

Islamic University of Gaza (IUG)
Deanery of Post Graduate Studies
Faculty of Engineering
Electrical Engineering Department



CONSTRUCTION OF LDPC CODES USING RANDOMLY PERMUTATED COPIES OF PARITY CHECK MATRIX

By:

Amr Yehia Lulu

Supervised By:

Dr. Ammar Abu Hadrous

Thesis submitted in partial fulfillment of the requirements for
the degree of *Master of Engineering Science*.

June, 2012

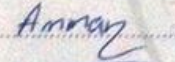




نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ عمرو يحيى محمود لولو لنيل درجة الماجستير في كلية الهندسة قسم الهندسة الكهربائية- أنظمة اتصالات وموضوعها:

Construction of LDPC codes using randomly permuted copies of parity check matrix

وبعد المناقشة التي تمت اليوم الأربعاء 23 رجب 1433 هـ، الموافق 2012/06/13م الساعة الثانية عشرة ظهراً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

	مشرفاً ورئيساً	د. عمار محمد رمضان أبو هديوس
	مناقشاً داخلياً	د. فادي إبراهيم النحال
	مناقشاً خارجياً	د. أنور محمد موسى

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية الهندسة/ قسم الهندسة الكهربائية- أنظمة اتصالات.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

عميد الدراسات العليا



أ.د. قسوة علي العاجز

Abstract

Low-density parity-check codes (LDPC) have been shown to have good error correcting performance, putting in mind the Shannon's limit approaching capability. This enables an efficient and reliable communication. However, the construction method of LDPC code can vary over a wide range of parameters such as rate, girth and length. There is a need to develop methods of constructing codes over a wide range of rates and lengths with good performance.

This research studies the construction of LDPC codes in randomized and structured form. The contribution of this thesis is introducing a method called "Randomly permuted copies of parity check matrix" that uses a base parity check matrix designed by a random or structured construction method such as Gallager or QC-LDPC codes respectively to get codes with multiple lengths and same rate of the base matrix. This is done by using a seed matrix with row and column weights of one, distributed randomly and can be addressed by a number in the base matrix. This method reduces the memory space needed for storing large parity check matrices, and also reduces the probability of failing to construct a parity matrix by random approaches. Numerical results show that the proposed construction performs similarly to random codes with the same length and rate as in Gallager's and Mackay's codes. It also increases the girth average of the Tanner graph and reduces the number of 4 cycles in the resulted matrix if exists in a base graph.

ملخص الأطروحة

استطاعت الأكواد ذات مصفوفات الفحص قليلة الكثافة أن تعطي أداء جيداً لتصحيح الأخطاء. مع الأخذ في الاعتبار قدرتها على الأداء قريبا من نهاية شانون. هذه الخاصية تمكن من الحصول على اتصال فعال وحقيقي. توجد العديد من الطرق لبناء هذا النوع من الاكواد وتتغير حسب معدل وطول الكود. تبقى هناك حاجة لتطوير طرق بناء هذا النوع من الاكواد للحصول على شريحه واسعه من المعدلات والاطوال ذات أداء فعال وجيد.

يتطرق هذا البحث إلى دراسه وعرض طرق بناء هذا النوع من الاكواد بطرقه المختلفه مثل الاكواد المبنية بطريقه عشوائية التركيب مثل أكواد جالاجر، والاكواد المبنية بطريقه منظمة ذات نسق ونموذج معروف مثل الاكواد ذات الطبيعة الدائرية. بالإضافة التي تضيفها هذه الاطروحة هي تقديم طريقة جديدة لبناء الأكواد ذات مصفوفات الفحص قليلة الكثافة تستخدم مصفوفات سابقة البناء كأساس للبناء وللحصول على أكواد ذات أطوال مضاعفة من نفس طول الكود الأساسي وبنفس المعدل. يتم ذلك عن طريق استخدام مصفوفه وحدة موزعه بشكل عشوائي، ويمكن تمييزها برقم عند إدراجها في المصفوفه الأساسية. هذا الأسلوب يقلل من الذاكرة المستخدمة ويقلل من احتمال حدوث فشل في البناء بالطريقة العشوائية. النتائج العديدة أيضاً تدل على ان البناء المقترح يزيد من متوسط طول الدوائر في رسومات تانر الدالة على المصفوفة، ويقلل من عدد الدورات ذات الطول الرباعي في المصفوفة الاساسية.

Dedication and Acknowledgment

To The Soul of My Beloved Father

Yehia M. Lulu

And

To my struggling Mother

I would also want to thank my supervisor Dr. Ammar Abu Hadrous for his support and trust during my work on this thesis, and also during his helpful courses. And Last but not least, I would like to thank my wife for standing beside me throughout all the crises we faced during writing this thesis.

Contents

1	Introduction	1
1.1	Communication Systems and Reliable Transmission.....	1
1.2	LDPC Codes.....	2
1.3	Problem Statement.....	4
1.4	Objectives.....	5
1.5	Thesis Outline.....	5
2	LDPC Codes	6
2.1	Linear Block Codes.....	6
2.2	Low-Density Parity Check Codes.....	8
2.3	Tanner Graph.....	8
2.4	Encoding of LDPC.....	9
2.5	Decoding of LDPC.....	9
2.5.1	Bit-Flip Algorithm (Hard Decision Decoding).....	10
2.5.2	Message Passing Algorithm (MPA).....	11
2.5.3	Log- domain Sum-Product Algorithm Decoder.....	14
2.6	Designing and Optimizing LDPC codes.....	18
2.6.1	Code Size.....	18
2.6.2	Code weight and Code Rate.....	18
2.6.3	Code Structure.....	19
2.6.4	Decoder Iterations.....	19
2.6.5	Minimum Distance.....	20
2.6.6	Girth.....	20
2.6.7	Density Evolution.....	21
2.7	Modeling and Measuring Decoding Performance of LDPC codes.....	21
3	Constructing LDPC codes	23
3.1	Randomly Constructed LDPC Codes.....	23
3.1.1	Gallager's Construction.....	24
3.1.2	MacKay's Constructions.....	28
3.1.3	Irregular LDPC codes.....	30
3.1.4	Progressive Edge-Growth Algorithm.....	32
3.1.5	Protograph LDPC codes.....	34

3.2 Structured Construction of LDPC Codes.....	35
3.2.1 Combinatorial Designs.....	36
3.2.2 Euclidean Geometry LDPC code Construction (EG-LDPC).....	37
4 Constructing Quasi-Cyclic LDPC Codes	40
4.1 Algebraic construction of LDPC codes based on circulant matrices.....	40
4.2 Construction of QC-LDPC Codes from Circulant Permutation Matrices by search algorithm.....	43
4.3 Design parameters of QC-LDPC codes.....	47
5 LDPC code construction using randomly permuted copies of parity check matrix.....	49
5.1 Construction of LDPC code using identity seed matrix	49
5.2 Graphical Perspective.....	53
5.3 Construction of LDPC code using randomly permuted identity seed matrix.....	53
5.4 Construction of LDPC code using seeds of circulant identity matrices..	56
5.5 Simulation and Results.....	57
6 Conclusions and Future Work.....	62
6.1 Conclusions.....	62
6.2 Future Work.....	62
Bibliography.....	63

List of figures

1.1:	Basic Communication system block diagram.....	2
1.2:	Bit Error Rates of different codes shows the effectiveness of LDPC code over other codes.....	4
2.1:	LDPC matrix example and Tanner graph representation.....	8
2.2:	Message Passing Decoding on Tanner Graph.....	14
2.3:	An iteration of message passing in Sum product algorithm. (a) The computation of $q_{ij}(b)$. (b) The computation of $r_{ji}(b)$	15
2.4:	Tanner graph of H matrix of irregular code to be decoded by log-domain SPA decoding algorithm.....	16
3.1:	An example of Gallager's Construction of a $(20, 3, 4)$ parity check matrix, where $\pi_i(H_1)$ denotes a column permutation of H_1	25
3.2:	BER Performance of different lengths of Gallager Codes with $R = 1/2$...	27
3.3:	BER Performance of different lengths of Gallager Codes with $R = 2/3$...	27
3.4:	BER Performance of ensemble 2 of MacKay LDPC codes with and without removing cycles of length 4.....	29
3.5:	BER Performance of ensemble 3 of MacKay LDPC codes with and without removing cycles of length 4.....	29
3.6:	H matrix with corresponding Tanner graph with $l_2 = 4/5$, $l_3 = 1/5$ and $r_3 = 1/3$, $r_4 = 2/3$, $\lambda_2 = 8/11$, $\lambda_3 = 3/11$, $\rho_3 = 3/11$, $\rho_4 = 8/11$	30
3.7:	A tree that shows the depth l for a variable node v_j	34
3.8:	An example of a protograph and the corresponding base matrix.....	34
3.9:	Illustration of the protograph copy and permute procedure with $Q = 4$ copies.....	35
3.10:	The corresponding parity check matrix of a $(9, 3, 1)$ -BIBD.....	36
3.11:	A graphical representation of a finite geometry with $\rho = 2$, $\gamma = 3$ and the corresponding incidence matrix.....	38
4.1:	A $[5219, 4300]$ QC-LDPC code, constructed with $m = 307$, $a = 9$, and $b = 17$ where $o(a) = 17$ and $o(b) = 3$	42
4.2:	An example of Identity matrix of dimensions 4×4 with columns permuted 3 positions to the left and denoted by $I(3)$	43
4.3:	Parity Check Matrix of code $(120, 40)$ of design rate $1/3$ constructed	

	with $J = 4, L = 6, p = 20$	46
4.4:	BER performance of QC-LDPC codes of different lengths with $R = 1/2$..	48
4.5:	BER performance of QC-LDPC codes of different lengths with $R = 2/3$..	48
5.1:	A graphical representation of cycles of girth $g = 6$ resulted from the construction in definition 5.1.....	51
5.2:	(a) Construction of H matrix with an identity seed of size 3×3 . (b) Corresponding Tanner graph of the H matrix. (c) The same Tanner graph in part (b) after rearrangement of nodes shows the separation of edges between the three copies.....	52
5.3:	(a) H matrix of LDPC code of length $N = 18, R = 1/3$, with $j = 2$ and $k = 3$ constructed by definition 5.2. (b) The corresponding Tanner graph of H	54
5.4:	The 6 possible random seeds generated when $p = 3$	54
5.5:	The model matrix \check{H}_c for H in figure 5.3a.....	55
5.6:	BER performance of (504, 3, 6) proposed code designed with base of Gallager H with random seed of $p = 3$, and with an identity seed of $p = 3$, compared to Gallager code with length $N = 504$	58
5.7:	BER performance of proposed codes with (504, 3, 6), with random seed of $p = 3$, and Mackay's and Gallager's codes of the same length.....	58
5.8:	BER performance of proposed codes with rate $R = 2/3$ and a classic random QC-LDPC code with the same rate and block length $N = 324$	59
5.9:	BER performance of proposed code with rate $R = 1/2$ and a classic random QC-LDPC code with the same rate and block length $N = 288$	59
5.10:	BER performance of (90, 6, 3) proposed code, shows the effect of reducing 4 cycles in corresponding Tanner graph of H matrix.....	60
5.11:	BER performance of proposed QC-LDPC code with $N = 360$ and code with random seeds, compared to Gallager code.....	60
5.12:	BER performance of proposed QC-LDPC code with $N = 360$ and code with random seeds, compared to MacKay code.....	61

List of Tables

3.1: A Summary of EG-LDPC codes.....	39
4.1: Examples of QC-LDPC codes constructed from (prime) circulant sizes.....	42
4.2: Smallest value for p for a (J, L) -regular QC-LDPC code with girth $g \geq 6$ found by computer search.....	47

List of Abbreviations

AWGN	Additive White Gaussian Noise
BCH	The Bose, Chaudhuri, and Hocquenghem
BER	Bit Error Rate
BF	Belief Propagation
BIBD	Balanced Incomplete Block Design
BPSK	Binary Phase Shift Key
ECC	Error Correcting Code
EG	Euclidean Geometry
GF	Galois Field
IRRWBF	Implementation Efficient Reliability Ratio Based Weighted Bit-Flipping
LDPC	Low Density Parity Check
LLR	Log-Likelihood Ratio
MPA	Message Passing Algorithm
PEG	Progressive Edge-Growth Algorithm
QC-LDPC	Quasi Cyclic-Low Density Parity Check
SNR	Signal to Noise Ratio
WER	Word Error Rate

Chapter 1

Introduction

1.1 Communication Systems and Reliable Transmission

The goal behind communication systems is to transmit data from a source to a destination. Transmission is done through different mediums like air, wires or optical fibers. These mediums can't keep the originality of data at the source, and they add noise that changes the data according to the level of noise in that medium. The noise added by mediums results in errors that lead to the unreliability problem in communication systems.

In 1948, a remarkable historical innovated work by Shannon [1] showed the limit of reliable transmission of data over noisy channels and suggestions to achieve that, he also defined the capacity of a channel as a number that reliable transmission can be obtained within it. The reliable transmission of data can be achieved by having a code with arbitrary data rate close to the capacity of the channel and it can correct all errors as code length approaches infinity [1].

The study of channel codes began with the pioneered work of Hamming [2] and as mentioned Shannon [1]. The error correcting process is achieved using error correcting codes (ECCs), which are constructed by adding extra symbols that make relations and bonds between the original data symbols. These bonds help in retrieving the original symbols in case they are changed by the effect of the transmission process in the channel. Without error correcting codes, data signals would be retransmitted in case an error occurred at the destination, which in this case adds delay and cost to the system. Another way than retransmission is increasing the power of transmission so the power of signal overcomes the noise in the channel. This approach causes more power waste and consumption in the system. Using the ECCs can increase both the speed and throughput of the system and reduce the power consumption in the system. The basic communication system is shown in figure 1.1; the information data is encoded at the resource before transmission by adding redundancy symbols to the original data based on an error correcting and detection algorithm. Then in the modulation block the data is carried on high frequency signals and sent through the

channel where the noise is added and it changes symbols of the encoded data. At the receiver, the received signal is demodulated and a stream of symbols including the redundancy symbols is now in hand but with some errors in the data. After that, the decoding block uses the bonds added by the coding algorithm to apply a decoding algorithm that detects and corrects the stream of the received data to some extent. The probability of having an error at the output of the system depends on many factors like the code characteristics, the type of modulation, noise, interference level of the channel and the signal power.

There is a tradeoff between the probability of having an error and the transmission power. It will always be a researching point to try to minimize the power consumption while maintaining a reliable communication. This point shows the need of having stronger codes with more error correcting capability.

Several error correcting codes have been developed over decades to perform encoding and decoding of data. They vary in their construction, performance, computation, and implementation complexity. Some well known error correcting codes are convolutional code, Reed Solomon, BCH, turbo and LDPC codes [3].

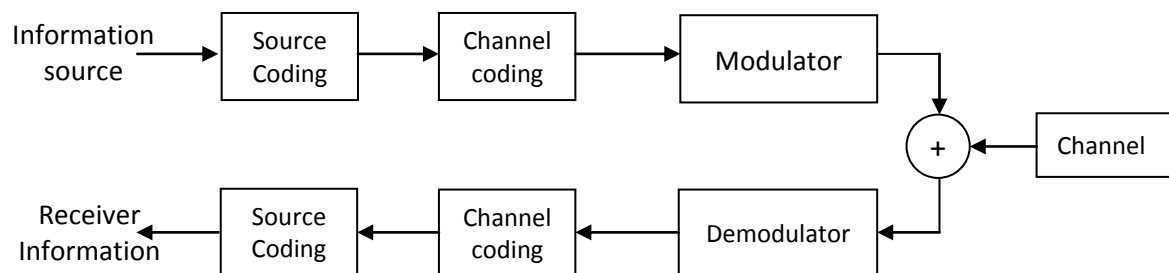


Figure 1.1: Basic Communication system block diagram.

1.2 LDPC Codes

Low density parity check codes are a special type of error correcting codes that is known for their good decoding performance and high throughput.

Low density parity check codes were first introduced by Robert Gallager in early 60's [4, 5]. His work was ignored for decades because of its high computational complexity for hardware implementation in that time.

LDPC codes are decoded using a subclass of message passing algorithms [6] introduced in Gallager's work named the belief propagation decoding algorithm. Its strength is in the inherent parallelism of the message passing and the iterative decoding behavior that is done by updating bit probabilities.

LDPC codes are designed starting from the parity check matrix, where two sets of separated nodes called check and variable nodes are connected to points in the other set based on some regulations and restrictions. The separation of sets allows parallel decoding computations. In contrary, the decoding operations of turbo codes which are the most competitors to LDPC codes, depends on each other in blocks or windows [7] which results in serial computations. LDPC codes have simple graphical representation based on Tanner graph [8] that leads to accurate analysis of performance, also it helps optimizing the designs of regular and irregular constructions.

The complexity problem of computations in LDPC decoding algorithms is reduced by approximation techniques without significant affecting on the performance. Another good property of LDPC codes is having good distance properties, which is one of the main challenges in designing of LDPC codes.

After many years MacKay [9] rediscovered LDPC codes and showed that they perform close to the capacity limits of turbo codes. The work of MacKay was extended by Luby [10] to show that irregular LDPC codes are capable of exceeding the performance of regular LDPC codes, and he also introduced approaches to design irregular codes. Richardson and Urbanke [6] extended that work to soft decision message passing decoding and introduce a method called density evolution. Chung [11] showed that, with carefully choosing an irregular LDPC code from an optimized ensemble, performance of LDPC codes can approach the Shannon limit.

The construction and decoding of LDPC codes results in a low error rates that is close to Shannon limit. A threshold of 0.0045 dB away from Shannon limit is achieved with a LDPC code of rate 1/2 and a block length of 10^7 bits with additive white Gaussian noise [11]. Figure 1.2 shows the effectiveness of different LDPC codes over other codes.

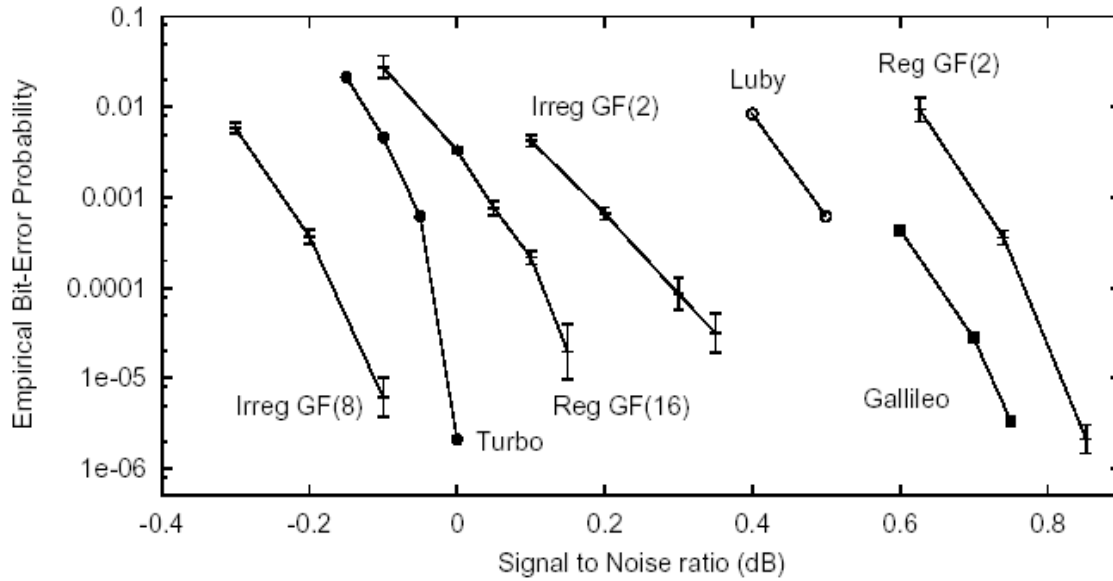


Figure 1.2 [12]: Bit Error Rates of different codes shows the effectiveness of LDPC code over other codes.

1.3 Problem Statement

Construction methods of LDPC codes are either random or structured. Random methods have unstructured row-column connections and generated by computer searches. Construction can be done by designing Tanner graphs that can produce undesirable cycles of four and may not produce a desired rate, but still can be optimized by post processing or by having constrains on the code. Random construction codes are desired to produce high rates and girths, also when long length codes are desired then random construction codes are to be used. They are characterized by flexibility in design and in construction. At the same time they lack regularity in row-column construction which increases the decoder interconnection complexity. Structured constructions have regular interconnection patterns that give good performance and are easier to implement in hardware but with limitations in rate, length and girth.

There are a lot of developed methods that include algebra [13], graph [14], combinatorial designs [15] and heuristic searching techniques [16-19]. The construction of LDPC code determines how good the decoding performance and hardware implementation will be.

1.4 Objectives

- Analyzing and comparing the performance of different LDPC randomized construction algorithms and methods.
- Analyzing and comparing the performance of different LDPC structured construction algorithms and methods.
- Constructing LDPC codes that have a good performance which are also easy to implement.
- Publishing the research results in prestigious IEEE journals and in proceedings of highly reputed, refereed international conferences.

1.5 Thesis Outline

This thesis is organized as follows. In chapter 2, we provide a brief review of LDPC codes, including representations of LDPC codes, encoding, decoding algorithms, and optimization of LDPC codes parameters. In chapter 3 we introduce different methods and schemes of LDPC codes construction. In chapter 4, Quasi-Cyclic LDPC codes with different construction methods are reviewed. In chapter 5 we examine a method of extending a base LDPC parity check matrix by a use of a randomly permuted identity matrix and finding the analyses of the performance of constructed codes under this modification. And finally, chapter 6 includes conclusions and future work for this thesis.

Chapter 2

LDPC Codes

2.1 Linear Block codes

The error correcting codes are obtained by adding additional bits to the original data to be transmitted. These bits are redundant and used to detect and correct errors on the received data. In linear block codes the original data are divided into blocks of fixed lengths of K bits. Based on certain rules and regulations each symbol or bit is expressed as a linear combination of other bits or symbols of block length N called codewords, where $N > K$, the redundancy, $m = N - K$, determines the code rate $R = K/N$, which is a measure on how much information is sent per codeword. The term *linear code* refers to the property where the sum of any subset of codewords is always equal to a codeword from the code space.

With K bits in the input message, there are 2^K possible messages encoded to the same number of possible codewords of length N . Thus there are $2^N - 2^K$ vectors of length N that are not codewords, which is a necessary and sufficient condition for error detection.

A simple encoding requires the encoder to store all possible combination of codewords. But this approach is not efficient for large K . The linear block codes can reduce the complexity of encoding by using a linear generator matrix that transforms original data of K bits to codewords of length N . The linearity leads to the construction of generator matrix that consists of K independent row vectors $g_0 \dots g_{K-1}$ of size N . It is expressed as:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \cdot \\ \cdot \\ \cdot \\ g_{K-2} \\ g_{K-1} \end{bmatrix} \quad (2.1)$$

Where two vectors are called linearly independent if the sum in $GF(2)$ of these vectors is non zero, and called the basis of the space since they can span out all code vectors in the space.

The codewords are generated by multiplying an input vector a by the generator matrix, each 1 in the input vector refers to the row of generator to be added to the combination of independent rows which will result in the unique codeword c , where $c = aG$. This reduces the space complexity from $2^{K \times N}$ which is all possible codewords in the space to $K \times N$, the generator matrix dimensions.

The dual code C^\perp with dimensions $N \times m$ is a linear code called the null space of code C where the cross product of some codes in C and codes in C^\perp is 0, the dual space of a linear code is used to derive the parity check matrix H that relates the generator matrix by $GH^T = 0$ and it is used to check if the received word v is indeed a word, this is guaranteed if

$$vH^T = 0 \quad (2.2)$$

This property is used to detect and correct errors. The construction of generator matrix can be done in systematic and nonsystematic form. In the first type $G = [I_K | P]$ and H is derived as

$$H = [-P^T | I_m] \quad (2.3)$$

where I_K is the $K \times K$ identity matrix, and P is a random matrix with dimensions $m \times N$. The generator matrix is reformed to get the systematic form by Gaussian elimination, and then the transpose of P is obtained to construct the parity check matrix H , also the generator matrix can be derived from the parity check matrix if G is given using the same way.

The *weight* of a code vector is the number of 1's entries. *Minimum distance* of a linear code d_{min} is defined as the minimum number of different bits between two codewords, which is a fundamental property that shows the error correcting capability of a code, hence increasing the minimum distance improves the error correcting capability, referring to the relations, $s \leq d_{min} - 1$ and $t \leq (d_{min} - 1)/2$ where s is number of detected errors, and t is the number errors to be corrected.

2.2 Low-Density Parity Check Codes

Low density parity check matrix are a class of linear block codes defined by a *sparse* parity check matrix H with dimensions $m \times N$. A matrix is called *sparse* if its density is less than 0.5 and *very sparse* if the density remains constant while $N \rightarrow \infty$ [20]. The term *density* refers to the average weight distribution taken over all row vectors in the matrix. The set of LDPC codewords $c \in C$ in the code space C of length N , spans the null space of the parity check matrix H in which: $cH^T = 0$, Thus LDPC codes are given as the null space of a sparse matrix, rather than as the space generated by the rows of that matrix. Considering only binary codes, the parity check matrix has a small number of 1's compared to the number of 0's. The row weight k is the number of 1's in a row, the column weight j is the number of 1's in a column, if weights of rows are equal to k , and weights of columns are equal to j , the parity check matrix is regular LDPC code, otherwise it is an irregular parity check matrix, with always $k > j$. the rate of regular LDPC code can be expressed as $K/N = 1 - m/N$, since the number of 1's in the matrix is given by mk or Nj , then $m/N = j/k$, hence the rate can be express as $1 - j/k$.

2.3 Tanner Graph

LDPC codes can be represented by a *bipartite* graph called *Tanner Graph* [8], the term bipartite refers to a set of nodes partitioned into two subsets in such a way that all edges have a vertex in the first subset and another one in the second, and no edges connect nodes within the same subset. The two subsets are called *check nodes* representing rows, and *variable nodes* representing columns of the LDPC parity check matrix. The entry $H_{a,b}$ is 1 if the check node a is connected to the variable node b as shown in figure 2.1

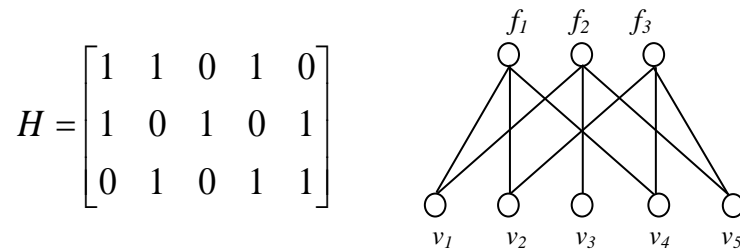


Figure 2.1: LDPC matrix example and Tanner graph representation.

From figure 2.1, f_1, \dots, f_3 represent the rows of H and v_1, \dots, v_5 are the columns of H , rows and columns weights are equal to the edges connected to corresponding check nodes and variable nodes in sequence. Tanner graph results in cycles of different lengths, a *cycle* is defined by the path of edges starting and ending at the same node, the shortest cycle in the graph is called *girth*, which is a fundamental property of LDPC code that affects the performance of decoding. If column and rows have fixed number of 'one' entries in each, then the code is a *regular* LDPC code, otherwise it is *irregular* LDPC code.

2.4 Encoding of LDPC

Encoding of LDPC codes has been considered the major problem in restricting the use of LDPC codes. LDPC encoding is done by the same way linear block codes are encoded. Since LDPC codes are designed starting by the parity check matrix, the generator is derived from H in systematic form $H = [-P^T \mid I_m]$, this needs rearranging H matrix by Gaussian reduction to obtain a dense P matrix part that results in rows and columns of non-fixed lengths. The generator matrix is derived and the data word is multiplied by the generator to get the code. The denseness of P determines the computational complexity of the encoder. The encoding complexity in this case is $O(N^2)$ which is the major concern in LDPC codes, since it is not proportional to the linear time decoding in LDPC. There are some innovated techniques to reduce complexity of encoding process that reaches linear complexity of $O(N)$ [21]. In designing parity check matrix, we may get dependent rows that are reduced in obtaining systematic forms without changing the code space. However, the existence of redundant rows helps in overdefining the code and gives additional indicators that help in the decoding process. A common approach for researches that are less concerned about encoding process, is dealing with the 'all zeros' codeword, which is valid in any linear code space, this helps to skip the encoding process altogether.

2.5 Decoding of LDPC

The process of decoding tries to recover the transmitted codeword c from the received codeword v using the parity check matrix H , since $cH^T = 0$ defines the set of equations that must be satisfied in order to retrieve the received codeword. The relations

between check nodes and variable nodes are taken from the Tanner graph where the (mod 2) sum of the values attached to variable nodes returns 0 to the check node in case the codeword is valid.

In his thesis, Gallager introduced two algorithms: the bit flipping algorithm based on hard decision decoding, and the belief propagation algorithm based on soft decision decoding, they are subclasses of message passing algorithms, which are an iterative decoding algorithms where the passed messages are probability estimators sent, updated and exchanged between variable and check nodes on the Tanner graph, check nodes measure the reliability of bit probability using estimations from adjacent variable nodes, the variable nodes estimate the probability that a given bit is 0 or 1 based on the estimation of the received bit from connected check nodes.

Operations in the decoding algorithm are simplified by implementing log domain to replace multiplication and division operations by adding and subtracting, which reduces the implementation complexity of the decoder.

2.5.1 Bit-Flip Algorithm (Hard Decision Decoding) [4]

As discussed above the bit flipping algorithm is a hard decision algorithm deals with simple messages, where a variable node sends a message to a check node with a value of 1 or 0, then check nodes send a message to its connected variable nodes with a value of 1 or 0 declaring if it is satisfied or not.

Steps of Bit Flipping Algorithm:

Step 1: Initialization. Variable nodes are assigned corresponding bit values from the received vector, and send these values as messages to the check nodes connected with each variable node.

Step 2: Check Nodes Update. Each check node calculates a response for each variable node connected assuming other bits are correct and sends a value that results with a sum of 0 to satisfy the parity check equations. If all equations are satisfied the algorithm terminates.

Step 3: Bit Update. The variable nodes receive values from check nodes and determine if the original received bit is correct, depending on the majority voting of

values received from check nodes. This process is repeated until satisfying all parity check equations or until number of suggested iterations is reached, if maximum number of iterations is reached the algorithm terminates and declares failure to converge. Note that the design of LDPC sparse H matrix ensures existence of one or no transmission error in each parity check equation.

Regardless of the simpleness of this algorithm, there is major drawback in the fact of operating on hard decision by the decoder ignoring all valuable information from the channel when we are dealing with continuous output channels.

2.5.2 Message Passing Algorithm (MPA) [22, 23]

MPA is an iterative decoding algorithm that is used with codes represented by factor graphs as in linear block codes and LDPC codes, it is also known by other names as max-product and sum-product algorithm (SPA).

The message passing algorithm uses estimations of bit probabilities such as intrinsic and extrinsic information of bits representing knowledge before and after an event in sequence. The extrinsic information is about data being depending only on other nodes, and it is not affected by the node it is sent to, in messages containing intrinsic information, nodes will be dominated by its current value.

There are two types of probabilities that express the relation between a variable and an event E , the first is *a-priori* probability of u with respect to E , which is the probability that u is equal to a , and is denoted by:

$$P_E^{priori}(u = a) = P(u = a) \quad (2.4)$$

The second probability is *a-posteriori* probability of u with respect to E , which is the probability of u given the outcome E and it is denoted by:

$$P_E^{post}(u = a) = P(u = a | E) \quad (2.5)$$

and it can be written as:

$$P(u = a | E) = \frac{1}{P(E)} P(E | u = a) P(u = a) \quad (2.6)$$

The term $P(E|u = a)$ is proportional to the extrinsic probability, that describes new information for u obtained from E . and the extrinsic probability is given by:

$$P_E^{ext}(u = a) = dP(E|u = a) \quad (2.7)$$

Where d is a constant that normalize the probability sum to 1. Thus the relation between these probabilities can be written as:

$$P_E^{post}(u = a) = P_E^{priori}(u = a)P_E^{ext}(u = a) \quad (2.8)$$

Since our concern is binary case, we can use log-likelihood ratio (LLR) in which probability of variable u is expressed in terms of a real number. Thus LLR of u is defined as:

$$LLR(u) = \log \frac{P(u = 1)}{P(u = 0)} = \log \frac{p}{1 - p} \quad (2.9)$$

where $p = P(u = 1)$. Now equation (2.6) can be rewritten as:

$$LLR_E^{post}(u) = LLR_E^{priori}(u) + LLR_E^{ext}(u) \quad (2.10)$$

where $LLR(u)$ is positive if $p \geq 0.5$ and negative if $p < 0.5$. The extrinsic information reflects the incremental gain in knowledge of *a-posteriori* information over *a-priori* information. The message passing algorithm is based on *a-priori*, *extrinsic* and *a-posteriori* probabilities. The *a-priori* information is taken from channel, where the extrinsic information comes from other nodes. The steps for MPA iterative decoding are listed below.

MPA iterative processes:

1. Initialization: decoder is initialized by giving variable nodes the values from the received vector y_n of n bits. The initial probability that the sent bit is 1 or 0 given the received vector is calculated for each variable node by:

$$L(u_i) = \ln \frac{P(u_i = 1 | y_i)}{P(u_i = 0 | y_i)} \quad (2.11)$$

where $0 < i < n - 1$. In AWGN channel $L(u_i) = 2y_i / \sigma^2$, where σ^2 is the noise variance. Messages to check nodes and variable nodes, and check nodes LLR s are initialized to zero by sending values on variable nodes to connected check nodes.

2. Updating Check nodes. In this step LLR and check to variable node messages are calculated for each check nodes based on variable node messages. LLR for check nodes of number m denoted by λ_j where $0 < j < m - 1$ is given by:

$$\lambda_j = \sum_{all\text{-}messages} \ln \left[\tanh \left(\frac{abs(\Omega_{i,j})}{2} \right) \right] \quad (2.12)$$

where $\Omega_{i,j}$ is the message from variable node i to check node j . check to variable messages are given by:

$$\Lambda_{j,i} = 2 \times atanh \left\{ \exp \left\{ \ln(\lambda_j) - \ln \left(\tanh \left(\frac{\Omega_{i,j}}{2} \right) \right) \right\} \right\} \quad (2.13)$$

3. Updating variable nodes. LLR and outgoing messages of variable nodes are calculated and LLR is given by:

$$\lambda_i = L(u_i) + \sum_{all\text{-}messages} \Lambda_{j,i} \quad (2.14)$$

LLR is the sum of all incoming messages with addition to the initial value of variable node. Messages from variable node to check nodes are given by check nodes LLR minus messages received on that edge as given:

$$\Omega_{i,j} = \lambda_i - \Lambda_{i,j} \quad (2.15)$$

4. Decision: The values of variable nodes are decided as 1 or 0 by the value λ_i . If $\lambda_i < 0$ then $LLR_i = 0$, and $LLR_i = 1$ if $\lambda_i \geq 0$, if $LLR \times H^T = 0$, then take the value of LLR as an estimation of codeword c_n at the decoder output, if not, go to step 2. Maximum number of iterations is decided so the algorithm stops in case the algorithm doesn't halt.

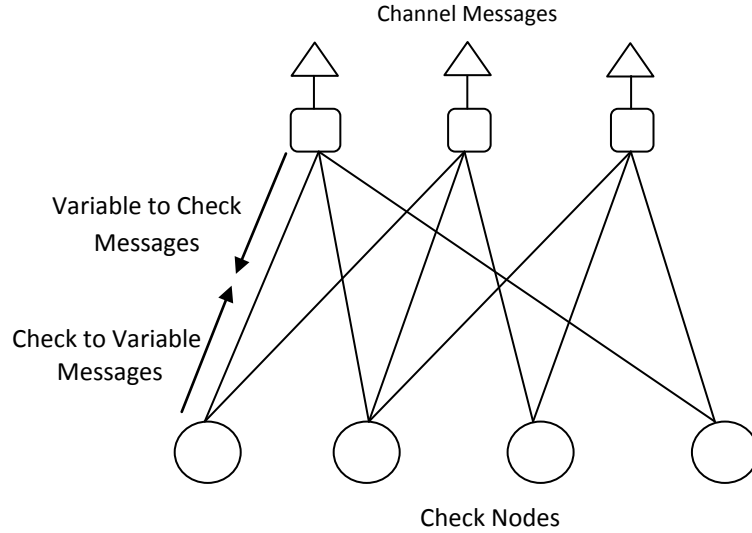


Figure 2.2: Message Passing Decoding on Tanner Graph.

2.5.3 Log- domain Sum-Product Algorithm Decoder

A simplified version of the sum product algorithm (SPA) that reduces the complexity of the parity check update at the cost of some loss in performance was proposed in [24]. This simplification has been derived by operating in the log-likelihood domain. In log domain SPA the following notations and *LLRs* are defined.

Notations:

$q_{ij}(b)$: Probability that variable node v_i having a bit value b satisfies all check equations except c_j .

$r_{ji}(b)$: Probability that check equation related to check node c_j is satisfied if variable node v_i has value b . (Meaning that from all the other nodes related to check node c_j we obtain an even ($b=0$) or odd ($b=1$) number of ones).

$Q_i(b)$: Probability that variable node v_i has value b , where $b = \{0,1\}$.

$V_j = \{\text{variable nodes connected to check node } c_j\}$ (neighborhood of c_j).

$V_{j/i} = \{\text{variable nodes connected to check node } c_j\} / \{\text{variable node } v_i\} = \{\text{variable nodes connected to check node } c_j \text{ except } v_i\}$.

$C_i = \{\text{check-nodes connected to variable node } v_i\}$ (neighborhood of v_i).

$C_{i/j} = \{\text{check nodes connected to variable node } v_i\} / \{\text{check node } c_j\} = \{\text{check nodes connected to variable node } v_i \text{ except } c_j\}$.

LLRs:

$$L(c_i) = \ln \left(\frac{\Pr(c_i = 0 | y_i)}{\Pr(c_i = 1 | y_i)} \right) \quad (2.16)$$

$$L(r_{ji}) = \ln \left(\frac{r_{ji}(0)}{r_{ji}(1)} \right) \quad (2.17)$$

$$L(q_{ji}) = \ln \left(\frac{q_{ij}(0)}{q_{ij}(1)} \right) = L(v_i) + \sum_{j' \in C_i / j} L(r_{ji'}) \quad (2.18)$$

$$L(Q_i) = \ln \left(\frac{Q_i(0)}{Q_i(1)} \right) = L(v_i) + \sum_{j \in C_i} L(r_{ji}) \quad (2.19)$$

where $L(r_{ji})$ can be approximated by [25]:

$$L(r_{ji}) = \sum_{i' \in V_j / i} \boxplus L(q_{i'j}) \quad (2.20)$$

Where \boxplus refers to box-plus operator defined as:

$$L(x_1) \boxplus L(x_2) = \text{sgn}[L(x_1)] \cdot \text{sgn}[L(x_2)] \cdot \min\{|L(x_1)|, |L(x_2)|\} \quad (2.21)$$

with rules:

$$L(x) \boxplus 0 = 0, \quad L(x) \boxplus \infty = L(x), \quad L(x) \boxplus -\infty = -L(x). \quad (2.22)$$

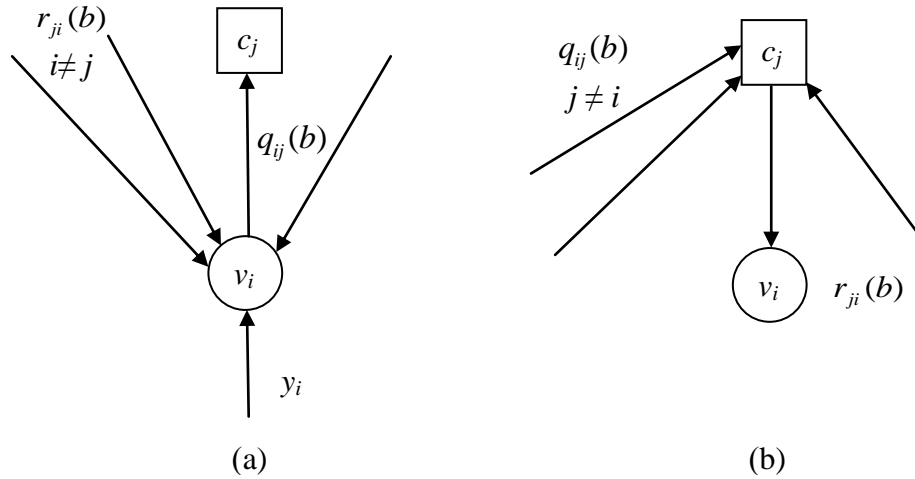


Figure 2.3: An iteration of message passing in Sum product algorithm. (a) The computation of $q_{ij}(b)$. (b) The computation of $r_{ji}(b)$.

Summary of the log-domain SPA Decoding Algorithm [26]:

- 1) For $i = 0, 1, \dots, n-1$, initialize $L(q_{ij}) = L(v_i) = 2y_i/\sigma_n^2$ for all i, j for which $h_{ji} = 1$.
- 2) Update $\{L(r_{ji})\}$.

- 3) Update $\{ L(q_{ij}) \}$.
- 4) Update $\{ L(Q_i) \}$.
- 5) For $i = 0, 1, \dots, n - 1$, set.

$$\hat{v} = \begin{cases} 1 & \text{if } L(Q_i) < 0 \\ 0 & \text{else} \end{cases}$$

- 6) If $\hat{v}H^T = 0$ or the number of iterations equals the maximum number of iterations set, stop, else, go to step 2.

Example:

Given Tanner graph of H matrix as in figure 2.4, and running first iteration of decoding process given the value of $L(v)$.

$$L(v) = [0.7 \quad -1.5 \quad 0.3 \quad 1.1 \quad -0.6]$$

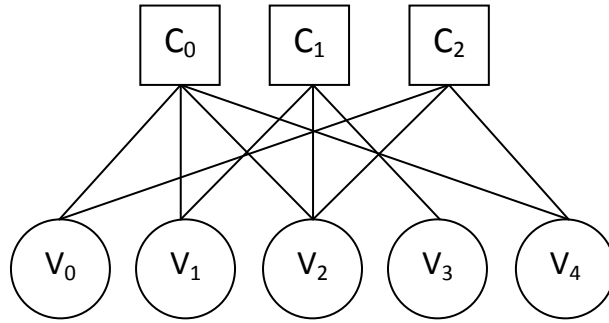


Figure 2.4: Tanner graph of H matrix of irregular code to be decoded by log-domain SPA decoding algorithm.

First step: Initializing the value of $L(q_{ij})=L(v_i)$ so,

$$L(q_{0j})_{j=0,2} = L(v_0) = 0.7$$

$$L(q_{1j})_{j=0,1} = L(v_1) = -1.5$$

$$L(q_{2j})_{j=1,2,3} = L(v_2) = 0.3$$

$$L(q_{3j})_{j=1,2} = L(v_3) = 1.1$$

$$L(q_{4j})_{j=0,2} = L(v_4) = -0.6$$

Second step: Updating $L(r_{ji})$.

$$L(r_{00}) = L(q_{10}) \boxplus L(q_{20}) \boxplus L(q_{40}) = -1.5 \boxplus 0.3 \boxplus -0.6 \approx 0.3$$

$$L(r_{01}) = L(q_{00}) \boxplus L(q_{20}) \boxplus L(q_{40}) = 0.7 \boxplus 0.3 \boxplus -0.6 \approx -0.3$$

$$L(r_{02}) = L(q_{00}) \boxplus L(q_{10}) \boxplus L(q_{40}) = 0.7 \boxplus -1.5 \boxplus -0.6 \approx 0.6$$

$$L(r_{04}) = L(q_{00}) \boxplus L(q_{10}) \boxplus L(q_{20}) = 0.7 \boxplus -1.5 \boxplus 0.3 \approx -0.3$$

$$L(r_{11}) = L(q_{21}) \boxplus L(q_{31}) = 0.3 \boxplus 1.1 \approx 0.3$$

$$L(r_{12}) = L(q_{11}) \boxplus L(q_{31}) = -1.5 \boxplus 1.1 \approx -1.1$$

$$L(r_{13}) = L(q_{11}) \boxplus L(q_{21}) = -1.5 \boxplus 0.3 \approx -0.3$$

$$L(r_{20}) = L(q_{22}) \boxplus L(q_{32}) \boxplus L(q_{42}) = 0.3 \boxplus 1.1 \boxplus -0.6 \approx -0.3$$

$$L(r_{22}) = L(q_{02}) \boxplus L(q_{32}) \boxplus L(q_{42}) = 0.7 \boxplus 1.1 \boxplus -0.6 \approx -0.6$$

$$L(r_{23}) = L(q_{02}) \boxplus L(q_{22}) \boxplus L(q_{42}) = 0.7 \boxplus 0.3 \boxplus -0.6 \approx -0.3$$

$$L(r_{24}) = L(q_{02}) \boxplus L(q_{22}) \boxplus L(q_{32}) = 0.7 \boxplus 0.3 \boxplus 1.1 \approx 0.3$$

Third step: Updating $L(q_{ij})$.

$$L(q_{00}) = L(v_0) + L(r_{20}) = 0.7 - 0.3 = 0.4$$

$$L(q_{02}) = L(v_0) + L(r_{00}) = 0.7 + 0.3 = 1$$

$$L(q_{10}) = L(v_1) + L(r_{11}) = -1.5 + 0.3 = -1.2$$

$$L(q_{11}) = L(v_1) + L(r_{01}) = -1.5 - 0.3 = -1.8$$

$$L(q_{20}) = L(v_2) + L(r_{12}) + L(r_{22}) = 0.3 - 1.1 - 0.6 = -1.4$$

$$L(q_{21}) = L(v_2) + L(r_{02}) + L(r_{22}) = 0.3 + 0.6 - 0.6 = 0.3$$

$$L(q_{22}) = L(v_2) + L(r_{02}) + L(r_{12}) = 0.3 + 0.6 - 1.1 = -0.2$$

$$L(q_{31}) = L(v_3) + L(r_{23}) = 1.1 - 0.3 = 0.8$$

$$L(q_{32}) = L(v_3) + L(r_{13}) = 1.1 - 0.3 = 0.8$$

$$L(q_{40}) = L(v_4) + L(r_{24}) = -0.6 + 0.3 = -0.3$$

$$L(q_{42}) = L(v_4) + L(r_{04}) = -0.6 - 0.3 = -0.9$$

Step four: Update $L(Q_i)$.

$$L(Q_0) = L(v_0) + L(r_{00}) + L(r_{20}) = 0.7 + 0.3 - 0.3 = 0.7$$

$$L(Q_1) = L(v_1) + L(r_{01}) + L(r_{11}) = -1.5 - 0.3 + 0.3 = -1.5$$

$$L(Q_2) = L(v_2) + L(r_{02}) + L(r_{12}) + L(r_{22}) = 0.3 + 0.6 - 1.1 - 0.6 = -0.8$$

$$L(Q_3) = L(v_3) + L(r_{13}) + L(r_{23}) = 1.1 - 0.3 - 0.3 = 0.5$$

$$L(Q_4) = L(v_4) + L(r_{04}) + L(r_{24}) = -0.6 - 0.3 + 0.3 = -0.6$$

Step five: Find v .

$$v = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Step Six: Check if $\hat{v}H^T = 0$ or maximum number of iterations is reached to stop, else go to step 2.

2.6 Designing and Optimizing LDPC codes

There are some parameters to be taken in consideration in the design of LDPC codes as code length and code rate. These parameters and others are affected by the application needs and affect the performance of the code. Also In designing LDPC codes, there are some parameters to be optimized in order to get better performance and better bit error rate, such as optimizing girth, increasing Hamming distance.

2.6.1 Code Size

One of the basic parameters of code design is *code size*, which is specified by the code length N and row-column weights j and k as (N, j, k) , unsurprisingly, codes with larger block length are better in performance than shorter ones [9], but in terms of cost and implementation they require larger memory.

2.6.2 Code weight and Code Rate

Another parameter is *code weight and rate*. Having codes with large row and column weights, results in increasing computations at each node, since nodes will be attached to more information bits in the decoding process. In the other hand, higher weights leads to more consistent decoding and more nodes to participate in estimating the probability of a bit which leads to faster convergence. Code rate describes the redundancy of bits where higher rates mean less redundancy, which results in high throughput of information data and less protection of bits that decreases the efficiency of decoding performance and increases bit error rate (BER) [27]. Low rates means

less throughput and better decoding performance. Column weight has been proven [4] to affect the minimum distance property as in codes with column weight of 2 grows logarithmically with N , where codes with $j \geq 3$ have a minimum distance that grows linearly with N that makes them more desired to be used. However, compared with $j \geq 3$ codes, codes with $j = 2$ are easier to implement and require less storage making them a target for many applications [28]. A careful adjusting of the bit and check nodes, with maximizing bit degree, minimizing check degree, and increasing the support of some bits, many well-designed Irregular codes with nonuniform columns distribution of 1's have achieved better error correcting performance than regular codes [10].

2.6.3 Code Structure

The next parameter in designing LDPC codes is the type of *code structure*, which is defined as the pattern of connections between bit and check nodes, the type of structure results in many tradeoffs as cost, decoder complexity and flexibility of design. The original introduced LDPC code [4] was a random code, where no pattern of interconnections defines the row-column connections. Some disadvantages of random construction appear when trying to implement it in real communication system. Random construction codes need to be stored in memory for decoding and encoding process. In case of long block codes, very large memory usage is needed to store the parity check matrix which deduces the computational efficiency of the code, to overcome this problem structured constructions appear to have predefined patterns that needs few inputs to generate a range of code words. These constructions have an advantage in reducing the cost, complexity, memory usage, and latency.

2.6.4 Decoder Iterations

Number of iterations in the decoding process is defined by the number of times the received bit is estimated before a hard decision is made by the decoding algorithm [29]. Increasing number of iterations increases the decoding convergence and lowers the bit error ratio until reaching the error floor, The parameter of iterations affects the power consumption and the decoder delay, that is in some applications like video broadcasting where decoding time is limited, optimum number of iterations is set to the maximum decoding delay allowed. Another point is that in case of soft values,

decoding may converge to a non valid state, also soft values tend to converge to a stable state after a few number of iterations, that's where increasing time of decoding does not improve the decoding process.

2.6.5 Minimum Distance

In general, Codes with larger minimum distance have better performance and correct larger number of errors. LDPC codes are found to have better minimum distance than linear codes with the same length and dimension. A good property of LDPC codes is that minimum distance is proportional to the size of code in case of random constructed LDPC codes [7]. In other types of structured construction, minimum distance is restricted to an upper bound depending on column weight, and in this case increasing the code size does not increase minimum distance [19]. Although LDPC codes performance depends on both the structure of its Tanner graph and its minimum distance, a LDPC code with better minimum distance may not outperform the performance of a code with worse minimum distance, that's because codes with better graph structure ease the decoding process in the belief propagation decoding algorithm, since BF is a suboptimum and graph dependent [23].

2.6.6 Girth

Girth of a code is also optimized to enhance its performance. Avoiding cycles of length 4 and trying to obtain girths that is greater than 4, provides sufficient feedback protection, since with small cycles a node gets a probability estimate depending mainly on its own probability contribution. With large girth, the probability estimation of bit decoding relies on the connected bits, which results in better estimation of the node. Bad topologies result in a low minimum distance. It is proved that there is dependence between girth and minimum distance of a code [8, 19]. In general, maximizing girth will improve code performance to some bound [30]. Also the girth distribution defined by the fraction of the symbol nodes with a given girth is proposed as an effective tool for designing short LDPC codes and it matters more than girth [31]. The average girth is defined by the sum of the smallest cycles in nodes divided by the number of nodes, where codes with larger average girths perform better than ones with small averages.

2.6.7 Density Evolution [6, 32]

Density evolution is a technique used for observing convergence of the overall bit error probability of decoding process and thus approximating the decoding bit error probability. It is used in Sum product algorithm messages at fixed SNR levels as the decoder iterates under the assumption of having free cycles. The density is observed and plotted in a graph to show the relation between number of iterations and convergence of density. Using this chart, one can optimize the number of iteration used in the decoding process and approximate the bit error probability of the code.

2.7 Modeling and Measuring Decoding Performance of LDPC codes

The decoding performance of LDPC codes is measured and evaluated *using Bit Error Rate (BER)*, which is the number of errors found per iteration over the code length at a given *Signal to Noise Ratio (SNR)*, calculated by

$$BER = \frac{\text{Number of Errors}}{\text{Numbers of Bits}} \quad (2.22)$$

The BER of a code is calculated under the assumption of *Additive White Gaussian Noise (AWGN)* channel. The AWGN channel is a simple-binary input, unquantized output channel- model, which subjects the transmitted vector of bits into random peaks of energy (noise) described as a random normally distributed variable added successfully to the transmitted symbols. The channel output is modeled as $y_i = s_i + n_i$, where s is the transmitter output, n is the AWGN, at any instant i . The randomness of the Gaussian noise is a one sided power spectral density (PSD) N_0 , that depends on the noise level or the variance σ^2 , where $N_0 = 2\sigma^2$.

The **errors** are the received bits with different values than the transmitted bits. The **SNR** is a measure that compares the level of a desired signal to the level of background noise, and it is modeled as:

$$SNR = 10 \log \frac{E_s}{N_0} \quad (2.23)$$

where E_s is the signal energy. The higher the SNR, the less obtrusive the background noise is. Thus increasing the SNR generally decreases number of errors. In simulating decoding performance, BER is calculated many times at the same level of SNR and

the average of BER is plot as an output of SNR level, this is to increase the level of confidence of BER results. Other parameters may be used in simulation like *Word Error Rate (WER)* which is the number of decoded words with errors compared to the number of transmitted words, the use of BER or WER is determined by the application used. In some cases it is essential that all of a word must be received correctly, that's where WER is preferred.

In case of binary Input, data is modulated using *Binary Phase Shift Key (BPSK)* technique as in the simulations of this thesis.

Chapter 3

Constructing LDPC Codes

The rediscovery of LDPC codes opened the doors wide to researchers to find efficient algorithms that construct efficient codes, especially after the remarkable improvement of computers processing speed. The construction of LDPC codes is affected by many parameters that should be taken care of, as discussed in Chapter 2. The main goal in constructing a code is determining the length and rate of the code to be used. Another aspect is the ability of getting a construction method that can produce a wide range of codes that vary in length and rate as desired. There always will be a tradeoff between good decoding performance and easier hardware implementations.

The construction of LDPC codes is categorized mainly into two main categories: **Random** constructions and **Structured** constructions. The type of construction is determined by the connections between check nodes and variable nodes in Tanner graph. Each type of constructions has their advantages over the other. In the following, we will review some of the important construction methods from both types and clarify their reflections over the performance of LDPC codes.

3.1 Randomly Constructed LDPC Codes

Random constructions refer to the unstructured row-column connections in the parity check matrix with no predefined pattern. The design of randomly constructed LDPC codes is done by computer searching algorithms to fulfill the design requirements. Actually the process of generating in this case is a pseudo-random process but it is written 'random' for brevity.

As previously mentioned in chapter 2, constructing LDPC codes is a backward process, starting by constructing the H matrix with dimensions of $N \times K$. The random search results in dependant rows that reduce the rate of the code. The rank of H is defines the number of K bits that will ensemble the code and $K = N - \text{Rank}(H)$, thus the rate of the designed code varies and become slightly higher than what is intended[20], which means obtaining a different code. So the rank of random codes

should be checked each time the H matrix is constructed. In case we want to construct a code with $R = 1/3$, we construct a random H matrix with dimensions of rate 3:2 as in (1200, 800) code, meaning to have all rows to be linearly independent, but the rank is to be less than or equal to the desired full rank, which increases the rate of the code. The sparsity of LDPC codes decreases the probability of producing linearly dependent rows as the code size grows large, especially when column weight of H is odd [33]. So repeating the generating of code with different seed may result in the desired rate.

Another important point of randomly constructed LDPC codes is that the generating of codes - based on some regulations of random filling of H matrix- results in *similar* codes that has different code space but they approximately have the same performance on the level of bit-error, this allows us to optimize the code design by focusing on design parameters without going on details discussing the structure of bits in a specific code.

The construction of random LDPC codes is done by adding random edges to a Tanner graph or '1' entries in the parity check matrix. Designing a code with a desired rate and girth can be achieved by post processing the connections of H matrix to maintain the desired rate and girth and to delete cycles of 4 which degrade the performance of the code. Also random construction is done by putting constraints on design parameters to satisfy the desired girth and rate.

The design and analysis of LDPC codes are based on *Ensembles*, an ensemble is a set of codes with certain properties. Usually it is easier to evaluate the performance of a code based on its ensemble than evaluating the performance for a particular randomly generated code, putting in mind that the average performance of the ensembles is well approximated to meet the performance of any generated code from the same ensemble. Richardson proved the assumption that the codes in an ensemble are equivalent [6]. Thus a randomly constructed code is chosen from an ensemble that satisfies the desired requirements and then it is fixed and built in the system.

Random codes have better performance compared to structured codes in case of long codes [11]. They are used in cases we want to increase the girth or rate of a given size [17]. A disadvantage of random LDPC codes appears when wanted to be

A step-by-step random construction method that produces Gallager code with girth $g > 4$ is introduced by Lin and Costello in [3], the introduced algorithm grows the parity check matrix of dimensions $m \times N$ column by column. The codes generated by this algorithm are regular and given by the framework (N, j, k) . The approach is done by computer search starting with the null matrix H_0 by picking a random vector h_i of the pool of length m and weight j at the i th step where $1 \leq i \leq N$, and each vector is checked if it satisfies the constraints of the ensemble to be added to a previously partially constructed parity check matrix H_{i-1} that satisfies the required constraint of j, k , and g . To add column h_i the following constraints must be satisfied, taking in mind that for constructing regular H matrix with constant row weight k the equation $j \times N = k \times (N - K)$ must hold, if N is not divisible by $(N - K)$ then $j \times N = k \times (N - K) + b$ which can be rearranged to $j \times N = (N - K - b)k + b(k + 1)$.

1. Choose h_i at random from the remaining binary $(N - K)$ -tuples that are not being used in H_{i-1} and that were not rejected earlier.
2. Check whether h_i has more than one 1-component in common with any column in H_{i-1} . If not, go to step 3, otherwise, reject h_i and go back to step 1 to choose another candidate column.
3. Add h_i to H_{i-1} to form a temporary partial parity check matrix H_i . check the row weights of H_i if all the top b rows of H_i have weights less than or equal to $k + 1$, and all the bottom $N - K - b$ rows of H_i have weights less than or equal to k , then permanently add h_i to H_{i-1} to form H_i and go to step 1 to continue the construction process. If any of the top b rows of H_i has weight exceeding $k + 1$, or any of the bottom $N - K - b$ rows of H_i has weight exceeding k , reject h_i , and go to step 1 to choose another candidate column.

From the analysis of this algorithm [34], it is recommended to increase the number of pool candidates compared to N , this improves the probability that the candidate will be acceptable and not rejected. Still, the algorithm is not suitable for handling a large number of candidates since the evaluation of candidates is repeated exponentially and the founding of a valid candidate becomes very difficult. A modification on this algorithm is to keep vectors already in use since rejecting will result in extensive bookkeeping and causes latency for looking up for matches in rejects table. In case

the candidate from the pool is already in use, it will not satisfy the conditions of step 2 and another candidate will be picked.

Figures 3.2 and 3.3 show the performance of Gallager codes with different rates and lengths over an AWGN channel.

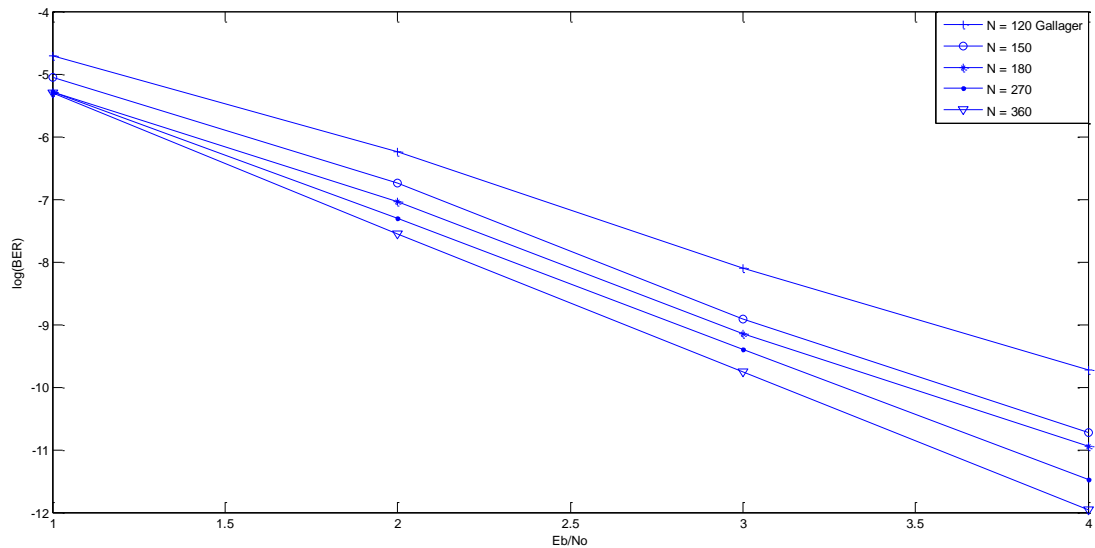


Figure 3.2: BER Performance of different lengths of Gallager Codes with $R = 1/2$.

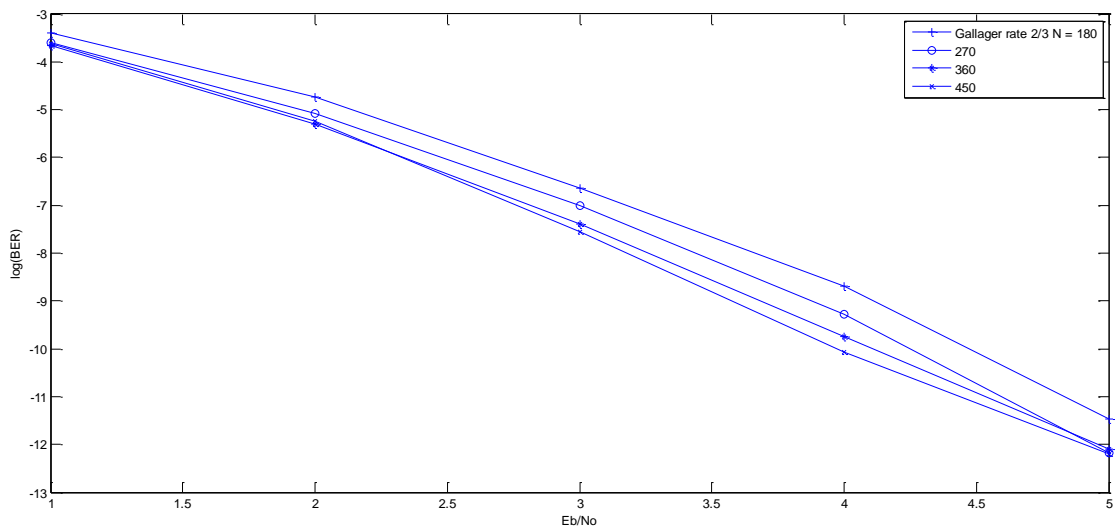


Figure 3.3: BER Performance of different lengths of Gallager Codes with $R = 2/3$.

3.1.2 MacKay's Constructions

MacKay was the first to introduce the benefit of designing codes with sparse parity check matrices and showed their ability to perform near capacity limits [20, 33].

MacKay introduced an order of ensembles of random LDPC codes in which he assumed will decrease the average probability of decoding error, but not necessarily improve performance [20] named by '*Ensembles of very Sparse Matrices*', though he didn't give a prove for his assumption.

MacKay's Ensembles of Very Sparse Matrices:

1. Matrix H is generated by starting from an all-zero matrix and randomly flipping j not necessarily distinct bits in each column.
2. Matrix H is generated by randomly creating weight j columns.
3. Matrix H is generated with weight j per column and (as near as possible) uniform weight per row.
4. Matrix H is generated with weight j per column and uniform weight per row, and no columns having overlap greater than 1 (meaning, 'no 4-cycles').
5. Matrix H is further constrained so its bipartite graph has girth greater than 6.
6. Matrix $H = [C_1 | C_2]$ is further constrained or slightly modified so that C_2 is an invertible $m \times m$ matrix.

Figures 3.4 and 3.5 show the BER performance of MacKay code from ensemble 2 where 1's are distributed uniformly in columns and ensemble 3 where 1's are distributed uniformly in columns and rows, the construction of MacKay code with obtaining these codes with random generation doesn't guarantee a cycle-4 free construction for short lengths codes, thus further processing can be applied in order to remove these loops which degrade the decoding performance of the code as shown in figures 3.4 and 3.5.

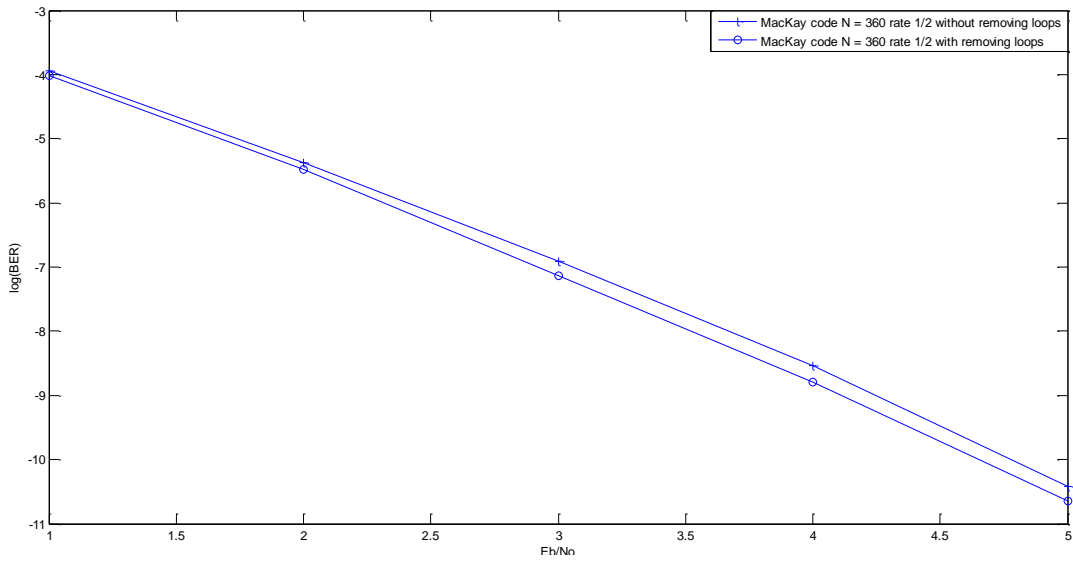


Figure 3.4: BER Performance of ensemble 2 of MacKay LDPC codes with and without removing cycles of length 4.

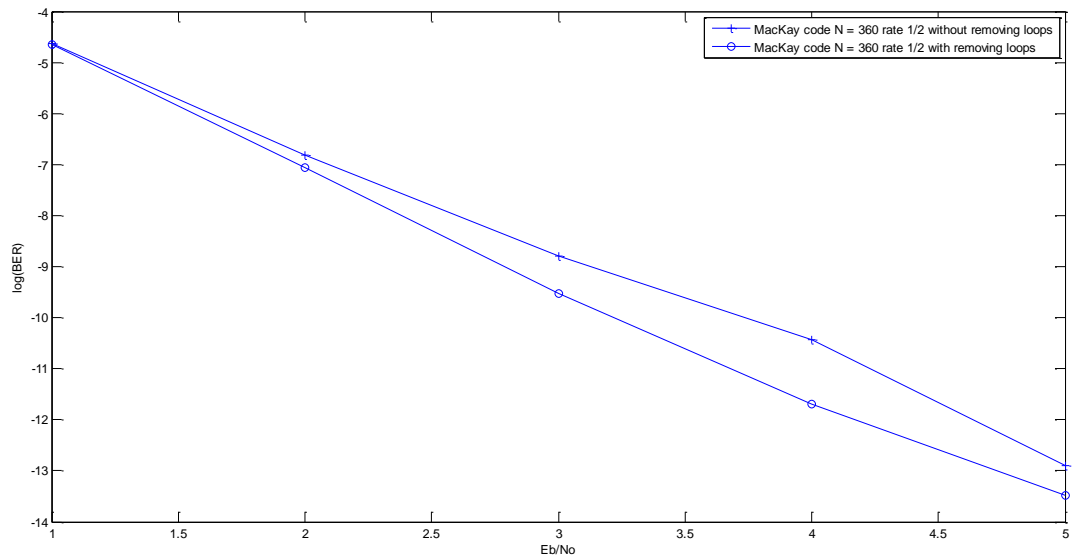


Figure 3.5: BER Performance of ensemble 3 of MacKay LDPC codes with and without removing cycles of length 4.

3.1.3 Irregular LDPC codes

Irregular LDPC codes are a generalization of regular LDPC codes. Parity check matrix of an irregular LDPC codes [32] has nonuniform column weights and non uniform row weights, thus, an irregular LDPC code is defined by an expression that describe the degree distribution of check nodes and variable nodes along the Tanner graph. The irregularity in weights results in variable nodes and check nodes with different degrees in the Tanner graph. Increasing the degree of a variable node means connecting to more check nodes and more information to be gathered about the state of the correct value of the variable node, thus more accurate values can be derived. In the other hand increasing the variable node degree increases the probability of sending wrong guesses to the variable node. These two factors are balanced in the design of irregular codes in order to enhance the performance of the code.

The procedure of decoding irregular LDPC codes is done by starting with higher degree variable nodes where they converge to their correct values faster with less number of iterations, such strong bits are called 'elite bits'. Then they pass their values to check nodes helping them to decode lower degree variable nodes. Now the remaining errors are now easier to detect.

In constructing irregular LDPC codes, the following parameters are defined.

- λ_i : The fraction of edges incident to variable nodes of degree i .
- ρ_j : The fraction of edges incident to check nodes of degree j .
- l_i : The fraction of variable node of degree i .
- r_j : The fraction of variable node of degree j .

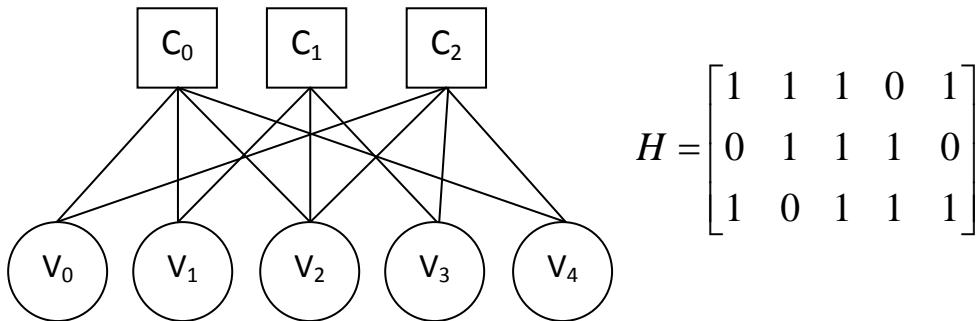


Figure 3.6: H matrix with corresponding Tanner graph with $l_2 = 4/5$, $l_3 = 1/5$ and

$$r_3 = 1/3, r_4 = 2/3, \lambda_2 = 8/11, \lambda_3 = 3/11, \rho_3 = 3/11, \rho_4 = 8/11.$$

In the literature, it is usual to specify the variable node and check node degree distribution polynomial, denoted by [10]:

$$\lambda(X) = \sum_{d=1}^{d_v} \lambda_d X^{d-1} \quad (3.1)$$

$$\rho(X) = \sum_{d=1}^{d_c} \rho_d X^{d-1} \quad (3.2)$$

Where d_v and d_c are the maximum variable node and check node degree respectively. By defining E as the total number of edges in the corresponding Tanner graph, the number of variable nodes of degree d by $N_v(d)$, and the number of check nodes of degree d by $N_c(d)$ then they can be expressed as proved [35] by:

$$E = \frac{n}{\int_0^1 \lambda(X) dX} = \frac{m}{\int_0^1 \rho(X) dX} \quad (3.3)$$

$$N_v(d) = \frac{E \lambda_d}{d} = \frac{n \lambda_d}{d \int_0^1 \lambda(X) dX} \quad (3.4)$$

$$N_c(d) = \frac{E \rho_d}{d} = \frac{m \rho_d}{d \int_0^1 \rho(X) dX} \quad (3.5)$$

Then from equation (3.3) one can easily conclude that code rate is bounded by

$$R \geq 1 - \frac{m}{n} = 1 - \frac{\int_0^1 \rho(X) dX}{\int_0^1 \lambda(X) dX} \quad (3.6)$$

A precise theory in optimizing the design of irregular codes is difficult to find, the studies about this field [14, 32] depends on computer searches to find the best optimization of check and variable degrees. Different approaches have been used in the design of irregular LDPC codes. One approach is to have one profile given as an input from which the other profile can be found, a simpler searching approach is to fix the distribution along one dimension, typically using constant row-weight.

3.1.4 Progressive Edge-Growth Algorithm

In their published paper [18] Hu *et al.* introduced a method for constructing LDPC codes based on progressive establishing of edges in Tanner graph between variable nodes and check nodes called Progressive Edge-Growth Algorithm (PEG). Tanner graphs are defined by (V, E) where V is a set of nodes consists of check nodes $V_v = \{v_0, v_1, \dots, v_{n-1}\}$ and variable nodes $V_c = \{c_0, c_1, \dots, c_{m-1}\}$, E of size $V_v \times V_c$ consists of all possible edges (c_i, v_j) between the set of V_v and V_c when $h_{ij} \neq 0$, $h_{ij} \in H$, $0 \leq i \leq m - 1$, $0 \leq j \leq n - 1$. The degree of variable nodes is stored in the array $D_v = \{d_{v_0}, d_{v_1}, \dots, d_{v_{n-1}}\}$, where d_{v_j} is the degree of the variable node v_j , in the same manner the degree of the check nodes is stored in the array $D_c = \{d_{c_0}, d_{c_1}, \dots, d_{c_{m-1}}\}$, where d_{c_i} is the degree of the check node c_i . Define E_{v_j} be the set of edges incident on the variable node v_j and $E_{v_j}^k$ is the k^{th} edge incident on v_j where $0 \leq k \leq d_{v_j} - 1$. For a variable node v_j , the set of check nodes connected through a cycle starting and ending at v_j is $N_{v_j}^l$ where l is the depth of its neighbors as shown in figure 3.7, the complementary set of $N_{v_j}^l$ is $\overline{N}_{v_j}^l$ where $N_{v_j}^l \cup \overline{N}_{v_j}^l = V_c$. for computing $N_{v_j}^l$ an indicator I_{c_i} is set for each check node c_i taking values from the set $\{0, 1\}$ and it is initialized to 0, as the tree starting at v_j to the determined depth l , the indicators of check nodes in the tree is also set to 1, which indicates that these nodes belongs to $N_{v_j}^l$, at the same time $\overline{N}_{v_j}^l$ is obtained by checking the indicator I_{c_i} is 0. For each variable node v_j , a local girth g_{v_j} is defined as the set $\{g_{v_j}\}$ such that the girth of the code $g = \min_j \{g_{v_j}\}$.

The construction of PEG consists of two procedures; the first is the expansion of the local graph, and the check node selection. The expansion of the variable node is to avoid short cycles when adding a new edge to the graph, by pruning the check nodes that will produce a short cycle or in case short cycles can't be avoided then only check nodes that will produce large cycles remain in the set as candidates. In the selection procedure is used to reduce the candidates according to the settings of the graph as degree of check nodes. The selection is done basically by choosing check nodes with the lowest degree.

PEG algorithm can construct regular and irregular LDPC codes with optimized performance. Codes obtained by PEG algorithm is known for the best performance in the case of short length codes [36]. A drawback of this algorithm is that it is not guaranteed to achieve bipartite graphs that has the largest possible girth with the given parameters n , m and D_v which leads to a problem of complexity in combinatorics. Thus PEG algorithm is considered as a sub-optimal algorithm in constructing high girth bipartite graphs. The performance of PEG LDPC codes is found to be better than codes constructed by MacKay's method having the same length in case of short codes [37].

Progressive Edge-Growth Algorithm:

for $j = 0$ to $n = 1$ **do**

begin

for $k = 0$ to $d_s = 1$ **do**

begin

if $k = 0$

$E_{v_j}^0 \leftarrow$ edge (c_i, v_j) , where $E_{v_j}^0$ is the first edge incident to v_j and c_i is a check node such that it has the lowest check-node degree under the current graph setting $E_{v_0} \cup E_{v_1} \cup \dots \cup E_{v_{j-1}}$.

else

expand a subgraph from symbol node v_j up to depth l under the current graph setting such that the cardinality of $N_{v_j}^l$ stops increasing but is less than m , or $\overline{N}_{v_j}^l \neq \Phi$ but $\overline{N}_{v_j}^{l+1} = \phi$, then $E_{v_j}^k \leftarrow$ edge (c_i, v_j) , where $E_{v_j}^k$ is the k^{th} edge incident to v_j and c_i is a check node picked from the set $\overline{N}_{v_j}^l$ having the lowest check-node degree.

end

end

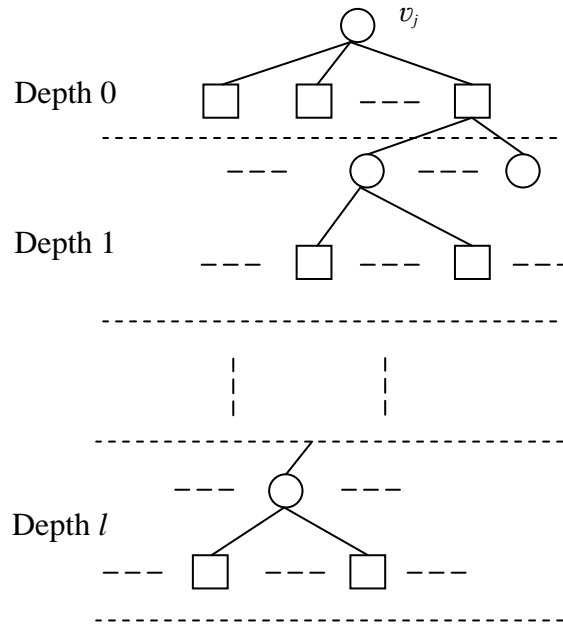


Figure 3.7: A tree that shows the depth l for a variable node v_j .

3.1.5 Protograph LDPC codes

The construction of protograph LDPC codes [38, 39] starts by having a relatively small bipartite graph that may include parallel edges called a protograph, which is used to obtain a large related graph by a copy-and-permute procedure. The protograph is copied Q times and then the edges are copied and connected among the copies of the protograph under some restrictions to obtain a large single Tanner graph of a new parity check matrix. Variable and check nodes are labeled so if a variable node V in the protograph is connected to a check node C , then variable node V in copies can only be connected to one check node C of the copies. The reason behind this is to preserve the decoding threshold properties of the protograph. The construction eliminates any parallel edges in the main protograph so a single Tanner graph that suits the constraints of parity check matrix can be obtained.

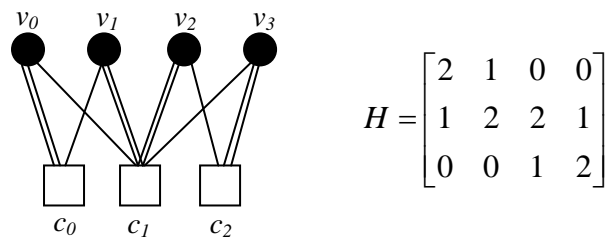


Figure 3.8: An example of a protograph and the corresponding base matrix.

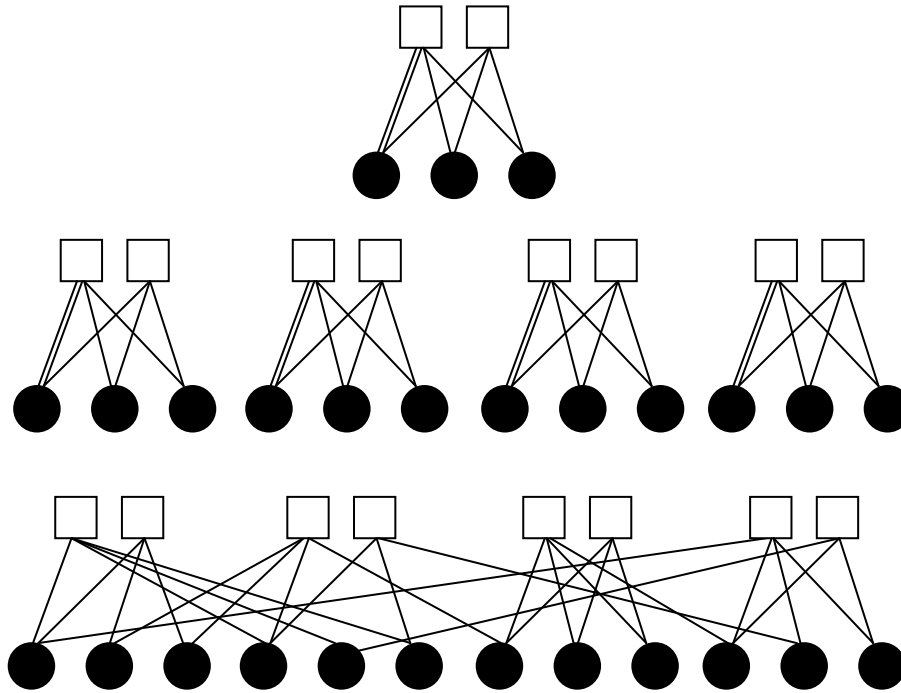


Figure 3.9: Illustration of the protograph copy and permute procedure with $Q = 4$ copies.

Figure 3.9 shows a protograph seed copied with $Q = 4$, the protograph seed is copied to 4 disconnected protographs of the same edges, the endpoints of the edges are permuted among the four copies at the same corresponding variable and check nodes. The resulted Tanner graph is called the derived graph and it is corresponding to a protograph LDPC code of length $N = 12$, $K = 4$ of rate $R = 1/4$. The protograph LDPC codes have many approaches and methods to optimize the construction of LDPC codes as in [40, 41].

3.2 Structured Construction of LDPC Codes

The uncertainty of guaranteeing an asymptotically optimum performance in random constructions leads to the use of structured construction of LDPC codes. Randomly constructed codes have some disadvantages as needing extra memory for the use of decoding and encoding. Also the randomness of construction affects the computational efficiency which is more crucial than the bit-error rate performance of the code. This leads to the need of some regularity in construction to overcome these problems.

In this type of LDPC constructions, predefined patterns of row-column connections are set following some constraints. Many methods have been developed as algebraic constructions, combinatorial designs, graph based constructions, and heuristic searching techniques.

3.2.1 Combinatorial Designs

Structured LDPC codes can be generated by constructing parity check matrix using combinatorial mathematics. A well structured, low complexity implemented codes using Balanced Incomplete Block Designs (BIBDs) [15] are designed by the inclusion of v points in b blocks according to some defined constraints. The two basic constraints are [42]:

- 1- A pair of points appear together only in λ blocks for a defined value of λ .
- 2- The number of points in each block is given by k and the number of blocks that a point appears at is r .

The construction is done by considering points and blocks and rows and columns, the design is balanced by having the covalency λ the same for all pair of points, from the definition, the dimensions of the code is given by $(b \times v)$. Row and column weights are given by k and r . The design parameters (v, k, λ, r, b) are considered and only three parameters of the five are independent, there for the notation (v, k, λ) -BIBD represents a BIBD code designed by v points with block size k and covalency λ .

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

Figure 3.10: The corresponding parity check matrix of a (9, 3, 1)-BIBD.

Example: LDPC code designed by (9, 3, 1)-BIBD.

Let $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and A the collection of 12 three-element blocks:

$A = \{(1, 2, 3), (4, 5, 6), (7, 8, 9), (1, 4, 7), (2, 5, 8), (3, 6, 9), (1, 5, 9), (2, 6, 7), (3, 4, 8), (1, 6, 8), (2, 4, 9), (3, 5, 7)\}$, the pair (X, A) is a (9, 3, 1)-BIBD.

The corresponding parity check matrix of the BIBD system is shown in figure 3.10.

The previous example is a Steiner triple system where $k = 3$, and $\lambda = 1$.

3.2.2 Euclidean Geometry LDPC code Construction (EG-LDPC)

Euclidean geometry is used to construct LDPC parity check matrix [16, 43] and it is similar to combinatorial design. Finite geometry is defined for n points intersecting by unique lines J with the conditions of the following structural properties.

- 1- Every line consists of ρ points.
- 2- Every point is intersected by γ lines.
- 3- Any two points are connected with one and only one line.
- 4- Any two lines are either disjointed, or they have only one point in common.

In this construction rows of H matrix represent lines and columns represent points, properties (1) and (2) ensure regularity of H matrix since two columns do not have more than one position with common '1's, and number of points in lines is constant. Property (3) keeps the girth at least 6 and avoids cycles of length 4. The sparsity of H can be obtained by choosing $\rho \ll n$ and $\gamma \ll J$ and thus the constructed matrix can be considered as a low density parity check matrix. The minimum distance of this construction can be calculated and lower bounded by $\gamma + 1$ in the case of decoding by one-step majority logic decoding since each bit has γ orthogonal check-sums. Compared to other types of construction as quasi-cyclic LDPC codes, finite geometry tends to have relatively large minimum distance.

A drawback of this type of construction is having a maximum girth of 6 [43] with existence of cycles of length 8 [44], which limits the improvement of the code in terms of optimizing and increasing the girth of the matrix. Another drawback of this construction is that the resultant H matrix is a square matrix of dimensions $N \times N$, so we have to choose $(N - K)$ rows to decode the code but this still degrade the performance of the code [45]. Also another drawback is that the row and column

weights are relatively large which increases the complexity of the decoding process. Also the structure of the code does not provide flexibility in the design to obtain wide range of lengths or rates.

In designing, one can omit the origin from the set of points, and lines intersecting the origin from the set of lines to obtain cyclic codes, which leads to lower-complexity linear time encoding [16].

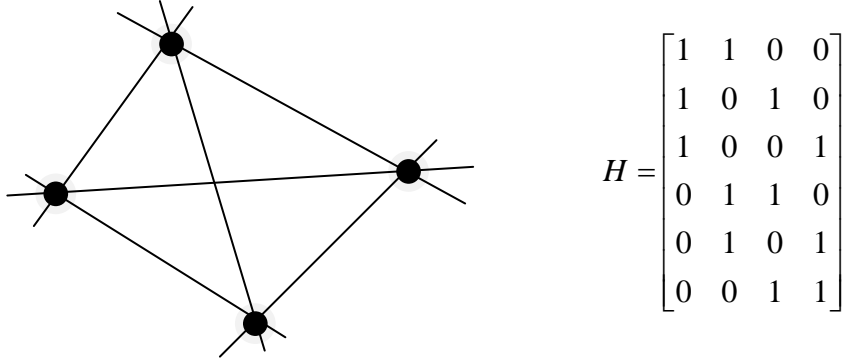


Figure 3.11: A graphical representation of a finite geometry with $\rho = 2$, $\gamma = 3$ and the corresponding incidence matrix.

LDPC codes constructed via Euclidian Geometry is denoted by EG-LDPC codes and defined over $\text{GF}(2^s)$ as $\text{EG}(m, 2^s)$. It consists of $N = 2^{ms}$ points represented by m-tuples. Number of lines in the geometry is $J = [2^{(m-1)s}(2^{ms} - 1)]/[2^s - 1]$, each point is intersecting $\gamma = (2^{ms} - 1)/(2^s - 1)$ lines that represents the column weight of the matrix. And each line includes $\rho = 2^s$ points which is the row weight of H . The density is given by $r = 2^{-(m-1)s}$. Table 3.1 shows variety of EG-LDPC codes found and listed in [26].

Design Parameters		Code length n	Code dim. K	Code Rate R	Minimum Distance	Rows in H J	Row Weight p	Col. Weight γ	Density of H $r = p/n$
m	s								
2	1	3	1	0.333	3	3	2	2	0.667
2	2	15	7	0.467	5	15	4	4	0.267
2	3	63	37	0.587	9	63	8	8	0.127
2	4	255	175	0.686	17	255	16	16	0.063
2	5	1,023	781	0.763	33	1,023	32	32	0.031
2	6	4,095	3,367	0.822	65	4,095	64	64	0.016
3	1	7	1	0.143	7	21	2	6	0.286
3	2	63	13	0.206	23	315	4	20	0.063
3	3	511	139	0.272	79	4,599	8	72	0.016
4	1	15	1	0.067	15	105	2	14	0.133
4	2	255	21	0.082	95	5,355	4	86	0.016
5	1	31	1	0.032	31	465	2	30	0.065
5	2	1,023	31	0.030	383	86,955	4	340	0.004
6	1	63	1	0.016	63	1,953	2	62	0.032
7	1	127	1	0.008	127	8,001	2	126	0.016

Table 3.1: A Summary of EG-LDPC codes.

Chapter 4

Constructing Quasi-Cyclic LDPC Codes

In the previous chapter, we reviewed different algorithms and methods to construct structured LDPC codes, these codes differ in their computational and implementation complexity. This belongs to the fact that structured codes may have many patterns for the interconnections between rows and columns in a single structured H matrix. The more row-column interconnection patterns the more storage needed for the decoder to store and manage.

Quasi-Cyclic (QC) LDPC codes are codes in which rows or columns in a sub matrix have similar and cyclic connections [29]. The structure of QC-LDPC codes allows them to be decoded using shift registers [46] and their decoders architectures require simple address generation mechanisms, less memory and localized memory accesses [47]. The construction of QC-LDPC codes is done by shifting identity sub-matrices. Numbers of cyclic shifts of columns in the identity sub-matrices are represented in a matrix which gives a compact representation of H matrix.

A linear code C is called a **Quasi-Cyclic (QC) LDPC code** (with circulant permutation matrices), if a parity-check matrix H of C has the following block form [48]:

$$H = \begin{bmatrix} I(c_{0,0}) & I(c_{0,1}) & \cdots & I(c_{0,L-1}) \\ I(c_{1,0}) & I(c_{1,1}) & \cdots & I(c_{1,L-1}) \\ \vdots & & \ddots & \vdots \\ I(c_{J-1,0}) & I(c_{J-1,1}) & \cdots & I(c_{J-1,L-1}) \end{bmatrix} \quad (4.1)$$

Let \check{H}_c denotes a matrix which consists of the indices of H , in other words,

$$\check{H}_c = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,L-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,L-1} \\ \vdots & & \ddots & \vdots \\ c_{J-1,0} & c_{J-1,1} & \cdots & c_{J-1,L-1} \end{bmatrix} \quad (4.2)$$

We call \check{H}_c the **model matrix** of H . It should be noted that a model matrix \check{H}_c characterizes a parity check matrix H of a quasi-cyclic LDPC code.

4.1 Algebraic construction of LDPC codes based on circulant matrices [49]

Quasi cyclic block codes can be constructed using multiplicative groups in term of integers mod m to refer to the number of circulations of identity matrices that construct various H matrix with a variety of length and rates.

The construction starts by choosing a prime number m , the element from 0 to $m - 1$ form a field under addition and multiplication (mod m), and thus the nonzero elements of this field represent a cyclic multiplicative group. Choose a and b to be nonzero elements with order of K and J respectively. And then form the P matrix with dimensions of $J \times K$ with elements from $GF(m)$ as the following:

$$P = \begin{bmatrix} 1 & a & a^2 & \cdots & a^{K-1} \\ b & ab & a^2b & \cdots & a^{K-1}b \\ \vdots & \vdots & \ddots & \cdots & \cdots \\ b^{J-1} & ab^{J-1} & a^2b^{J-1} & \cdots & a^{K-1}b^{J-1} \end{bmatrix} \quad (4.3)$$

Then a $(Jm \times Km)$ H matrix is constructed by having an $m \times m$ identity matrices (I_x) inserted with their rows circularly shifted to the left by $(x - 1)$ positions according to the values of P matrix as the following,

$$H = \begin{bmatrix} I & I_a & I_{a^2} & \cdots & I_{a^{K-1}} \\ I_b & I_{ab} & I_{a^2b} & \cdots & I_{a^{K-1}b} \\ \vdots & \vdots & \ddots & \cdots & \cdots \\ I_{b^{J-1}} & I_{ab^{J-1}} & I_{a^2b^{J-1}} & \cdots & I_{a^{K-1}b^{J-1}} \end{bmatrix} \quad (4.4)$$

The rate R of the constructed code is up to $1 - J/K$ due to the independence between some of resulting rows of H matrix. The number of '1's in columns and rows is J and K in sequence, this makes it a regular LDPC code. The construction also can be

extended to use nonprime integers with some modifications to sustain the regularity of the H matrix.

Block Length N	Design Parameters		Design Rate R	Actual Rate R	Circulant Size m
	J	K			
21	2	3	1/3	0.3809	7
93	2	3	1/3	0.3441	31
129	2	3	1/3	0.3411	43
155	3	5	2/5	0.4129	31
186	5	6	1/6	0.1882	31
305	3	5	2/5	0.4065	61
755	3	5	2/5	0.4423	151
905	3	5	2/5	0.4022	181
1055	3	5	2/5	0.4018	211
1205	3	5	2/5	0.4016	241
1477	3	7	4/7	0.5727	211
1477	5	7	2/7	0.2884	211
1703	5	13	5/8	0.6177	131
1928	3	8	5/8	0.626	241
1928	5	8	3/8	0.3771	241
1967	5	7	2/7	0.2877	281
2041	3	13	10/13	0.7702	157
2248	5	8	3/8	0.3768	281
2947	3	7	4/7	0.5721	421
2947	4	7	3/7	0.4296	421
3641	3	11	8/11	0.7278	331
3641	5	11	6/11	0.5465	331
5219	3	17	14/17	0.8239	307
11555	3	5	2/5	0.4001	2311

Table 4.1 [49]: Examples of QC-LDPC codes constructed from (prime) circulant sizes.

$$H = \begin{bmatrix} I_1 & I_9 & I_{81} & I_{115} & I_{114} & I_{105} & I_{24} & I_{216} & I_{102} & I_{304} & I_{280} & I_{64} & I_{269} & I_{272} & I_{299} & I_{235} & I_{273} \\ I_{17} & I_{153} & I_{149} & I_{113} & I_{96} & I_{250} & I_{101} & I_{295} & I_{199} & I_{256} & I_{155} & I_{167} & I_{275} & I_{19} & I_{171} & I_4 & I_{36} \\ I_{289} & I_{145} & I_{77} & I_{79} & I_{97} & I_{259} & I_{182} & I_{103} & I_6 & I_{54} & I_{179} & I_{76} & I_{70} & I_{16} & I_{144} & I_{68} & I_{305} \end{bmatrix}$$

Figure 4.1 [48]: A [5219, 4300] QC-LDPC code, constructed with $m = 307$, $a = 9$, and $b = 17$ where $o(a) = 17$ and $o(b) = 3$.

4.2 Construction of QC-LDPC Codes from Circulant Permutation Matrices by search algorithm

In his work, M. Fossier [19] has shown that codes with large girths are easy to obtain by QC-LDPC codes. He derived sufficient conditions to obtain these high girth codes using a search algorithm.

Starting by H matrix with rows and columns of weight J and L respectively, A number of circularly permuted $(J \times L)$ identity matrices of dimension $(p \times p)$ are inserted in the matrix to have a code length $N = Lp$. The first column and row are set to identity matrix, thus we only need $(J - 1)(L - 1)$ integers to describe the code.

The parity check matrix H of a (J, L) - regular QC-LDPC code of length $N = pL$ can be represented by:

$$H = \begin{bmatrix} I(0) & I(0) & \cdots & I(0) \\ I(0) & I(p_{1,l}) & \cdots & I(p_{1,L-1}) \\ \vdots & & \ddots & \vdots \\ I(0) & I(p_{J-1,l}) & \cdots & I(p_{J-1,L-1}) \end{bmatrix} \quad (4.5)$$

Where $1 \leq j \leq J - 1$, $1 \leq l \leq L - 1$, and $I(p_{j,l})$ is the Identity matrix with columns circularly shifted to the left by $p_{j,l}$ positions. And $I(0)$ represents the identity matrix, another approach is to set just the first row by identity matrices and other rows have circularly shifted identity matrices [50].

$$I(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad I(3) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 4.2: An example of Identity matrix of dimensions 4×4 with columns permuted 3 positions to the left and denoted by $I(3)$.

Defining:

$$\Delta_{j_x, j_y}(l) = p_{j_x, l} - p_{j_y, l} \quad (4.6)$$

A general rule for obtaining H matrix containing a cycle of length at least $2(i + 1)$ is giving by (4.7) if and only if:

$$\sum_{k=0}^{m-1} \Delta_{j_k, j_{k+1}}(l_k) \neq 0 \pmod{p} \quad (4.7)$$

for all m , $2 \leq m \leq i$, all j_k , $0 \leq j_k \leq J - 1$, all j_{k+1} , $0 \leq j_{k+1} \leq J - 1$, and all $0 \leq l_k \leq L - 1$, with $j_0 = j_m$, $j_k \neq j_{k+1}$, and $l_k \neq l_{k+1}$.

A condition for constructing QC-LDPC codes with girth $g \geq 6$ is given by (4.8) as:

$$p_{j_1, l_1} - p_{j_1, l_2} + p_{j_2, l_2} - p_{j_2, l_1} \neq 0 \pmod{p} \quad (4.8)$$

Where $0 \leq j_1 \leq j_2 < J$ and $0 \leq l_1 \leq l_2 < L$.

In the choosing of parameters J , L , and p , some conditions must be taken in consideration in order to achieve girth $\geq 6, 8, 10$ or 12 [19].

The indices of the model matrix can be found by a search algorithm as in [50], an example of constructing QC-LDPC codes is represented.

Example:

Consider the construction of H matrix with $J = 4$, $L = 6$, $p = 20$, given the following incomplete model matrix.

$$\check{H}_c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 16 & 20 & 7 & 8 & 6 & 4 \\ 9 & 14 & 4 & 10 & 13 & 12 \\ 13 & 5 & 10 & 2 & 15 & ? \end{bmatrix}$$

To find the missing entry the following calculations is set.

A random value $x = 11$ is picked from a vector of remaining values other than picked numbers in the current row, Now the algorithm checks the difference between this value and the previous elements in the same row with $\text{mod}(p)$. So in this case, by checking the condition (4.8) between the current row and the (current row - 1) we get:

$$(11 - 15) \text{mod}(20) = 16$$

$$(12 - 13) \text{mod}(20) = 19$$

$$\text{Reject} = 0$$

$$(11 - 2) \text{mod}(20) = 9$$

$$(12 - 10) \bmod(20) = 2$$

$$\text{Reject} = 0$$

$$(11 - 10) \bmod(20) = 1$$

$$(12 - 4) \bmod(20) = 12$$

$$\text{Reject} = 0$$

$$(11 - 5) \bmod(20) = 6$$

$$(12 - 14) \bmod(20) = 18$$

$$\text{Reject} = 0$$

$$(11 - 13) \bmod(20) = 18$$

$$(12 - 9) \bmod(20) = 3$$

$$\text{Reject} = 0$$

The flag 'Reject' returns 1 if the two values are equal. Now check the condition (4.8) between the current row and the (current row - 2).

$$(11 - 15) \bmod(20) = 16$$

$$(4 - 6) \bmod(20) = 18$$

$$\text{Reject} = 0$$

$$(11 - 2) \bmod(20) = 9$$

$$(4 - 8) \bmod(20) = 16$$

$$\text{Reject} = 0$$

$$(11 - 10) \bmod(20) = 1$$

$$(4 - 7) \bmod(20) = 17$$

$$\text{Reject} = 0$$

$$(11 - 5) \bmod(20) = 6$$

$$(4 - 20) \bmod(20) = 4$$

$$\text{Reject} = 0$$

$$(11 - 13) \bmod(20) = 18$$

$$(4 - 16) \bmod(20) = 8$$

$$\text{Reject} = 0$$

And the algorithm continues until the value satisfies condition (4.8). The result will be a matrix with girth $g = 6$ with dimensions 80 by 120 with 480 ones distributed regularly.

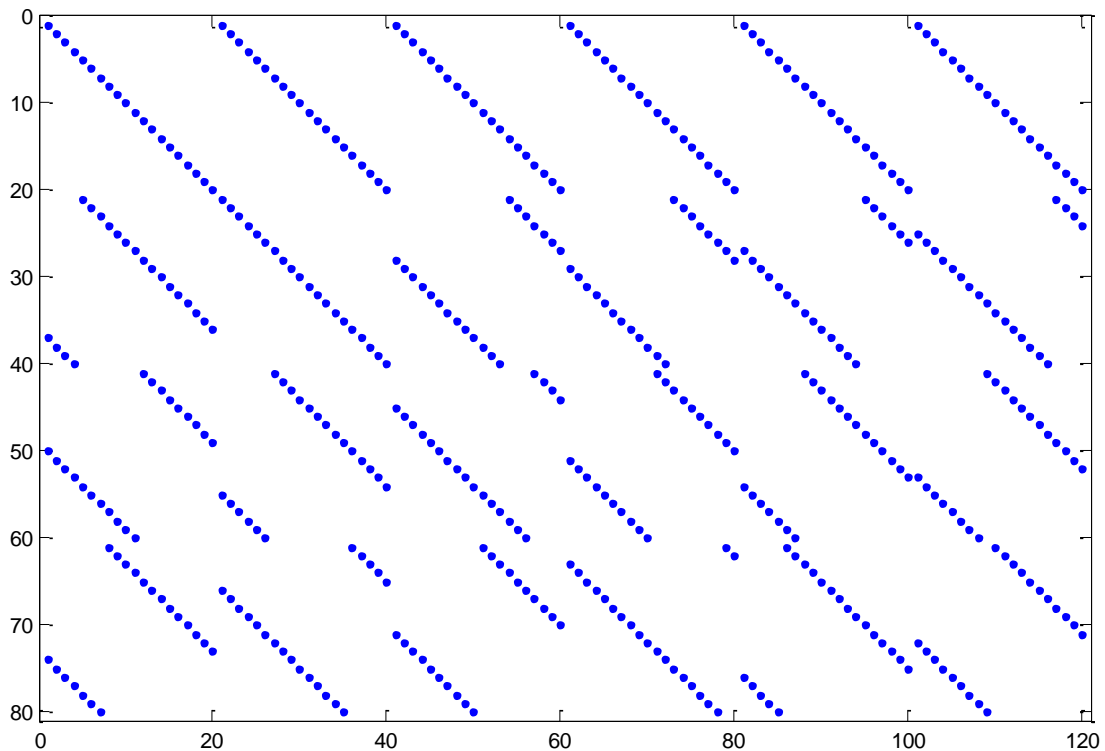


Figure 4.3: Parity Check Matrix of code (120, 40) of design rate 1/3
constructed with $J = 4$, $L = 6$, $p = 20$.

The rank of the generated parity check matrix is found to be in this case 77, thus the actual rate is in this case $R = 43/120 = 0.3583$.

There are some observations and comments about random construction of QC-LDPC codes based on column permutation method listed below:

- This method does not guarantee a full rank parity check matrix, thus it is hard to construct a code with specific length.
- The construction of the parity check matrix is not guaranteed from the first trial of running the algorithm; therefore the amount of time needed cannot be predetermined.
- Enlarging the block length for a given row and column weights can help avoiding the occurrence of overlapping of circulant matrices.

4.3 Design Parameters of QC-LDPC codes

QC-LDPC codes are characterized into two families, Random constructions as in 4.2 and structured constructions as in 4.1. In random construction LDPC codes, the smallest value of p to be chosen for a (J, L) regular QC-LDPC is found by computer search as in table 4.1. a theoretical limit was derived in [19] as the following: For codes with girth ≥ 6 a necessary condition for finding a code is $p \geq L$, or $N \geq L^2$ if L is odd, and $p \geq L + 1$ or $N \geq L(L - 1)$ if L is even. For obtaining a code with girth ≥ 8 a necessary condition is $p > (L - 1)(J - 1)$ or $N > (L - 1)(J - 1)L$.

J	L	4	5	6	7	8	9	10	11	12
3		5	5	7	7	9	9	11	11	13
4		-	5	7	7	9	10	11	11	13
5		-	-	7	7	9	10	11	11	13

Table 4.2: smallest value for p for a (J, L) - regular QC-LDPC code with girth $g \geq 6$ found by computer search [19].

Another important parameter is the minimum distance, in [51] the upper bound for the minimum Hamming distance of a (J, L) regular QC-LDPC code is given by $d_H \leq (J + 1)!$. Therefore, the minimum distance cannot increase with increasing the code length N , which leads to the suggestion that QC-LDPC codes are compared only to random codes of short and medium lengths.

The following figures 4.4-4.5 show the BER performance of different lengths and rates of QC-LDPC codes, the simulation shows the efficiency of QC-LDPC codes with short and moderate lengths.

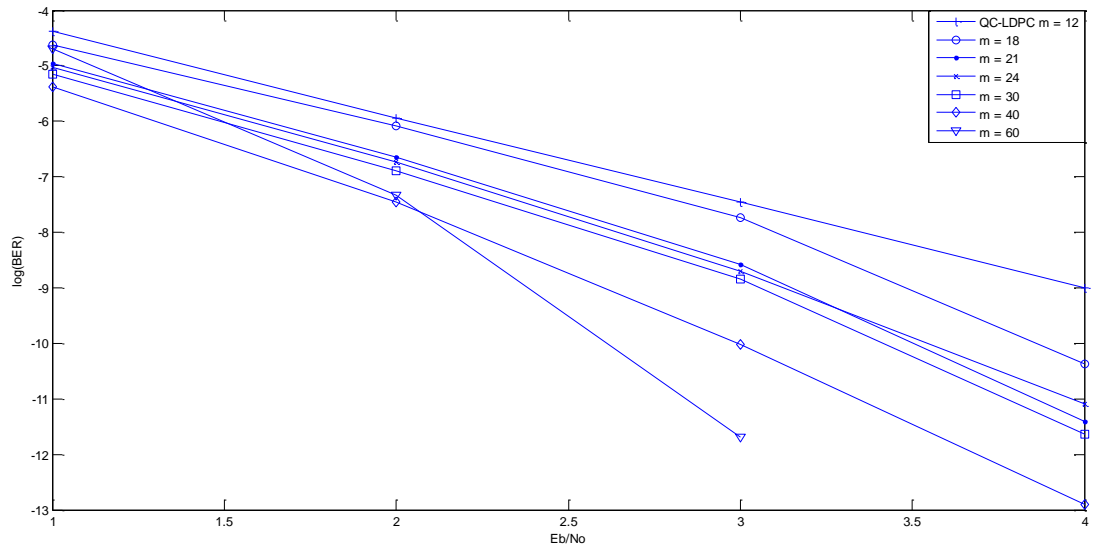


Figure 4.4: BER performance of QC-LDPC codes of different lengths with $R = 1/2$.

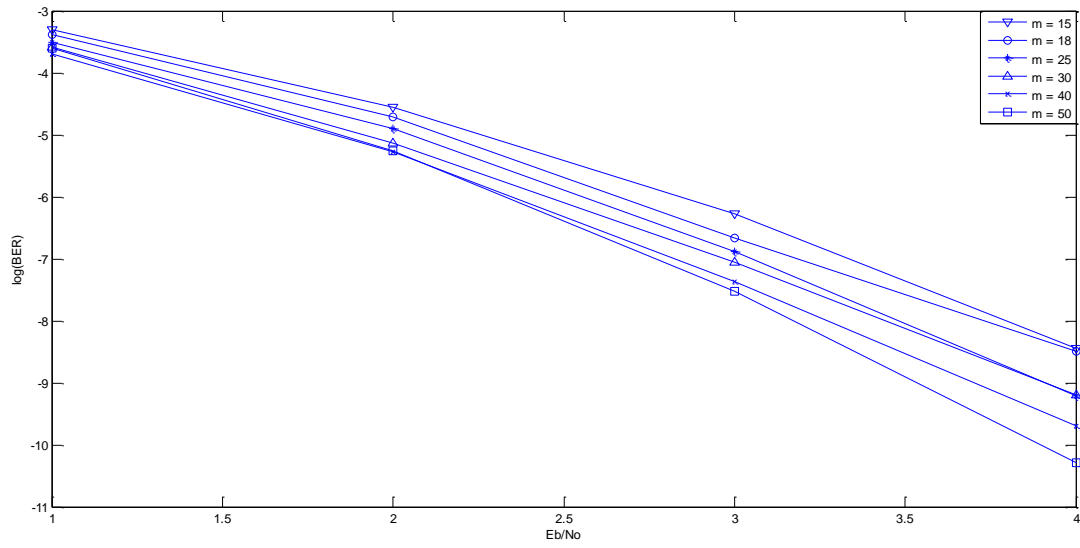


Figure 4.5: BER performance of QC-LDPC codes of different lengths with $R = 2/3$.

Chapter 5

LDPC code construction using randomly permuted copies of parity check matrix

In this chapter, a method to generate LDPC codes of multiple lengths starting from previously constructed codes is introduced. The idea starts from the same idea of QC-LDPC codes. In constructing QC-LDPC codes, we first construct the model matrix \check{H}_c where its entries refer to the number of shifts applied to an identity matrix placed at the same address in the parity check matrix. The design is introduced in the following sections.

5.1 Construction of LDPC code using identity seed matrix

In the following design, we consider having a base parity check matrix H_b of dimensions $N \times m$ represented by:

$$H_b = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,N-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,N-1} \\ \vdots & & \ddots & \vdots \\ c_{m-1,0} & c_{m-1,1} & \cdots & c_{m-1,N-1} \end{bmatrix} \quad (5.1)$$

H_b must satisfy the constraints of LDPC design, where $g \geq 6$, $\lambda = 0$ or 1 . Entries of H_b are binary data denoted by $c_{i,j}$ where $c_{i,j} \in \{0, 1\}$, $0 \leq j \leq m-1$ and $0 \leq i \leq N-1$

Definition 5.1: An identity sub-matrix I_p called *identity seed* and all-zeros square sub-matrix O_p with dimensions $p \times p$ are defined, where p is an integer and $p \geq 2$. The construction is done by constructing an all-zeros parity check matrix H of dimensions $Np \times mp$ divided into a number of (Nm) sub-matrices of dimensions $p \times p$, and then the base matrix H_b is used as a model matrix to construct H by replacing each 1 and 0 in H_b by I_p and O_p in H respectively. The resulted H has the same row and column weights of H_b .

In the following, we show an arbitrary example of the construction.

Example 5.1: Consider a (2, 3)-regular code of parity check matrix of dimensions (6 × 4) obtained from Euclidean geometry construction and represented by the parity check matrix:

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

This matrix defines the regular-LDPC code which has a length of $N = 6$ with a rate $R = 1/3$. Let us assume that we want to construct a regular code with length $N = 12$ of the same rate, thus we define I_2 and O_2 by:

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad O_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Now we replace each 1 by I_2 and each 0 by O_2 , and the resulted H matrix is:

$$H = \begin{bmatrix} I_2 & I_2 & I_2 & O_2 & O_2 & O_2 \\ I_2 & O_2 & O_2 & I_2 & I_2 & O_2 \\ O_2 & I_2 & O_2 & I_2 & O_2 & I_2 \\ O_2 & O_2 & I_2 & O_2 & I_2 & I_2 \end{bmatrix},$$

which is equivalent to:

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

The resultant matrix represents a (2, 3)-regular LDPC code of length of $N = 12$, and rate $R = 1/3$.

Theorem 5.1: A parity check matrix constructed by definition 5.1 has a girth equal to the girth of the base matrix H_b .

Proof: the girth is calculated by counting the straight edges that connect between '1's starting and ending at the same 1 entry in the parity check matrix, since the construction expands the parity check matrix with identity and all zeros matrices, then the construction do not change horizontal and vertical positions of the ones, nor adds ones between existed '1's in the matrix, where the length of the edges just increases by multiples equal to p . The insertion of identity seed results in new cycles of the same length which keeps the girth equal to the same girth in the base matrix.

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

Figure 5.1: A graphical representation of cycles of girth $g = 6$ resulted from the construction in definition 5.1.

Theorem 5.2: The rank of H is equal to $\text{Rank}(H_b) \times p$.

Proof: The rank of matrix H_b is defined by the maximum number of independent rows in the matrix, the construction of H in definition 5.1 keeps the positions of horizontal and vertical '1's fixed without adding '1's between the original '1's, which preserves the relation of dependency or independency between rows fixed. Since row reduction method operations between rows will result in the same rows in H_b but with zeros from the expansion added to the row. The shifted expanded copies of the original rows are independent of original rows which results in $(p - 1)$ copies of original rows independent from the original and have the same relations with their rows, which leads to the relation:

$$\text{Rank}(H) = p \cdot \text{Rank}(H_b)$$

Corollary 5.1: The rate R of H is equal to the rate R_b of H_b .

The rate of H_b is given by $R_b = (N - \text{Rank}(H_b))/N$, the rate of H is given by $R = (pN - p \cdot \text{Rank}(H_b))/(pN) = p(N - \text{Rank}(H_b))/(pN) = (N - \text{Rank}(H_b))/N$. Thus $R = R_b$.

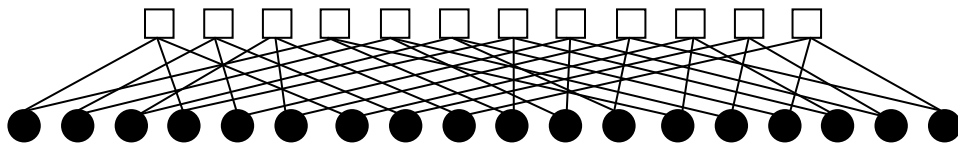
Sparsity of H :

The proposed construction keeps the degree of variable nodes and check nodes fixed. The insertion of the identity matrices leads to sparser H matrix. Suppose that in H_b the total number of '1's is given by e , and the total number of entries is s , then the density D_b of H_b is e/s . The number of '1's in H matrix is then equal to (pe) and the total entries is now equal to p^2s , the density of H is $pe/(p^2s) = e/(ps)$. So the relation between D and D_b is given by:

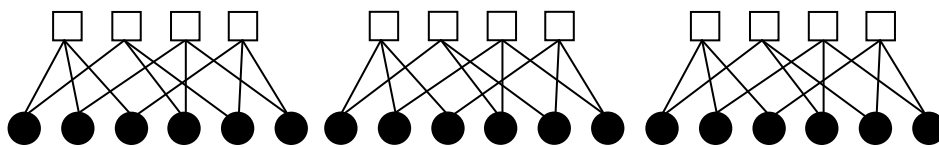
$$D = D_b/p$$

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

(a)



(b)



(c)

Figure 5.2: (a) Construction of H matrix with an identity seed of size 3×3 . (b) Corresponding Tanner graph of the H matrix. (c) The same Tanner graph in part (b) after rearrangement of nodes shows the separation of edges between the three copies.

5.2 Graphical Perspective

The proposed code construction is similar to protograph construction describe in 3.1.5. The corresponding Tanner graph results in p separated copies of the base matrix H_b . The copies are not connected by any edge between their check and variable nodes as shown in figure 5.2c. The advantage of this construction is decreasing the time of decoding to about $1/p$ of the decoding time, since each copy can be implemented individually and semi parallel with other copies as in [52]. The problem in this type of construction is in decoding process when a data bit is received at a variable node that belongs to one of the copies, the outgoing messages of that bit are sent only among its parent copy, and at the same time it receives messages only from the check nodes from the same parent copy, thus it cannot benefit from the information sent from bits in the other copies with strong probabilities of receiving a correct bit. Also the girth average of variable nodes is not changed where increasing the number of columns and rows with decreasing the density of ones in the matrix is expected to increase the girth average of its Tanner graph, where increasing the girth average enhances the decoding performance [31].

5.3 Construction of LDPC code using randomly permuted identity seed matrix

To solve the problem of independency between copies we introduce an enhancement in definition 5.2 that interchanges the connections of edges between all variable nodes and check nodes all over the copies and increases the girth average of the code.

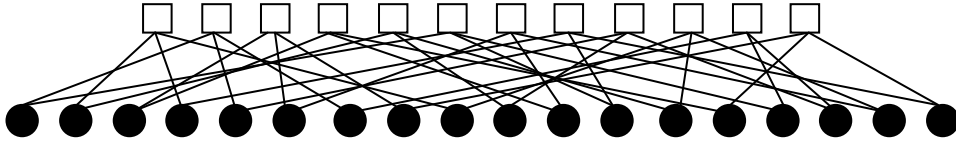
Definition 5.2: The identity seed matrix I_p is replaced by a randomly permuted identity matrix of dimensions $p \times p$ with regular column and row weights of 1, in other words each row contains 1 in a unique random column. And each 0 in the base matrix is replaced as in definition 5.1.

The graphical correspondence of permutations in definition 5.2 is that for a variable node V_1 connected to check node C_1 by edge E , the edges of the p copies of V_1 are permuted across the p copies of C_1 where each copy of V_1 is connected with only one copy of C_1 .

The proposed construction can be represented by a model matrix since each 1 in the original matrix is replaced by a randomly permuted identity matrix $I_p(r)$ called *random seed* selected from a space of $p!$ different matrices, where $0 \leq r \leq p! - 1$. Each permuted identity matrix is assigned a number r which indicates which $I_p(r)$ to be replaced.

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

(a)



(b)

Figure 5.3: (a) H matrix of LDPC code of length $N = 18$, $R = 1/3$, with $j = 2$ and $k = 3$ constructed by definition 5.2. (b) The corresponding Tanner graph of H .

Figure 5.4 shows an example of a 6 random seeds generated by choosing $p = 3$, each assigned by a number, and figure 5.5 shows the model matrix for H in figure 5.3

$$I_3(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, I_3(1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, I_3(2) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$I_3(3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, I_3(4) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, I_3(5) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Figure 5.4: The 6 possible random seeds generated when $p = 3$.

$$\check{H}_c = \begin{bmatrix} I_3(4) & I_3(0) & I_3(2) & O_3 & O_3 & O_3 \\ I_3(5) & O_3 & O_3 & I_3(4) & I_3(4) & O_3 \\ O_3 & I_3(2) & O_3 & I_3(1) & O_3 & I_3(2) \\ O_3 & O_3 & I_3(2) & O_3 & I_3(3) & I_3(4) \end{bmatrix}$$

Figure 5.5: The model matrix \check{H}_c for H in figure 5.3a.

The proposed construction differs from protograph LDPC code construction is the ability of storing the addresses of permuted matrices and calling the addresses by a simple address generation mechanism from the model matrix. This helps in reducing the memory for storing the $m \times N$ matrix. Another advantage of construction from definition 5.2 is when constructing starting by a bad seed matrix generated with a random method containing 4 cycles, the construction can reduce the number of removed bit resulted by any method of removing loops algorithms [53] in order to enhance the decoding performance of the code as illustrated in the following theorem..

Theorem 5.3 The Girth of code constructed by definition 5.3 is greater than or equal to the girth of the base matrix.

Proof: The girth of the proposed code is guaranteed to be greater than or equal to the girth of the base matrix, this belongs to the fact that the insertion of random seeds in the base matrix, adds cycles of the same length of the existed cycles in the base matrix distributed by the randomness of the positions of 1's in each seed matrix which allows the path of old cycles to pass over the possible position of old ones to construct larger cycles. So the random distribution of ones in each seed results in enlarging the length of cycles in corresponding Tanner graph and keeps the girth of the base matrix which is the reason why the proposed construction can reduce number of 4 cycles in bad base matrices.

Theorem 5.4: The memory storage needed for storing entries of a LDPC parity check matrix constructed by random seed method is equal to $1/p^2$ of the memory storage needed for entries of a random LDPC code with the same length.

Proof: Suppose a base H matrix with dimensions $a \times b$ is designed and a seed of dimension $p \times p$ is inserted to construct a LDPC code by definition 5.2 that has the

same length of a random LDPC code. The new number of entries is then $(ap)(bp) = abp^2$. The model matrix has a number of entries equal to ab . Thus the memory storage needed for the model matrix is $1/p^2$ of entries of H .

5.4 Construction of LDPC code using seeds of circulant identity matrices

The proposed construction proposed in definition 5.2 helps in reducing the memory to store and construct random entries of the LDPC H matrix by a factor of $1/p^2$. An example is when a construction of a LDPC code with of length $N = 3000$ and rate $R = 1/2$ is done using a random seed with $p = 100$, it results in a model matrix with dimensions of 15 by 30 instead of 1500 by 3000, and the construction can be achieved by a simple random generation of numbers that refers to a predefined random seeds. It also preserves the performance of same length code constructed by the same method the base matrix is constructed by. A drawback of the proposed construction in definition 5.2 is when choosing a large seed matrix in order to get a long length code from a small base matrix, the number of all possible random seeds of dimensions $p \times p$ is growing to exceed the number of 1's in the base matrix. And each 1 in the base matrix has the possibility to be replaced with a different seed matrix, which leads to storing a number of seeds that are probably greater than the number of ones in the sparse base matrix, and that reduces the efficiency of the code design. Recalling the base matrix represented in example 5.1, if p is chosen to be 4, the number of possible matrices to be generated is $4! = 24$, but the number of matrices to be used is upper-bounded by the total number of ones in the base matrix which is 12 in this case, so to overcome this problem we can limit the number of seeds by the total number of 1's in the base matrix in order not to store unused seeds. Another problem in this type of construction appears when using a large p seeds with a relatively large base matrix with many 1's entries, in this case the random generation will result in too many different seeds to be stored, and the advantage of reducing the memory space of H matrix will be violated by the large number of stored seeds. So it is recommended to use of codes from definition 5.2 only when the dimensions of base matrix are small with large p seeds, or when the base matrix is large using small p seeds

In order to overcome the restrictions in constructing codes by definition 5.2, we suggest decreasing the number of choices to be equal to p instead of $p!$ by taking the

seed matrix I_p with its $p - 1$ circulant matrices to be the only available choices, the insertion of only p circulant matrices turns the code into a QC-LDPC code similar to [54]. The randomness in choosing the seed to be inserted keeps the girth of the constructed code greater than or equal to the original girth of the base matrix as in theorem 5.2. The quasi cyclic property of the code allows linear encoding with shift registers [55, pp. 256–261]. The performance of the QC-LDPC codes generated by this method is found to be similar in BER performance of codes from definition 5.2.

5.5 Simulation and Results

In this section, we represent the performance results for the proposed LDPC codes by comparing the BER of codes designed by the proposed construction with codes from random constructions such as Gallager codes and with classic random QC-LDPC codes. In additive white Gaussian noise (AWGN) channel, we use Implementation-efficient Reliability Ratio Based Weighted Bit-Flipping (IRRWBF) [56] to decode LDPC codes. And all simulations use maximum iteration number of 80. In the first example we compare the performance of a LDPC Gallager code (504, 3, 6) with a LDPC code constructed using a base matrix H_b constructed by a LDPC Gallager code with (168, 3, 6) and a random seed with $p = 3$. The resulted H matrix is of a LDPC code of length $N = 504$ of the same rate. It can be seen in figure 5.6 and figure 5.7 that the BER performance of the proposed codes is very close to Gallager's and MacKay's codes. Since both of constructions have similar BER performance, the proposed code still has an advantage over random codes such as Gallager's and MacKay's is that it reduces the memory usage by having addresses of each random seed. The construction with identity seeds shows degradation of the performance of the extended code.

The second example represents a comparison between the proposed code with rate $R = 1/2$, $j = 3$, $k = 6$ constructed with classic random QC-LDPC base matrix of $m = 9$ with a seed $I_3(r)$ to give a block length $N = 162$, and with a classic random QC-LDPC of the same length with circulant identity matrices with $m = 27$. The results in figure 5.7 show that proposed codes constructed by a QC-LDPC base, outperform the original code constructed with the same QC-LDPC matrix.

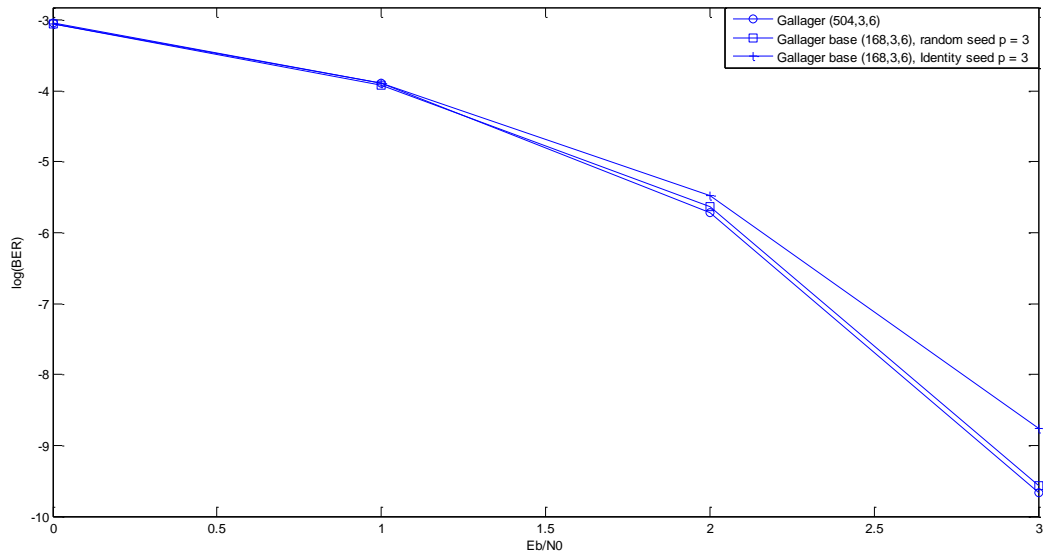


Figure 5.6: BER performance of $(504, 3, 6)$ proposed code designed with base of Gallager H with random seed of $p = 3$, and with an identity seed of $p = 3$, compared to Gallager code with length $N = 504$.

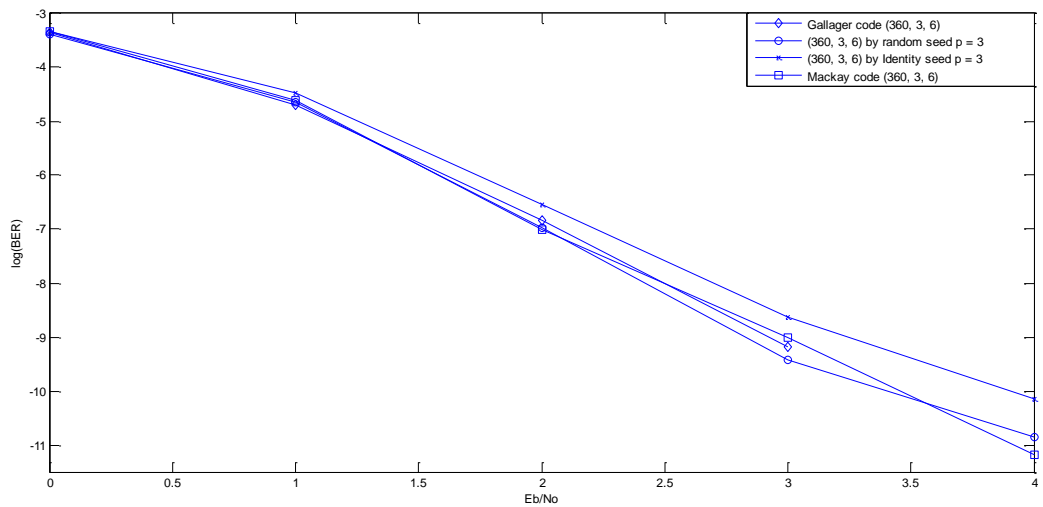


Figure 5.7: BER performance of proposed codes with $(504, 3, 6)$, with random seed of $p = 3$, and Mackay's and Gallager's codes of the same length.

The proposed code also outperforms the classic QC-LDPC codes with different rates and block lengths as shown in figure 5.8 and figure 5.9.

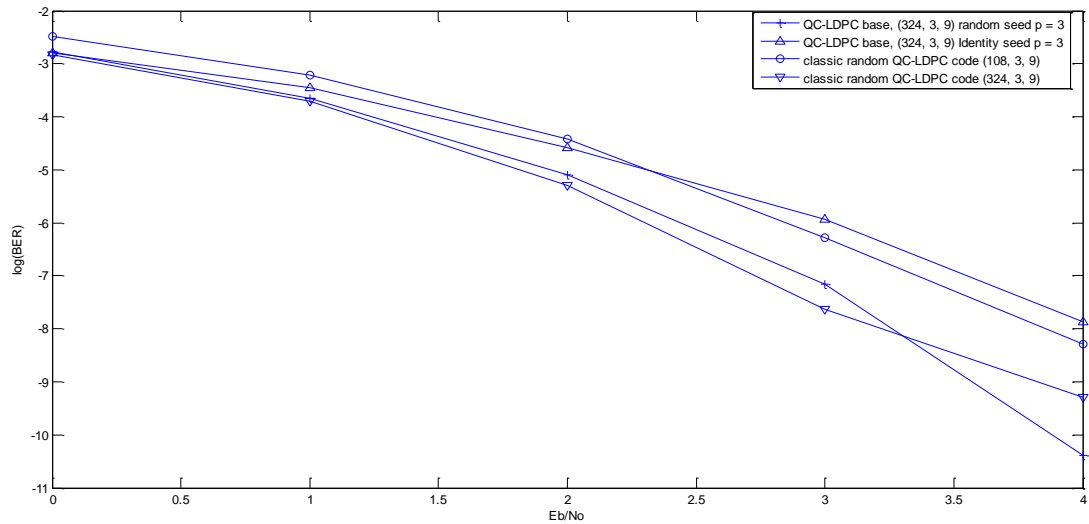


Figure 5.8: BER performance of proposed codes with rate $R = 2/3$ and a classic random QC-LDPC code with the same rate and block length $N = 324$.

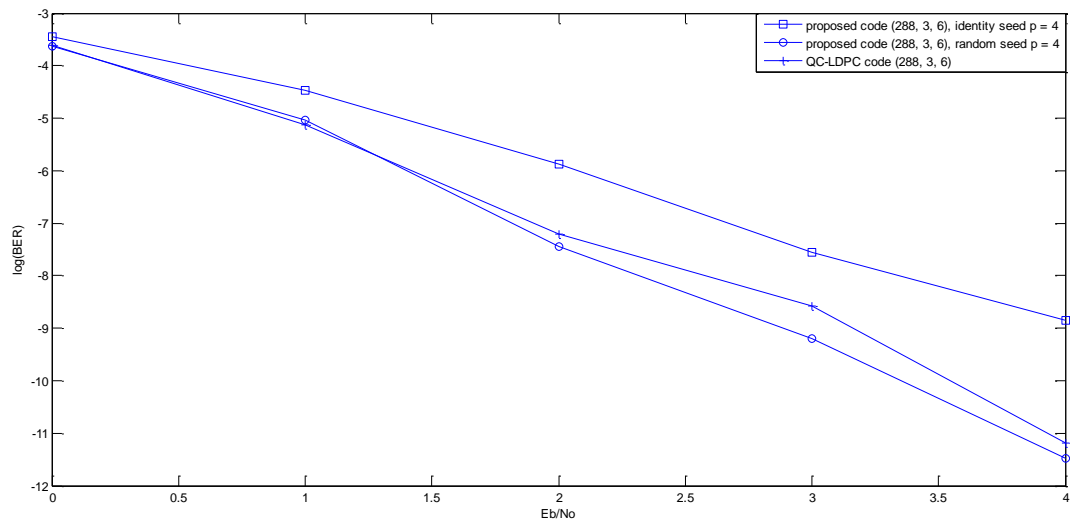


Figure 5.9: BER performance of proposed codes with rate $R = 1/2$ and a classic random QC-LDPC code with the same rate and block length $N = 288$.

The following figure shows the effect of proposed code on decreasing the number of 4 cycles in a bad constructed H_b , the removing of bits in order to get a free 4 cycle graph degrade the decoding performance, so removing less ones with respect to the constructed size of H reduce the degrading in decoding performance.

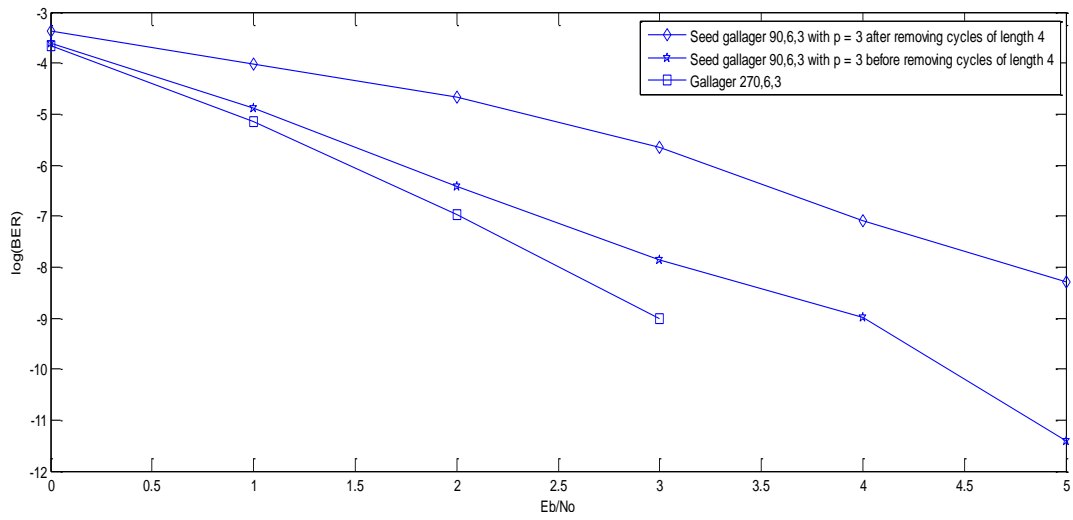


Figure 5.10: BER performance of (90, 6, 3) proposed codes, shows the effect of reducing 4 cycles in corresponding Tanner graph of H matrix.

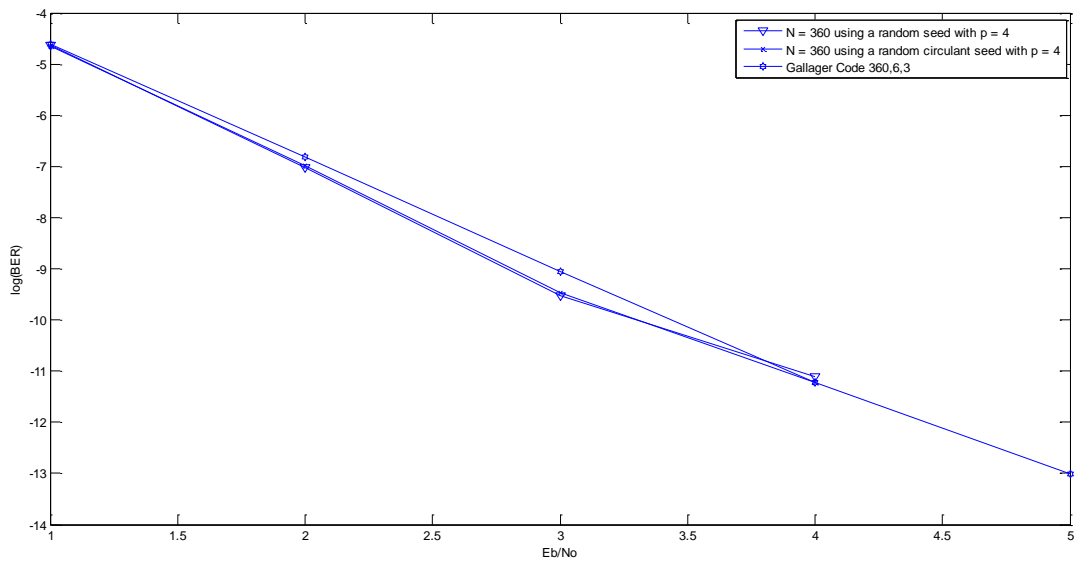


Figure 5.11: BER performance of proposed QC-LDPC code with $N = 360$ and code with random seeds, compared to Gallager code.

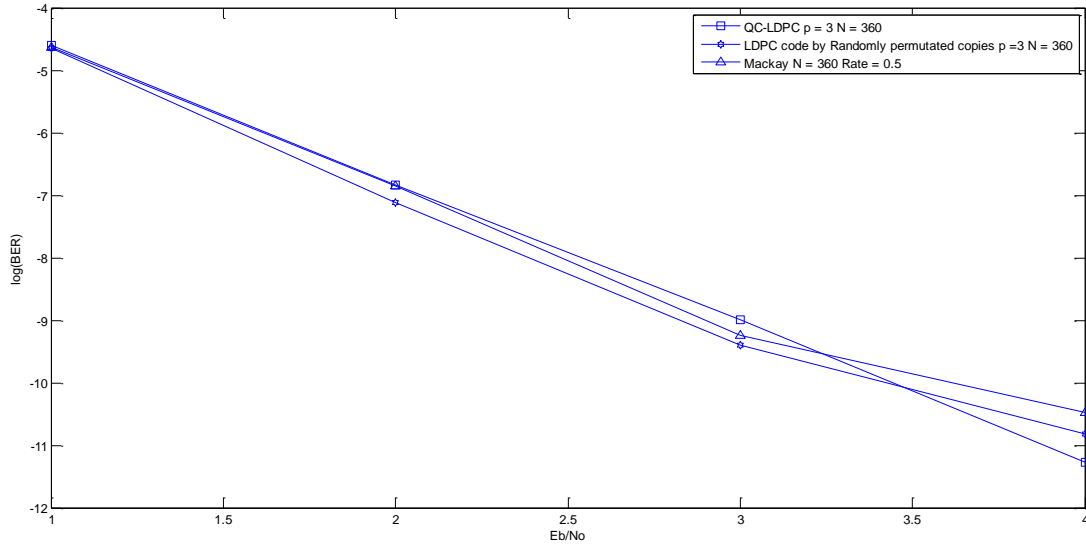


Figure 5.12: BER performance of proposed QC-LDPC code with $N = 360$ and code with random seeds, compared to MacKay code.

Figure 5.11 shows the performance of proposed QC-LDPC code compared to code generated by definition 5.2. Both constructions appear to have the same BER performance, taking in mind the advantage of QC-LDPC codes over the other proposed design in encoding process and in decreasing the number of stored seed matrices. Figure 5.12 shows comparison between constructions using a seed of MacKay code.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The main subject of this thesis is to study various construction methods of LDPC codes and to understand the advantages and disadvantages of each type. The main issue between random construction methods and structured methods is the trade-off between the need of high memory storage for randomly constructed codes and the easy implementation of structured codes, taking in account the outperforming of random codes in large block lengths over structured codes.

Thus, the objective that comes up is to construct LDPC codes that have good BER performance, and are also easy to be implemented in hardware. We achieved our objectives by developing new LDPC code construction method that reduces the memory usage for storing H matrix by a factor of $1/p^2$ and performs similar to fully random codes such as Gallager's and MacKay's. Also experimental results show that the proposed codes outperform classic random QC-LDPC codes. Another advantage is that it can reduce the number of ones to be removed in order to get rid of 4 cycles which degrade the decoding performance. We constructed LDPC codes with various lengths and rates and we show that the proposed method works well in designing block type LDPC codes.

6.2 Future work

The proposed code showed that it performs as good as random codes. Further analysis of the obtained codes is needed to improve the performance in terms of girth, and how to improve girth of the constructed codes. Also analysis of the obtained codes in terms of minimum distance is required. Also further BER simulations at different lengths and rates will be necessary to evaluate the stability of performance of these codes for some applications.

A detailed study of hardware implementations of the proposed code is necessary to have a better comparison with different constructions.

Bibliography

- [1] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423 (Part I) and 623-656 (Part II), 1948.
- [2] R. Hamming, "Error detecting and error correcting codes," *Bell Systems Technical*, vol. 29, pp. 147-160, 1950.
- [3] S. Lin and D. Costello, *Error-Control Coding: Fundamentals and Applications*. 2nd ed. Upper Saddle River, NJ: Pearson/Prentice-Hall, 2004.
- [4] R. Gallager, "Low-Density Parity Check Codes," Cambridge, MA: MIT Press, 1963.
- [5] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, pp. 21-28, Jan. 1962.
- [6] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, pp. 599-618, 2001.
- [7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo codes," *Proc. IEEE Intl. Conf. Commun. (ICC 93)*, pp. 1064-1070, 1993.
- [8] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, pp. 533-547, 1981.
- [9] D. MacKay and R. Neal, "Near Shannon limit Performance of Low-Density Parity-Check Codes," *Electron. Lett.*, vol. 32, pp. 1645-1646, Aug. 1996.
- [10] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," *Proc. 29th Symp. Theory of Computing*, pp. 150-159, 1997.

- [11] S. Chung *et al.*, “On the design of low density parity-check codes within 0.0045 dB of the Shannon limit,” *IEEE Comm. Lett.*, vol. 5, pp. 58-60, Feb. 2001.
- [12] M. Davey and D. MacKay, “Low density parity check codes over GF(q),” *IEEE Commun. Lett.*, vol. 2, pp. 165–167, June 1998.
- [13] Q. Huang *et al.*, “Quasi-cyclic LDPC codes: an algebraic construction,” *IEEE Trans. Commun.*, vol. 58, pp. 1383-1396, May 2010.
- [14] M. Luby *et al.*, “Improved Low-Density Parity-Check Codes Using Irregular Graphs,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 585-598, Feb. 2001.
- [15] B. Vasic and O. Milenkovic, “Combinatorial constructions of low-density parity-check codes for iterative decoding,” *IEEE Trans. Inf. Theory*, vol. 50, pp. 1156-1176, June 2004.
- [16] Y. Kou, S. Lin, and M. Fossorier, “Low-Density Parity-Check Codes Based on Finite Geometries: A Rediscovery and New Results,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 2711-2736, Nov. 2001.
- [17] J. Campello, D. Dodha, and S. Rajagopalan, “Designing LDPC codes using Bit-Filling,” *Proc. Int. Conf. Communications (ICC)*, Helsinki, Finland, 2001.
- [18] X.Y. Hu, E. Eleftheriou, and D.M. Arnold, “Progressive edge growth Tanner Graphs,” *IEEE Global Telecommunications Conf. 2001*, vol. 2, pp. 995-1001, Nov. 2001.
- [19] M. Fossorier, “Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices,” *IEEE Trans. on Inf. Theory*, vol. 50, pp. 1788-1793, Aug. 2004.
- [20] D. MacKay and R. Neal, “Good codes based on very sparse matrices,” *Proc. IMA Conf. Cryptography, Coding*, vol. 1025, pp. 100-111, 1995.

- [21] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, pp.638-656, Feb. 2001.
- [22] S.B. Wicker and S. Kim, "*Fundamentals of Codes, Graphs and Iterative Decoding*," Kluwer Academic Publishers, Norwell, MA, 2003.
- [23] F. Kshinschang, B. Frey, and H. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498-519, Feb. 2001.
- [24] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999.
- [25] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, pp. 429- 445, Mar. 1996.
- [26] J. Attili, "An Investigation of Low Density Parity Check Codes," San Diego State Univ. Report, May 2006.
- [27] D. MacKay and M. Davey, "Evaluation of Gallager Codes for Short Block Length and High Rate Applications," *Proc. IMA Workshop on Codes, Systems and Graphical Models*. vol. 123, pp. 113-130, Aug. 1999.
- [28] G. Malema, and M. Liebelt, "Low Complexity Regular LDPC codes for Magnetic Storage Devices," *World Academy of Science, Engineering and Technology*. July 2005.
- [29] G. Malema, "Low density Parity-Check Codes: Construction and Implementation," Ph.D. dissertation, Faculty of Eng. Comp. and Math. Sci., Univ. of Adelaide, Australia, 2007.
- [30] M. O'Sullivan, "Algebraic Construction of Sparse Matrices With Large Girth," *IEEE Trans. Inf. Theory*, vol. 52, pp. 718-727, Feb 2006.

- [31] Y. Mao and AH Banihashemi, "A Heuristic Search for Good Low-Density Parity-Check Codes at Short Block Lengths," *Proc. IEEE Int. Conf. Commun.*, Helsinki, Finland, June 2001.
- [32] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, pp. 619-637, 2001.
- [33] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 399-431, 1999.
- [34] J. Knudsen, "Randomised Construction and Dynamic Decoding of LDPC Code," M.S. thesis, dep. Inf., Univ. Bergen. 2005.
- [35] N. Traore, S. Kant, and T. Jensen, "Message Passing Algorithm and Linear Programming Decoding for LDPC and Linear Block Codes," Faculty Eng. Sci., Aalborg Univ., 2007.
- [36] D. MacKay, S. Wilson, M. Davey, "Comparison of Construction of Gallager codes," *IEEE Trans. Comm.*, vol. 47, pp. 1449-1454, Oct. 1999.
- [37] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, pp. 386-298, Jan. 2005.
- [38] J. Thorpe, "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs," JPL INP, Tech. Rep., pp. 42-154, Aug. 2003.
- [39] T. Richardson, "Multi-Edge Type LDPC Codes," presented at the Workshop honoring Prof. Bob McEliece on his 60th birthday (but not included in the proceedings), California Institute of Technology, Pasadena, California, May 24-25, 2002.

[40] K. Wang, Y. Xiao, and K. Kim, "Construction of protograph LDPC codes with circular generator matrices," *Journal of Systems Engineering and Electronics*, vol. 22, no. 5, pp. 840-847, Oct. 2011.

[41] D. Mitchell, R. Smarandache, M. Lentmaier, and D. Costello, "Quasi-Cyclic Asymptotically Regular LDPC Codes," Proc. IEEE Inf. Theory Workshop (ITW), 2010.

[42] S. Johnson, S. Weller,
"Construction of low-density parity-
check codes from Kirkman triple
systems," *Proc. IEEE Globecom
Conf.*, vol. 2, pp. 970-974, Nov
2001.

[43] Y. Kou, S. Lin, and M. Fossorier, "Low-Density Parity-Check Codes: Construction based on finite Geometry," *IEEE Globecom conf. 2000*, San Francisco, CA, vol. 2, no.7, pp. 825-829, Nov. 2000.

[44] M. Flanagan *et al.*, "A Euclidean Geometry Based Algebraic Construction Technique for Girth-8 Gallager LDPC Codes," Proc. IEEE Inf. Theory Workshop (ITW), 2006.

[45] W. Ryan, "An introduction to LDPC codes," Dep. Elect. Comp. Eng., Univ. Arizona, Aug. 2003.

[46] H. Fujita and K. Sakaniwa, “Some Classes of Quasi-Cyclic LDPC Codes: Properties and Efficient Encoding Methods,” *IEICE Trans. Fundam. Electr. Commun. Comp. sci.*, vol. 88, pp. 3627-3635, 2005.

[47] Y. Chen and K. Parhi, “Overlapped Message Passing For Quasi-Cyclic Low Density Parity Check Codes,” *IEEE Trans. Circuits syst.*, vol. 51, pp. 1106-1113, June 2004.

[48] M. Hagiwara and H. Imai, “Quantum Quasi-Cyclic LDPC Codes,” *Proc. ISIT'07*, pp. 806-811, Nice, June 2007.

[49] R. Tanner *et al.*, “LDPC Block and Convolutional Codes Based on Circulant Matrices,” *IEEE Trans. Inf. Theory*, vol. 50, pp. 2966– 2984, Dec. 2004.

[50] G. Malema, “Constructing Quasi-Cyclic LDPC codes Using a Search Algorithm,” *Signal Processing and Information Technology. IEEE Int. Symp.*, pp. 956-960, 2007.

[51] D. MacKay and M. Davey, “Evaluation of Gallager codes for short block length and high rate applications,” *Proc. IMA Workshop Codes, Systems and Graphical Models*, Aug. 1999.

[52] J. Lee *et al.*, “A Scalable Architecture of a Structured LDPC Decoder,” *Proc. Int. Symp. Information Theory*, June 27-July 2, 2004.

[53] J. McGowan and R. Williamson, “Removing Loops from LDPC Codes,” *Australian Commun. Theory Workshop Proceedings*, 2003.

[54] R. Tanner, “On Graph Constructions for LDPC Codes by Quasi-Cyclic Extension,” in *Itgomlation, Coding and Matzematics* (M. Elaum, P. Farrell, and H. van Tilborg, eds.), pp. 209-220, Kluwer, June 2002.

[55] W. Peterson and E. Weldon, *Error-Correcting Codes*. 2nd ed. Cambridge, MA: MIT Press, 1972.

[56] C. Lee, and W. Wolf, “Implementation-efficient reliability ratio based weighted bit-flipping decoding for LDPC codes,” *Electronics Letters*, vol. 41, no. 13, pp. 755-757, 23 June 2005.