COMPARISON OF DECODING ALGORITHMS FOR LOW-DENSITY PARITY-CHECK CODES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERT KOLAYLI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet ERKMEN

Head of Department

This is to certify that i have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Melek D. YÜCEL

Supervisor

Examining Committee Members

Prof. Dr. Yalçın TANIK (METU, EEE) _____

Assoc. Prof. Dr. Melek D. YÜCEL (METU, EEE) _____

Assoc. Prof. Dr. T. Engin TUNCER (METU, EEE) _____

Asst. Prof. Dr. Ali Özgür YILMAZ (METU, EEE) _____

Asst. Prof. Dr. Murat H. SAZLI (Ankara University, EE) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required, I have fully cited and referenced all material and results that are not original to this work.

Name Lastname : Mert KOLAYLI

Signature :

ABSTRACT

COMPARISON OF DECODING ALGORITHMS FOR LOW-DENSITY PARITY-CHECK CODES

KOLAYLI, Mert
M.Sc., Department of Electrical and Electronics Engineering
Supervisor : Assoc. Prof. Dr. Melek D. YÜCEL
September 2006, 53 pages

Low-density parity-check (LDPC) codes are a subclass of linear block codes. These codes have parity-check matrices in which the ratio of the non-zero elements to all elements is low. This property is exploited in defining low complexity decoding algorithms. Low-density parity-check codes have good distance properties and error correction capability near Shannon limits.

In this thesis, the sum-product and the bit-flip decoding algorithms for low-density parity-check codes are implemented on Intel Pentium M 1,86 GHz processor using the software called MATLAB. Simulations for the two decoding algorithms are made over additive white gaussian noise (AWGN) channel changing the code parameters like the information rate, the blocklength of the code and the column

weight of the parity-check matrix. Performance comparison of the two decoding algorithms are made according to these simulation results.

As expected, the sum-product algorithm, which is based on soft-decision decoding, outperforms the bit-flip algorithm, which depends on hard-decision decoding. Our simulations show that the performance of LDPC codes improves with increasing blocklength and number of iterations for both decoding algorithms. Since the sum-product algorithm has lower error-floor characteristics, increasing the number of iterations is more effective for the sum-product decoder compared to the bit-flip decoder. By having better BER performance for lower information rates, the bit-flip algorithm performs according to the expectations; however, the performance of the sum-product decoder deteriorates for information rates below 0.5 instead of improving. By irregular construction of LDPC codes, a performance improvement is observed especially for low SNR values.

Keywords: LDPC codes, sum-product algorithm, bit-flip algorithm, iterative decoding algorithms

ÖZ

DÜŞÜK YOĞUNLUKLU EŞLİK DENETİM KODLARI İÇİN KOD ÇÖZME ALGORİTMALARININ KARŞILAŞTIRILMASI

KOLAYLI, Mert
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Melek D. YÜCEL
Eylül 2006, 53 sayfa

Düşük yoğunluklu eşlik denetim (DYED) kodları doğrusal blok kodların bir alt sınıfıdır. Bu kodların eşlik denetim matrislerinde sıfırdan farklı olan elemanların sayısı tüm elemanların sayısına oranla daha küçüktür. Bu özellik sayesinde düşük yoğunluklu eşlik denetim kodları için karmaşık olmayan kod çözme algoritmalarının kullanılabilmesi mümkün olmaktadır. Düşük yoğunluklu eşlik denetim kodları, kod kelimeleri arasında istenen uzaklık özeliklerini sağlamakta ve Shannon limitine yaklaşan hata düzeltme yeteneği göstermektedirler. Bu tezde düşük yoğunluklu eşlik denetim kodları için topla-çarp ve ikil-değiştir kod çözme algoritmaları, MATLAB yazılımı kullanılarak Pentium M 1,86 GHz işlemcili bir bilgisayarda gerçeklenmiştir.

İki kod çözme algoritmasının başarımları bilgi oranı, blok uzunluğu ve eşlik denetim matrisinin kolon ağırlığı gibi parametrelerin değişmesi durumlarında toplamsal beyaz gürültülü kanal üzerinde karşılaştırılmıştır.

Beklendiği gibi gelen bilgiyi daha etkin kullanan topla-çarp algoritması, benzetimlerde ikil-değiştir algoritmasına göre daha iyi sonuç vermiştir. Blok uzunluğunun ve kullanılan kod çözme algoritmalarının yineleme sayılarının artması DYED kodlarının başarımını arttırmaktadır. Ancak topla-çarp algoritması daha düşük bir hata alt sınırına sahip olduğundan, yineleme sayısının artması ikil-değiştir algoritmasına göre daha etkili olmaktadır. Kontrol ikillerinin bilgi ikillerine oranı arttıkça, ikil-değiştir algoritmasının başarımı beklendiği gibi artmasına rağmen, topla-çarp algoritması için aynı durum gözlenememiştir. Düzenli olmayan eşlik denetim matrislerine sahip olan DYED kodları, düşük sinyal-gürültü oranları için, düzenli olanlara göre daha başarılı sonuç vermiştir.

Anahtar sözcükler: DYED kodları, topla-çarp algoritması, ikil-değiştir algoritması, yinelemeli kod çözme algoritmaları

ACKNOWLEDGEMENTS

I would like to express my thanks to my supervisor Assoc. Dr. Melek Yücel for her understanding, guidance and support during the preparation of this thesis.

I also would like to thank my mother, who supported and encouraged me whenever i needed. It would be impossible for me to complete this study without her positive energy.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Error correcting codes are the most important tools for reliable communication in any noisy communication channel. Noisy Channel Coding Theorem [Shannon1948] proves that, probability of decoding error can be made to approach zero exponentially with the code length, if properly coded information is transmitted at a rate below the channel capacity. This theorem proves that there are capacity-approaching codes but gives no explicit scheme to construct these codes. Although long code lengths are needed to achieve low probability of error, using long blocks results in longer computation time and higher equipment cost.

There are two main challenges for communicating reliably at rates close to the channel capacity. First we have to properly design distinct codewords and second find reliable methods for estimating transmitted messages at the output of a noise contaminated channel without excessive complexity. A great amount of research has been made and is still being made for finding efficient coding and decoding algorithms which can utilize long blocklengths, without needing excessive computation time or equipment cost.

The use of parity-check codes makes coding relatively simple to implement, however decoding of parity-check codes is not simple. With ordinary parity-check matrices, decoding complexity of parity-check codes grows quadratically with the blocklength. Therefore decoding algorithms with reasonable decoding complexity are needed for the parity-check codes of large blocklengths.

In 1962, Gallager [Gallager1962] proposed a special class of linear block codes with sparse parity-check matrices, which are called low-density parity-check (LDPC) codes. He also proposed two decoding algorithms for decoding these codes,

which are the sum-product and the bit-flip algorithms. When sparse parity-check matrices are used, it is observed that the decoding complexity grows linearly with the blocklength rather than quadratically. Low complexity allows efficient decoding algorithms to operate.

Low-density parity-check (LDPC) codes are not optimum in the sense of minimizing probability of error for a given blocklength. Maximum rate at which these codes can be used, is bounded below the channel capacity [Gallager1960]. Although these codes are not optimum from the point of probability of error, simple and efficient decoding algorithms compensate for the situation. These codes also have good distance properties, which is one of the main challenges in code design.

In [Gallager1962], it was also proved that probability of error for these codes decreases exponentially with the blocklength and the exponent is the same as the optimum code with slightly higher rate. Gallager in his paper [Gallager1962] claimed that at those rates, no other known coding and decoding scheme could correct errors as many as his codes could do. Unfortunately, the idea behind his codes could not be understood at that time. Due to the computational incapabilities at his time, he could not prove the performance of his codes to the community and precious work of Gallager was forgotten for more than thirty years.

MacKay rediscovered low-density parity-check codes [MacKay1999] and showed that they perform almost as close to capacity limits as turbo codes. Luby [Luby2001] extended MacKay's work and showed that irregular LDPC codes are capable of outperforming regular LDPC codes. He also introduced tools for designing irregular code ensembles for which the maximum allowed crossover probability of the binary symmetric channel is optimized. Then Richardson and Urbanke extended Luby's work to any binary input memoryless channel and to soft decision message passing decoding [Richardson2001]. They determined the capacity of message passing decoders applied to LDPC code ensembles by a method called density evolution. Chung [Chung2001] have demonstrated that, with carefully choosing an irregular LDPC code from an optimized ensemble, performance of LDPC codes can approach the Shannon limit.

Today because of its significant bit-error-rate performance, LDPC codes are gaining increased attention in communication standards and literature. The high definition television (HDTV) satellite standard, known as Digital Video Broadcasting (DVB-S2) transmission system standard includes irregular LDPC codes. In DVB-S2 standard an inner irregular LDPC code is concatenated with an outer BCH code .

In the content of this thesis, two decoding algorithms, the bit-flip and the sum-product decoding algorithms are implemented and simulated over an Additive White Gaussian Noise (AWGN) channel for different code parameters. In order to examine the performance of the two algorithms blocklength, column weight and information rate of the code is changed. Some simulation results are also given to show the effects of changing the number of iterations on the performance of the two algorithms. The sum-product algorithm is a relatively complex algorithm compared to the bit-flip algorithm. So the steps of the sum-product algorithm is re-explained thoroughly by supplying the necessary proofs and facts. Some derivations are given whenever it is needed. Similarly for the bit-flip algorithm some examples are provided along with the schematic illustrations. Regular and irregular parity-check matrices are constructed according to the construction schemes given by [MacKay1999]. Performance comparison between the regular and irregular LDPC codes is given in simulation results.

The thesis is organized as follows. In Chapter 2, after a brief description of LDPC codes, factor graphs used for decoding LDPC codes are explained. Then two decoding algorithms, the bit-flip and the sum-product algorithms are described in detail along with some examples. We describe the construction schemes of the parity-check matrices for LDPC codes in Chapter 3. There are simulation results for comparison of the performances of the two decoding algorithms in Chapter 4 and lastly Chapter 5 is composed of our conclusions and comments on the simulation results.

CHAPTER 2

LOW-DENSITY PARITY-CHECK CODES

In this chapter, after a brief description of low-density parity-check codes, decoding algorithms for these codes and factor graphs, which are used as tools of decoding are summarized using the references [Gallager1962], [Loeliger2001] and [Johnson2002]. Examples 2.3, 2.4 and the details between the equations from (2.3) to (2.5), (2.15) to (2.16) and (2.17) to (2.18) are original to this thesis.

2.1 Low-Density Parity-Check (LDPC) Codes

Low-density parity-check codes are special examples of linear block codes. The codewords of a parity-check code are formed by combining a block of binary information digits with a block of check digits. Each check digit is the modulo 2 sum of a prespecified set of information digits. These formation rules for the check digits can be represented by a parity-check matrix, as in Figure 2.1. This matrix represents a set of linear homogeneous modulo 2 equations, which are called parity-check equations, and the set of codewords is the set of all solutions of these equations. We call the set of digits contained in a parity-check equation a parity-check set.

$$\begin{array}{cccccccc} \overbrace{}^{INFO} & & & & \overbrace{}^{CHECK} & & & \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & x_5 = x_1 \oplus x_2 \oplus x_3 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & \Leftrightarrow \quad x_6 = x_1 \oplus x_2 \oplus x_4 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & x_7 = x_1 \oplus x_3 \oplus x_4 \end{array}$$

Figure 2.1: Example of a parity-check matrix. (First 4 digits, which are denoted by "*INFO*", are the information digits, check digits are placed at the end of each parity-check set.)

Low-density parity-check codes are the codes specified by a matrix containing mostly 0's and only a small number of 1's. In particular (n, w_c, w_r) low-density code is a code of blocklength n with a parity-check matrix, where each column contains a small number, w_c of 1's and each row contains a small number w_r of 1's. Note that this type of matrix does not have the check digits appearing in diagonal form as in Figure 2.1. For coding purposes, the equations represented by these matrices can always be solved to give the check digits as explicit sums of information digits.

These codes are not optimum in the sense of minimizing probability of decoding error for a given blocklength. Maximum rate at which these codes can be used is bounded below the channel capacity. However, simple decoding schemes exist for low-density codes and this compensates for their lack of optimality.

The analysis of a low-density code of long blocklength is difficult because of the immense number of codewords involved. It is simpler to analyze a whole ensemble of such codes because the statistics of an ensemble lets us average over quantities that are not tractable in individual codes. From the ensemble behavior, one can make statistical statements about the properties of the member codes.

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	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	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Figure 2.2. Parity-check matrix of a (20, 3, 4) LDPC code given by Gallager.

In order to define an ensemble of (n, w_c, w_r) low-density codes, consider the matrix given in Figure 2.2 defined by Gallager in his paper [Gallager1962]. The matrix can be divided into w_c submatrices, each containing a single 1 in each column. The first of these submatrices contains all its l's in descending order; i.e., the i'th row contains l's in columns $(i - 1)w_r + 1$ to iw_r. The other submatrices are merely column permutations of the first. We define an ensemble of (n, w_c, w_r) codes as the ensemble resulting from random permutation of the columns of each of the bottom $w_c - 1$ submatrices of a matrix such as Figure 2.2, with equal probability assigned to each permutation.

There are two interesting results that can be proven using this ensemble, the first concerning the minimum distance of the member codes, and the second concerning the probability of decoding error.

If we note the ratio of the minimum distance to blocklength by δ, for large n all codes in the ensemble of low-density (n, w_c, w_r) codes have a minimum distance of at least $n\delta_{cr}$, where δ_{cr} is a constant determined by w_c and w_r [Gallager1960]. In Table 2.1 comparison of this ratio of typical minimum distance to blocklength for an LDPC code is compared to that for a parity-check code chosen at random, i.e. with a matrix filled in with equiprobable independent binary digits. The "rate" mentioned in Table 2.1 is the code rate, $R = k / n$, where k is the number of information bits. In terms of w_c and w_r the code rate can be defined as $R = 1 - w_c / w_r$.

Table 2.1. [Gallager1962] Comparison of δ_{cr}, the ratio of typical minimum distance to blocklength for an (n, w_c, w_r) LDPC code, to δ, the same ratio for an ordinary parity-check code of the same rate.

w_c	w_r	Rate	δ_{cr}	δ
5	6	0.167	0.255	0.263
4	5	0.2	0.210	0.241
3	4	0.25	0.122	0.214
4	6	0.333	0.129	0.173
3	5	0.4	0.044	0.145
3	6	0.5	0.023	0.11

For a binary symmetric channel it was shown that [Gallager1960] over a reasonable range of channel transition probabilties, the low-density parity-check code has a probability of decoding error that decreases exponentially with the blocklength and the exponent is the same as that for the optimum code of slightly higher rate.Table 2.2 compares the rate values of LDPC codes and equivalent optimum codes.

Table 2.2. [Gallager1962] Loss of rate associated with LDPC codes.

w_c	w_r	Rate for LDPC	Rate for Equivalent Optimum Code
3	6	0.5	0.555
3	5	0.4	0.43
4	6	0.333	0.343
3	4	0.25	0.266

2.2 Factor Graphs

Calculation of marginal probability density function of a random variable from a given complicated global function is an important problem in many different branches of science and technology. Algorithms that must deal with complicated functions of many variables, often make use of the manner in which the given functions factor as a product of local functions. Each local function depends on a subset of the random variables. Such a factorization can be visualized with a bipartite graph which is called a factor graph. These graphs are widely used for all message passing algorithms. Detailed information about factor graphs can be obtained from [Loeliger2001].

Suppose that $g(x_1,...,x_n)$ is a function which factors into a product of several local functions, each having some subset of $\{x_1,...,x_n\}$ as arguments; i.e., suppose that

$$g(x_1,...,x_n) = \prod_{j \in J} f_j(X_j) \tag{2.1}$$

where J is a discrete index set, X_j is a subset of $\{x_1, \ldots, x_n\}$, and $f_j(X_j)$ is a function having the elements of X_j as arguments.

Definition 2.1 : A factor graph is a bipartite graph that expresses the structure of the factorization given by (2.1). A factor graph has a variable node for each variable x_i, a factor node for each local function f_j, and an edge which connects variable node x_i to factor node f_j if and only if x_i is an argument of f_j .

Example 2.1 (A Simple Factor Graph) : Let $g(x_1, x_2, x_3, x_4, x_5)$ be a function of five variables, and suppose that g can be expressed as a product

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5) \tag{2.2}$$

of five factors, so that $J = \{A, B, C, D, E\}$, $X_A = \{ x_1 \}$, $X_B = \{x_2\}$, $X_C = \{x_1, x_2, x_3\}$, $X_D = \{x_3, x_4\}$ and $X_E = \{x_3, x_5\}$. The factor graph that corresponds to (2.2) is shown in Figure 2.3.

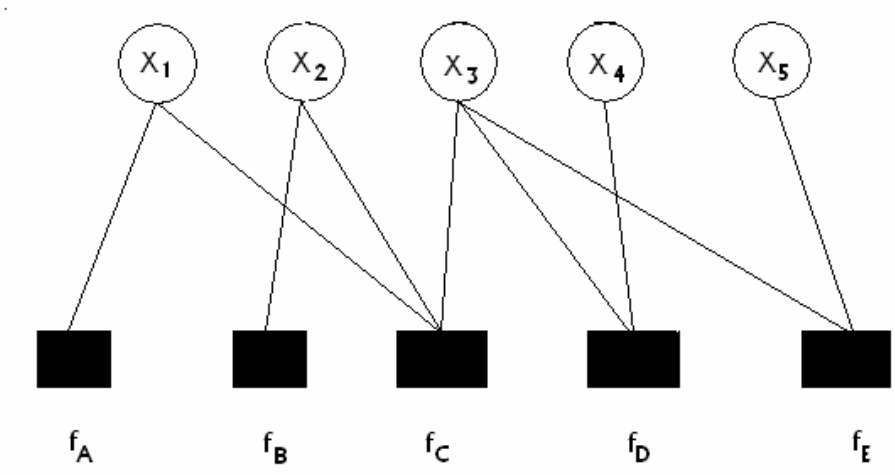


Figure.2.3.A factor graph for the product $f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5)$.

It should be obvious that a factor graph for any (n, k) linear block code may be obtained from a parity-check matrix H for the code. Such a parity-check matrix has n columns and at least ($n-k$) rows. Variable nodes correspond to the columns of H and factor nodes (or checks) to the rows of H, with an edge connecting factor node i to variable node j if and only if $h_{ij} \neq 0$, where $H = \{h_{ij}\}$.

Example 2.2 (Factor Graph of a Linear Code): Let us consider a (6,3) linear code C defined by the parity-check matrix

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

then C is the set of all binary 6-tuples $\boldsymbol{x} = (x_1, x_2, ..., x_6)$ that satisfy three simultaneous equations expressed in matrix form as $H\boldsymbol{x}^{\mathrm{T}} = 0$.

Membership in C is completely determined by checking whether each of the three equations is satisfied. Therefore,

$$X_C(x_1, x_2, ..., x_6) = [(x_1, x_2, ..., x_6) \in C]$$
$$= [x_1 \oplus x_2 \oplus x_5 \equiv 0] \wedge [x_2 \oplus x_3 \oplus x_6 \equiv 0] \wedge [x_1 \oplus x_3 \oplus x_4 \equiv 0]$$

where $\oplus$ denotes the sum in GF(2). The corresponding factor graph is shown in Figure 2.2.

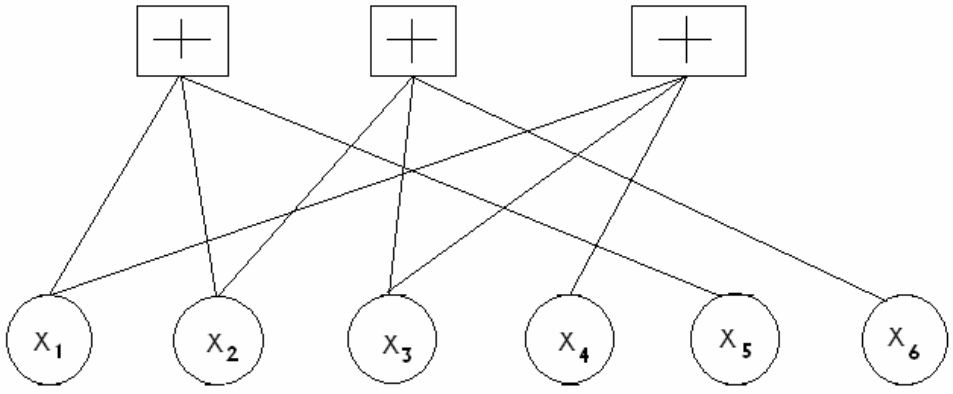


Figure 2.4. Factor graph for the binary linear linear code of Example 2.2.(Variable nodes are represented with circles and factor nodes are represented with squares.)

A cycle in the factor graph is a sequence of connected nodes which start and end at the same node in the graph, and which contain other nodes no more than once. The length of a cycle is the number of edges it contains, and the girth of a graph is the size of the smallest cycle. A linear code with a cycle in its factor graph is shown in Example 2.3.

Tanner introduced [Tanner1981] bipartite graphs, which are called "Tanner graphs", to describe low-density parity-check codes. In Tanner's original formulation, all variables are codeword symbols. From factor graph perspective, a Tanner graph for a code represents a particular factorization of the characteristic function of the code.

Example 2.3 (A Factor Graph with Cycle): Consider a (6,2) linear code C defined by the parity-check matrix

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

The factor graph corresponding to C is given in Figure 2.3.

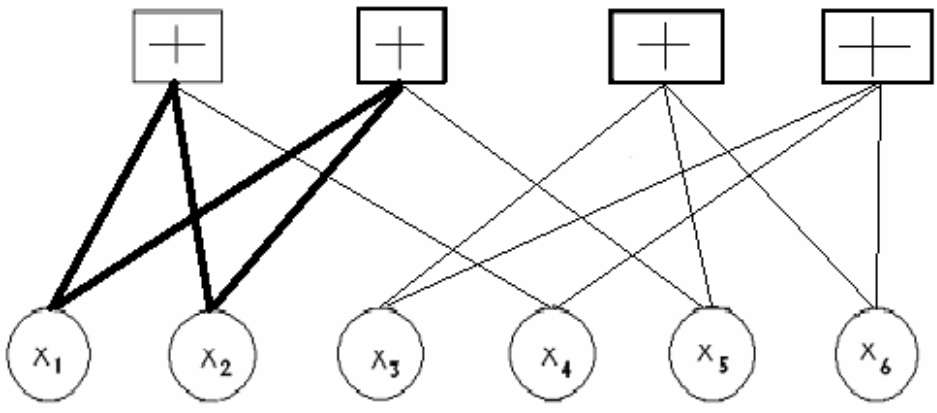


Figure 2.5. Factor graph representation of the parity-check matrix in Example 2.3. (A cycle of length 4 is shown in bold.)

2.3 Encoding of LDPC Codes

LDPC codes are a subclass of linear block codes so their encoding schemes are the same. Encoding scheme of linear block codes is summarized from [Blahut1984]. A linear block code can be represented by a *k* by *n* matrix G, which is called the generator matrix of the code. The rows of the generator matrix are linearly independent and any codeword is the linear combination of the rows. There are q^k codewords, and the q^k distinct k-tuples over GF(q) can be mapped onto the set of codewords.

The following procedure is used for one-to-one pairing of k-tuples and codewords:

$$c = i\,G,$$

where i, the information word, is a k-tuple of information symbols to be encoded and c is the codeword n-tuple. Encoding procedure can be understood with the following example.

Example 2.4 [Blahut1984] : A binary linear code with the generator matrix G is given below.

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

The information vector i = [0 1 1] is encoded into the codeword

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

2.4 Decoding Algorithms

Two decoding algorithms were considered by Gallager [Gallager1962]. First one is called the bit-flip algorithm and it is based on hard decision decoding. Second

one is the sum-product (belief propagation) algorithm which is based on soft decision decoding. Both algorithms are members of a general class of decoding algorithms called message passing algorithms. Message passing algorithms are iterative algorithms. At each iteration of the algorithm, messages are sent from variable nodes to factor nodes and from factor nodes back to variable nodes on the factor graph..

2.4.1 Bit-Flip Algorithm (Hard Decision Decoding)

Since algorithm operates on hard decisions (0 or 1), an initial hard decision is made for each received bit. Decoder computes all the parity-checks and then flips any digit that is contained in more than some predetermined number of unsatisfied parity-check equations. Using these new values parity-checks are recomputed and this process is repeated until all parity-checks are satisfied.

As we stated above, algorithms operate on factor graphs and messages are sent between variable nodes and factor nodes iteratively. For the bit-flip algorithm, these messages are simple: a variable node sends a message to each of the factor nodes to which it is connected declaring if it is a 1 or a 0, and each factor node sends a message to each of the variable nodes to which it is connected, declaring whether the parity check is satisfied or not.

The bit-flip algorithm is as follows:

- Step 1 *Initialization:* Each variable node is assigned the bit value received from the channel, and sends messages to the factor nodes to which it is connected indicating this value.
- Step 2 *Parity Update:* Using the messages from the variable nodes, each factor node calculates whether or not its parity-check equation is satisfied. If all parity-check equations are satisfied the algorithm terminates, otherwise each factor node sends messages to variable nodes to which it is connected indicating whether or not the parity-check equation is satisfied.
- Step 3 *Bit Update:* If the majority of the messages received by each variable node are “not satisfied”, the variable node flips its current value, otherwise the value is retained. If the maximum number of allowed iterations is reached, the algorithm

terminates and a failure to converge is reported; otherwise each variable node sends new messages to the factor nodes to which it is connected, indicating its value and the algorithm returns to Step 2.

Since this algorithm works on sparse parity-check matrices, each check equation will contain either one transmission error or no transmission errors. If the number of parity-check equations is small then this procedure is reasonable. When most of the parity-check equations containing a bit are unsatisfied, this strongly indicates that bit is in error.

Example 2.5 (Operation of the Bit-Flip Algorithm): Consider the code defined by the parity-check matrix H given below. Assume the codeword c = [001011] is sent, and the word r = [101011] is received.

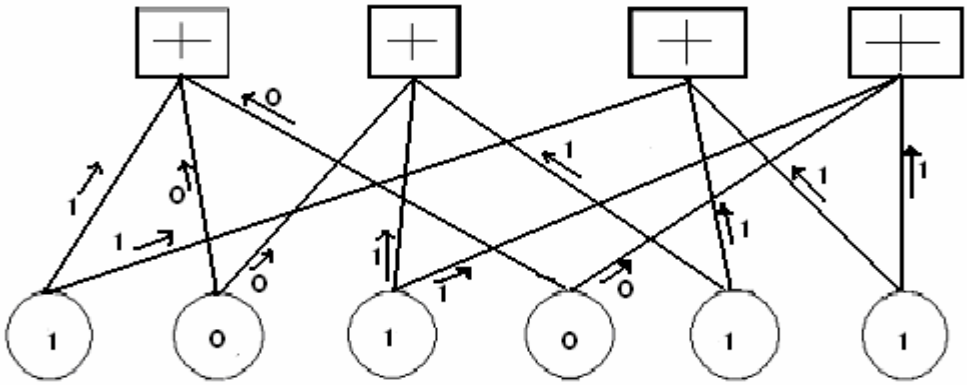
$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$r.H^T = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

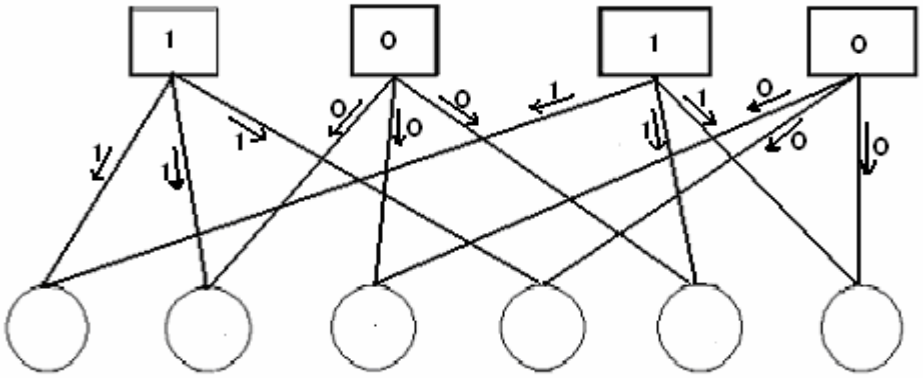
The vector $r.H^T$ indicates that the first and third check equations are not satisfied for the received word r. The bit which appears on both unsatisfied parity-check equations is the first bit of the received word r. We flip the first bit of the received word r and call the resulting word r'. Word r' is multiplied with the transpose of the parity-check matrix H. All elements of resulting vector $r'H^T$ are 0. This shows that all parity-check equations are satisfied. This is one of the stopping conditions of the algorithm. The decided word is r' and it is equal to the codeword sent from the transmitter, for this example.

Steps of the algorithm are explained on the factor graph given in Figure 2.6 for Example 2.5 .

- **Step1:** *Initial values of the variable nodes are sent from variable nodes to factor nodes.*



- **Step 2:** *Each factor node calculates whether or not its parity-check equation is satisfied, using the messages from variable nodes.*



- **Step 3:** *If all parity-check equations are satisfied, that is, all factor nodes send message 0 to variable nodes, algorithm stops .*

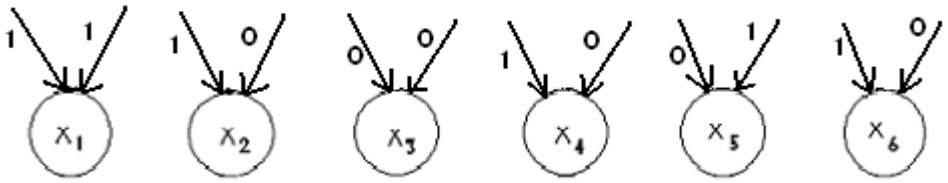


Figure 2.6. Operation of the bit-flip algorithm.

Only variable nodes of the factor graph are shown in step 3. As "1" represents 'not satisfied' first bit of the received word will flip its value. Both of the two messages it received are 'not satisfied'. These new variable node values are sent to factor nodes and step 2 will be repeated until all parity-checks are satisfied or predetermined number of iterations is reached .

2.4.1.1 Effect of a Cycle on the Bit-Flip Algorithm

The existence of cycles in the factor graph of a code reduces the effectiveness of the iterative decoding process. To illustrate the detrimental effect of a cycle of length 4, consider the code given in Example 2.3.(see Figure 2.5)

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

A valid codeword for this code is [001001] but again we assume that the first bit is corrupted, so that r = [101001] is received from the channel. In Step 1 initial bit values are 1,0,1,0,0 and 1 respectively, and messages are sent to factor nodes indicating these values. Step 2 reveals that the first and second parity-check equations are not satisfied. In Step 3 both first and second bits have the majority of their messages indicating "not satisfied" and so both flip their values. When step 2 is repeated, we see that the first and second parity-check equations are again not satisfied. Further iterations at this point simply cause the first two bits to flip their values in such a way that one of them is always incorrect so algorithm fails to converge. As a result of the cycle of length 4, each of the first two codeword bits are involved in the same two parity-check equations, and so when neither of these parity-check equations are satisfied, it is not possible to determine which bit is causing the error.

Although the bit-flip algorithm is relatively simple, major drawback of this algorithm is that it operates on hard decisions made by the decoder at the channel

output. This throws away valuable information coming from the channel especially when we are dealing with continuous-output channels.

Soft decision decoding algorithms have better BER performance. On the other hand, hard decision decoding algorithms are simple, fast and their hardware implementations are easy. One important fact is that when we have only hard decisions at the channel output, hard decision decoding is the only way to decode low-density parity-check codes. Next generation of optical communication systems are operating at 40 Gb/s. The current circuit technology does not allow soft decision decoding algorithms to operate at these data rates. There are some research on improving hard decision decoding algorithms [FossorierMiladinovic2005] to get better performance.

2.4.2 Sum-Product Algorithm

Sum-product is a general name for a class of maximum likelihood decoding algorithms. Wiberg in his Ph.D. thesis [Wiberg1996] proved that famous BCJR [BahlCocke1974], Turbo [Berrou1993] and Gallager's algorithm [Gallager1962] are all maximum likelihood decoding algorithms. The algorithm uses the channel information and the values coming from the channel. It forms a probabilistic value for each received bit and iteratively refreshes this value to find an estimate for that bit. We introduce Gallager's algorithm as it was given in his original paper [Gallager1962].

Consider the tree structure given in Figure 2.7. Digit *d* is represented by the node at the base of the tree, and each line rising from this node represents one of the check sets containing the digit *d*. The other digits in these parity-check sets are represented by the nodes on the first tier of the tree. The lines rising from tier-1 to tier-2 represent the other digits in those parity-check sets.

Assume both digit *d* and some of the digits in the first tier are transmission errors. Then on the first decoding attempt, the error-free digits in the second tier and their parity-check constraints will allow correction of digit *d* on the second decoding attempt. Thus digits and parity-check equations can be used in decoding a digit which

seems unconnected to them. Gallager's probabilistic decoding algorithm makes use of these extra digits and parity-check constraints systematically.

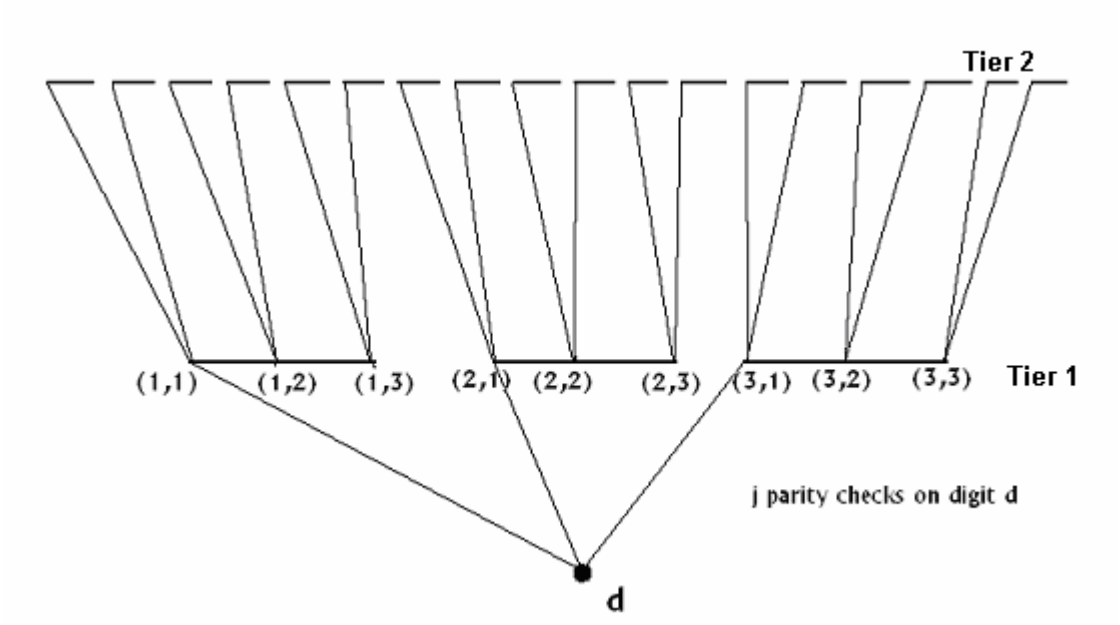


Figure 2.7. Parity-check set tree.

Assume that the codewords from an (n, w_c , w_r) code are used with equal probability on an arbitrary binary-input channel. For any digit d, using the notation in Figure 2.7, an iteration process will be derived that on the m'th iteration computes the probability that the transmitted digit in position d is a 1, conditional on the received symbols out to and including the m'th tier. For the first iteration, consider digit d and the digits in the first tier to form a subcode. All sets of these digits are satisfying w_c parity-check equations and these sets have equal probability of transmission. Consider the ensemble of events, in which transmitted digits in position d and the digits in the first tier are independent equiprobable binary digits where their probabilities are determined by channel transition probabilities $P_x(y)$.

Let $P(x_d = 1)$ and $P(x_d = 0)$ be probabilities that digit in position d is 1 and 0 respectively. We want to find an expression for the ratio of these probabilities conditional on the set of received symbols $\{y\}$ and S = {transmitted digits satisfy w_c parity-check equations}. Using the definition of conditional probabilities for the three events A, B and C:

$$P(A \mid B,C) = \frac{P(A,B,C)}{P(B,C)} = \frac{P(C \mid A,B)P(A,B)}{P(C \mid B)P(B)} = \frac{P(C \mid A,B)P(A \mid B)P(B)}{P(C \mid B)P(B)}$$

$$= \frac{P(C \mid A,B)P(A \mid B)}{P(C \mid B)} \quad . \qquad (2.3)$$

If A' is the complement of A, using (2.3) one finds

$$\frac{P(A' \mid B,C)}{P(A \mid B,C)} = \frac{P(A' \mid B)P(C \mid A',B)}{P(A \mid B)P(C \mid A,B)} . \qquad (2.4)$$

Let us define the events A,B and C by $A = \{x_d = 1\}$, $B = \{y\}$ and $C = S$, so $A' = \{x_d = 0\}$. Then using (2.4) we get

$$\frac{P(A' \mid B,C)}{P(A \mid B,C)} = \frac{P(x_d = 0 \mid \{y\},S)}{P(x_d = 1 \mid \{y\},S)} = \frac{P(x_d = 0 \mid \{y\})}{P(x_d = 1 \mid \{y\})} \frac{P(S \mid x_d = 0,\{y\})}{P(S \mid x_d = 1,\{y\})} . \qquad (2.5)$$

Calling $P(x_d = 1 \mid \{y\}) = P_d$ and $P(x_d = 0 \mid \{y\}) = 1 - P_d$, (2.5) can be written as

$$\frac{P(x_d = 0 \mid \{y\},S)}{P(x_d = 1 \mid \{y\},S)} = \frac{1 - P_d}{P_d} \frac{P(S \mid x_d = 0,\{y\})}{P(S \mid x_d = 1,\{y\})} . \qquad (2.6)$$

We now try to calculate $P(S \mid x_d = 0,\{y\})$ term in (2.6), using the following lemma.

Lemma 2.1: Consider a sequence of m independent binary digits, in which the l'th digit is 1 with probability P_l. The probability of even number of digits are 1 in this sequence is

$$\frac{1 + \prod_{l=1}^{m} (1 - 2P_l)}{2} \tag{2.7}$$

Proof : Consider the function

$$\prod_{l=1}^{m} (1 - P_l + P_l t)$$

Observe that if this is expanded into a polynomial in t, the coefficient of t^i is the probability of i 1's. The function $\prod_{l=1}^{m} (1 - P_l - P_l t)$ is identical except that all the odd powers of t are negative. Adding these two functions, all the even powers of t are doubled, and the odd terms cancel out. Finally, letting t=1 and dividing by 2, the result is the probability of an even number of ones. So

$$\frac{\prod_{l=1}^{m} (1 - P_l + P_l) + \prod_{l=1}^{m} (1 - P_l - P_l)}{2} = \frac{1 + \prod_{l=1}^{m} (1 - 2P_l)}{2} \tag{2.8}$$

thus proving the lemma.

Corollary 2.1: Consider a sequence of m independent binary digits. If the l'th digit is 1 with probability P_l , then the probability of odd number of digits are 1 in this sequence is

$$\frac{1 - \prod_{l=1}^{m} (1 - 2P_l)}{2} \tag{2.9}$$

Given that $x_d = 0$, a parity-check on the digit d is satisfied if the other $(w_r - 1)$ positions in the parity-check set contain an even number of 1's. Using Lemma 2.1, the probability that i'th parity-check is satisfied when d'th bit is known to be 0 is

$$P_{id} = \frac{1}{2} + \frac{1}{2} \prod_{d' \in B_i, d' \neq d} (1 - 2P_{d'}). \tag{2.10}$$

B_i represents the set of column locations of the bits in the i'th parity-check equation. Similarly, A_d is the set of row locations of the parity check equations which check on the d'th bit of the code. Since all digits in the ensemble are independent, the probability that all w_c parity-checks are satisfied is the product of the probabilities of the individual checks being satisfied. That is

$$P(S \mid x_d = 0, \{y\}) = P_{id} = \prod_{i \in A_d} \left(\frac{1}{2} + \frac{1}{2} \prod_{d' \in B_i, d' \neq d} (1 - 2P_{d'}) \right) \tag{2.11}$$

Similarly using Corollary 2.1

$$P(S \mid x_d = 1, \{y\}) = P_{id} = \prod_{i \in A_d} \left(\frac{1}{2} - \frac{1}{2} \prod_{d' \in B_i, d' \neq d} (1 - 2P_{d'}) \right) \tag{2.12}$$

Substituting (2.11) and (2.12) into (2.6) we get

$$\frac{P(x_d = 0 \mid \{y\}, S)}{P(x_d = 1 \mid \{y\}, S)} = \frac{1 - P_d}{P_d} \frac{\prod_{i \in A_d} \left(\frac{1}{2} + \frac{1}{2} \prod_{d' \in B_i, d' \neq d} (1 - 2P_{d'}) \right)}{\prod_{i \in A_d} \left(\frac{1}{2} - \frac{1}{2} \prod_{d' \in B_i, d' \neq d} (1 - 2P_{d'}) \right)} \tag{2.13}$$

We can take the logarithm of both sides of (2.13) as it was given in [Johnson2002]. One benefit of the logarithm is that whereas probabilities need to be multiplied, log-likelihood ratios need only to be added, reducing implementation complexity. Taking

the logarithm of both sides of (2.13) we have

$$\log_e\left(\frac{P(x_d=0\,|\,\{y\},S)}{P(x_d=1\,|\,\{y\},S)}\right)=\log_e\left(\frac{1-P_d}{P_d}\right)+\log_e\left(\frac{\prod_{i\in A_d}\left(\frac{1}{2}+\frac{1}{2}\prod_{d'\in B_i,d'\neq d}(1-2P_{d'})\right)}{\prod_{i\in A_d}\left(\frac{1}{2}-\frac{1}{2}\prod_{d'\in B_i,d'\neq d}(1-2P_{d'})\right)}\right) \tag{2.14}$$

Since

$$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$$

it can be shown that

$$\tanh\left(\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)\right)=1-2P_{d'}. \tag{2.15}$$

Proof of (2.15):

$$\tanh\left(\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)\right)=\frac{e^{\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)}-e^{-\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)}}{e^{\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)}+e^{-\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)}}$$

$$=\frac{e^{\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)^{\frac{1}{2}}}-e^{\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)^{-\frac{1}{2}}}}{e^{\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)^{\frac{1}{2}}}+e^{\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)^{-\frac{1}{2}}}}$$

$$=\frac{\left(\frac{1-P_{d'}}{P_{d'}}\right)^{1/2}-\left(\frac{1-P_{d'}}{P_{d'}}\right)^{-1/2}}{\left(\frac{1-P_{d'}}{P_{d'}}\right)^{1/2}+\left(\frac{1-P_{d'}}{P_{d'}}\right)^{-1/2}}$$

$$= \frac{\left(\frac{1-P_{d'}}{P_{d'}}\right)^{1/2}\left(1-\left(\frac{1-P_{d'}}{P_{d'}}\right)^{-1}\right)}{\left(\frac{1-P_{d'}}{P_{d'}}\right)^{1/2}\left(1+\left(\frac{1-P_{d'}}{P_{d'}}\right)^{-1}\right)}$$

$$= \frac{1-\frac{P_{d'}}{1-P_{d'}}}{1+\frac{P_{d'}}{1-P_{d'}}} = \frac{\frac{1-P_{d'}-P_{d'}}{1-P_{d'}}}{\frac{1-P_{d'}+P_{d'}}{1-P_{d'}}} = 1-2P_{d'}$$

Then we can substitute (2.15) into (2.14) to obtain the log-likelihood ratio of the estimated a posteriori probability of the *d'*th bit as

$$\log_e\left(\frac{P(x_d=0\,|\,\{y\},S)}{P(x_d=1\,|\,\{y\},S)}\right) = \log_e\left(\frac{1-P_d}{P_d}\right) + \sum_{i\in A_d}\left(\log_e\frac{1+\prod_{d'\in B_i,d'\neq d}\tanh\left(\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)\right)}{1-\prod_{d'\in B_i,d'\neq d}\tanh\left(\frac{1}{2}\log_e\left(\frac{1-P_{d'}}{P_{d'}}\right)\right)}\right) \quad (2.16)$$

The sum-product algorithm is described as follows [Johnson2002]:

- Step 1 *Initialization:* The initial message sent from variable node *d* to the factor node *i* is the log-likelihood ratio of the (soft) received signal y_d given knowledge of the channel properties. If we represent the log-likelihood ratio of the *d*'th bit in the *i*'th parity-check with $L_{d,i}$. $L_{d,i}$ for an AWGN channel with signal-to-noise ratio E_b/N_0 is:

$$L_{d,i} = R_d = 4y_d\ \frac{E_b}{N_0}\ . \quad (2.17)$$

We use BPSK modulation, where −1 is transmitted for 1 and +1 is transmitted for 0. For AWGN channel, probability density function of *y* given the received digit is 0, is

$$p_y(y\,|\,x_d=0) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(y-(1))^2}{2\sigma^2}}\ .$$

For the same channel, probability density function of *y* given the received digit is 1, is

$$p_y\left(y \mid x_d = 1\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-(-1))^2}{2\sigma^2}}$$

Then likelihood ratio is given by

$$\frac{p_y\left(y \mid x_d = 0\right)}{p_y\left(y \mid x_d = 1\right)} = \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-(1))^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-(-1))^2}{2\sigma^2}}}$$

$$= e^{\frac{-\left(y^2-2y+1\right)+y^2+2y+1}{2\sigma^2}}$$

$$= e^{\frac{4y}{2\sigma^2}}$$

For an AWGN channel

$$\sigma^2 = \frac{N_0}{2}.$$

$$= e^{\frac{4y}{2\frac{N_0}{2}}}$$

If we take the logarithm to get the log-likelihood ratio we reach

$$= \log_e\left(e^{\frac{4y}{N_0}}\right) = \frac{4y}{N_0}.$$

So for an AWGN channel with signal-to-noise ratio E_b/N_0, the log-likelihood ratio of the *d*'th bit is given as

$$4y_d \ \frac{E_b}{N_0}.$$

- Step 2 *Factor-to-variable:* The extrinsic message from factor node *i* to variable node *d* is the probability that parity-check *i* is satisfied if bit *d* is assumed to be a 1 expressed as a log-likelihood ratio denoted by $E_{d,i}$. Then $E_{d,i}$ is

For So

$$E_{d,i} = \log_e \left(\frac{1 + \prod_{d' \in B_i, d' \neq d} \tanh\left(\frac{\left(L_{d',i}\right)}{2} \right)}{1 - \prod_{d' \in B_i, d' \neq d} \tanh\left(\frac{\left(L_{d',i}\right)}{2} \right)} \right). \tag{2.18}$$

- Step 3 *Codeword test:* The combined log-likelihood ratio for *d*'th digit,which is denoted by L_d, is the sum of the extrinsic messages and the original log-likelihood ratio calculated in Step 1:

$$L_d = \sum_{i \in A_d} E_{d,i} + R_d \tag{2.19}$$

For each bit a hard decision is made:

$$z_d = \begin{cases} 1, & L_d \leq 0 \\ 0, & L_d > 0. \end{cases}$$

If $z = [\, z_1,...,z_n]$ is a valid codeword ($Hz^T = 0$), or if the maximum number of allowed iterations have been completed, the algorithm terminates.

- Step 4 *Variable- to-factor:* The message sent by each variable node to factor nodes to which it is connected is similar to (2.19), except that variable *d* sends to factor node *i* a log-likelihood ratio calculated without using the information from factor node *i*:

$$L_{d,i} = \sum_{i' \in A_d, i' \neq i} E_{d,i'} + R_d \,. \tag{2.20}$$

Return to Step 2.

The extrinsic information passed from a factor node to a variable node is independent of the probability value for that bit. The extrinsic information from the check nodes is then used as a priori information for the variable nodes in the subsequent iterations.

The existence of an exact termination rule for the sum-product decoding algorithm has two important benefits; the first is that failure to converge is always

detected, and the second is that additional iterations once a solution has been found are avoided.

There are variations to the sum-product algorithm presented here. The *min-sum algorithm*, for example, simplifies the calculation of (2.18) by recognizing that the term corresponding to the smallest $L_{d'i}$ dominates the product term and so the product can be approximated by a minimum; the resulting algorithm thus requires calculation of only minimums and additions. Example 2.4, which shows the operation of the sum-product algorithm is taken from [Johnson2002].

Example 2.5 (Operation of the Sum-Product Algorithm)[Johnson2002]: Suppose codeword [0 0 1 0 1 1] is sent with BPSK modulation, where 0 is mapped to 1 and 1 is mapped into –1, using AWGN channel with $E_b/N_0 = 1.25$. Received signal is $y = -0.1 \quad 0.5 \quad -0.8 \quad 1.0 \quad -0.7 \quad 0.5$. Two bits (bits 1 and 6) are in error if hard decision is considered. For these values the algorithm operates as follows.

Iteration 1

$$R = \begin{bmatrix} -0.5000 & 2.5000 & -4.0000 & 5.0000 & -3.5000 & 2.5000 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \text{ as a hard decision}$$

$$E = \begin{bmatrix} 2.4217 & -0.4930 & . & -0.4217 & . & . \\ . & 3.0265 & -2.1892 & . & -2.3001 & . \\ -2.1892 & . & . & . & -0.4217. & 0.4696 \\ . & . & 2.4217 & -2.3001 & . & -3.6869 \end{bmatrix}$$

$$L = \begin{bmatrix} -0.2676 & 5.0334 & -3.7676 & 2.2783 & -6.2217 & -0.7173 \end{bmatrix}$$

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

$$Hz^T = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T \Rightarrow \text{Continue}$$

$$L = \begin{bmatrix} -2.6892 & 5.5265 & . & 2.6999 & . & . \\ . & 2.0070 & -1.5783 & . & -3.9217 & . \\ 1.9217 & . & . & . & -5.8001 & -1.1869 \\ . & . & -6.1892 & 4.5783 & . & 2.9696 \end{bmatrix}$$

Iteration 2

$$E = \begin{bmatrix} 2.6426 & -2.0060 & . & -2.6326 & . & . \\ . & 1.4907 & -1.8721 & . & -1.1041 & . \\ 1.1779 & . & . & . & -0.8388 & -1.9016 \\ . & . & -2.7877 & -2.9305 & . & -4.3963 \end{bmatrix}$$

$$L = \begin{bmatrix} 3.3206 & 1.9848 & -3.0845 & -0.5603 & -5.4429 & -3.7979 \end{bmatrix}$$

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

$$Hz^T = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \Rightarrow \text{Continue}$$

$$L = \begin{bmatrix} 0.6779 & 3.9907 & . & 2.0695 & . & . \\ . & 0.4940 & -1.2123 & . & -4.3388 & . \\ 2.1426 & . & . & . & -4.6041 & -1.1863 \\ . & . & -5.8721 & 2.3674 & . & 0.5984 \end{bmatrix}$$

Iteration 3

$$E = \begin{bmatrix} 1.9352 & -0.5180 & . & 0.6515 & . & . \\ . & 1.1733 & -0.4808 & . & -0.2637 & . \\ 1.1832 & . & . & . & -1.3362 & -2.0620 \\ . & . & 0.4912 & -0.5948 & . & -2.3381 \end{bmatrix}$$

$$L = \begin{bmatrix} 3.2684 & 4.1912 & -3.9896 & 5.0567 & -5.0999 & -1.9001 \end{bmatrix}$$

$$z = [0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1]$$

$$Hz^T = [0 \quad 0 \quad 0 \quad 0]^T \Rightarrow \text{Terminate}$$

Low-density parity-check codes are sparse graph codes, that means each constraint involves only small number of variables in the factor graph of LDPC codes. There are other sparse garph codes like turbo codes, repeat accumulate codes and digital fountain codes. All these codes can be decoded by the sum-product algorithm.

For various implementations of the sum-product algorithm, instead of log-likelihood ratio of soft received signal, likelihood difference and signed log-likelihood difference could be used as message content (see [Loeliger2001]).

The bit-flip algorithm is relatively simple compared to the sum-product algorithm, but it throws away the valuable information especially when we are dealing with continuous-output channels. The sum-product is more complex than the bit-flip algorithm. It makes use of the soft received signal, which is vital when continuous-output channels are used. It is important to understand these two decoding algorithms for LDPC codes to compare their BER performances. In Chapter 4 we change code parameters like rate, blocklength and number of iterations in our simulations to see the performance difference between the two algorithms.

CHAPTER 3

PARITY-CHECK MATRICES OF LDPC CODES

3.1. Previous Work

Low-density parity-check codes are a class of linear block codes with the requirement that vast majority of the entries in their parity-check matrices are zero. Each column of the parity-check matrix corresponds to a codeword bit, and each row of the parity-check matrix defines a parity-check set. A parity-check matrix is regular if each code bit is contained in a fixed number, w_c, of parity checks and each parity-check equation contains a fixed number, w_r, of code bits. w_r is called the row weight and w_c is called the column weight of the code. If a low-density parity-check code is described by a regular parity-check matrix, it is called a *regular* low-density parity-check code; otherwise it is called an *irregular* low-density parity-check code.

The matrix proposed by Gallager in his paper [Gallager1962] is a regular matrix with a column weight of 3 and a row weight of 4. This matrix is also cycle-free to make decoding algorithms operate properly. MacKay increased the number of columns of the parity-check matrix proposed by Gallager from 504 bits to 16000 bits [MacKay1999]. He performed simulations with LDPC codes based on these matrices and showed that they can perform close to the capacity of turbo codes. MacKay also used higher fields to define parity-check matrices for LDPC codes [MacKay1998]. He acquired better performance by using higher order fields. Then Luby [Luby22001]

introduced irregular matrices for LDPC codes. Simulations with these irregular parity-check matrices show that LDPC codes with irregular parity-check matrices for blocklengths greater than 10^5 bits can outperform turbo codes.

3.2 Properties of a Regular Parity-Check Matrix

A regular parity check matrix is given below, with $w_r = 3$ and $w_c = 2$.

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

Consider the factor graph of this code. Assume m represents the number of rows and n represents the number of columns in the parity-check matrix. In a factor graph the number of edges leaving the variable nodes must equal the number of edges leaving the factor nodes, thus for a regular code,

$$m.\, w_r = n.\, w_c .$$

k representing the information bits, there are $n-k$ check bits in each codeword. A linear block code can be represented by a parity-check matrix of $n-k$ rows and n columns. The code rate of a block code is defined as the ratio of the number of information bits to the total number of bits.

$$R = \frac{k}{n}$$

The code rate of a linear block code can be determined by using the number of columns and the number of rows of the parity-check matrix. Also the knowledge of the column weight and the row weight can be used to determine the rate as follows:

$$\frac{m}{n} = \frac{n-k}{n} = \frac{w_c}{w_r} = 1 - \frac{k}{n} = 1 - R .$$

Then it can be written that

$$1 - \frac{w_c}{w_r} = R .$$

3.3 Construction Schemes

There are several construction schemes for constructing sparse parity check matrices. These construction schemes are named construction 1A, 2A, 1B and 2B in [MacKay1999]. Construction 1A and 1B defines regular parity-check matrices where Construction 2A and 2B defines irregular parity-check matrices. We describe these construction schemes as they are given in [MacKay1999].

Construction 1A

A matrix with m rows and n columns created randomly with column weight w_c and row weight is uniform as possible. The matrix contains no cycles. The matrix proposed by Gallager in his paper [Gallager1962] is created according to construction 1A. It has a column weight of 3 and a row weight of 4.

Construction 2A

Up to $m/2$ columns have weight 2 and overlap between any two of these weight 2 columns is zero. Remaining columns are made random with weight 3 and rows have weights as uniform as possible. Contains no cycles.

Construction 1B

Because matrices are created randomly, in some cases we can not get rid of cycles. In these cases columns, which cause cycles are deleted from the matrix. After constructing a matrix according to construction 1A, some of the columns are deleted from the matrix in order to prevent short cycles. This method is called construction 1B.

Construction 2B

After constructing a matrix according to the construction 2A, some columns of the matrix is deleted in order to prevent cycles.

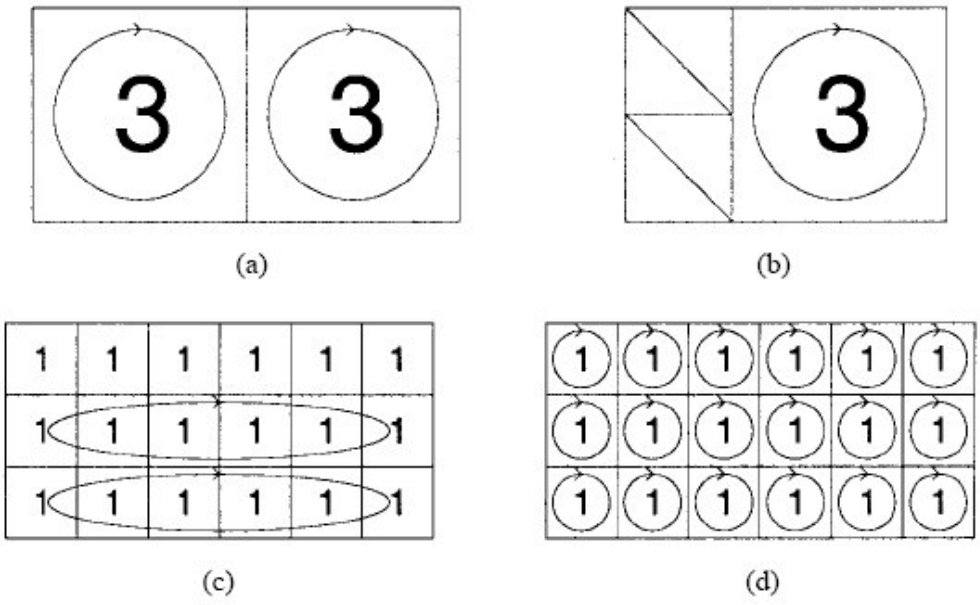


Figure 3.1 [MacKay1999] Schematic illustrations of construction methods for parity-check matrices.((a) Construction 1A for an LDPC code with w_c = 3, w_r = 6 and R = 1/2 ,(b) Construction 2A for an LDPC code with rate 1/3, (c)(d) two constructions similar to Construction 1A)

Notation in the figure: an integer represents a number of permutation matrices superposed on the surrounding square. A diagonal line represents an identity matrix. A rotating ellipse imposed on a block within the matrix represents random permutation of all the columns in that block.

To see the performance difference between the regular and irregular LDPC codes, check Figure 3.2, where BER performances of Turbo codes, convolutional codes, regular and irregular LDPC codes are reproduced from [Johnson2002]. The best performance, close to the Shannon limit shown in Figure 3.2, is obtained by irregular construction of LDPC codes with very large blocklengths like 10^7 bits.

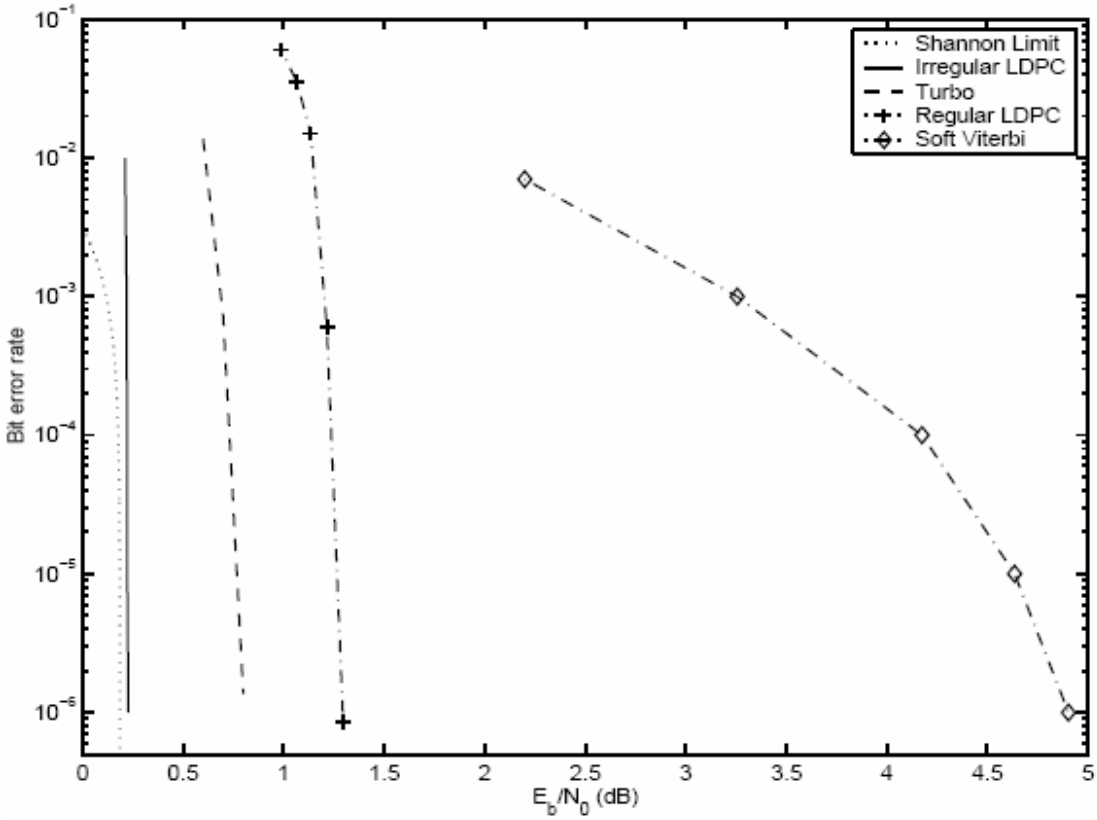


Figure 3.2 [Johnson2002] BER performance of rate-1/2 error correction codes on AWGN channel. (From right to left, soft Viterbi decoding of a constraint lenght 7 convolutional codes; sum-product decoding of regular LDPC code with blocklength 65389; a turbo code with 2+32 states, 16384 bit interleaver and 18 iterations, sum-product decoding of a blocklength 10^7 irregular code, shanon limit at rate 1/2.)

Even with sparse parity-check matrices encoding complexity of LDPC codes is quadratic in the blocklength. The H matrix should be put into a form of $[P^T\ I]$ and then using P, the generator matrix $G = [I\ P]$ should be constructed. Bringing H into the desired form requires n^3 operations of preprocessing. Then the actual encoding requires n^2 operations since, after the preprocessing the matrix H is no longer sparse. To overcome this problem several solutions are offered. One of them [Lubys22001] is using cascaded graphs rather than bipartite graphs. By determining the number of stages carefully one can construct codes which are decodable and encodable in linear time. This method comes with a performance loss compared to the standard LDPC code [Luby22001]. Another solution [MacKay1998] is to force the parity-check

matrix in having lower triangular form. This restriction guarantees a linear time encoding complexity, but it also results in some loss of performance.

In Figure 3.3 a parity-check matrix which is put into a lower triangular form can be seen. After some column and row permutations a sparse parity-check matrix can changed into this form and encoding can be performed efficiently [Richardson22001].

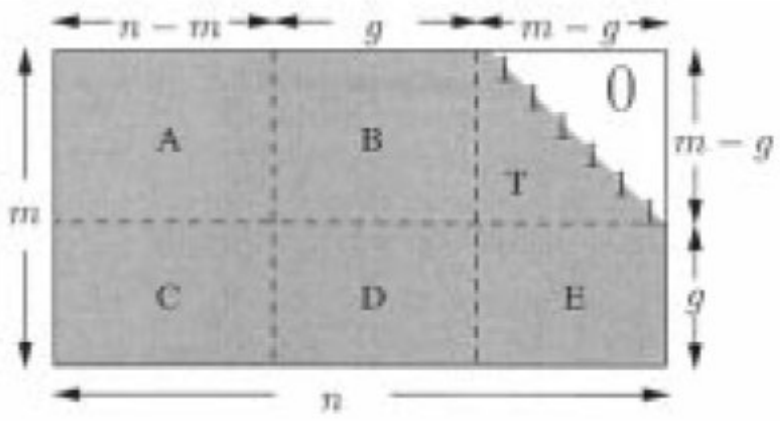


Figure 3.3. [Richardson22001]The parity-check matrix in approximate lower triangular form.

The grey area shown in Figure 3.3 are full of binary elements. Since column and row permutations are used on a sparse parity-check matrix to obtain the form depicted in Figure 3.3. Matrices A, B, C, D, T and E are also sparse matrices.

CHAPTER 4

SIMULATION RESULTS

In this chapter, we investigate the bit-error-rate performances of LDPC codes over AWGN channels using either the sum-product or the bit-flip decoding algorithms. We examine the effects of changing the code parameters like blocklength, column weight and information rate on the performance of LDPC decoding algorithms. We give our simulation for the performance comparison of regular and irregular LDPC codes in Section 4.5. The last section is a literature review [Davey1998] to give an idea about the use of LDPC codes over higher order finite fields.

4.1 Dependence on the Blocklength

Shannon's noisy channel coding theorem states that, when properly coded information is sent through a noise contaminated channel, using long blocklengths, at rates below the channel capacity, error probability can be made to approach zero. Depending on this theorem we can say that the blocklength of the code has to be large to achieve low error probability when information rate is below the channel capacity. For LDPC codes it was shown [Gallager1960] that for $w_c > 3$ and below the channel

capacity, probability of error decreases at least exponentially with the root of the blocklength, using the sum-product decoder on binary symmetric channel.

In Figure 4.1 and Figure 4.2 we see the BER performances of the sum-product and the bit-flip decoding algorithms respectively on rate 1/ 2 regular LDPC code with blocklengths of 100, 250, 500 and 1000 on AWGN channel.

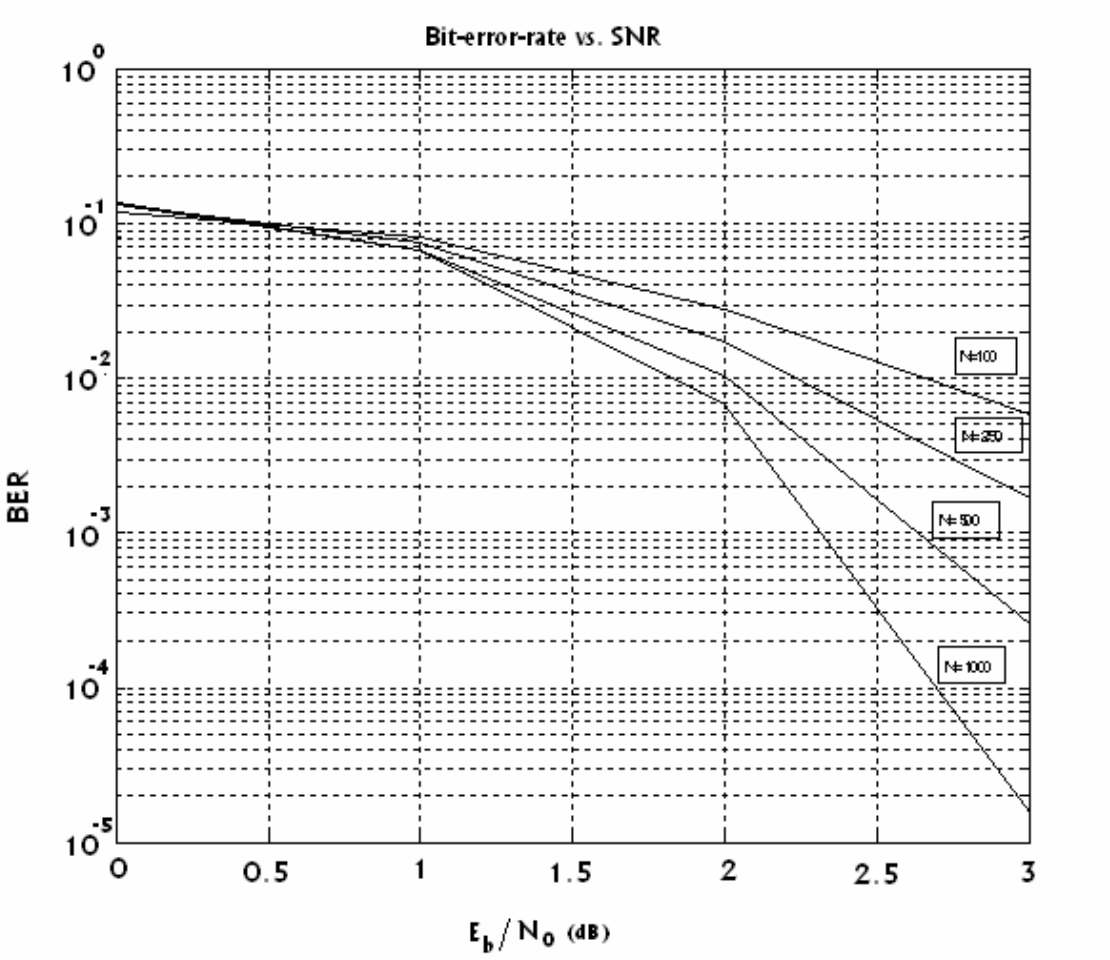


Figure 4.1. BER performance comparison of LDPC codes on blocklengths 100, 250, 500 and 1000 using the sum-product decoding algorithm. (Maximum iteration number is set to 10).

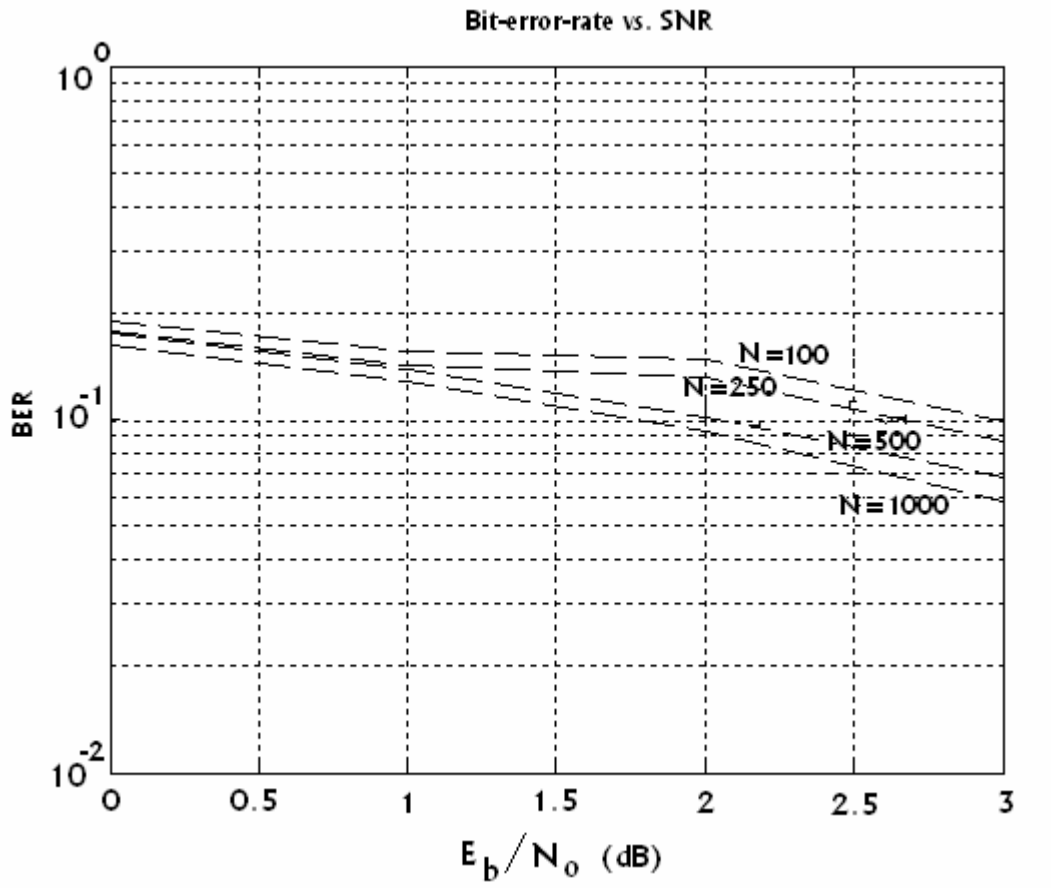


Figure 4.2. BER performance comparison of LDPC codes on blocklengths 100, 250, 500 and 1000 using the bit-flip decoding algorithm.

As it can be seen from Figure 4.1 and Figure 4.2, increasing blocklength of the code resulted in a performance improvement for both algorithms. This is the result we expect, depending on Shannon's theorem. For better comparison between two algorithms Figure 4.1 and Figure 4.2 are plotted on the same figure, Figure 4.3. It is clear in Figure 4.3 that for equal blocklengths the sum-product algorithm outperforms the bit-flip algorithm.

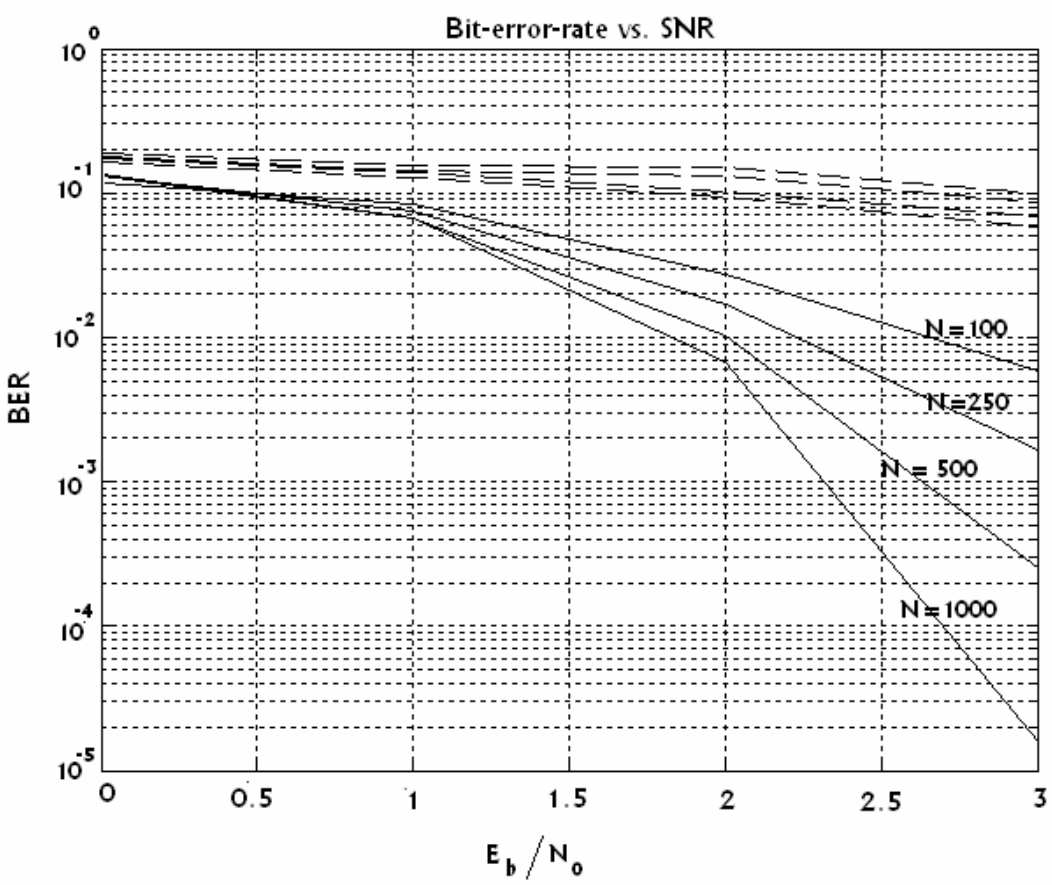


Figure 4.3 BER performance comparison of LDPC codes for the two decoding algorithms on different blocklengths. (Dashed lines show the performance of the bit-flip algorithm where solid lines are for the sum-product algorithm.)

4.2 Dependence on the Column Weight

Given an optimal decoder, best peformance would be obtained for the codes closest to random codes [MacKay2004]. The codes which are closest to random codes have largest column weight, w_c. However, the sum-product decoder makes poor progress in dense graphs [MacKay2004], so the best performance is obtained for a small value of w_c. This result gave the motivation to construct some columns of the parity-check matrix with weight 2. New parity-check matrices were proposed by Luby [Luby2001] with varying column weights and irregular LDPC codes were introduced.

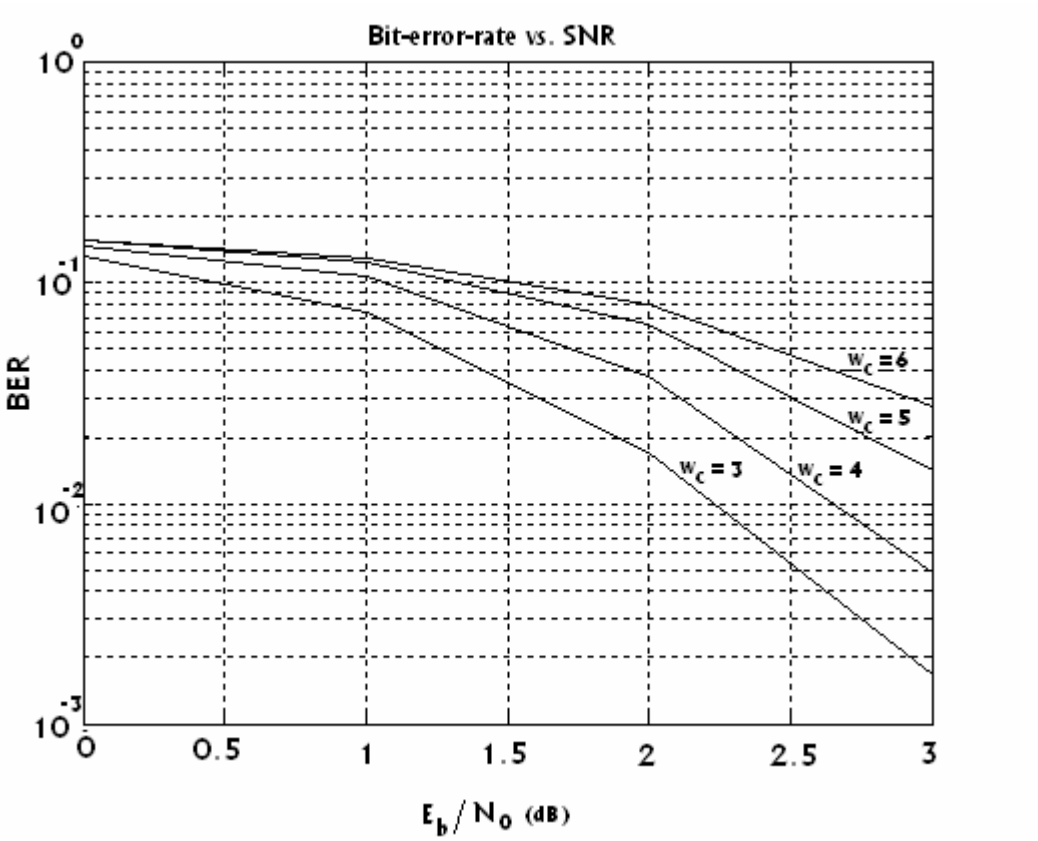


Figure 4.4 Dependence on column weight w_c for codes of blocklength 250 with rate 1 / 2 using the sum-product decoding algorithm . (Maximum number of iterations is set to 10).

As it can be seen from Figure 4.4 increasing column weight, w_c , results in a BER performance loss. This result is consistent with [MacKay2004] where it is stated that the sum-product algorithm can not operate on dense graphs.

Let us investigate the BER performance of the bit-flip algorithm for increasing column weights by examining Figure 4.5. It can be seen from Figure 4.5 that higher column weights result a deterioration in BER performance of LDPC codes. Increasing the column weight of the parity-check matrix increases the probability of having short cycles in the factor graph. We explained the effects of short cycles on the bit-flip decoding algorithm in section 2.4.1.1. So our simulation results are consistent with the theory.

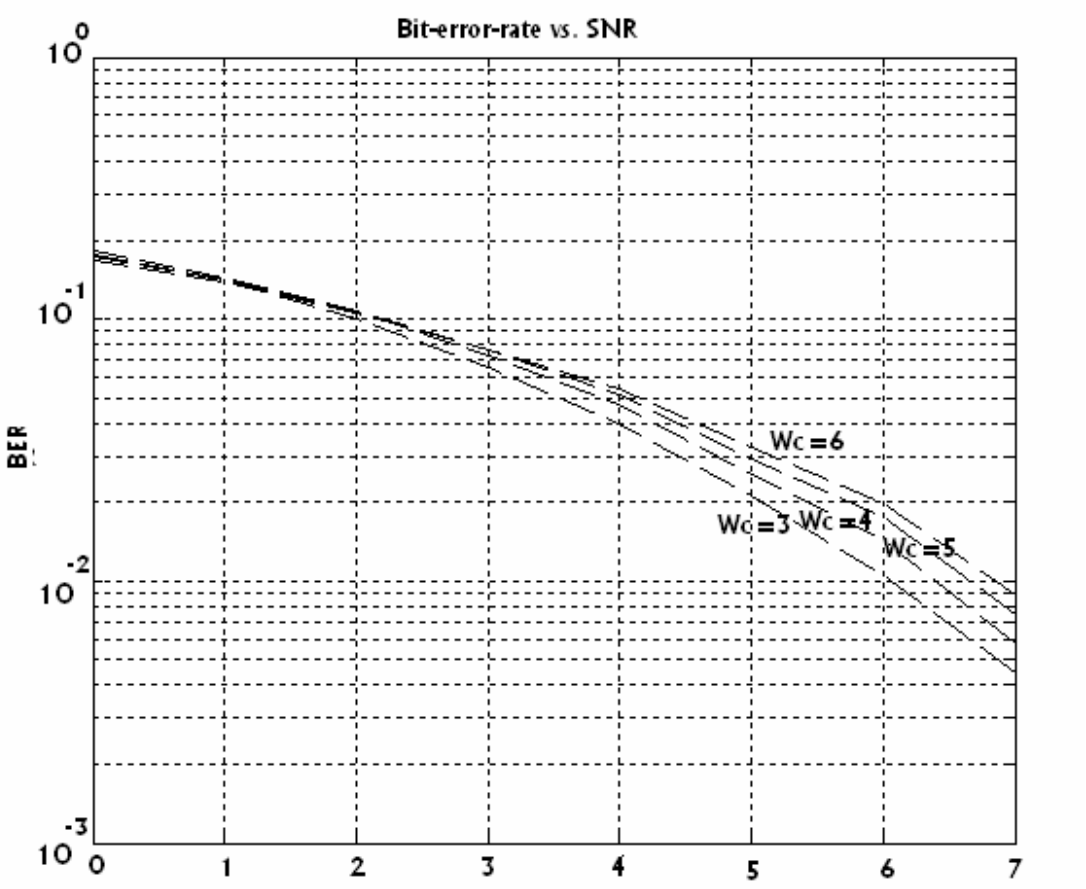


Figure 4.5 Dependence on column weight w_c for codes of blocklength 250 rate 1 / 2 using the bit- flip algorithm. (Maximum number of iterations is 10)

4.3 Dependence on the Number of Iterations

Since both the sum-product and the bit-flip algorithms are iterative decoding algorithms we expect to see better BER performance for both decoding algorithms when we increase the number of iterations. In Figures 4.6 and 4.7 we can see the effect of changing the number of iterations for the sum-product decoder and the bit-flip decoder respectively. The number of iterations is changed between 5 and 25 for both of the algorithms.

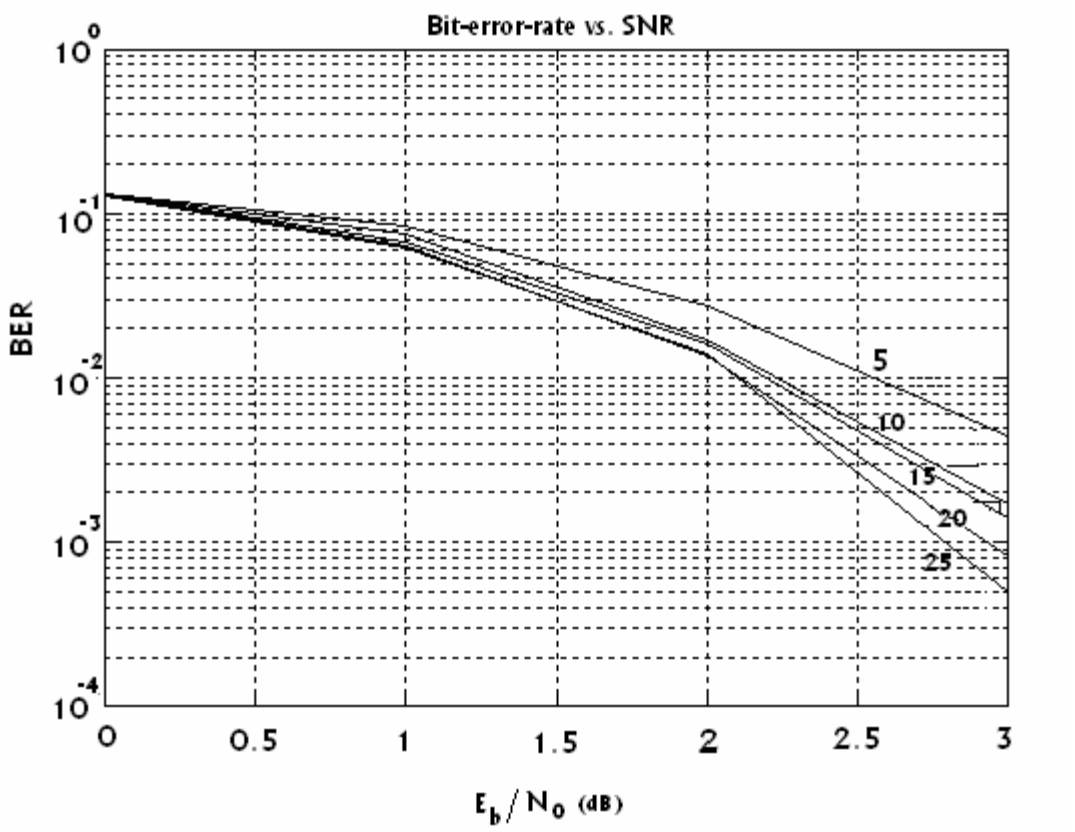


Figure 4.6 Dependence on the number of iterations for blocklength 250 rate 1/2 regular LDPC code for the sum-product decoder. (Number of iterations is changed from 5 to 25. Column weight, $w_c = 3$.)

Increasing the number of iterations results in an improvement in the BER performance of the sum-product decoder as we expect. The more we increase the number of iterations, the lower BER we achieve until reaching the error floor. But in some applications where decoding time is limited like video broadcasting we have to find the optimum number of iterations, since excessive decoding times are not allowed.

Figure 4.7 shows the BER performance of LDPC codes using the bit-flip algorithm. Note that for equal number of iterations the sum-product algorithm outperforms the bit-flip algorithm. After 15 iterations performance of the bit-flip decoder does not improve for this blocklength and rate.

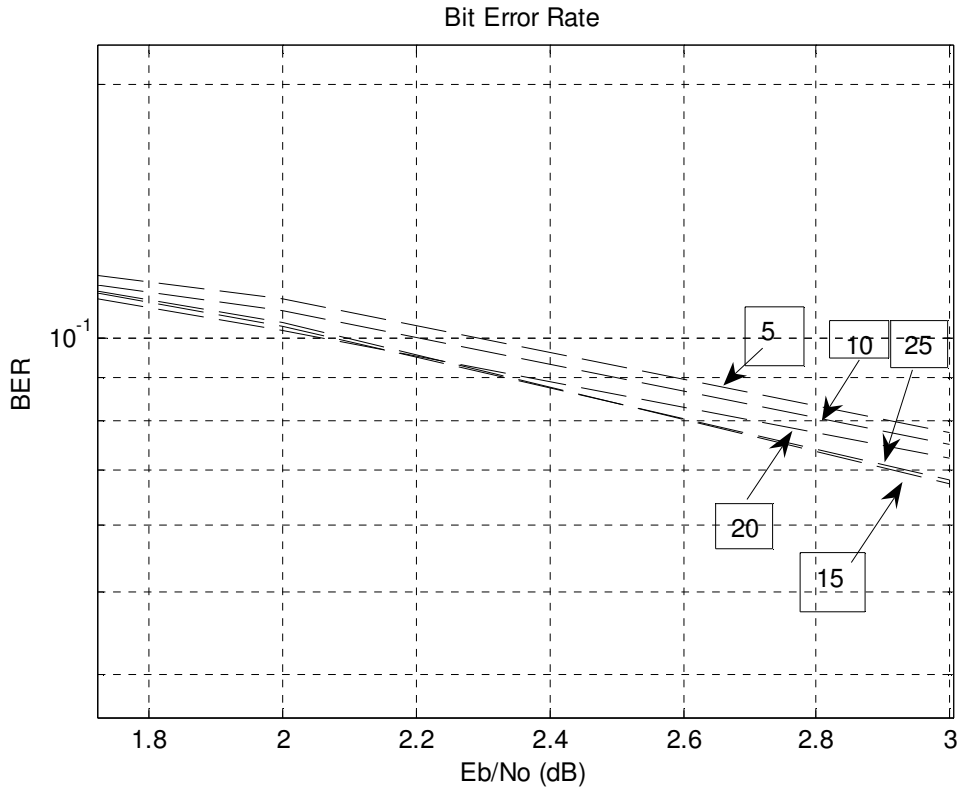


Figure 4.7 Dependence on number of iterations for blocklength 250 rate 1/2 regular LDPC code using the bit-flip decoding algorithm. (Number of iterations is changed from 5 to 25. Column weight of H is constant and 3.)

4.4 Dependence on the Information Rate

Construction of parity-check matrices is described in Chapter 3. We can specify the rate of an LDPC code by using the number of rows and columns. Define m as the number of rows and n as the number of columns (blocklength of the code) of the parity check matrix. The information rate, R , of the code can be calculated by using m and n as

$$\frac{m}{n} = \frac{n-k}{n} = 1 - R \rightarrow R = 1 - \frac{m}{n}$$

where k is the number of information bits. So we can change the information rate of the code by constructing a parity-check matrix, having suitable number of rows and columns. In our simulations we changed the information rate with this method.

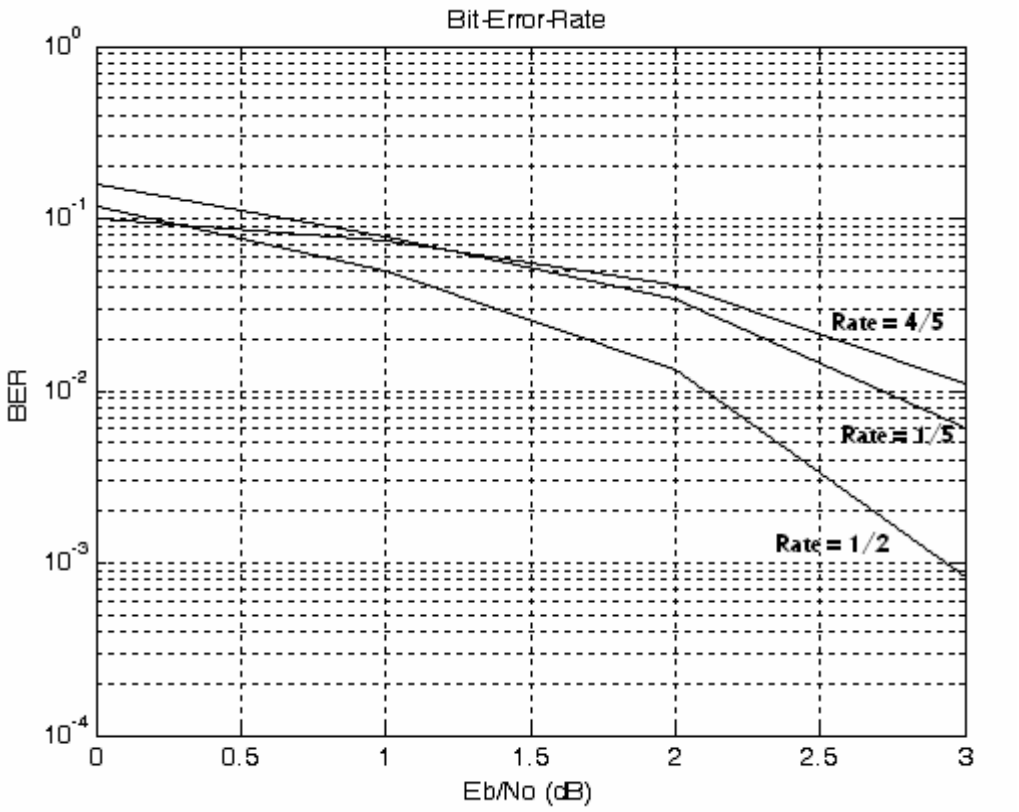


Figure 4.8 Dependence on information rate with sum-product decoding. (Blocklength 250. Maximum number of iterations is set to 20.)

In Figure 4.8 we can see the BER performance of LDPC codes at rates 1/2, 1/5 and 4/5 with the sum-product decoding algorithm. Rate 1/2 LDPC code has better BER performance than rate 4/5 code. But rate 1/5 LDPC code can not perform better than the rate 1/2 LDPC code. Although we increase the number of check bits in our code we can not achieve better BER performance for rate 1/5. In Figure 4.9 and 4.10 which are taken from [Luby22001], we see BER performances of regular and irregular LDPC codes and turbo codes for rates 1/2 and 1/4.

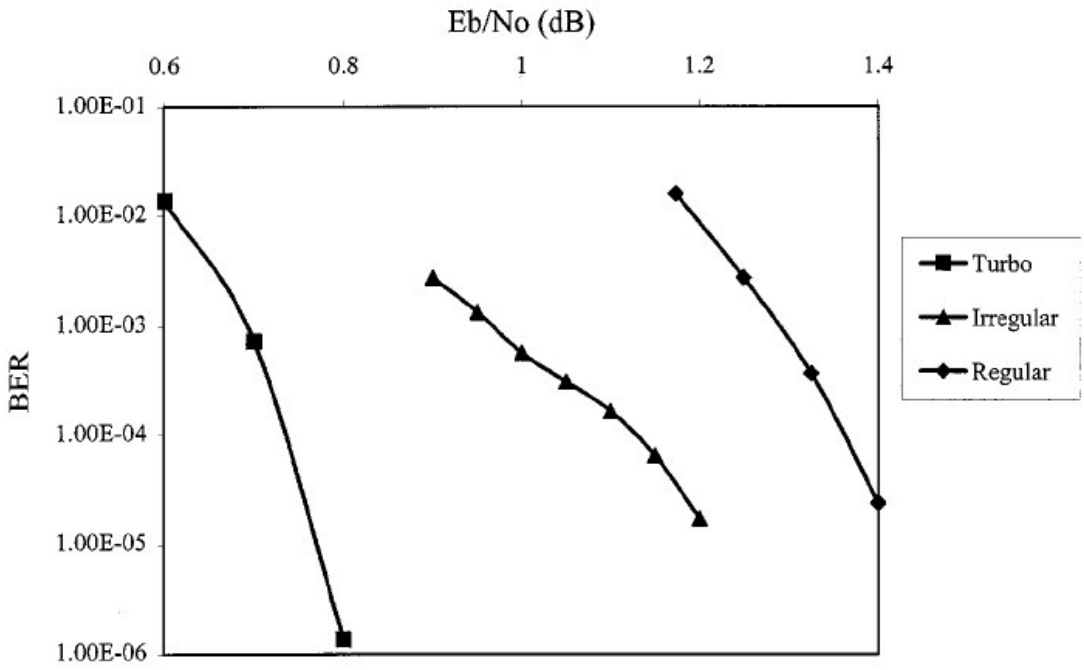


Figure 4.9 [Luby22001] Rate 1/2, blocklength 16000 regular and irregular LDPC codes and turbo codes.

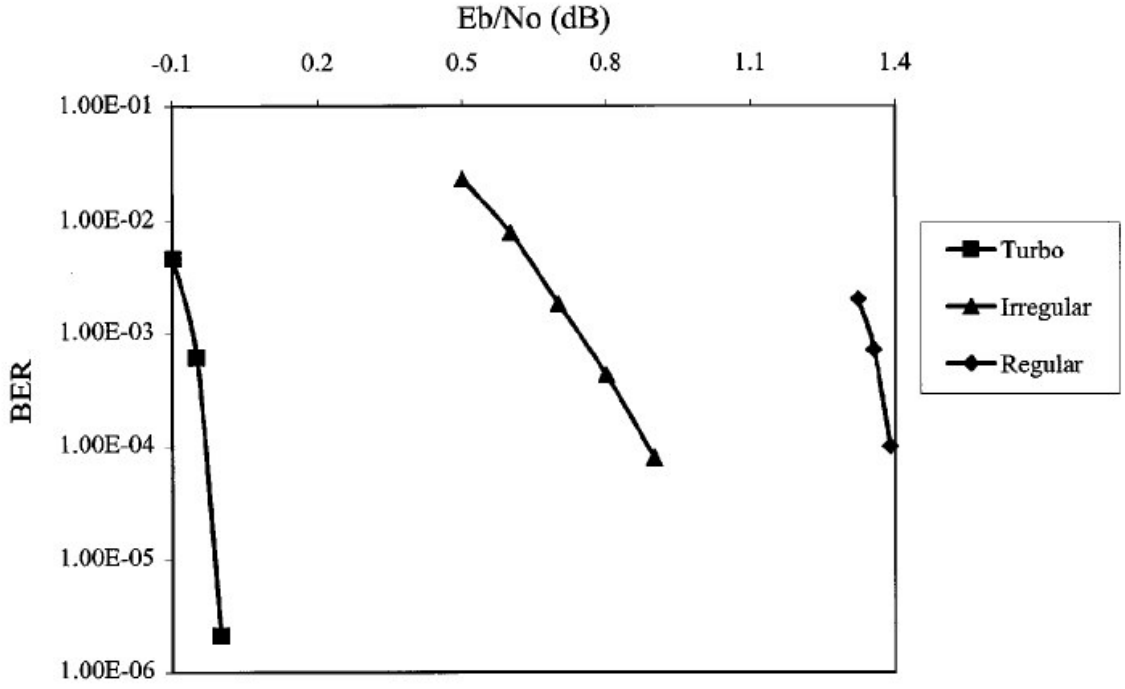


Figure 4.10 [Luby22001] Rate 1/4, blocklength 16000 regular and irregular LDPC codes and turbo codes.

Figure 4.9 and 4.10 verify our results given in Figure 4.8. Note that rate 1/2 regular LDPC code outperforms rate 1/4 regular LDPC code for a blocklength of 16000. This means that reducing the information rate of the regular LDPC code below 1/2 we can not get better BER performance.

Although we find similar results in literature this situation is quite unusual. There is no explanation in [Luby22001] to justify why rate 1/2 code outperforms rate 1/4 code. We increase the number of check bits which results a high redundancy but the BER performance of the code does not improve. This situation needs further investigation to fully understand the reason causing this behaviour.

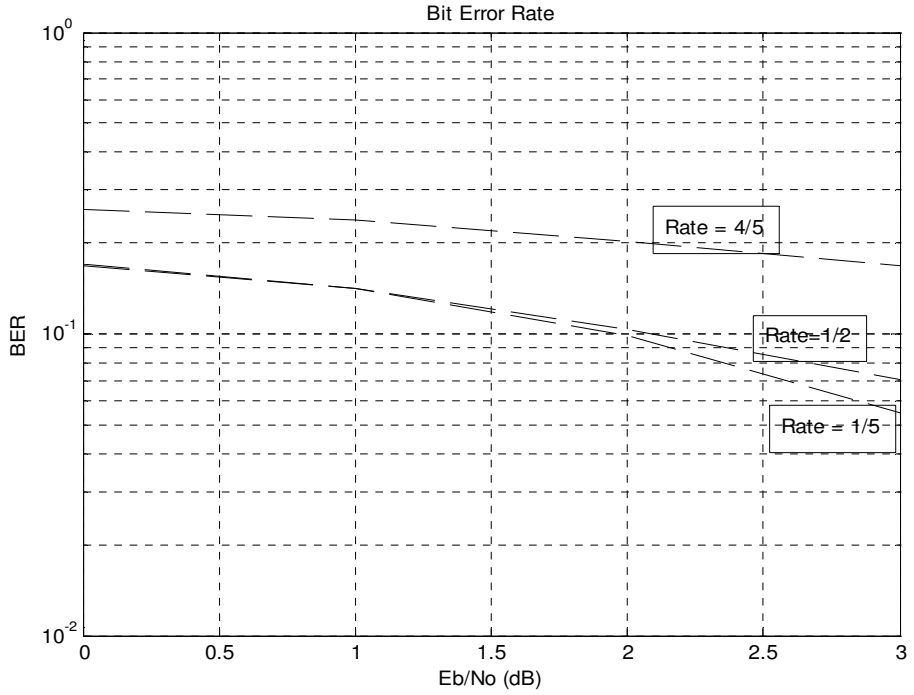


Figure 4.11 Dependence on the information rate with bit-flip decoding(Blocklength 250.Maximum iteration number is set to 20)

In Figure 4.11 we can see that the rate 1/5 LDPC code has the best BER performance whereas the LDPC code with rate 4/5 has the worst one. Performance of rate 1/2 LDPC code is between them. By reducing the information rate we actually increase

the number of check bits in our code so we expect to have better BER performance as the rate decreases. This is the case for the bit-flip algorithm.

4.5 Performance Comparison of Regular and Irregular LDPC Codes

Luby showed that [Luby2001] irregular construction of parity-check matrices, improves performance of LDPC codes for the AWGN channel. Instead of giving equal degree, w_c , to every variable node, we can have some variable nodes with degree 2, some 3, some 4 and a few with degree 20. Check nodes can also have unequal degrees but it is shown that for AWGN channel the best graphs have regular check connectivity [Luby2001].

In Figure 4.12 we compare BER performance of a regular LDPC code at rate 1/2 with blocklength 1000 with an irregular LDPC code at the same rate and the blocklength. We construct the parity-check matrix for the irregular code according to construction 2A, and use construction 1A is for constructing the regular code.

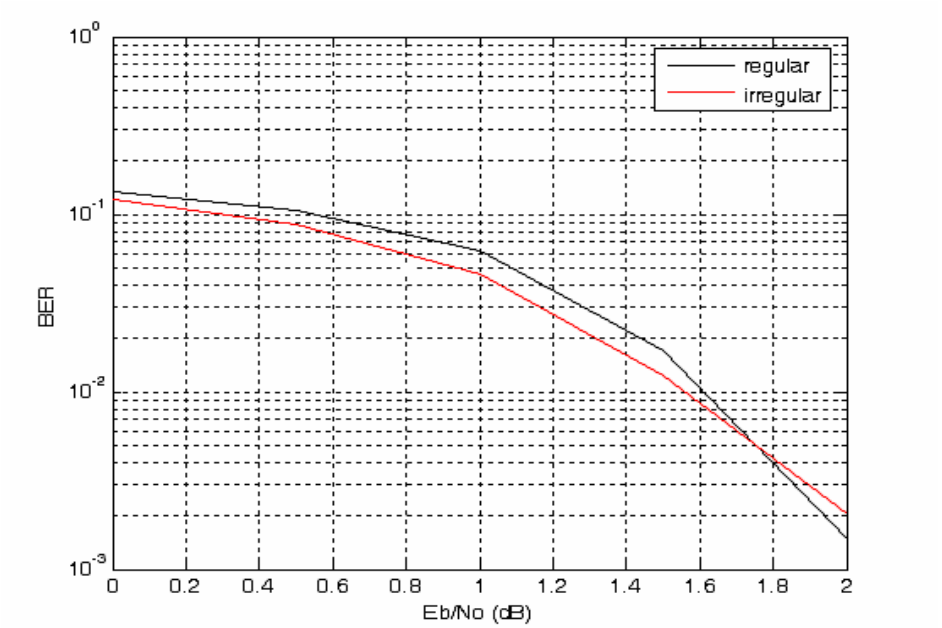


Figure 4.12 BER performances of rate-1/2 regular and irregular LDPC codes of blocklength 1000.

Luby stated that because the irregular LDPC codes have higher error floor characateristics, irregular LDPC codes have worse BER performance than those of regular ones at high SNR values [Luby2001].

In Figure 4.12 one can see that for low SNR values irregular LDPC code outperforms the regular LDPC code but after an SNR value of 1.7 dB, regular code outperforms the irregular code.

4.6 Performance of LDPC Codes Over GF(q)

MacKay and Davey investigated the performance of LDPC codes over higher order fields [Davey1998]. They proposed a method, in which the variable nodes in the factor graphs of LDPC codes are grouped together into metavariables and check nodes are similarly grouped into metachecks. The edge locations between metavariables and metachecks are determined with using a finite field GF(q) like GF(4) or GF(8). With this method parity-check matrices of LDPC codes are defined over GF(q) and binary messages are translated into GF(q) using a mapping. Mappings can be defined as in tables 4.1 and 4.2.

Table 4.1 [MacKay2004] Translation between binary and GF(4) for message symbols.

$GF(4)$	$\Leftrightarrow$	$Binary$
0	$\Leftrightarrow$	00
1	$\Leftrightarrow$	01
2	$\Leftrightarrow$	10
3	$\Leftrightarrow$	11

When messages are passed in the factor graph of the code during decoding, those messages are likelihoods of many binary variables. For example if three binary variables are grouped together, likelihoods will describe eight alternative states of the corresponding bits.

Table 4.2 [MacKay2004] Translation between binary and GF(4) for matrix entries. (An $m \times n$ parity-check matrix over GF(4) can be translated into a $2m \times 2n$ binary parity-check matrix in this way.)

$GF(4) \Leftrightarrow$		$Binary$
0	$\Leftrightarrow$	$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$
1	$\Leftrightarrow$	$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$
2	$\Leftrightarrow$	$\begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}$
3	$\Leftrightarrow$	$\begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix}$

In Figure 4.14 performance comparison of LDPC codes over GF(2), GF(4), GF(8) and GF(16) is given. Performance of turbo codes at rates of 1/3 and 1/4 can also be seen. As it can be seen from the figure an LDPC code defined over GF(16) can perform nearly one decibel better than a binary LDPC code.

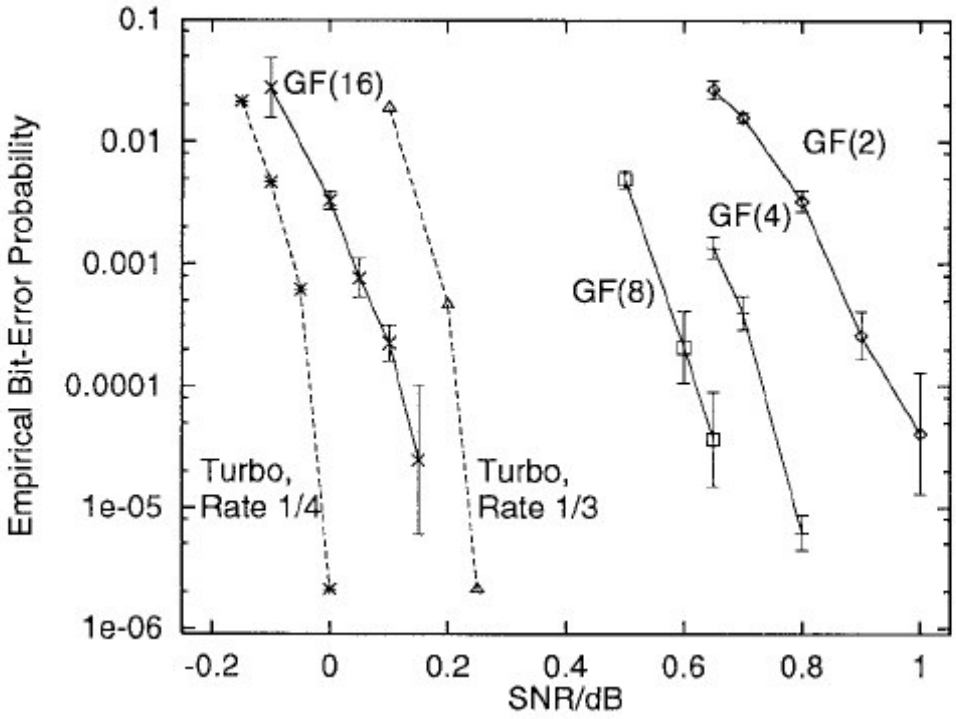


Figure 4.13 [Davey1998] Performance comparison of LDPC codes over a binary Gaussian channel over GF(2), GF(4) and GF(8). (From left to right: JPL Turbo code, rate 1/4, blocklength 65536; LDPC code with rate 0.26, average column weight 2.3, blocklength 6000 over GF(16); JPL Turbo code, rate 1/3, blocklength 49152; LDPC codes with rate 1/3, average column weight 2.5, blocklengths of 6000, 9000 and 18000, over GF(8), GF(4) and GF(2) respectively.

There is not always a monotonic improvement with increased field order. Column weight 3, rate 1/2 LDPC codes over GF(4) can outperform GF(8) codes [Davey1998]. It is a hot topic for research to understand the effects of changing the code parameters on performance. We can conclude that defining LDPC codes over higher order fields results in a significant amount of performance improvement.

CHAPTER 5

CONCLUSION

In this thesis we study LDPC codes, their construction schemes and algorithms that are used for decoding these codes. We provide simulation results to compare the BER performance of these codes under the sum-product and the bit-flip decoding algorithms for code parameters like blocklength, information rate, column weight and iteration number.

The bit-flip decoding algorithm operates on hard decisions. It is computationally simple compared to the sum-product algorithm. When we are dealing with continuous-output channels this algorithm is inferior to the sum-product algorithm because it throws away valuable information coming from the channel. The sum-product is a more complex algorithm than the bit-flip algorithm. It makes use of the soft received signal, which is vital when continuous-output channels are used. In our simulations we see that the sum-product algorithm has better BER performance compared to the bit-flip algorithm.

We observe that for both decoding algorithms, the BER performance of LDPC codes increase with increasing the blocklength of the code. But for the same blocklength, performance of the sum-product decoder is better than the performance of the bit-flip decoder. Increasing the column weight of the parity-check matrix of LDPC code resulted a performance deterioration for the bit-flip and the sum-product decoding algorithms. Main reason for this situation is the short cycles in the parity-check matrix. We can not get rid of short cycles for high column weights. Number of iterations is another important parameter for the two decoding algorithms. We observed that both the sum-product and the bit-flip algorithms are performing better with increasing number of iterations, where the sum-product decoder performs better

than the bit-flip decoder for the same number of iterations. Some simulations are performed in order to see the effects of changing the information rate for both decoding algorithms. For the bit-flip algorithm we see a better performance when we reduce the information rate of the code. Sum-product decoder performed unexpectedly when we reduce the information rate below 1/2. Instead of having better BER performance the performance of sum-product decoder deteriorated. We came across similar results in other papers, which supports our experimental results. This situation should be investigated further in future work. There are two methods in literature for increasing the performance of LDPC codes. One of them is constructing irregular matrices. We performed simulations for observing the difference between the regular and irregular LDPC codes in Section 4.5. For low SNR values irregular LDPC code performed better than the regular one. For increasing SNR values regular LDPC code outperformed the irregular one because of the high error floor characteristics of the irregular LDPC codes. The other method for better BER performance is defining LDPC codes over higher order fields. We summarized this method using the reference [MacKay2004] with some simulation results taken from the same reference.

The sum-product decoder performs better than the bit-flip decoder. The price we have to pay is the high complexity of the decoder along with the decoding delay. In applications where decoding time is not important but error correction capability is the first priority, like deep space communication, the sum-product decoder should be employed. The bit-flip is a relatively simple algorithm but can not perform as good as the sum-product algorithm. When decoding time is limited and high error correction capability is not needed, decoders with short decoding delay, like the bit-flip decoder, should be employed. The selection between the two algorithms should be done by carefully determining the needs of the communication system.

Today a serious amount of research is being made on sparse graph codes like digital fountain codes, repeat-accumulate codes, tornado codes and LDPC codes. For LDPC codes two main research topics are popular. First, computationally efficient algorithms should be developed to overcome the encoding complexity for LDPC codes. Second, because factor graphs with cycles deteriorate the performance of the

sum-product algorithm, efficient algorithms should be introduced for the codes containing cycles in their factor graphs.

REFERENCES

[Berrou1993] BERROU Claude, ADDLE Patrick, ANGUI Ettiboua, FAUDEIL Stephane A Low-Complexity Soft-Output Viterbi Decoder Architecture 0-7803-0950-2/93/3.00©1993IEEE

[Blahut1984] BLAHUT Richard E. "Theory and practice of error control codes", Addison-Wesley 1984.

[Chung2001] CHUNG S.Y., FORNEY G.D., RICHARDSON T.J., URBANKE R. "On the design of Low-density parity-check codes within 0.0045 dB of the Shannon limit" IEEE Communication Letters, vol.5, no.2, pp,58-60 February 2001.

[Davey1998] DAVEY Matthew, MacKay David "Low-density parity-check codes over GF(q)" IEEE Communications Letters, vol. 2, no.6, June 1998

[Fossorier2005] FOSSORIER M P.C. "Improved Bit-Flipping Decoding of Low-Density Parity-Check Codes" IEEE Transactions on Information Theory, vol.51, no.4, pp. 1594-1606, February 2005

[Gallager1962] GALLAGER R.G., "Low-Density Parity-Check Codes." IRE Transactions on Information Theory, vol IT-8,no1,pp,21-28,January 1962.

[Gallager1960] GALLAGER R.G. "Low-Density Parity-Check Codes" Sc.D.thesis MIT, Cambridge.September 1960

[Johnson2002] JOHNSON Sarah J. Low-Density Parity-Check Codes Design a Decoding Wiley Encyclopedia of Telecommunications 2002

[Loeliger2001] LOELIGER Hans-Andrea Factor Graphs and the Sum-Product Algorithm IEEE Transactions on Information Theory, vol.47,no.2, pp. 498-519, February2001

[Luby1997] LUBY M.G.,MITZENMACHER M.,SHOKROLLAHI M.A. SPIELMAN D.A. “Practical loss-resilient codes” in Proc. 29th Annual ACM Symp.Theory of Computing, 1997, pp.150-159.

[Luby2001] LUBY M.G.,MITZENMACHER M.,SHOKROLLAHI M.A. and SPIELMAN D.A. “Efficient erasure correcting codes” IEEE Transactions on Information Theory,vol.47,no.2, pp. 569-584, February 2001.

[Luby22001] LUBY M.G.,MITZENMACHER M.,SHOKROLLAHI M.A. and SPIELMAN D.A. “Improved Low-Density Parity-Check Codes Using Irregular Graphs ” IEEE Transactions on Information Theory,vol.47,no.2, pp. 585-598, February 2001.

[MacKay1998] MacKAY David J.,WILSON S.T., DAVEY M.C.”Comparison of Constructions of Irregular Gallager Codes” in Proc. 36th Allerton Conf. Communication, Sept. 1998

[MacKay1999] MacKAY David J. “Good Error-Correcting Codes Based on Very Sparse Matrices.” IEEE Transactions on Information Theory, vol.45, no.2, pp. 399-431, March 1999

[MacKay2004] MacKAY David J. “Information Theory, Inference and Learning Algorithms” Cambridge University Press 2004.

[Richardson2001] RICHARDSON Thomas, SHOKROLLAHI Amin, URBANKE Rüdiger “The Capacity Low-Density Parity-Check Codes under message passing decoding” IEEE Transactions on Information Theory, vol. 47, no.2, pp.599-618, February 2001

[Richardson22001] RICHARDSON Thomas, URBANKE Rüdiger “Efficient Encoding of Low-Density Parity-Check Codes” IEEE Transactions on Information Theory, vol. 47, no.2, pp.638-656, February 2001

[Tanner1981] TANNER R.Michael “A Recursive Approach to Low Complexity Codes” IEEE Transactions on Information Theory, vol.it-27, no.5, September 1981