initialized to zero. Specifically for BPSK signals transmitted in the AWGN channel,

$$\lambda_{ch} = 2y_n / \sigma 2$$

y_n being the received BPSK signal value, and σ^2 being the noise variance.

2. Iterative decoding: During the kth iteration, we observed that:

$$\lambda_{nm} = \lambda_k - \Lambda_k - 1$$

As a result of, messages from variable node to check-node could be generated using the S_O and the memory for λ_{nm} is saved. When λ_k is extracted, update messages from variable node n to check-node m using Eqn.(3).

3. Check stop rules: compute the hard decoding output for current iteration:

$$C = \{{}^{0}_{1} \text{if } \lambda \geq 0\}$$

If constraint function H · ˆ C = 0 is satisfied or the maximum number of iteration has performed, stop the iteration and output decoded results. Otherwise, continue the iteration. Noted that in Eqn.(7) of the proposed algorithm the variable-node message update function has been revised by assigning more weight to current LLR . We get that,

$$\lambda_n = \lambda_{nm} + \Lambda_{nm} + \omega * (\lambda_n - \Lambda_{nm})$$

Compared with the standard layered decoding, a new term ($\Lambda_{nm} - \Lambda_{n(k-1)}$)is added. As the check-node to variable-node message becomes more reliable as theiteration proceeds, the term indicates the convergence direction for variable-node n. Therefore, adding such a term to the variable-node message update function will accelerate the convergence speed.

3.4 Log-Likelihood Algorithm

We could directly maximize $l(\theta) = \sum_z \log p(x, z|\theta)$ using a gradient method (e.g., gradient ascent, conjugate gradient, quasi-Newton) but sometimes the gradient is hard to compute, hard to implement, or we do not want to bother adding in a black-box optimization routine.

Consider the following inequality,

$$l(\theta) = \log p(x|\theta) = \log \sum z \; p(x, z|\theta)$$

$$\log \sum z \; q(z|x, \theta)\, p(x, z|\theta) / \, q(z|x, \theta)$$

$$\leq \sum z \; q(z|x, \theta) \log(p(x, z|\theta) / \, q(z|x, \theta)) = F(q, \theta)$$

Where $q(z|x, \theta)$ is an arbitrary density over Z. This inequality is foundational to what are called "variation methods" in the machine learning literature2. Instead of maximizing $l(\theta)$ directly, EM maximizes the lower-bound F(q, θ) via coordinate ascent:

E-step: $q(t+1) = \arg\max F(q, \theta(t))$

M-step: $\theta(t+1) = \arg\max F(q(t+1), \theta)$

Starting with some initial value of the parameters $\theta(0)$, one cycles between the E and M-steps until $\theta(t)$ converges to a local maxima. Computing equation 4 directly involves fixing $\theta = \theta(t)$ and optimization over the space of distributions, which looks painful. However, it is possible that $q(t+1) = p(z|x, \theta(t))$. We can stop worrying about q as a variable over the space of distributions, since we know the optimal q is a distribution that depends on $\theta(t)$. To compute equation 5 we filed to show q and note that,

$$l(\theta) \geq F(q, \theta)$$

$$= \sum z \; q(z|x, \theta) \log(p(x, z|\theta) / q(z|x, \theta))$$

$$= \sum z \; q(z|x, \theta) \log(p(x, z|\theta) - \sum z \; q(z|x, \theta) \log q(z|x, \theta)$$

$$= Q(\theta|\theta(t)) + H(q)$$

so maximizing F(q, θ) is equivalent to maximizing the expected complete log-likelihood. Obscuring these details, which explain what EM is doing, we can re-express equations 4 and 5 as,

E-step : Compute $Q(\theta|\theta(t)) = Ep(z|x, \theta(t))[\log p(x, z|\theta)]$

M-step : $\theta(t+1) = \arg\max Ep(z|x, \theta(t))[\log p(x, z|\theta)]$

Sometimes the M-step is a constrained maximization, which means that there are constraints on valid solutions not encoded in the function itself.

CHAPTER 4

PROPOSED ALGORITHM

4.1 INTRODUCTION

Density evolution is an algorithm for computing the capacity of low-density parity-check (LDPC) codes under message-passing decoding. For memoryless binary-input continuous-output additive white Gaussian noise (AWGN) channels and sum-product decoders, we use a Gaussian approximation for message densities under density evolution to simplify the analysis of the decoding algorithm. We convert the infinite-dimensional problem of iteratively calculating message densities, which is needed to find the exact threshold, to a one-dimensional problem of updating the means of the Gaussian densities. This simplification not only allows us to calculate the threshold quickly and to understand the behavior of the decoder better, but also makes it easier to design good irregular LDPC codes for AWGN channels. For various regular LDPC codes we have examined, thresholds can be estimated within 0.1 dB of the exact value. For rates between 0.5 and 0.9, codes designed using the Gaussian approximation perform within 0.02 dB of the best performing codes found so far by using density evolution when the maximum variable degree is 10. We show that by using the Gaussian approximation, we can visualize the sum-product decoding algorithm. We also show that the optimization of degree distributions can be understood and done graphically using the visualization

The sum–product algorithm is a soft decision message passing algorithm. It is similar to the bit-flipping algorithm ,but the messages representing each decision (whether the bit value is 1 or 0) are now probabilities. Whereas bit-flipping decoding accepts an initial hard decision on the received bits as input, the sum–product algorithm is a soft decision algorithm that accepts the probability for each received bit as input. The input channel or received bit probabilities are also called the priori probabilities for the received bits, because they were known in advance before the LDPC decoder was operated. For the sum–product decoder, the extrinsic information passed between nodes is also given as probabilities rather than hard decisions.

There are many variants and applications of the sum-product algorithm. The most straight-forward application is to a posteriori probability (APP) decoding. When applied to a trellis, it becomes the celebrated BCJR decoding algorithm. In the field of statistical inference, it becomes the even more widely known “belief propagation” (BP) algorithm. For Gaussian state-space models, it becomes the Kalman smoother.

There is also a "min-sum" or maximum-likelihood sequence detection (MLSD) version of the sum-product algorithm. When applied to a trellis, the min-sum algorithm gives the same result as the Viterbi algorithm.

4.2 SUM PRODUCT ALGORITHM

let,

$$\lambda(c_n|\mathrm{r}) = \log\frac{p(c_n=\mathrm{r}\,|\,1)}{p(c_n=\mathrm{r}\,|\,0)}$$

$$= \log\frac{p\{c_n = \,|\,1 r_n, (r_i, \mathrm{i} \neq 1)\}}{p\{c_n = 0\,|\,r_n, (r_i, \mathrm{i} \neq 1)\}}$$

By application of Bayes' rule, the numerator can be expressed as

$$p\{c_n = 1|r_n, (r_i, \mathrm{i} \neq 1)\} = \frac{p(r_n, c_n = 1, \{r_i, \mathrm{i} \neq 1\})}{p(r_n, \{r_i, \mathrm{i} \neq 1\})}$$

$$= \frac{p(r_n\,|\,c_n = 1, \{r_i, \mathrm{i} \neq 1\})p(c_n = 1, \{r_i, \mathrm{i} \neq 1\})}{p(r_n\,|\,\{r_i, \mathrm{i} \neq 1\})p\{r_i, \mathrm{i} \neq 1\}}$$

$$= \frac{p(r_n\,|\,c_n = 1)p(c_n = 1, \{r_i, \mathrm{i} \neq 1\})}{p(r_n\,|\,\{r_i, \mathrm{i} \neq 1\}p(\{r_i, \mathrm{i} \neq 1\})}$$

$$= \frac{p(r_n\,|\,c_n = 1)\mathrm{P}(c_n = \,|\,1\{r_i, \mathrm{i} \neq 1\}}{p(r_n\,|\,\{r_i, \mathrm{i} \neq 1\}}$$

The likelihood ratio can be written as,

$$\lambda(c_n|\mathrm{r}) = \log\frac{p(r_n \mid c_n = 1)}{p(r_n \mid c_n = 0)} + \log\frac{\mathrm{P}(c_n = \mid 1\{r_i, \mathrm{i} \neq 1\})}{\mathrm{P}(c_n = 0 \mid \{r_i, \mathrm{i} \neq 1\})}$$

For Gaussian channel we have seen that,

$$\frac{\log\left(p(r_n|c_n = 1)\right)}{p(r_n|c_n = 0)} = L_c r_n$$

where, $L_c = 2\sqrt{\frac{E_c}{\sigma^2}}$.the channel reliability.The terms in the sum can be identified as,

$$\lambda(c_n|\mathrm{r}) = L_c r_n + \log\frac{\mathrm{P}(c_n = 1, \{r_i, \mathrm{i} \neq 1\})}{\mathrm{P}(c_n = 0, \{r_i, \mathrm{i} \neq 1\})}$$

where the intrinsic term is determined by the explicit measurement r, affecting the bit c_n and the extrinsic term is determined by the information provided by all the orher observations and the code structure.

Let us now express the probabilities in the extrinsic term in terms of the parity checks. Let $z_{m,n}$denote the parity check computed using the mth check associated with c_n except for c_n. That is,

$$z_{m,n} = \sum_{i \in N_{m,n}} c_i$$

If $c_n = 1$, then$z_{m,n} + c_n = 0$; that is, $z_{m,n} = 1$ for all the checks $m \in M_n$, in which $c_n$participates. Similarly, if $c_n = 0$, then $z_{m,n} = 0$ for all $m \in M_n$.We can write,

$$\lambda(c_n|\mathrm{r}) = L_c r_n + \log\frac{P(z_{m,n} = \text{for1 all m} \in M_n \mid \{r_i, \mathrm{i} \neq 1\})}{P(z_{m,n} = \text{for0 all m} \in M_n \mid \{r_i, \mathrm{i} \neq 1\})}$$

We now invoke the assumption that the graph associated with the code is cycle-free. Then the set of bits associated with $z_{m,n}$ are independent of the bits associated with $z_{m,n}$form $' \neq$ m.We thus have

$$\lambda(c_n|\mathrm{r}) = L_c r_n + \log\frac{\prod_{m'\in M_n} P(z_{m,n} = \,\big|\, 1\{r_i, \mathrm{i} \neq 1\})}{\prod_{m'\in M_n} P(z_{m,n} = 0 \,\big|\, \{r_i, \mathrm{i} \neq 1\})}$$

$$= L_c r_n + \sum_{m'\in M_n} \log\frac{P(z_{m,n} = 1 \,\big|\, \{r_i, \mathrm{i} \neq 1\})}{P(z_{m,n} = 0 \,\big|\, \{r_i, \mathrm{i} \neq 1\})}$$

Let us define the log likelihood ratio,

$$\lambda\,(z_{m,n} \,\big|\, \{r_i, \mathrm{i} \neq \mathrm{n}\}) = \frac{P(z_{m,n} = 1 \,\big|\, \{r_i, \mathrm{i} \neq 1\})}{P(z_{m,n} = 0 \,\big|\, \{r_i, \mathrm{i} \neq 1\})}$$

Then,

$$\lambda(c_n|\mathrm{r}) = L_c r_n + \sum_{m'\in M_n} \lambda\,(z_{m,n} \,\big|\, \{r_i, \mathrm{i} \neq \mathrm{n}\})$$

$$= L_c r_n + \sum_{m'\in M_n} \lambda \sum_{j\in N_{m,n}} c_j \,\big|\, \{r_i, \mathrm{i} \neq \mathrm{n}\}$$

Under the assumption that the checks in $z_{m,n}$ are conditionally independent , we invoke the tanh rule to write,

$$\lambda(c_n|\mathrm{r}) = L_c r_n - 2\sum_{m'\in M_n} tanh^{-1}\left(\prod_{j\in N_{m,n}} \tanh\left(-\frac{\lambda(c_j \,\big|\, \{r_i, \mathrm{i} \neq \mathrm{n}\})}{2}\right)\right)$$

Computation now requires knowing the $\lambda(c_j \,\big|\, \{r_i, \mathrm{i} \neq \mathrm{n}\})$ the conditional likelihoods of the bits which connect to the checks of c_n**.** How are these obtained? They are obtained the same way as $\lambda(c_n)$:that is, we remove fromc_n its distinguished role, and treat all bit nodes alike. However, we must be careful to deal only with the extrinsic information.

Let,

$$\eta_{m,n} = -2tanh^{-1}\left(\prod_{j\in N_{m,n}} \tanh\left(-\frac{\lambda(c_j \mid \{r_i, \mathrm{i} \neq \mathrm{n}\})}{2}\right)\right)$$

This can be thought of as the "message" which is passed from the check node m to the bit node n .Then we can write,

$$\lambda(c_n|\mathrm{r}) = L_c r_n + \sum_{m'\in M_n} \eta_{m,n}$$

This can be thought of as a message that the bit node c_n sends to its check nodes.

4.3 ITERATIVE LOG LIKELIHOOD DECODING ALGORITHM FOR BINARY LDPC CODES

Input: A ,the received vector r, the maximum number of iterations L, and the channel reliability L_c.

Initialization: Set $\eta_{m,n}^{[0]} = 0$ for all (m, n) with A(m, n) = 1.

Set $\lambda_n^0 = L_c r_n$

Set the loop counter l = 1.

After intialization we have to update the check node

Check node update: For each (m, n) with A(m, n) = 1: Compute,

$$\eta_{m,n}^{[l]} = -2tanh^{-1}\left(\prod_{j\in N_{m,n}} \tanh\left(-\frac{\lambda_i^{[l-1]} - \eta_{m,i}^{[l-1]}}{2}\right)\right)$$

Bit node update: For n = 1, 2, . . . , N , Compute

$$\lambda_n^{[l]} = L_c r_n + \sum_{m' \in M_n} \eta_{m,n}^{[l]}$$

Make a tentative decision: Set $\hat{c_n} = 1$ if $\lambda_n [l] > 0$, else set $\hat{c}_n = 0$.

If $A\hat{c_n} = 0$, then Stop. Otherwise, if number ofiterations >L , loop to Check node update

Otherwise, declare a decoding failure and Stop.

if $H.x^T = 0$, x is a vlaid codeword

otherwise , x is not a valid codeword

4.4 MODIFIED SUM PRODUCT LOG LIKELIHOOD DECODING ALGORITHM FOR BINARY LDPC CODES

Our proposed algorithm introduces a factor, k in the check to bit node updating process in the algorithm described in the step check node update. This factor,k lies between zero 0 and 1 i.e. k ∈ [0,1]and is computed as the ratio of the difference of the estimated code word to the sum of the estimated code word in present and preceding iteration and is termed as the modified factor. A value of k=1 signifies that the modified algorithm is equivalent to the sum product log domain algorithm. modified factor, k is introduced to minimize the probability of sign change during check node update in later iterations. The modification step is described as:

Check node update: For each (m, n) with A(m, n) = 1: Compute,

$$\eta_{m,n}^{l} = K \times \left(\prod_{i \in N_{m,n}} \lambda_i^{[l-1]} \right) \times \left(\sum_{i \in N_{m,n}} \eta_i^{[l-1]} \right)$$

K=modified factor

The prime objective of this modification is to attain better performance by combining the effectiveness of the two algorithms by imposing hard decision to the decoded vector, x. The steps are described as follows:

i. At the end of the iteration for sum product log domain algorithm, syndrome vector S is found by multiplying the decoded vector with the transpose of the parity check matrix: $\eta = \lambda \times H^T$. If all the elements of η are zero, the decoding is declared to be a success, otherwise go to step (ii).

ii. For each check equation equal to zero, find the position of the variable nodes in the parity check matrix where it assumes 1 and these nodes are called valid variable nodes. Form a column vector with the positions of the valid variable nodes.

iii. Each bit in the codeword, which does not belong to any position given in the column vector in step (ii), is flipped.

4.5 SIMULATION RESULT

MATLAB simulation results are shown below:

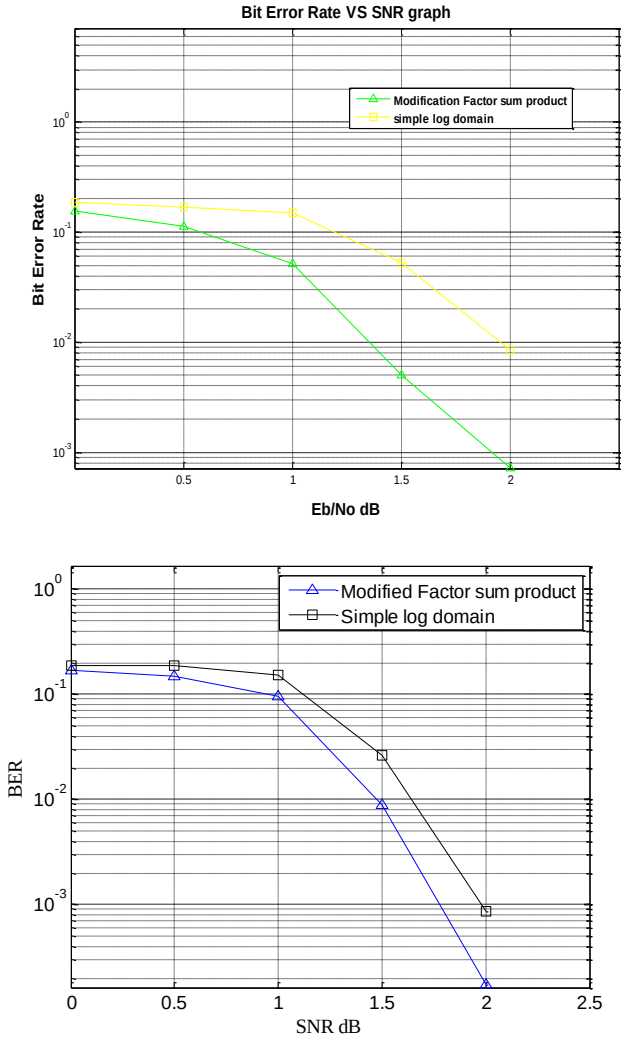


Fig.4.1: BER performance of proposed algorithm with ½ rate (972, 1944) LDPC codes with iteration number = 15 and frame=5

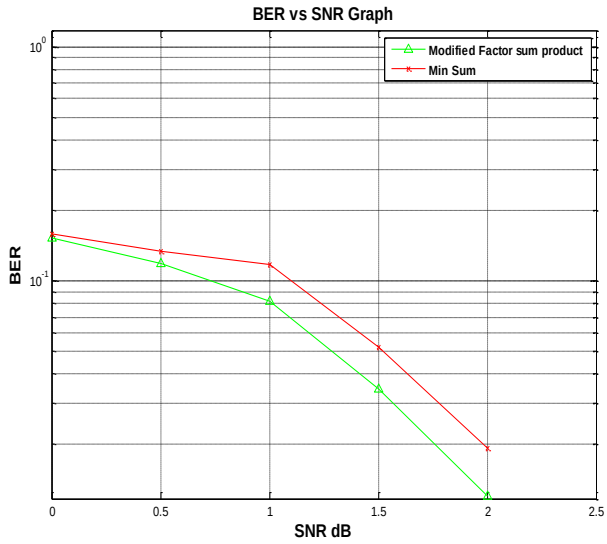


Fig.4.2: BER performance of proposed algorithm with ½ rate (972, 1944) LDPC codes with iteration number = 10 and frame=3

In this section the effectiveness of the proposed algorithms are verified by considering a regular (5, 13) LDPC code. The LDPC code has a codeword length, N = 1944 and information length, K = 972 which corresponds to a code rate of 0.5. Fig 4.1 was obtained using a maximum of 15 iterations. It is observed that modified factor sum product log domain algorithm gives a better performance than simple log domain sum product algorithm for the given SNR range of 1 to 2.5dB. A significant improvement is observed in modified factor sum product log domain above 2 dB. Fig 4.2 was obtained for the modified factor sum product log domain and minsum decoding but computed for a maximum of 10 iterations. This also shows better curve for modified factor sum product log domain than the minsum decoding.

in fig.2 we can see that the modified factor sum product log domain algorithm decreases the bit error rate compared to the simple log domain algorithm to an extent of 10^{-2} to 10^{-3}.also the SNR improved drastically.

In fig.3 we again see that the modified factor sum product log domain algorithm out performs the minsum decoding.

If we compute the complexity of the codes than it reveals that our propsed algorithm the modified factor sum product log domain has the same complexity level as the simple log domain algorithm.On the other hand modified factor sum product log domain algorithm has a bit comlexity than the min sum algorithm.But in the both case we can see that the BER performance of modified factor sum product log domain algorithm outperforms both the simple log domain and min sum algorithm.So we can trade of some of the complexity for better BER performance.

4.6 COMPLEXITY CALCULATION

Algorithm	Addition	Multiplication	Relative Complexity
Sum product log domain simple	478690	6340490	11.2
Sum product log domain proposed	488690	7642180	12
Attenuated Min Sum	457520	6466460	10

Fig 4.3:Comperative Comlexity Calculation of Different Algorithms.

Here we can see that the relative coplexity of our proposed algorithm is only 0.8 and 2 greater than the Sum product log domain simple and Attenuated Min Sum decoding algorithm respectively.So we can say that our algorithm gives better performance with optimum complexity.We have to make trade off between bit error rate and complexity.In that case our proposed algortihm Sum product log domain decoding is the perfect choice for the optimum case.

Next we have plotted the complexity in a graph and the result is found satisfactory enough to choose our algorithm as the optimum one.Hence we have proved here that with little bit of increase in complexity we can achieve better bit error rate as a result better sound to noise ratio(SNR).

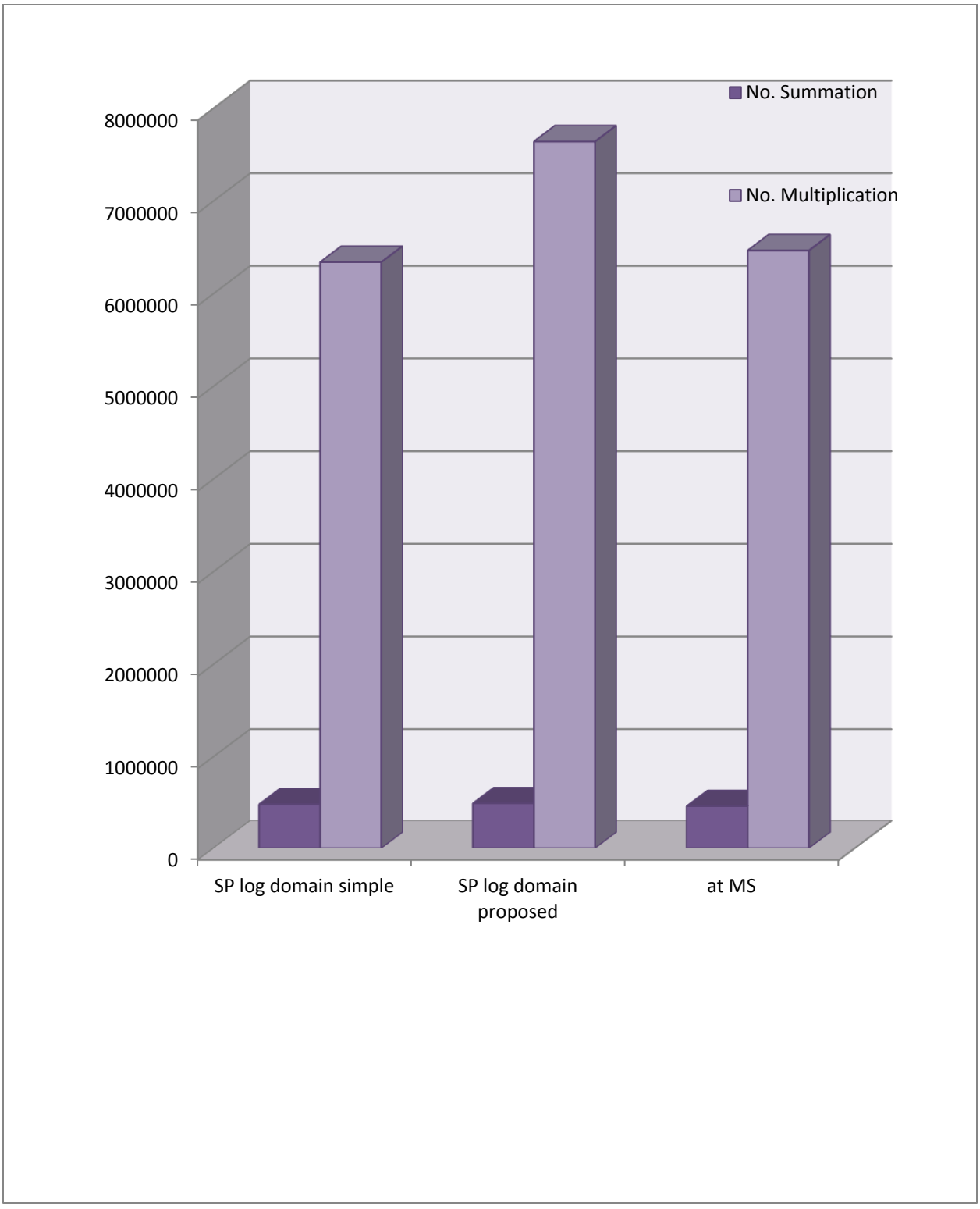


Fig 4.4: Plot of the number of summation and number of multiplication of the different algorithms.

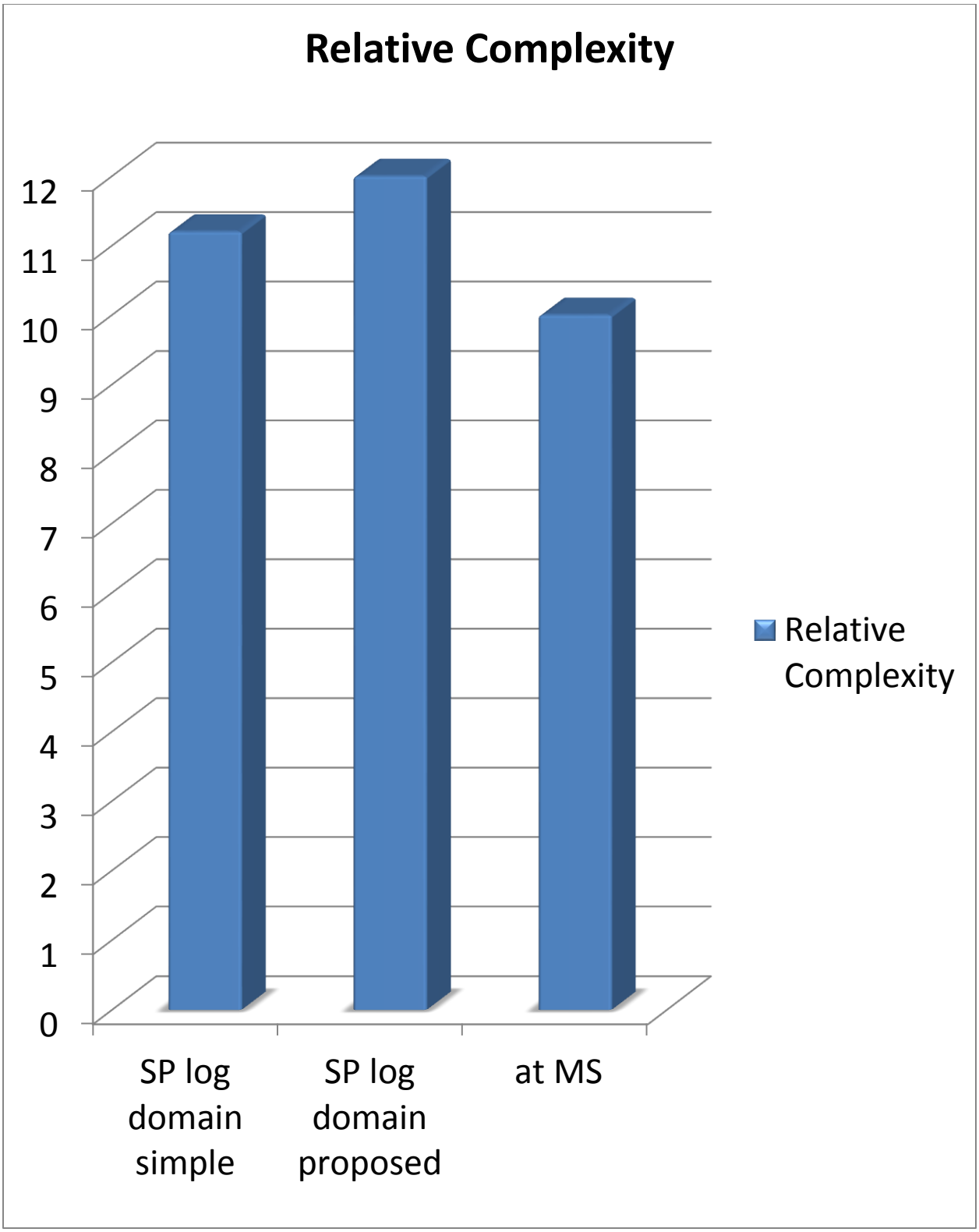


Fig 4.5:Plot of the relative complexity of the different algorithms.

Here during the complexity calculation we have calculated the number of addition and substraction under the heading of number of addition and we have calculated the number of multiplication and divsion under the heading of number of multiplication.Then we have assumed the complexity level of the each addition is equal to 1 and the coplexity level of each multiplication is 2.Then we have sum up the total coplexity of the addition and multiplication.After that we have found the time which is needed to run the simulation for one iteration.Then with the help of those two we have found the relative complexity of the algorithms.

So in this chapter we have seen that, trade off has to be done for better bit error rate in the cost of complexity.But we have to ensure an optimum condition where the coplexity is just high enough to to provide us very good bit error rate.Different tables and plots have shown in this chapter that the complexity of our proposed algorithm is not very high compared to that of the other algorithms.

CHAPTER 5

CONCLUSION

Low-density-parity-check codes have been studied a lot in the last years and huge progresses have been made in the understanding and ability to design iterative coding systems. The iterative decoding approach is already used in turbo codes but the structure of LDPC codes give even better results. In many cases they allow a higher code rate and also a lower error floor rate. Furthermore they make it possible to implement parallelizable decoders. The main disadvantages are that encoders are somehow more complex and that the code length has to be rather long to yield good results.

The LDPC codes are capable of nearing Shannon capacity performance when decoded with iterative message passing decoders such as sum product decoding in log domainalgorithms which are soft decoding approach. In this paper, decoding algorithms have been proposed combining the hard and soft decoding approach. Our modification allowed us to minimize the effect of sudden sign changes in the decoded vector during the check to bit node update in successive decoding iterations. In this paper we have introduced a new unique factor named modified factor (k). This factor was constantly updated every iteration and it was computed as a vector rather than a single valued variable whose vector dimension was equal to the parity check matrix. The modified factor ensures minimal sign flipping for the codeword computed in later iterations. The latter portion of the algorithm imposes hard decision on the codeword after the soft decision has already been implemented.

The simulation result reflects on the fact, modifications meliorate BER performance as well as improve the decoder's convergence behavior by trading off with complexity. It further infers that even with a very small number of iterations the proposed algorithms significantly outperforms the standard minsum decoder and also existing loglikeli hood decoding.

Analyses of such techniques are required for irregular LDPC codes and for non-binary instances. It has been proven that non-binary LDPC codes have shown performance which is advantageous over its binary counterpart; however, this gives rise to a much higher computational complexity. The sumproduct log domain decoding algorithm that was introduced as a hard decoding approach trades off between performance and complexity. Development of these algorithms is still an open challenge for non-binary cases. Further advancement in these iterative decoding algorithms is possible for the proposed algorithms either in terms of performance or computational complexity.

REFERENCES

[1] J. C. Mackay and R. M. Neal, “Near Shannon limit performance of low-density parity-check codes,” Electron. Lett., vol. 32, pp. 1645-1646, Aug. 1996.

[2] R. G. Gallager, “Low-Density Parity-Check Codes”. Cambridge, MA: MIT Press, 1963

[3] S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications. Englewood Cliffs, NJ: Prentice Hall, 1983.

[4] M. Fossorier, M. Mihaljevic, and H. Imai, “Reduced complexity Iterative decoding of low-density parity-check codes,” IEEE Trans. Commun., vol. 47, pp. 673-680, May 1999.

[5] J. C. MacKay, “Good error-correcting codes based on very sparse Matrices,” IEEE Trans. Inform. Theory, vol. IT-45, pp. 399-432, Mar. 1999.

[6] Chung, k. and Huo, J. “Improved Belief Propagation (BP) Decoding for LDPC Codes with a large number of short cycles” IEEE 63rd Vehicular Technology Conference, 2006. VTC 2006-Spring.

[7] Varnica, N.; Fossorier, M.P.C.; Kavcic, A, "Augmented Belief Propagation Decoding of Low-Density Parity Check Codes," Communications, IEEE Transactions on , vol.55, no.7, pp.1308- 1317, July 2007

[8] Yuan-Mao Chang; Vila Casado, A.I.; Chang, M.-C.F.; Wesel, R.D. “Lower-Complexity Layered Belief-Propagation Decoding of LDPC Codes” IEEE International Conference on Communications, 2008.ICC'08.

[9] Gounai, S.; Ohtsuki, T.; Kaneko, T.;” Modified Belief Propagation Decoding Algorithm for Low-Density Parity Check Code Based on Oscillation” IEEE 63rd Vehicular Technology Conference, 2006.VTC 2006-Spring. Volume: 3 Page(s): 1467 – 1471.

[10] VarnicaNedeljko; Fossorier Marc; , "Improvements in beliefpropagation decoding based on averaging information from decoder and correction of clusters of nodes," Communications Letters, IEEE , vol.10, no.12, pp.846-848, December 2006

[11] S.Y. Chung, J. G. D. Forney, T. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 dB ofthe Shannon limit,” IEEE Commun. Lett., vol. 5, Feb. 2001, pp. 58-60.

[12] Richardson, T. J.; Shokrollahi, A.; and Urbanke R., “Design of Capacity-approaching low-density parity-check codes,” IEEE Trans. Inform. Theory, vol. 47, Feb. 2001, pp. 619-637.

[13] Miladinovic, N.; Fossorier, M.P.C "Improved bit flipping Decoding of low-density parity check codes" IEEE International Symposium on Information Theory, 2002.

[14] Guo F. and. Hanzo L., "Reliability ratio based weighted bitflipping Decoding for low-density parity-check codes," IEEE Electron.Lett., vol. 40, pp. 1356-1358, Oct. 2004.

[15] Ming Jiang; Chunming Zhao; Zhihua Shi; Yu Chen; , "An Improvement on the modified weighted bit flipping decoding Algorithm for LDPC codes," Communications Letters, IEEE , vol.9, no.9, pp. 814- 816, Sep 2005

[16] Zhang J. and Fossorier M., "A modified weighted bit-flipping Decoding of low density parity-check codes," IEEE Commun. Lett., vol. 8, pp. 165-167, Mar. 2004.

[17] Shan M.; Zhao C.; Jiang M., "Improved weighted bit-flipping Algorithm for decoding LDPC codes," IEE Proc.-Commun., vol. 152, pp. 919922, Dec. 2005.