POLITECNICO DI TORINO

III Facoltà di Ingegneria

Ingegneria delle Telecomunicazioni

Search for Improvements in Low Density Parity Check Codes for WiMAX (802.16e) Applications

Author:
Carlos Dirube García

Supervisor:
Prof. Guido Masera

July 2014

This thesis is dedicated to my parents.

For their endless love, support and encouragement

Acknowledgments

First and foremost, I have to thank my parents for their love and support throughout my life. Also my aunt who always treated me like a son, and my brother and sister for getting me to improve day by day.

I would like to sincerely thank my supervisor, Prof. Masera, for his guidance and support throughout this study and for his confidence in me. And also to PhD student Carlo Condo for his selfless help and patience.

To my classmates who made each day different, who helped me countless times and made me push myself to the maximum.

I need to especially thank my friend Álvaro, who let me use his computer to run some simulations for this thesis when my computer crashed.

Finally I would like to thank my Erasmus friends for making this exchange experience one of the best years of my life, for motivating me, caring about me and supporting me in my bad moments. I cannot list all the names here, but you will always be on my mind.

Contents

Figures List

Tables List

1 Introduction

In this chapter the reasons that have led to the proposal and development of this project are introduced. Its objectives are also defined in base of this. Finally, in a third section, the structure of the report will be explained, showing a brief explanation of each one of the chapters in which it is divided.

1.1 Problem statement

Since some decades ago, human life is bound unconditionally to technology, specially for communications, such as TV, Internet, etc. At present, almost all this technology and devices are based on digital systems, that work with binary digits (bits). The transmission of these bits between different devices is done along cables (phone line between modems...) or electromagnetic waves (Wi-Fi, 3G...) and in both cases the transmission is affected by environmental characteristics such as obstacles, temperature, rain, etc. To fix the possible errors in the transmission, error-correction codes are needed. These codes introduce redundant bits in the transmitter to help the receptor to get the original information.

Low Density Parity Check (LDPC) codes are a type of forward error-correction codes, first proposed in the 1962 PhD thesis of Robert G. Gallager at *Massachusetts Institute of Technology* (MIT) [1]. This codes remained undiscovered during a few decades, due to the computational demands of simulation. During those years structured algebraic block and convolutional codes were used but they showed a significant gap to Shannon limit.

Shannon theory tells us that all “random” codes are “good codes”. Unfortunately "random" codes are not easily decodable. For many years, researchers worked under the assumption that they could not find structured codes with good decoding performance and efficiency.

In 1993 structured “turbo codes” were proposed by Berrou, Glavieux and Thitimajshima [2]. “Turbo codes” involved very little algebra and employed iterative and distributed al-

gorithms, but the most important point, they reduced the performance gap to Shannon limit.

In 1997, McKay and Neal rediscovered LDPC codes while studying "turbo codes" [7], [8], and demonstrated that LDPC codes could also show good *Bit Error Rate* (BER) performance for low *Signal to Noise Ratio* (SNR) values, although not as good as "turbo codes" at that moment. These developed codes showed similar properties than "turbo codes", but it was soon recognized that they were a rediscovery of Gallager's codes [1]. Other researchers produced later new irregular LDPC codes which outperformed the best turbo codes.

Nowadays LDPC are used in several radio and wired communication systems. The *Digital Video Broadcasting by Satellite - Second Generation* (DVB-S2) uses them since, in 2003, they beat Turbo Code proposals using a much more efficient decoder architecture. LDPC codes are also used for 10GBase-T Ethernet and are also part of the Wi-Fi (*Institute of Electrical and Electronics Engineers* (IEEE) 802.11) standard as an optional part of IEEE 802.11n and IEEE 802.11ac.

Quasi-Cyclic LDPC (QC-LDPC) codes are a special type of LDPC codes that consist of *Circulant Permutation Matrices* (CPMs) and zero matrices of the same size. These LDPC matrices are highly structured and can be characterized by the submatrix size and the circular-shift value of each CPM. QC-LDPC codes are receiving significant attention by researches since their structures ease the hardware implementation (parallel decoder architectures with high decoding throughput) and have excellent error performance over noisy channels.

In this thesis the research is focused on the QC-LDPC matrices used in standard IEEE 802.16e, *Worldwide Interoperability for Microwave Access* (WiMAX). The QC-LDPC codes used in this technology use identity matrices as CPMs.

The simplicity of the structure of the previously mentioned WiMAX matrices, that are formed by circularly shifted right smaller identity matrices, suggests the possibility of generating alternative QC-LDPC matrices with no much more complex structure or decoding hardware than WiMAX ones but with better error performance. Besides, lots of researchers have studied the properties of LDPC codes and they have also experimented with them in some ways (most of them increasing minimum cycles level) in order to improve the performance, but almost none of them has applied these modifications to this specifically WiMAX technology.

Even if this WiMAX technology uses "simple" QC-LDPC codes because the CPM is a simple identity matrix, these codes are characterized for using a structured right part that limits the possible successful modifications over the code matrix. So, it is interesting to see how some of the modifications from other research papers could be applied over this special structured matrix. The motivations of this work are then the possibility of

implementing some of these modifications over this specific WiMAX technology and also to test some new modifications such as submatrix modifications, barely seen in previous works.

1.2 Project objectives

The main objective of this thesis is to make a wide study of the possibilities of generating new LDPC matrices with good error performance by modifying WiMAX ones, changing their structure or the internal distribution of their submatrices, without making the decoding hardware much more complex.

A wide range of modifications in the matrices will be analyzed, implemented and simulated to see the error performance in transmission along noisy channel. The study will be focused on the smallest matrices used in decoding of IEEE 802.16e (M=288 rows x N=576 columns) with ratio $R = 0.5$, but some of the modifications will be also studied in bigger matrices and matrices with different code ratio.

The environment used to develop the new matrices will be MATLAB® [1] due to the fact that it is an environment specially designed to work and operate over matrices and has many special functions and features to do it. Besides it is also good to plot the results of the simulation processes. Even so, for some of the algorithms developed during this thesis, a faster compiled language as C could have been better in order to save some time.

To simulate the BER performance over SNR values, a MATLAB simulator is not suitable since MATLAB does not show good performance when working with loops, so a much faster C software developed by the *Department of Electronics and Telecommunications* (DET) of the *Politecnico di Torino* [2] will be used.

1.3 Structure of the report

In the next lines the structure of the report is presented, with a brief description of the chapters it is composed of:

- In the Chapter 2 the main theoretical concepts needed to understand this report will be described, as well as the principal bases that support the line of research of interest to this work. Besides, some different scientific papers carried out previously

[1] *MATrix LABoratory. http://www.mathworks.com/products/matlab*
[2] *http://www.det.polito.it/*

by various researchers will be referenced to get an idea of the state of the art and progress of the research in this field of study.

- The Chapter 3 will expose a wide sequence of modifications done over the WiMAX original code matrices in order to get better BER performance and the BER simulation results will be exposed as well. The BER simulation results will be briefly analyzed for each test and some ideas about the hardware cost will also be introduced.
- At the end, in the Chapter 4, the conclusions reached after the analysis and simulations will be presented. After that, a summary of the thesis and the processes developed in it will be added and finally, a series of interesting future lines of research will be exposed.
- Finally, in the Annex A a short explanation of each one of the MATLAB codes developed to work over the matrices will be also incorporated.

2

State of the Art

This chapter contains the introduction of each of the principal elements of this thesis. In this way, the reader can acquire the necessary knowledge that allows him to understand and interpret correctly the different points covered through the following pages. Besides the actual situation of this line of research will be presented and analyzed.

2.1 Noisy channel

All information carried through a channel is affected by noise. In radio communications, it is easy to understand that environment can affect data by damaging the transmission (white noise, rain...), but also in cable communications such as phone line, the signal can suffer cross-talk interference with other lines. Because of that, every time some information is transmitted there is a certain probability of not receiving exactly the same information that was sent.

The *Binary Symmetric Channel* (BSC) model says that each transmitted bit has got a probability *1-p* of being transmitted correctly and a probability p of being transmitted incorrectly.

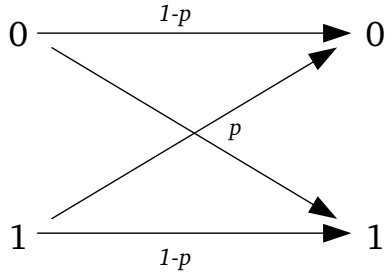


Figure 2.1: BSC

A possible solution would be using more or better physical resources to transmit our signals (more transmitted power, more telephone cables), but these improvements usually increase the cost of the communication channel. The advances in information theory and coding theory give us another alternative that allows us to detect and correct some errors introduced in the transmission by the channel. It basically consists on adding redundant information to our message before sending and discarding it after receiving.

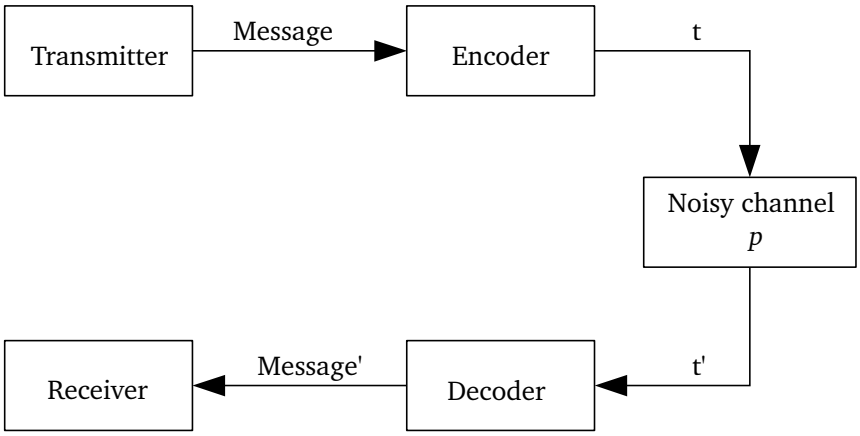


Figure 2.2: Transmission System

As shown in Figure 2.2 an encoder is added before sending the message through the channel. This encoder introduces redundant information to the message by following some coding theory, to get the transmitted message *t*. This message travels through the channel which adds noise to the message and the transmitted message *t* becomes into *t'*. Then the decoder works to recover the original message using the redundant information added by the encoder. At the end, the message received may have some errors, but the BER will be lower than the probability of error p of the channel.

2.2 Error-Correcting Codes

One of the most simplest error-correcting codes to introduce redundant information in the message could be, for example, one that replicates each bit of the original information several times. Even if this code would work, the relation between the improvement in the BER and the decrease of the information throughput would not be very good, and researchers soon noticed it could be much better.

To get better solutions, the redundant information is not applied to each particular bit, but to blocks of bits. The codes that work over blocks are called *block codes*. These

codes add parity-check bits in such a way that codewords (parts of the coded messages) are sufficiently distinct, one from another, that the transmitted message can be correctly inferred at the receiver, even when some bits in the codeword are corrupted during the transmission over the channel.

Block codes convert pieces (sequences of bits) of size K into pieces of size N adding $N - K$ extra bits. These extra bits are the result of calculating the parity of some of the bits in the information word. The bits included in each parity calculation are designed by the code. Each code chooses some or other bits, so there are lots of different possible codes. Also each code has got a size of information word K, a size of codeword N, and a code rate that is $R = K/N$.

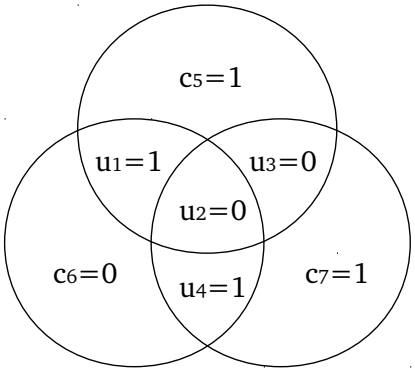


Figure 2.3: Encoding representation

In Figure 2.3 a simple encoding is shown. The information word of $K = 4$ bits $[u_1\ u_2\ u_3\ u_4]$ is coded with $N - K = 3$ redundant bits $[c_5\ c_6\ c_7]$ to get the codeword of $N = 7$ bits $[c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7]$ where $[c_1\ c_2\ c_3\ c_4] = [u_1\ u_2\ u_3\ u_4]$ and $[c_5\ c_6\ c_7]$ are the result of the calculation to keep the parity in each circle. These bits are calculated with the next parity-check equations (all the thesis work will be done over *Galois Field of two elements* (GF(2))[1]):

$$\begin{aligned} c_5 &= u_1 \oplus u_2 \oplus u_3 \\ c_6 &= u_1 \oplus u_2 \oplus u_4 \\ c_7 &= u_2 \oplus u_3 \oplus u_4 \end{aligned} \tag{2.1}$$

These linear codes exposed here, can be also written in terms of matrices. For the previous example the encoder process would be:

[1]Galois Field of two elements means 0+0=0, 0+1=1, 1+0=1 and 1+1=0

$$[c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7] = [u_1\ u_2\ u_3\ u_4] \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}}_{\text{G}} \tag{2.2}$$

Where G is the *generator matrix*. As shown, the original string of information u times the generator matrix G gives us the code word c:

$$c = uG \tag{2.3}$$

The check process in the decoder would be:

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}}_{\text{H}} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.4}$$

Where H is called the *parity-check matrix* and can be obtained from G. Each row of H corresponds to a parity-check equation and each column of H corresponds to a bit in the codeword. The codeword $t' = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7]$ is a valid codeword for the code with parity-check matrix H if and only if it satisfies the matrix equation:

$$Ht'^T = 0 \tag{2.5}$$

Depending on the code rate, it means, the code N and K chosen, the code *Hamming distance* (minimum number of bit positions in which two codewords differ) is greater or lower. A code with minimum *Hamming distance* d_{min} can always detect t errors whenever:

$$t < d_{min} \tag{2.6}$$

The number of bit flips that a decoder can correct depends on the strategy of decoding. The most intuitive strategy is to compare the codeword received with all the codewords accepted and choose the one with minimum binary distance to the one received. This is called *Maximum likelihood* (ML) and in general it can always correct codewords with e bit flips if:

$$e \leqslant \lfloor (d_{min} - 1)/2 \rfloor \tag{2.7}$$

This decoding strategy requires to compare the codeword received with all accepted codewords but for codes with thousands of message bits in a codeword it becomes too computationally expensive. To reduce that complexity other decoding strategies were developed. One of them is the one for LDPC codes.

2.2.1 LDPC Codes

Low Density Parity Check codes are, as their name suggests, block codes with parity-check matrices that contain only a few number of non-zero entries. This property of LDPC codes makes both the encoding and the decoding complexity increase only linearly with code length.

The biggest difference between LDPC codes and classical block codes is how they are decoded. Instead of using ML method, LDPC codes are decoded iteratively using graphical representations of their parity-check matrix H.

A LDPC code is characterized by its *row weight* (w_r) and its *column weight* (w_c). In *regular* codes this values are constant for all rows and columns respectively. If they are not, the code is called *irregular*. As it is said in [3], LDPC codes are good codes, even their simple construction, while w_r/N goes to zero, because of their good *Hamming distance*.

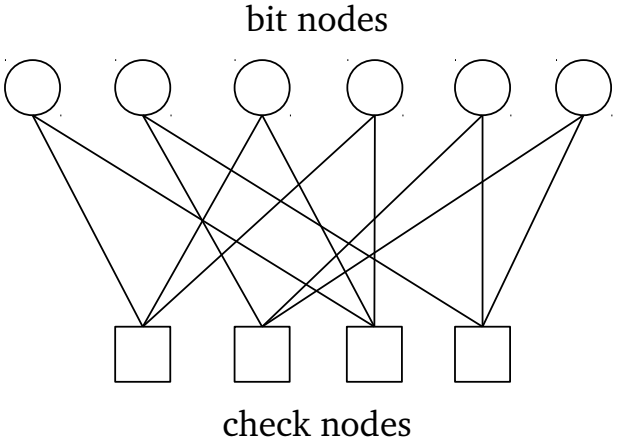


Figure 2.4: Tanner graph

LDPC codes are often represented in graphical form by *Tanner graph*. In this graph, defined by H matrix, the bits of codeword received (called *bit nodes*) and the parity-check

equations (called *check nodes*) are represented. An edge joins a bit node to a check node if that bit is included in the corresponding parity-check equation and so the number of edges in the Tanner graph is equal to the number of ones in the parity-check matrix.

In Figure 2.4 an example of Tanner graph is shown. It corresponds to a code of code length $N = 6$ and parity check bits $N - K = 4$.

Researchers in LDPC codes look for good codes performance, but they also look for code structures that allow to reduce the hardware and software encoding and decoding complexity. In this line of research QC-LDPC appeared.

2.2.2 QC-LDPC Codes

A QC-LDPC code is a type of LDPC constructed with circulant permutation matrices of size p x p [4]. These codes have the possibility of being encoded with shift registers in linear time. An example of a QC-LDPC code is shown in Figure 2.5, where $A_{p(i,j)}$ is the matrix A, of size p x p, circularly shifted $p(i, j)$ (mod p).

$$\begin{bmatrix} A_{p(0,0)} & A_{p(0,1)} & A_{p(0,2)} & \cdots & A_{p(0,N/p-1)} \\ A_{p(1,0)} & A_{p(1,1)} & A_{p(1,2)} & \cdots & A_{p(1,N/p-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{p((N-K)/p-1,0)} & A_{p((N-K)/p-1,1)} & A_{p((N-K)/p-1,2)} & \cdots & A_{p((N-K)/p-1,N/p-1)} \end{bmatrix}$$

Figure 2.5: Example of QC-LDPC code H matrix

2.2.3 WiMAX LDPC codes

The WiMAX standard (IEEE 802.16e) is a wireless communications standard used in small to medium distances in urban areas (bellow 10 Km range). This standard uses LDPC codes to encode and decode the signal. LDPC codes can be very demanding from a computational perspective, so they are still implemented using dedicated hardware based on *Application-Specific Integrated Circuit* (ASIC) solutions.

The *Forward Error Correcting* (FEC) system of WiMAX standard is based on a special class of LDPC codes [IEEE P802.16e/D12, 2005] characterized by a sparse binary block parity-check matrix H of the form shown in Figure 2.6.

$H_{(N-K)xN} = [H1|H2] =$

$$\left[\begin{array}{ccc|cccccccc}
I_{p(0,0)} & \cdots & I_{p(0,K/p-1)} & I_{p(0,K/p)} & I & 0 & 0 & 0 & 0 & \cdots & 0 \\
I_{p(1,0)} & \cdots & I_{p(1,K/p-1)} & I_{p(1,K/p)} & I & I & 0 & 0 & 0 & \cdots & 0 \\
I_{p(2,0)} & \cdots & I_{p(2,K/p-1)} & I_{p(2,K/p)} & 0 & I & I & 0 & 0 & \cdots & 0 \\
I_{p(3,0)} & \cdots & I_{p(3,K/p-1)} & I_{p(3,K/p)} & 0 & 0 & I & I & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & & \ddots & \ddots & & \vdots \\
\vdots & \ddots & \vdots & \vdots & 0 & 0 & \cdots & 0 & I & I & 0 \\
I_{p((N-K)/p-2,0)} & \cdots & I_{p((N-K)/p-2,K/p-1)} & I_{p((N-K)/p-2,K/p)} & 0 & 0 & \cdots & 0 & 0 & I & I \\
I_{p((N-K)/p-1,0)} & \cdots & I_{p((N-K)/p-1,K/p)} & I_{p((N-K)/p-2,K/p)} & 0 & 0 & \cdots & 0 & 0 & 0 & I
\end{array}\right]$$

Figure 2.6: Structure of a WiMAX LDPC H matrix

The H matrix of Figure 2.6 is composed of two differentiated parts: $H1$ and $H2$. The $H1$ part is a sparse pseudo randomly designed matrix of $(N-K)$ x K composed of quasi-random circularly shifted right identity sub-matrices with dimension p x p, or zero matrices of the same size. The distribution of zero and non-zero matrices, and also the shift indices are collected in the Base-Matrix. The dimension of the identity matrices goes from $p = 24$ in the smallest matrix of $N = 576$ to $p = 96$ in the biggest matrix of $N = 2304$. This $H1$ matrix correspond to the information bits.

The $H2$ part is formed by an special column and a double-diagonal *staircase* structured matrix and correspond to parity bits. The special column, the one in the position (i, K) is the first column of the parity-check bits and has got a weight of 3 for all WiMAX LDPC matrices. It is formed of two identity matrices shifted by the same index in the first and last row (positions $(0, K)$ and $((N-K)-1, K)$) and another matrix in a position between $(1, K)$ and $((K-N)-2, K)$ with an unpaired shift value. For more information, see [6].

The special *staircase* structure of $H2$ simplifies the encoding process and allows to encode with a simple accumulate process using the previous calculated values.

$$c_n = c_{n-1} + \sum_{k=0}^{K} H_{n,k} s_k \tag{2.8}$$

Where c_n is the desired parity-check value and c_{n-1} is the parity-check value calculated before. $H_{n,k}$ are the values of H matrix of row n and s_k are the information bits of source information word.

In the Table 2.1, the properties of all the 76 different IEEE 802.16e LDPC codes are shown. The submatrices size grows 4 for each different H matrix size.

Table 2.1: Properties of LDPC codes of IEEE 802.16e WiMAX standard

Code length (N)	Submatrix size (p)	Information bits (K)			
		$R = 1/2$	$R = 2/3$	$R = 3/4$	$R = 5/6$
576	24	288	384	432	480
672	28	336	448	504	560
768	32	384	512	576	640
864	36	432	576	648	720
960	40	480	640	720	800
1056	44	528	704	792	880
1152	48	576	768	864	960
1248	52	624	832	936	1040
1344	56	672	896	1008	1120
1440	60	720	960	1080	1200
1536	64	768	1024	1152	1280
1632	68	816	1088	1224	1360
1728	72	864	1152	1296	1440
1824	76	912	1216	1368	1520
1920	80	960	1280	1440	1600
2016	84	1008	1344	1512	1680
2112	88	1056	1408	1584	1760
2208	92	1104	1472	1656	1840
2304	96	1152	1536	1728	1920

2.3 Sum Product Algorithm (SPA)

The WiMAX codes used in this thesis are decoded using the *Sum-Product Algorithm* (SPA). This is a soft decision message-passing algorithm which accepts the probability of each received bit as input.

The input bit probabilities are called *a priori* probabilities for the received bits because they were known before start decoding. The bit probabilities returned after the decoding process are called *a posteriori* probabilities. In SPA both probabilities are expressed as *Log Likelihood Ratio* (LLR)s.

$$L(x) = log_e \left(\frac{p(x=0)}{p(x=1)} \right) \tag{2.9}$$

Where $p(x = 0)$ is the probability of the bit received of being a 0 and $p(x = 1)$ the

probability of being a 1. The benefit of doing this is that when probabilities need to be multiplied, log-likelihood ratios need only be added, reducing implementation complexity.

If $p(x = 0)$ is greater than $p(x = 1)$, $L(x)$ is positive and the greater the difference between the probabilities, the more sure of being a 0 and the bigger the positive value for $L(x)$. If $p(x = 1)$ is greater than $p(x = 0)$, $L(x)$ is negative and the more sure of being a 1, the bigger the negative value of $L(x)$.

From equation 2.9 it can be easily extracted:

$$e^{L(x)} = \frac{p(x = 0)}{p(x = 1)} = \frac{p(x = 0)}{1 - p(x = 0)} \tag{2.10}$$

$$(1 - p(x = 0))e^{L(x)} = p(x = 0) \tag{2.11}$$

$$e^{L(x)} - p(x = 0)(1 + e^{L(x)}) = 0 \tag{2.12}$$

$$p(x = 0) = \frac{e^{L(x)}}{1 + e^{L(x)}} \tag{2.13}$$

In the same way it is not difficult to get:

$$p(x = 1) = \frac{e^{-L(x)}}{1 + e^{-L(x)}} = \frac{1}{1 + e^{L(x)}} \tag{2.14}$$

The iterative algorithm send messages between variable nodes (or bit nodes) and check nodes and make calculations in both of them. In the variable nodes the calculations are these:

$$L_{out} = L_c + \sum_{j=1}^{j=d_v-1} L_j \tag{2.15}$$

Where L_{out} is the value sent from the variable node to the check nodes connected to it. L_c is the log-likelihood ratio obtained from the channel for that variable, L_j are the values received from the check nodes connected to it, and d_v is the degree of the node (the number of parity-check nodes connected to it). In the first iteration $L_{out} = L_c$. A representation of it is shown in Figure 2.7.

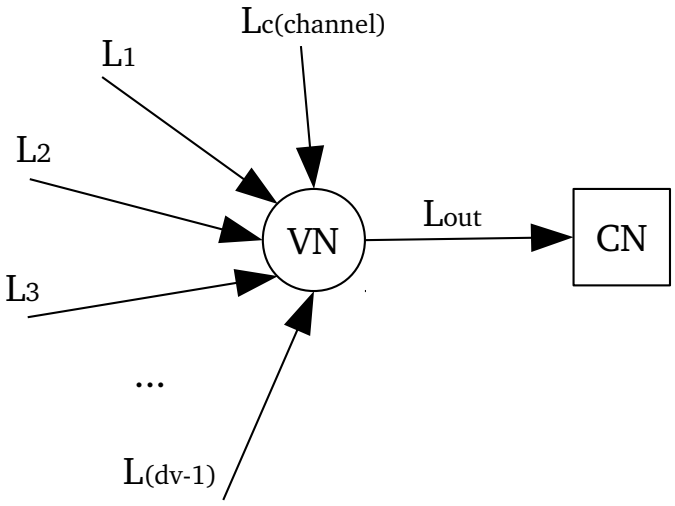


Figure 2.7: Variable Node

The computation of the check node is more complex. Lets do a change of variable:

$$\widehat{x} = \begin{cases} +1, & x = 0 \\ -1, & x = 1 \end{cases} \tag{2.16}$$

The parity check before the changing was:

$$x_1 + x_2 + x_3 + ... + x_{dc-1} + x_{out} = 0 \tag{2.17}$$

After the changing it is:

$$\widehat{x}_1\widehat{x}_2\widehat{x}_3...\widehat{x}_{dc-1}\widehat{x}_{out} = 1 \tag{2.18}$$

What is the same:

$$\widehat{x}_1\widehat{x}_2\widehat{x}_3...\widehat{x}_{dc-1} = \widehat{x}_{out} \tag{2.19}$$

In terms of probabilities, the check node calculates the expected value:

$$E[\widehat{x}_{out}] = E[\widehat{x}_1\widehat{x}_2\widehat{x}_3...\widehat{x}_{dc-1}] = E[\widehat{x}_1]E[\widehat{x}_2]E[\widehat{x}_3]...E[\widehat{x}_{dc-1}] \tag{2.20}$$

Then:

$$E[\widehat{x}] = (+1)p(\widehat{x} = 1) + (-1)p(\widehat{x} = -1) = p(\widehat{x} = 1) - (1 - p(\widehat{x} = 1)) \tag{2.21}$$

Using the results of equations 2.13 and 2.14:

$$E[\widehat{x}] = p(\widehat{x} = 1) - (1 - p(\widehat{x} = 1)) = \frac{e^L - 1}{e^L + 1} = \frac{e^{L/2} - e^{-L/2}}{e^{L/2} + e^{-L/2}} = tanh\left(\frac{L}{2}\right) \tag{2.22}$$

Using equation 2.20:

$$tanh\left(\frac{L_{out}}{2}\right) = \prod_{j=1}^{j=d_c-1} tanh\left(\frac{L_j}{2}\right) \tag{2.23}$$

So:

$$L_{out} = 2arctanh \prod_{j=1}^{j=d_c-1} tanh\left(\frac{L_j}{2}\right) \tag{2.24}$$

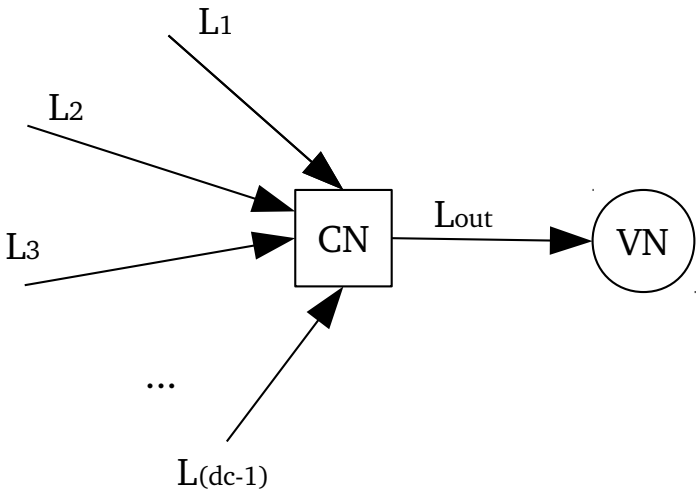


Figure 2.8: Check Node

A representation of this is shown in Figure 2.8, where L_{out} is the value sent from the check node to the variable nodes connected to it. This value is different depending the destiny of the L_{out} because the variable node which is going to receive the L_{out} is not included in the calculation of the value L_{out} it receives. L_j are the values received from the variable nodes connected to the check node (unless the one which it sends the L_{out} to), and d_c is the degree of the check node (the number of variable nodes connected to it unless the one which it sends the L_{out} to).

The iterative process of decoding finishes because one of these events:

- Valid codeword. If all parity-check equations are satisfied, it means the codeword sent has been found. The decoding process has been successfully.
- Fixed point. The iterative process gets stuck and the values don't change anymore. It means the process failed.
- Maximum iterations. The iterative process reaches the limit of maximum allowed iterations and it fails.

  In the simulations of this report, the iterative processing used will be a normalized version of SPA with factor $\rho = 0.75$.

2.4 Previous work

Since MacKay and Neal rediscovered the Gallager's LDPC codes [1] in the 1990s [7], [8], and showed that they could approach very near the Shannon's capacity[2] outperforming the best "turbo codes", lots of researchers started to work with these codes.

Most LDPC code designs rely on random construction of the parity check matrix [8]. However, these codes are not structured and that makes the code difficult to describe efficiently and hard to implement.

Opposed to random construction of LDPC codes, structured QC-LDPC codes were proposed in early 2000s by Tanner in [9] and [10]. This particular type of codes is of interest because they provide good performance and are hardware friendly. It has been shown that the regular quasi-cyclic LDPC codes can achieve comparable performance to randomly constructed codes if the codeword length is less than 10000 bits [11]. Moreover, the decoder implementation of QC-LDPC codes significantly simplifies the wire interconnections and memory address generation. These codes can also be partitioned and implemented with partially parallel decoder architectures, which achieve an efficient trade off between

[2]Shannon's theorem tells there is a maximum channel capacity C to be able to transmit information through a channel with an acceptable BER

Very Large Scale Integration (VLSI) complexity and decoding throughput.Because of these reasons, nowadays lots of researchers are working with these special codes.

Some research work shows improvements in the performance of WiMAX codes by modifying the decoder hardware or the decoding structures. Some other works also show improvements in BER performance by modifying the shift values or distribution of the WiMAX code Base Matrix [12] and also by modifying row and column weights [13].

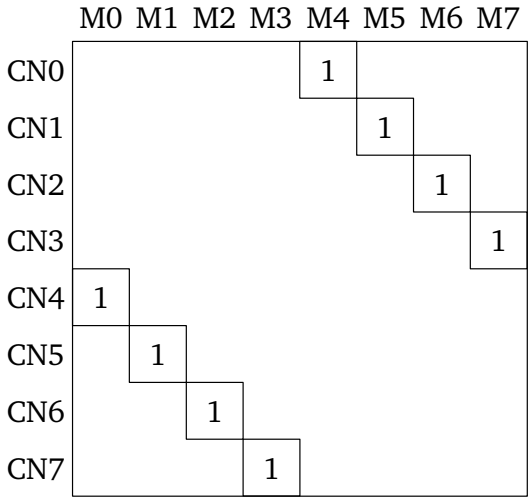


Figure 2.9: QC-LDPC submatrix with $shift\_value = 4$

In this report, some of those ways would be explored but working specifically over the WiMAX codes. Besides an study of the possible modifications of submatrices structure will also be done. The submatrix distribution used nowadays is shown in Figure 2.9. The distribution is an ordinary identity matrix of p x p circularly shifted right by some value between 0 and $p - 1$. This structure has got a very simple pattern that matches the information values stored in a memory $M0$,$M1$,$M2$... with the parity check nodes $CN0$, $CN1$, $CN2$... The simplicity of this pattern allows to implement this match network in hardware by simple barrel shifters[3].

In Figure 2.10 a very simplified sketch of the data path of information bits to arrive to the parity check nodes is shown. The mentioned barrel shifters would be the "Shift Network" of the sketch.

In this thesis the submatrix distribution would be studied and several modifications in its structure would be tested to try to outperform the actual WiMAX codes by transforming that "Shift Network" of Figure 2.10 without making it much more complex than simply

[3]A barrel shifter is a digital circuit that can shift a data word by a specified number of bits in one clock cycle

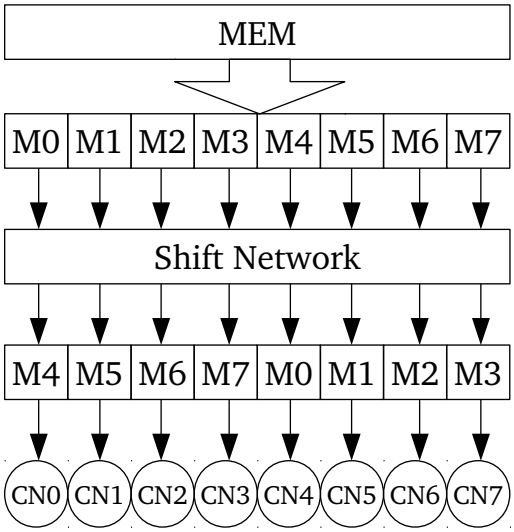


Figure 2.10: QC-LDPC data path

barrel shifters. This specific line of research has not been extensively covered by researchers, so the simulation results of these modifications are totally unknown.

3

Matrices modifications and simulations

This is the main chapter of this thesis and contains a sequence of studies of some of the possible modifications that can be made on the submatrix distribution and Base Matrix distribution of one of the smallest matrices of code length $N = 576$ and code rate $R = 1/2$ of the QC-LDPC codes of IEEE 802.16e (WiMAX) standard in order to outperform the codes used nowadays for this technology. Some of the modifications that get successful for code length of $N = 576$ and rate $R = 1/2$ would also be tested for other code lengths and rates. A very simple explanation of the hardware modifications that would be needed to get each matrix modification will be included.

The Base Matrix of the mentioned code that will be tested is the one on Figure 3.1 (the matrix values represent the shift index of each respective submatrix. The value -1 represents an empty submatrix)

$$\left[\begin{array}{cccccccccccc|cccccc}
-1 & 22 & 1 & -1 & -1 & -1 & -1 & -1 & 7 & 11 & -1 & -1 & 7 & 0 & -1 & -1 & & -1 \\
-1 & 3 & -1 & -1 & -1 & 22 & 7 & 9 & -1 & -1 & -1 & 12 & -1 & 0 & 0 & -1 & & -1 \\
-1 & -1 & -1 & 0 & 22 & 9 & -1 & 9 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & & -1 \\
13 & -1 & 23 & -1 & -1 & -1 & -1 & -1 & 17 & 1 & -1 & -1 & -1 & -1 & -1 & 0 & & -1 \\
-1 & -1 & 15 & -1 & -1 & -1 & 12 & -1 & -1 & 17 & 0 & -1 & -1 & -1 & -1 & -1 & & -1 \\
-1 & -1 & -1 & -1 & 22 & 16 & -1 & 10 & -1 & -1 & -1 & 7 & 0 & -1 & -1 & -1 & \cdots & -1 \\
-1 & -1 & 23 & 5 & -1 & -1 & -1 & -1 & -1 & 14 & 18 & -1 & -1 & -1 & -1 & -1 & \cdots & -1 \\
-1 & 11 & 1 & -1 & -1 & -1 & 2 & -1 & -1 & 23 & -1 & -1 & -1 & -1 & -1 & -1 & & -1 \\
12 & -1 & -1 & -1 & 11 & 0 & -1 & 19 & -1 & -1 & -1 & 3 & -1 & -1 & -1 & -1 & & -1 \\
-1 & -1 & 7 & 17 & -1 & -1 & -1 & -1 & 15 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & & 0 \\
19 & -1 & -1 & -1 & -1 & 18 & -1 & 17 & -1 & -1 & -1 & 2 & 7 & -1 & -1 & -1 & & 0
\end{array}\right]$$

Figure 3.1: Base Matrix of WiMAX LDPC code of N=576, R=1/2, $p = 24$

The right part of the matrix prepared specifically to increase the encoding perform-

ance (see Section 2.2.3) will not be touched by the modifications even if it reduces the possibilities of being successful.

This chapter would be divided in the next sections:

- Removing girth cycles.
- Modifications on Base Matrix distribution (creation of new Base Matrices).
- Modifications on the submatrix distribution.

At the beginning of each one of these sections a brief introduction will be presented, so the concepts and motivations of the exposed matrices modifications will be explained.

The results of the simulations will be exposed and compared to WiMAX original code BER performance. The simulation parameters will be:

- Decoding approach: Normalized Sum Product Algorithm
- Scaling factor for Normalized SPA: 0.75
- Maximum number of iterations: 10
- Max number of frames simulated: 10^7
- Max number of wrong frames simulated: 10^6
- All simulations will be done with the same random seed

3.1 Removing girth cycles

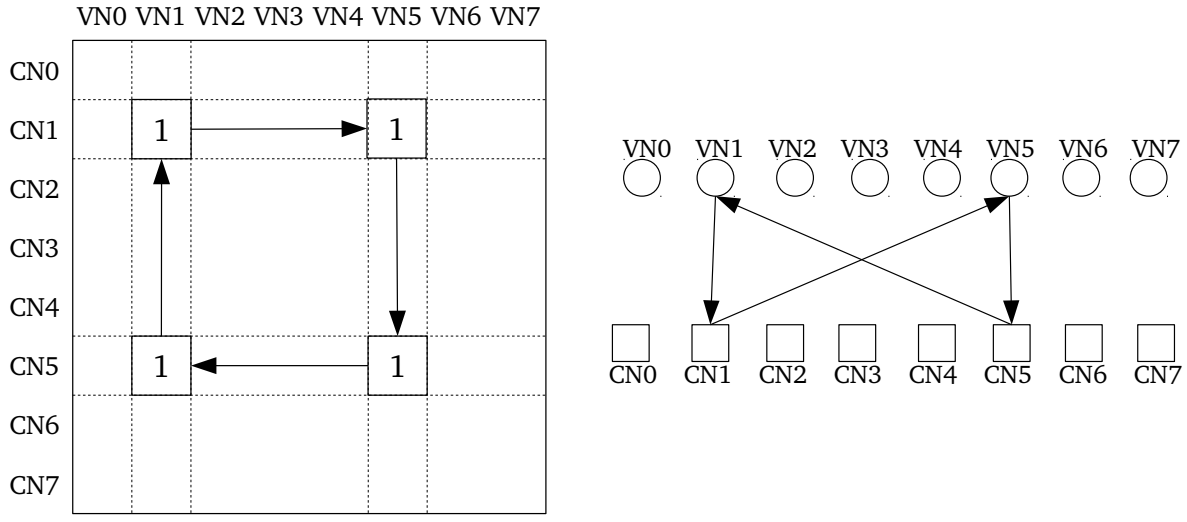
The aim of this section is to remove the *girth* cycles of H matrix. So to begin with this section, the concept of girth needs to be explained.

As mentioned in the section 2.3 when the check node updates its L_{out} value to send it to the variable nodes, it does not include in the computation the L value received from the variable node that is going to be the destiny. This makes that the extrinsic information obtained from a parity check constraint in the first iteration is independent of the *a priori* probability information for that bit (it does of course depend on the *a priori* probabilities of the other codeword bits). The extrinsic information provided from check node to variable node in subsequent iterations remains independent of the original *a priori* probability of that variable node until the *a priori* probability is returned back to the variable node via a cycle in the Tanner graph. The correlation of the extrinsic information with the original

a priori bit probability is what prevents the resulting *a posteriori* probabilities from being exact.

In other words, the more iterations the original *a priori* information of a variable node takes to return to the same variable node because of a cycle in the Tanner graph, the better the result is. These cycles are called *girth cycles*. The smallest cycle possible is a cycle of 4 iterations, called a girth4 cycle, the next is a girth6 cycle, girth8 cycle, etc.

The representation of a girth4 cycle is shown in Figure 3.2.

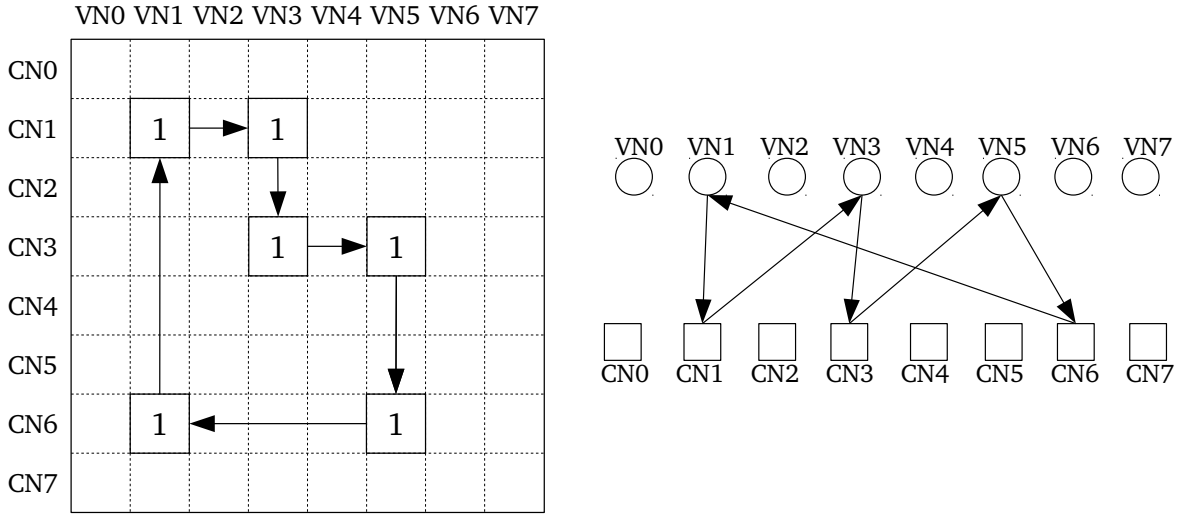


(a) Girth 4 matrix

(b) Girth 4 Tanner graph

Figure 3.2: Girth 4 cycle

The representation of a girth 6 cycle is shown in Figure 3.3.



(a) Girth 6 matrix

(b) Girth 6 Tanner graph

Figure 3.3: Girth 6 cycle

Higher girth cycles will not be discussed in this thesis, but its structure can be easily deduced. The cycles can appear into a submatrix or between different submatrices in the whole H matrix. For the submatrices of WiMAX codes treated in this report, it is only possible that girth cycles appear between different submatrices, because the submatrices are formed of shifted identity matrices and they can not create any cycle by themselves.

There are lots of research reports that talk about girth cycles. It is well known that increasing the minimum level of the girth cycles, the BER performance improves.

For the case of this thesis, the chances of finding a girth-free matrix are reduced because of the special structured right half part of the H matrix presented in Section 2.2.3, that will not be touched because it would impact on the encoding algorithm. This right part of the H matrix does not contain any girth4 or girth6 cycles but because of its structure it is easy that it creates girth cycles when combined with the left part of H matrix.

The first step is to develop an algorithm to detect if a H matrix has got girth4 cycles or not. The proposed algorithm is shown in Figure 3.4.

This algorithm checks if there are two columns that have more than one common column with value 1 in the H code matrix. If this is the case, the H matrix has girth4 cycles.

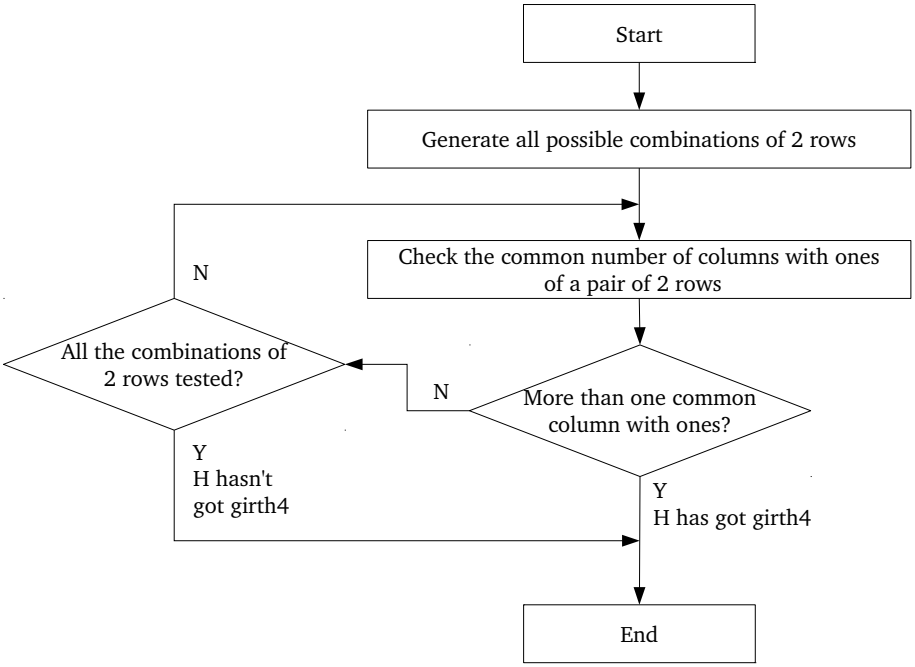


Figure 3.4: Flow chart of the proposed algorithm to detect girth4 cylces

Running this algorithm over the WiMAX original H matrix ($N = 576, R = 1/2$), the result obtained shows that this matrix contains girth4 cycles.

3.1.1 Removing girth 4 cycles

Taking a deeper look to the particular rows and columns involved in the girth4 cycles of the WiMAX matrix it is easy to see that the fact of using identity matrices is a disadvantage because if there is a girth4 cycle, it is common that there are more than one. An example of this problem is shown in Figure 3.5, where all ones in the image are forming girth4 cycles.

The first approach used to try to remove as many girth4 cycles as possible is the total symmetric inversion of some submatrices. As it can be seen in Figure 3.6, by inverting the structure of a submatrix that forms the cycles in a horizontal symmetric way, lots of cycles can be destroyed. In the Figure 3.6 it is shown that all the cycles have disappear, but not always all of them are destroyed with this method.

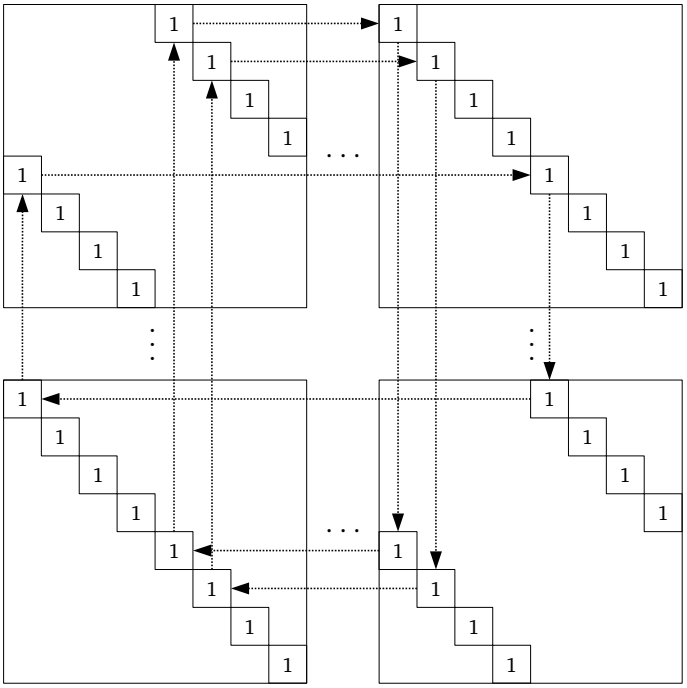


Figure 3.5: Multiple girth4 cycles

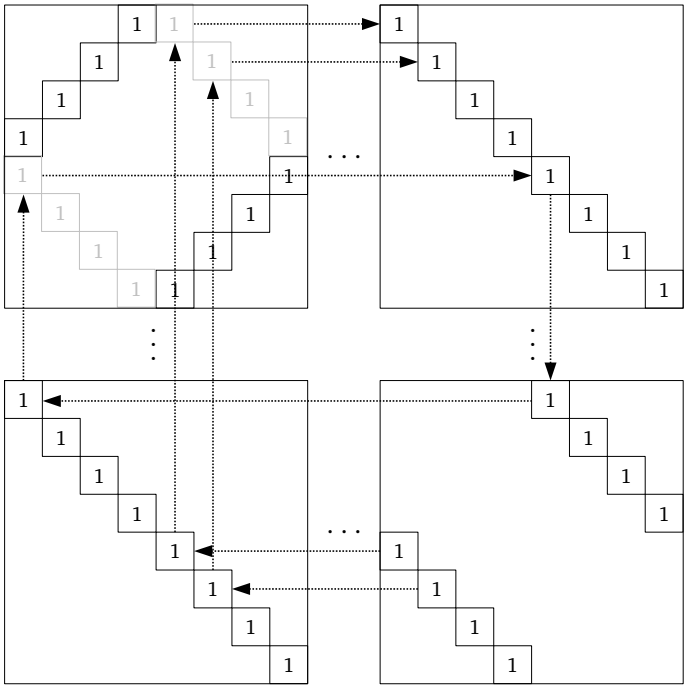
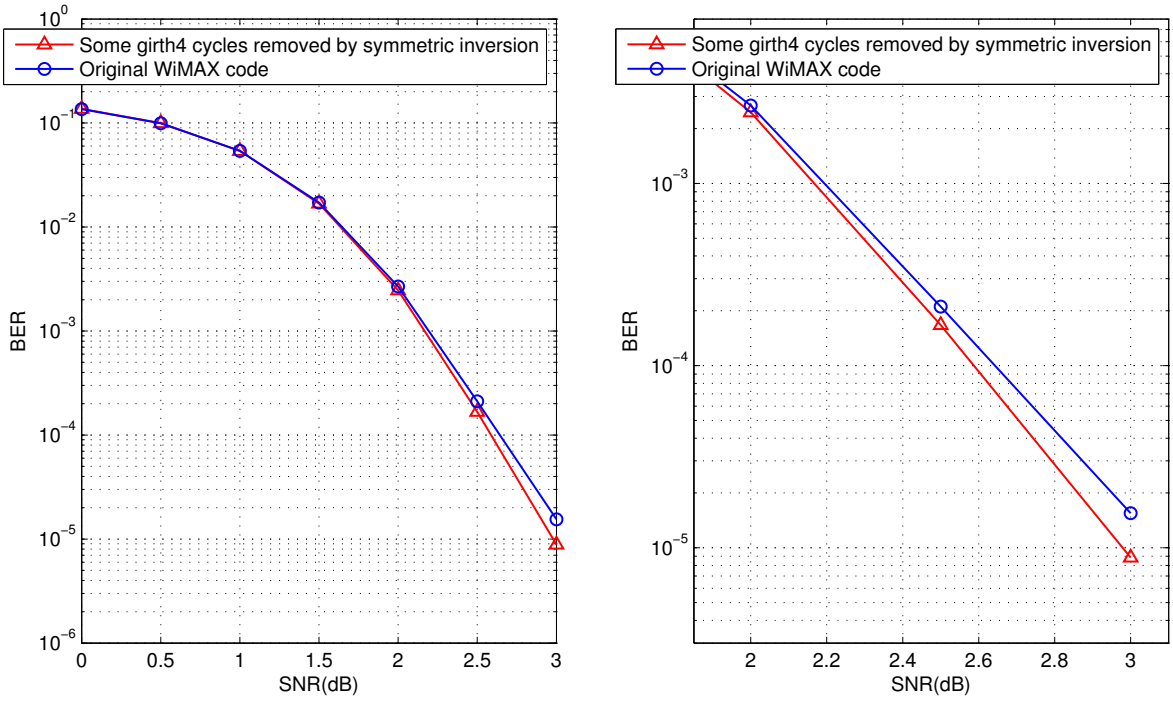


Figure 3.6: Girth4 cycles removed by symmetric inversion

In terms of hardware complexity of this implementation in the decoder, this solution would need a not very complex multiplexors network (p 2x1 multiplexors) working at the output of each barrel shifter in order to redirect the LLR values to the new destiny check nodes, and also a very small memory of a few bytes to store the positions of the submatrices that would redirect the outputs.

The proposed idea is applied to the WiMAX original matrix, where the algorithm inverts the left top submatrix of each group of submatrices that form one or more girth4 cycles, and the report shows that the number of rows involved in girth4 cycles has been reduced from 96 in the original matrix to 44 in the new one. The simulation results of this proposal over the WiMAX original code are shown in Figure 3.7 and Table 3.1. Also the *Frame Error Rate* (FER) performance is shown in Figure 3.8.



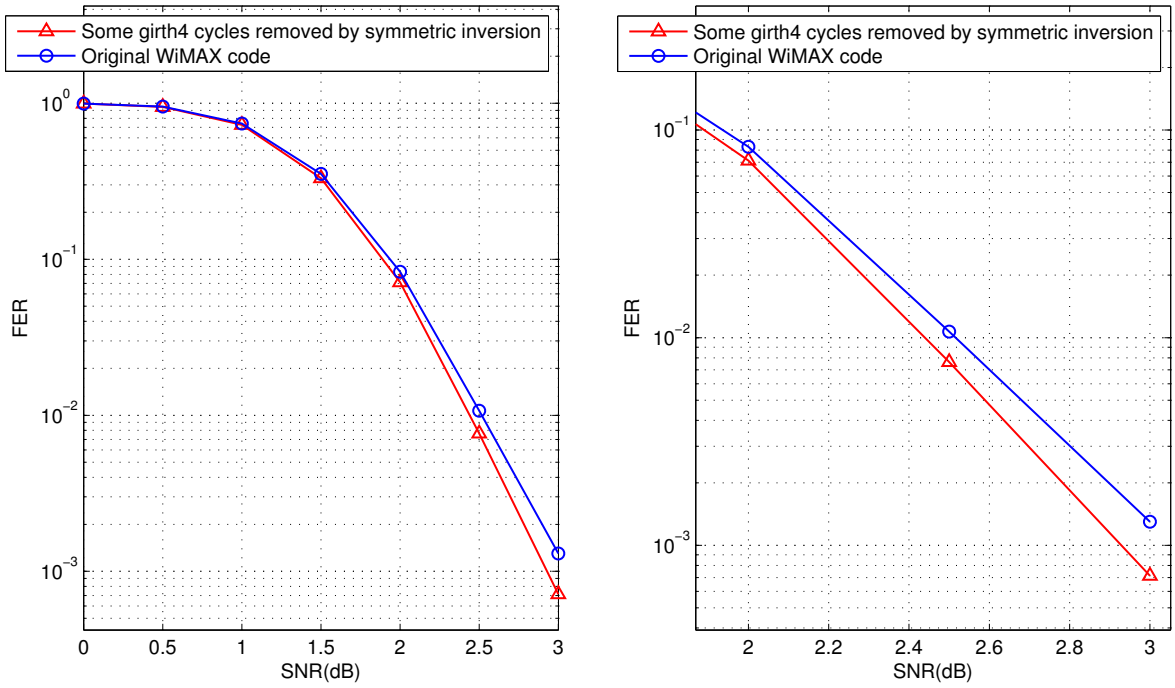
(a) Complete plot

(b) High SNR values zoom

Figure 3.7: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with some girth4 cycles removed by symmetric inversion of some submatrices

Table 3.1: Simulation results values of WiMAX code ($N = 576$, $R = 1/2$) with some girth4 cycles removed by symmetric inversion of some submatrices

SNR	Original WiMAX code BER	Modified matrix by inverting
0.00	1.36e-01	1.37e-01
0.50	9.93e-02	9.98e-02
1.00	5.39e-02	5.39e-02
1.50	1.72e-02	1.68e-02
2.00	2.68e-03	2.46e-03
2.50	2.11e-04	1.67e-04
3.00	1.55e-05	8.84e-06



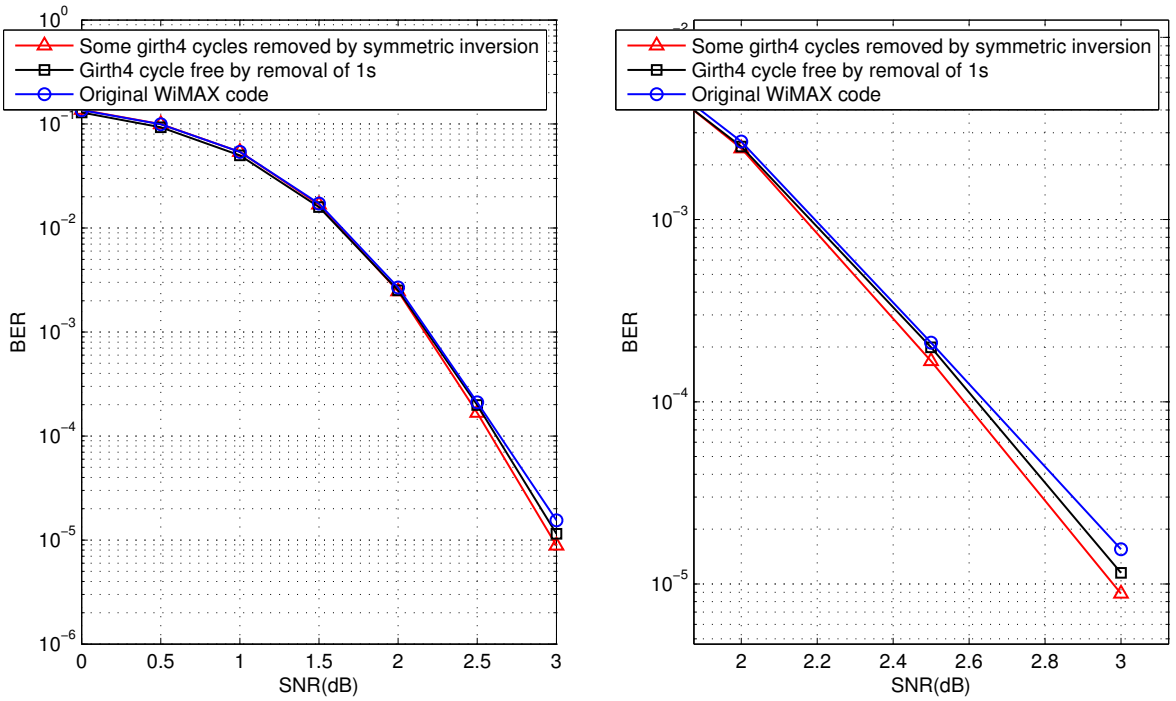
(a) Complete plot

(b) High SNR values zoom

Figure 3.8: FER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with some girth4 cycles removed by symmetric inversion of some submatrices

As seen, the results outperform a little the WiMAX original code for high SNR values in BER and FER. For low SNR values the performance is a little bit worst, but almost equal. Is not unreasonable to assume that the improvement is due to the removal of some girth4 cycles. The results suggest that by removing all the girth4 cycles the performance would be better.

In order to remove more girth4 cycles, all of them if possible, another way is taken. By removing one 1 of each girth4 cycle, it is sure all the girth4 cycles will be removed. Working only over the left part of original WiMAX matrix the algorithm is developed and applied. The results are shown in Figure 3.9 and Table 3.2.



(a) Complete plot

(b) High SNR values zoom

Figure 3.9: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with all girth4 cycles removed by removing 1s

Table 3.2: Simulation results values of WiMAX code ($N = 576$, $R = 1/2$) with all girth4 cycles removed by removal of 1s

SNR	Original WiMAX code BER	Modified matrix by removal of 1s
0.00	1.36e-01	1.29e-01
0.50	9.93e-02	9.30e-02
1.00	5.39e-02	4.99e-02
1.50	1.72e-02	1.59e-02
2.00	2.68e-03	2.52e-03
2.50	2.11e-04	1.99e-04
3.00	1.55e-05	1.15e-05

The results show a little improvement over the WiMAX original matrix, but not over the previous symmetric inversion algorithm shown before. Besides the hardware requirements of this idea are higher because it would need a large memory to store all the positions of the 1s to remove and a network working at the output of barrel shifters to select if each value is used or not. New ideas to remove all the girth4 cycles can be found.

Looking at the submatrices in Figure 3.5 the idea of changing the shifting values comes to the mind. By changing the shift value of one of the submatrices that form a girth4 cycle, all the girth4 cycles between those submatrices are broken. It can be seen in Figure 3.10, where the top left submatrix has changed its shift value from 4 to 5, destroying all the previous cycles.

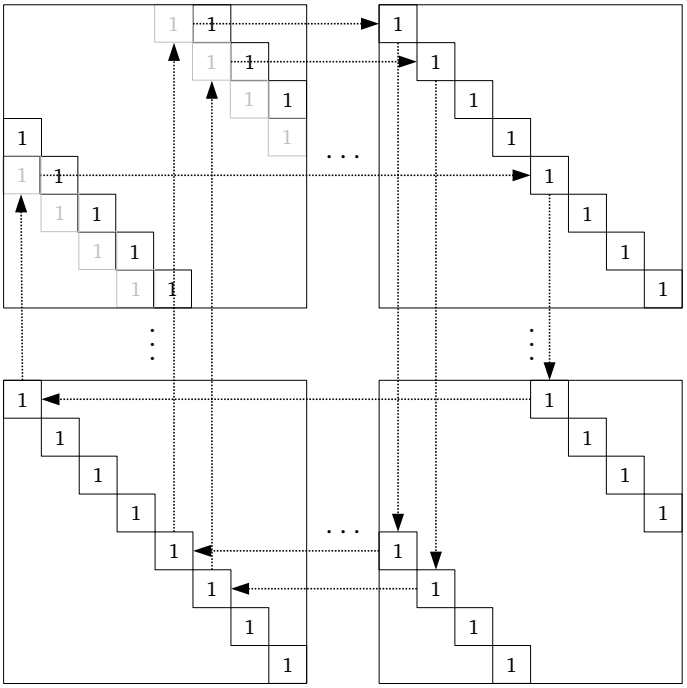


Figure 3.10: Girth4 cycles removed by submatrix circular shift

The algorithm proposed to remove girth4 in the whole H matrix by changing submatrices shift values is shown in Figure 3.11.

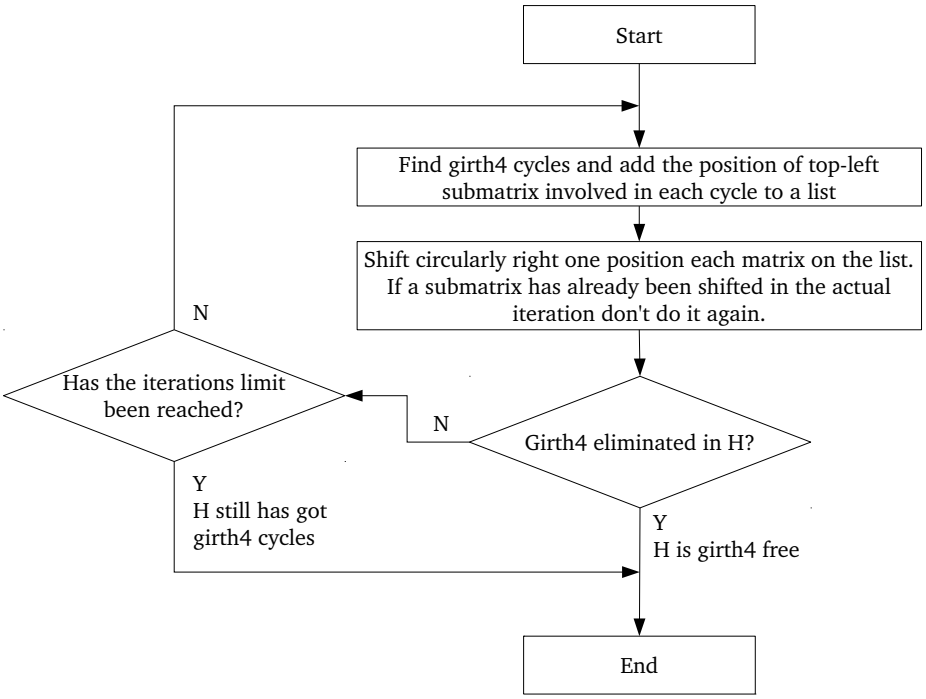
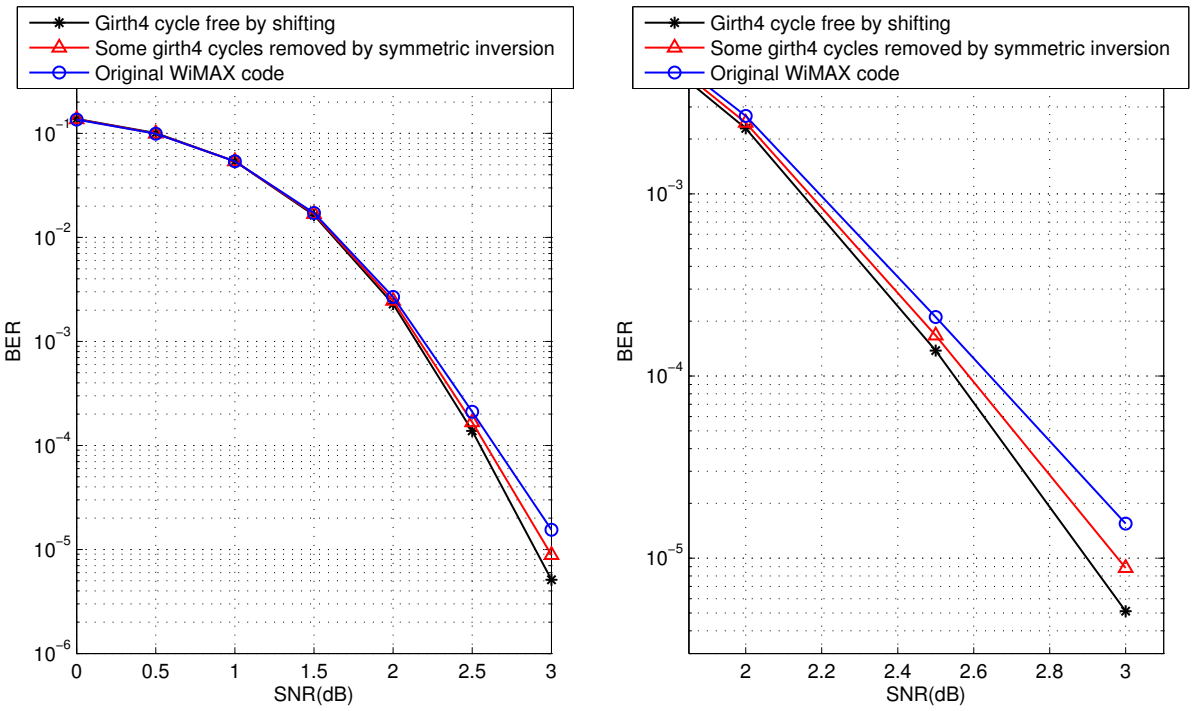


Figure 3.11: Flow chart of the proposed algorithm to remove girth4 cylces by changing shift values

This proposed algorithm is applied to the WiMAX original matrix and the report shows that all the girth4 cycles have been removed. The algorithms gets successful by changing the shift values of only 7 submatrices. The simulation results of this proposal over the WiMAX original code are shown in Figures 3.12 and 3.13 and Table 3.3.



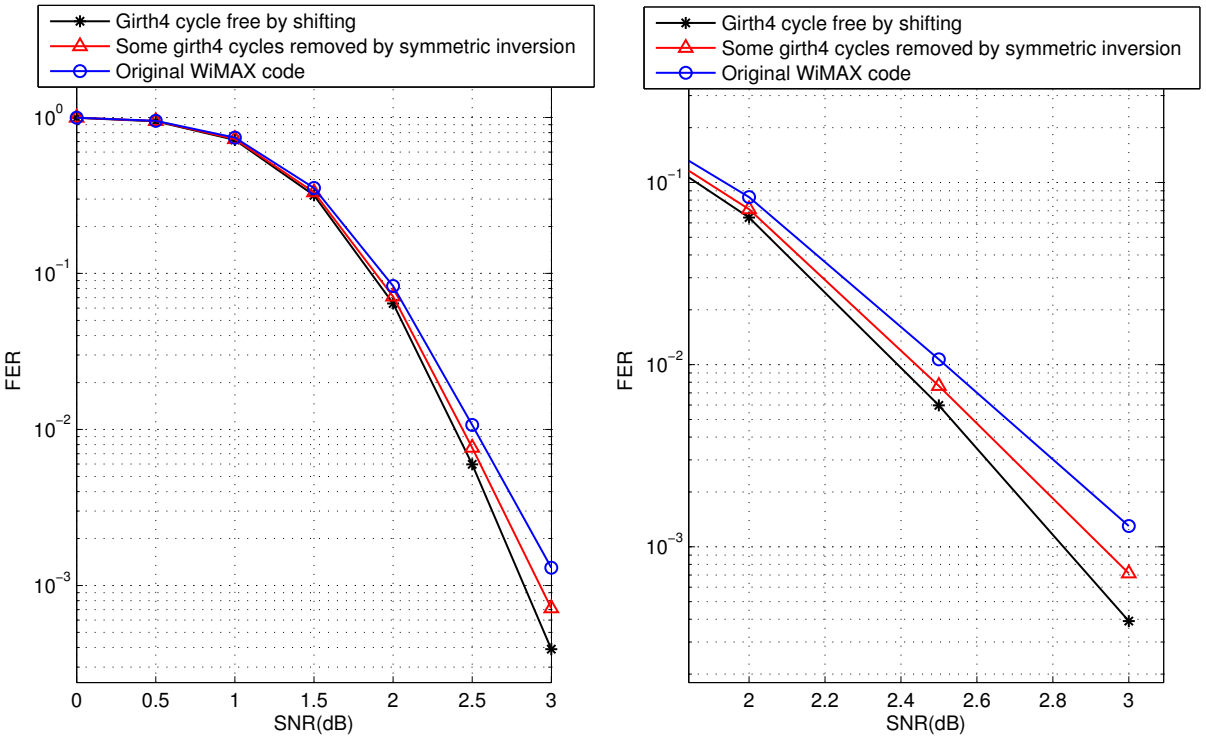
(a) Complete plot

(b) High SNR values zoom

Figure 3.12: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with all girth4 cycles removed by changing shift values of some submatrices

Table 3.3: Simulation results values of WiMAX code ($N = 576$, $R = 1/2$) with all girth4 cycles removed by shifting some submatrices

SNR	Original WiMAX code BER	Modified matrix by shifting
0.00	1.36e-01	1.38e-01
0.50	9.93e-02	1.01e-01
1.00	5.39e-02	5.39e-02
1.50	1.72e-02	1.64e-02
2.00	2.68e-03	2.29e-03
2.50	2.11e-04	1.38e-04
3.00	1.55e-05	5.12e-06



(a) Complete plot

(b) High SNR values zoom

Figure 3.13: FER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with all girth4 cycles removed by changing shift values of some submatrices

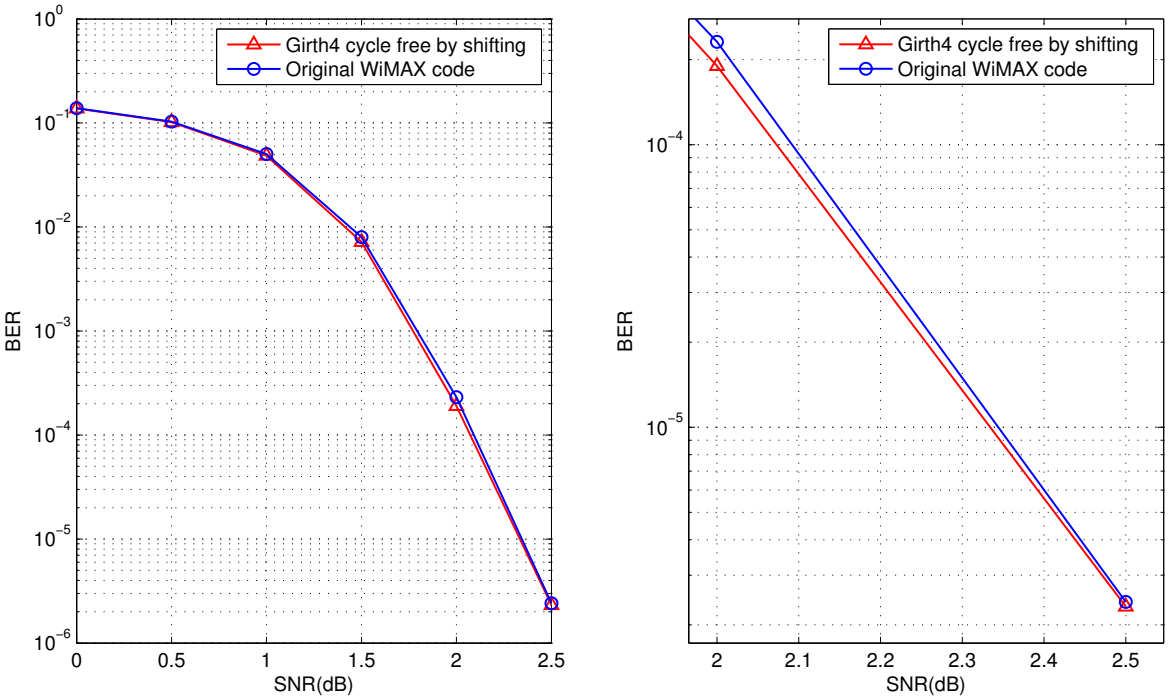
The results presented in Figures 3.12 and 3.13 show an important improvement in the BER and FER performance of the WiMAX code when the girth4 cycles are all removed. This approach outperform in BER and FER the original WiMAX code and also the first approach tested in this section (symmetric inversion). The improvement is almost 0 for low SNR values but it is relevant for high SNR values where it improves the fall of BER in almost $0.18dB$ for $BER = 0.2E10^{-4}$ over WiMAX original code (see Figure 3.12b).

In terms of hardware complexity, this solution would not need additional hardware, since this proposal only needs to modify the shift values of 7 submatrices in the whole H code matrix. A modification in the values stored in the memory that stores the shift values would be enough.

It has been shown that the relation of results between different approaches (better or worst) is the same in BER results than in FER results so, from now, no more values tables or FER graphics will be shown. If the reader is interested in reviewing the next simulation results deeply, he can go to the summary at the end of each section or to the attached material where the simulation output files are stored.

The algorithm was also tested for the medium size WiMAX code of $N = 1440$ and $R = 1/2$ and it got successful after shifting 4 submatrices. It was not tested in the biggest code matrix because it does not contain girth4 cycles.

The results for the girth4 cycle free matrix of code size $N = 1440$ and $R = 1/2$ are shown in Figure 3.14.



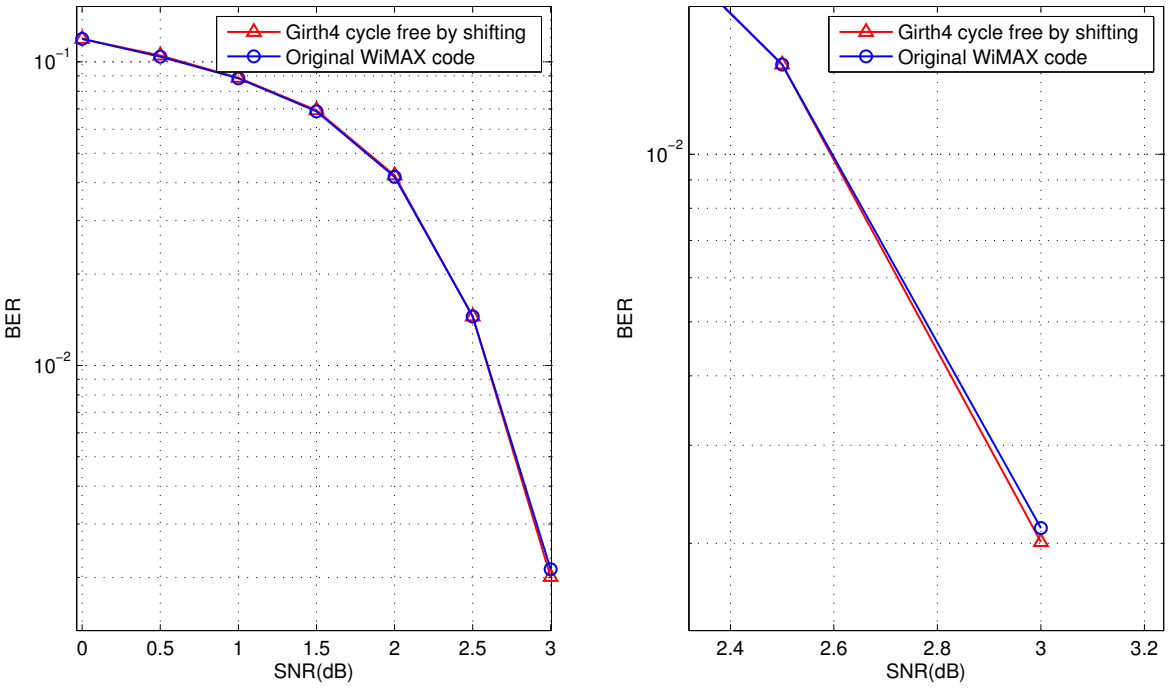
(a) Complete plot

(b) High SNR values zoom

Figure 3.14: BER simulation results plots of WiMAX code ($N = 1440$, $R = 1/2$) with all girth4 cycles removed by changing shift values of some submatrices

The results show that in this case the removal of girth4 cycles improves the BER performance a little bit for medium SNR values, for high SNR values the performance is almost equal tan WiMAX original code.

The algorithm is also tested for the small matrix of $N = 576$ but now of $R = 3/4$. The BER results are shown in Figure 3.15.



(a) Complete plot

(b) High SNR values zoom

Figure 3.15: BER simulation results plots of WiMAX code ($N = 576$, $R = 3/4$) with all girth4 cycles removed by changing shift values of some submatrices

The results show in this case that there is no improvement in the performance, but neither a deterioration. This result is unexpected due to the fact that the algorithm removed lots of girth4 cycles during the creation of the new matrix and the expected results were better.

To finish this section, a summary of the results shown in previous figures is presented in Tables 3.4, 3.5 and 3.6. In this summary the simulation values of each approach can be seen and the arrows and colors show the relation to the WiMAX original code simulation values. The down arrow and green color mean the result is better than original for that SNR value, an up arrow and red color mean the value is worst than original. This exposition also allows a quick comparison of the results between different approaches.

Table 3.4: Summary of results of Section 3.1.1 (Removing girth 4 cycles), $N = 576$, $R = 1/2$

SNR(dB)	Original WiMAX $N = 576$, $R = 1/2$	Some Inversions Fig. 3.7	Girth4 free by removing 1s Fig. 3.9	Girth4 free by shifting Fig: 3.12
0.0	1.36e-01	↑ 1.37e-01	↓ 1.29e-01	↑ 1.38e-01
0.5	9.93e-02	↑ 9.98e-02	↓ 9.30e-02	↑ 1.01e-01
1.0	5.39e-02	≈ 5.39e-02	↓ 4.99e-02	≈ 5.39e-02
1.5	1.72e-02	↓ 1.68e-02	↓ 1.59e-02	↓ 1.64e-02
2.0	2.68e-03	↓ 2.46e-03	↓ 2.52e-03	↓ 2.29e-03
2.5	2.11e-04	↓ 1.67e-04	↓ 1.99e-04	↓ 1.38e-04
3.0	1.55e-05	↓ 8.84e-06	↓ 1.15e-05	↓↓ 5.12e-06

The summary of the results of $N = 576$ and $R = 1/2$ shows all the approaches tested improves the BER for high SNR values (higher than SNR=1.0). In these high values the best approach is by far, the girth4 free matrix by changing shift indices. For low SNR values the results are in all cases similar to the originals.

Table 3.5: Summary of results of Section 3.1.1 (Removing girth 4 cycles), $N = 1440$, $R = 1/2$

SNR(dB)	Original WiMAX $N = 1440$ $R = 1/2$	Girth4 free by shifting Fig: 3.14
0.0	1.39e-01	↓ 1.38e-01
0.5	1.03e-01	↓ 1.02e-01
1.0	5.03e-02	↓ 4.85e-02
1.5	8.00e-03	↓ 7.19e-03
2.0	2.31e-04	↓ 1.90e-04
2.5	2.41e-06	↓ 2.32e-06

The summary of the results of $N = 1440$ and $R = 1/2$ shows that the approach tested of removing girth4 cycles by shifting improves the BER values for all SNR values but not enough.

Table 3.6: Summary of results of Section 3.1.1 (Removing girth 4 cycles), $N = 576$, $R = 3/4$

SNR(dB)	Original WiMAX $N = 576$ $R = 3/4$	Girth4 free by shifting Fig: 3.15
0.0	1.19e-01	$\approx$ 1.19e-01
0.5	1.04e-01	$\uparrow$ 1.05e-01
1.0	8.83e-02	$\uparrow$ 8.86e-02
1.5	6.87e-02	$\uparrow$ 6.93e-02
2.0	4.18e-02	$\uparrow$ 4.23e-02
2.5	1.45e-02	$\approx$ 1.45e-02
3.0	2.13e-03	$\downarrow$ 2.01e-03

The summary of the results of $N = 576$ and $R = 3/4$ shows that the approach tested of removing girth4 cycles by shifting only improves the original values for SNR=3.0 and this improvement is very small. For the rest of SNR values it is worst but almost equal.

Once the girth4 cycles have been removed from the target small and medium WiMAX matrices, the next step in this chapter would be about trying to eliminate girth6 cycles.

3.1.2 Removing girth 4 and girth 6 cycles

In the previous section, it was shown that the removal of girth4 cycles in some matrix sizes and rates leads to an important improvement in BER and FER performance so, it is encouraging that the improvement could be still better by removing girth6 cycles.

Even so, it is needed to point out that the possibilities of being successful are once again drastically reduced due to the structured right part of the H matrix. This part does not contain any cycle but it does easier that cycles appear when combining with left part. This structure also makes it impossible to use most of the algorithms seen in some research works to build large girth codes.

The first approach proposed in this report to eliminate girth6 cycles is to apply almost the same algorithm of Figure 3.11, the one that was successful for girth4. Of course it needs some modifications. First a girth6 detector is needed.

The proposed algorithm to detect girth6 is shown in Figure 3.16. This algorithm works over an already girth4 free H code matrix.

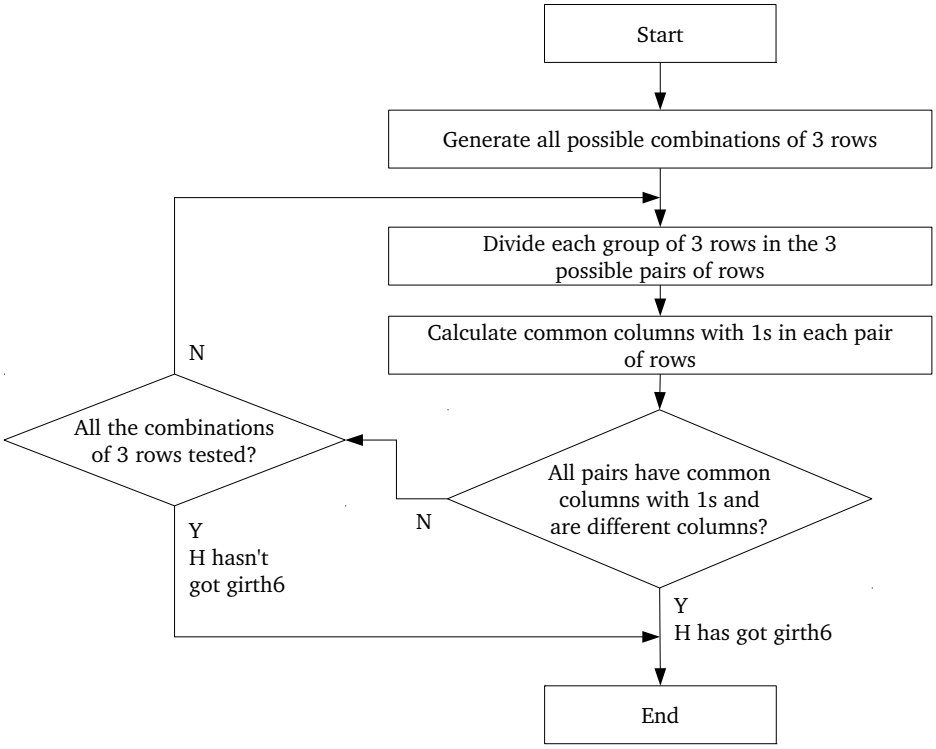


Figure 3.16: Flow chart of the proposed algorithm to detect girth6 cylces in a girth4 free code matrix

The girth6 detector is used to detect girth6 cycles and to identify the submatrices involved. Besides, instead of selecting the top left submatrix to be shifted, in this case, one of the submatrix that form the girth6 cycle, and that is not part of the structured right part, is selected randomly. The girth4 detector is also used so the algorithm does not create girth4 cycles while trying to remove girth6 cycles.

Unfortunately, the algorithm presented gets stuck when trying to remove girth6 cycles, because most of the shifting changes that it does create girth4 cycles in the H matrix, so the changes are reversed and the algorithm never gets successful. So, other ways need to be explored in order to eliminate girth6 cycles. In the next attempts, the algorithms designed to remove girth6 cycles will work over the girth4 matrix obtained with shifting algorithm (Figure 3.12).

One possible idea would be to remove one 1 of each girth6 cycle in the H matrix, so the girth6 cycle is broken, as it was done for girth4 removing in the test of Figure 3.9. This proposal is tested over the mentioned girth4 free matrix of Figure 3.12 and the results show a disaster in the BER performance. This is surely due to the high amount of 1s removed

from the matrix. There are so many girth6 cycles that the algorithm removes too many 1s and the row and column weights get unbalanced.

For the next algorithm tested, instead of changing the WiMAX original shift values, new shift values will be designated. The Base Matrix of the WiMAX original matrix is used to know the distribution and position of the non-empty submatrices and to replicate the structured right part, but the algorithm designates new shift values in the left part in order to create a H matrix that does not contain girth4 and girth6 cycles. The flow chart of the algorithm is presented in Figure 3.17.

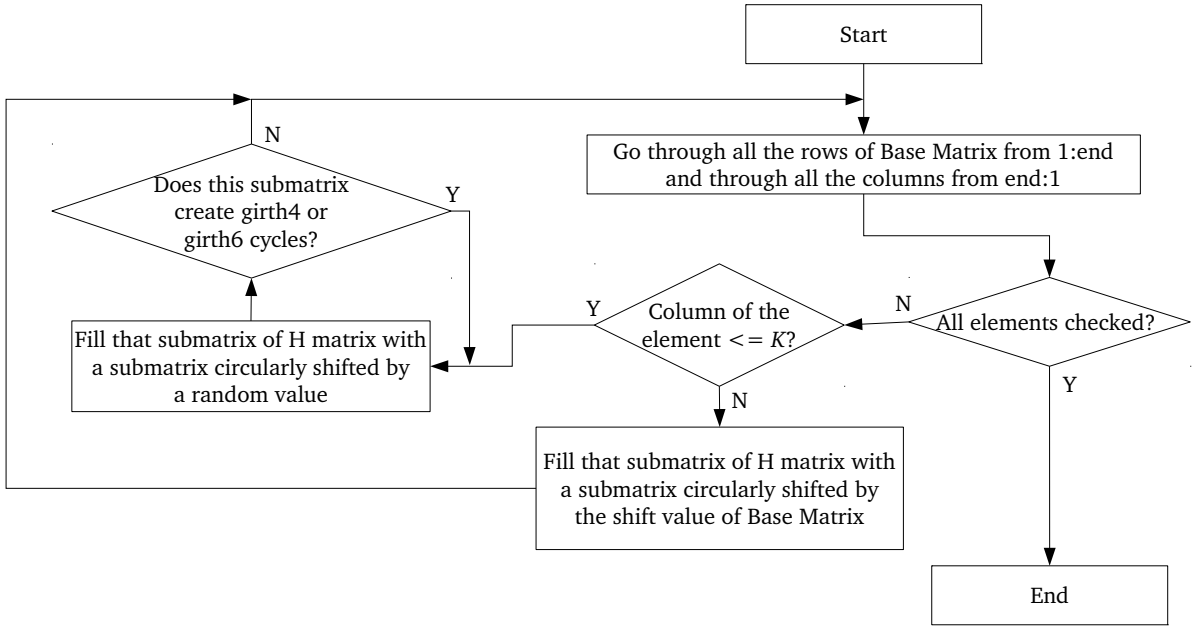
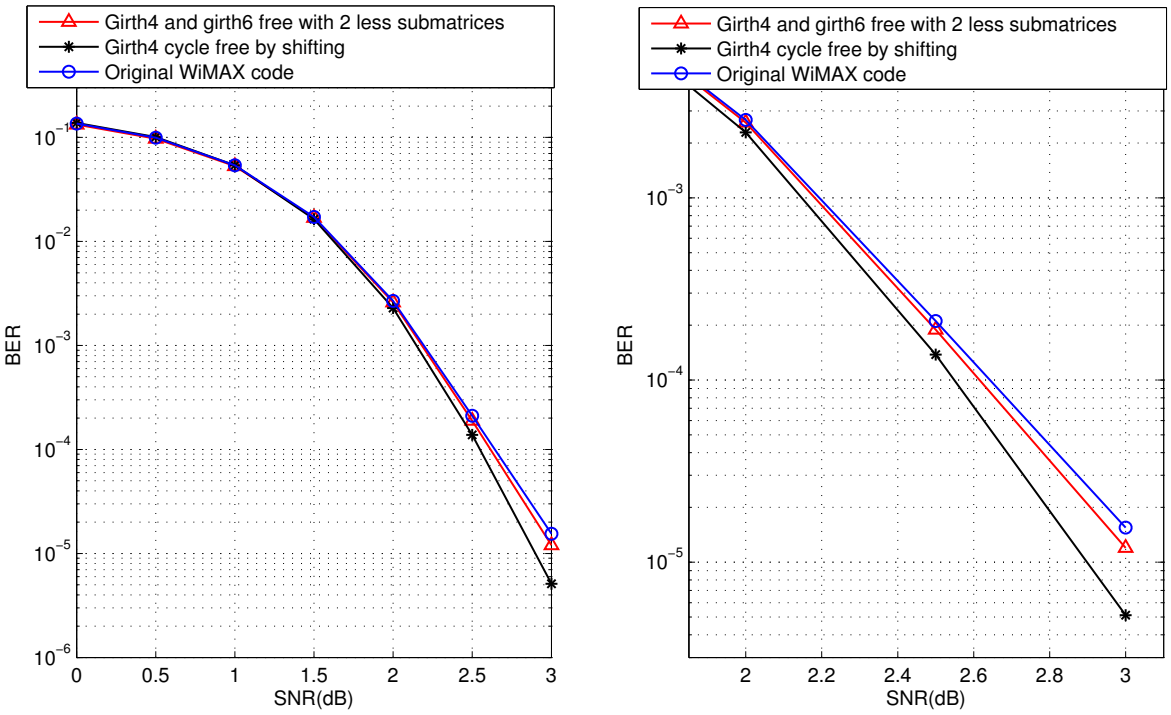


Figure 3.17: Flow chart of the proposed algorithm to generate a new H matrix girth4 and girth6 free

First, it is run over the original WiMAX code of $N = 576$ and $R = 1/2$. The algorithm finishes after a long time of processing and it shows that a new matrix has been created without girth4 and girth6 cycles. Even so, two submatrices has not been placed because any shift rotation was possible without creating girth6 cycles. So, two cases would be simulated, the one without girth4 and girth6 cycles, but with two less submatrices in H matrix, and another with those problematic submatrices filled with some shifted submatrices that do not create girth4 cycles (although they create a few girth6 cycles).

As said, the first BER simulation is a code of $N = 576$ and $R = 1/2$ with same Base Matrix distribution than WiMAX original code but different shift values, with two less submatrices (positions: row=12,column=6 and row=12,column=8), but without girth4 and girth6 cycles. The BER results are exposed in Figure 3.18.

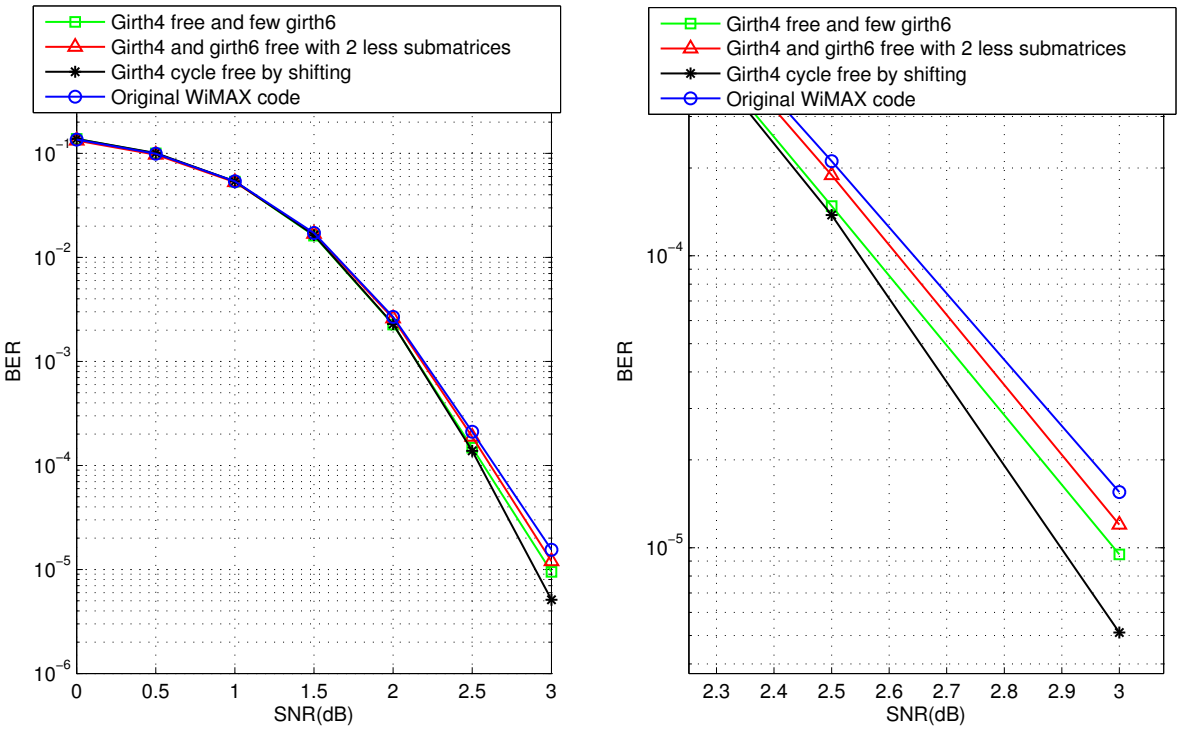


(a) Complete plot

(b) High SNR values zoom

Figure 3.18: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with all girth4 and girth6 cycles removed by generating new shift values but with two less submatrices

The results show a little improvement over the WiMAX original code but not over the girth4 free by shifting obtained in previous section. This can be due to those two less matrices that unbalance the code weights. It will be check in the next simulation where those two less matrices has been filled with shifted submatrices that do not create girth4. The results can be seen in Figure 3.19.



(a) Complete plot

(b) High SNR values zoom

Figure 3.19: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with all girth4 removed and almost all girth6 cycles removed by generating new shift values

As seen the results show an improvement over the previous case, even if the previous case was girth6 free, because in this case the weights are the original of WiMAX. Seems to be clear that the weight distribution is more important than the girth cycles. Anyway, this case has worst performance than the girth4 free code of the previous section. It does not make much sense that a code with all girth4 removed and almost all girth6 cycles removed is not better than one with only the girth4 cycles removed.

The algorithm is also tested in the small submatrix of $N = 576$, $R = 3/4$ but it does not converge well. The algorithm finishes but a huge amount of submatrices are empty because the algorithm did not find any shift value that was good not to create girth cycles. That was expected, because that small matrix has a high concentration of submatrices, much more than the $R = 1/2$ treated before, so if the algorithm had some troubles (2 submatrices) with $R = 1/2$ it as expected it had more troubles for $R = 3/4$.

The algorithm could be tested on the bigger matrix of $N = 1440$ and the biggest one of $N = 2304$ but the algorithm is too low and it would take ages. So another solution is found.

Reviewing some research papers on this field, an interesting algorithm is developed by Yejun He and Jie Yang in [17]. In this work an algorithm to eliminate girth cycles is proposed. The algorithm is based on the fact that the girth cycles can be detected very easily analyzing the Base Matrix in the way explained in the next lines.

If there are four submatrices forming a girth4 cycle in the Base Matrix as shown in Figure 3.20 (remember the value -1 represents an empty submatrix), there will exist a girth4 cycle in the H matrix that corresponds to that Base Matrix if the value of $A - C + D - B$ is equal to a multiple of p (size os submatrix).

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & A & -1 & C & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & B & -1 & D & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Figure 3.20: Girth4 in Base Matrix of Yejun He and Jie Yang algorithm

For girth6 cycles case, if there are 6 submatrices forming a girth6 cycle (see Figure 3.21), there will be a girth6 cycle in the H matrix if the value of $A-C+D-E+F-B$ is equal to a multiple of p.

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & A & -1 & C & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & D & -1 & E & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & B & -1 & -1 & -1 & F & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Figure 3.21: Girth6 in Base Matrix of Yejun He and Jie Yang algorithm

In Figure 3.22 their algorithm is presented with some modifications to adapt it to this work.

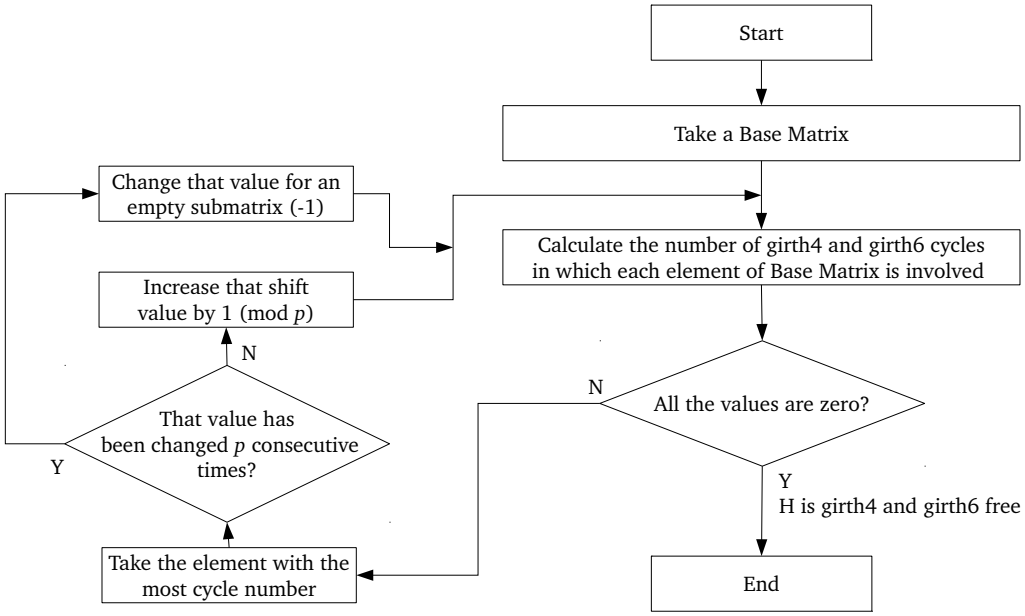
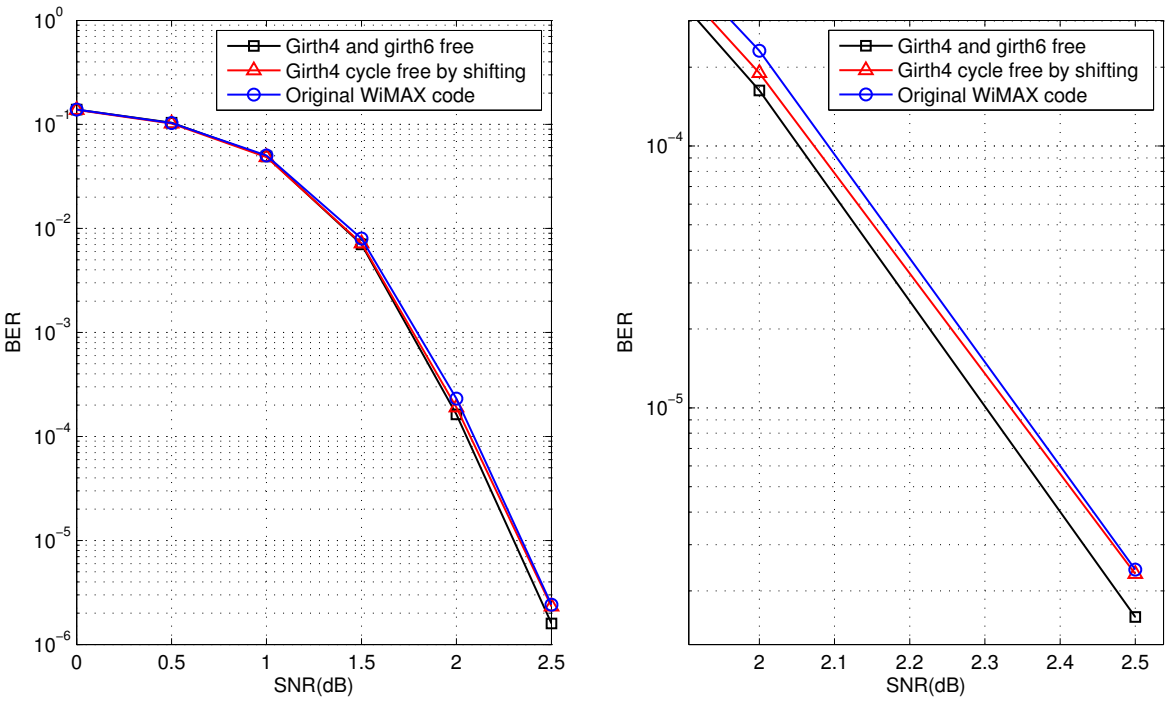


Figure 3.22: Flow chart of the proposed algorithm to detect girth6 cylces in a girth4 free code matrix

As seen, this algorithm needs a code to detect the number of girth4 and girth6 cycles in which each element of Base Matrix is involved. Two codes are developed to carry out this function. These codes go through all the elements in the Base Matrix, analyze all the possible combinations that could create girth4 and girth6 cycles and apply the equations seen a few lines before to detect if there will be girth cycles when the H matrix is created. If the algorithm shifts the same submatrix p consecutive times, it means that submatrix can not be there without creating girth cycles, so it is replaced with an empty submatrix.

The algorithm is run over all the submatrices treated before: $N = 576$ for $R = 1/2$ and $R = 3/4$, $N = 1440$ for $R = 1/2$ and $N = 2304$ for $R = 1/2$. For the first two cases of different rates of $N = 576$, the algorithm does not converge. It was expected due to the fact that the algorithm of Figure 3.17 could not find a combination to create a girth4 and girth6 free code. Anyway, for $N = 576$ and $R = 1/2$ a solution without girth4 cycles and only a few girth6 cycles was found. When run over the code of $N = 1440$ and $R = 1/2$, the algorithm gets successful and gets a girth4 and girth6 free matrix. The BER results are compared with the WiMAX original code for that size and rate and also the girth4 free results of that code obtained in Figure 3.14. The results are shown in Figure 3.23.



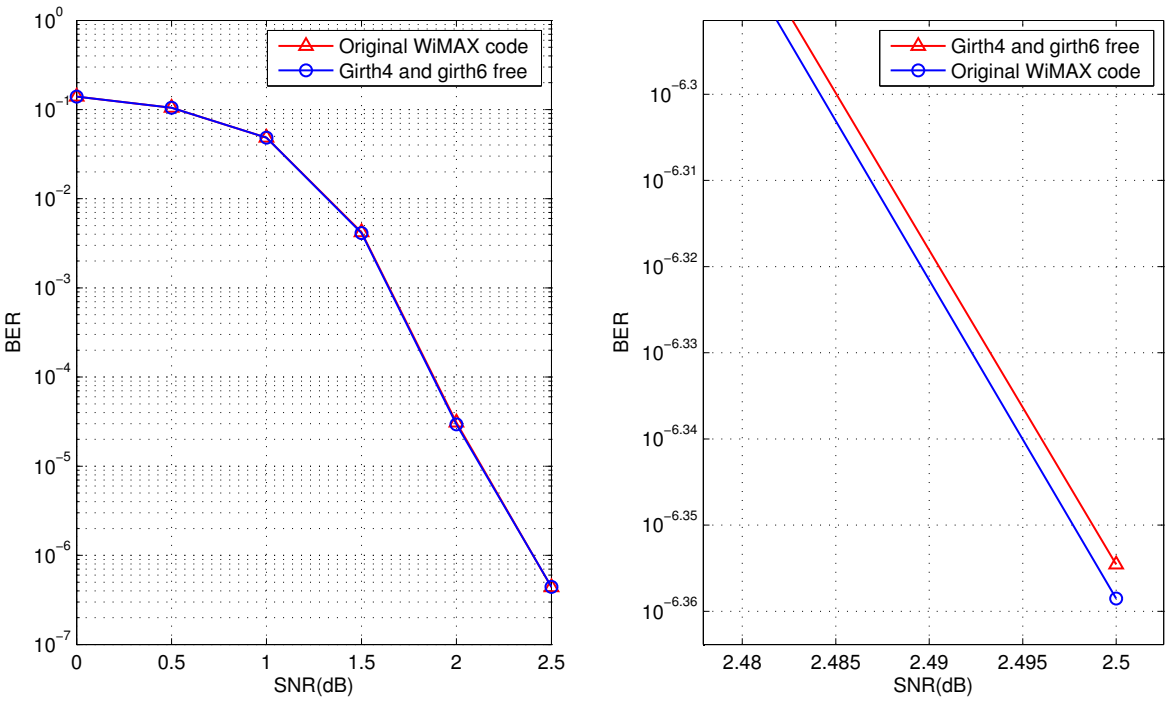
(a) Complete plot

(b) High SNR values zoom

Figure 3.23: BER simulation results plots of WiMAX code ($N = 1440$, $R = 1/2$) girth4 and girth6 free obtained with the algorithm of Figure 3.22

The results show an improvement in BER performance over both other cases. Besides as it was said before, these modifications do not need any hardware alteration since they are only shift values changes.

When the algorithm is run over the biggest matrix of $N = 2304$ and $R = 1/2$ it also gets successful and it gets a girth4 and girth6 free matrix. The results are compared with the WiMAX original code for that size and rate that was already girth4 free. The BER simulation results can be seen in Figure 3.24.



(a) Complete plot

(b) High SNR values zoom

Figure 3.24: BER simulation results plots of WiMAX code ($N = 2304$, $R = 1/2$) girth4 and girth6 free obtained with the algorithm of Figure 3.22

On this occasion, the difference seen is almost non-existent. The zoom needed to see any difference is too high. This was expected because this code matrix was already girth4 free, and the number of girth6 cycles was not high.

Lots of more algorithms could be developed and tested to eliminate higher girth cycles but the right structure of the matrix makes it almost impossible, at least in the size of matrix focused in this thesis ($N = 576$).

To finish this section, a summary of the results shown in previous figures is presented in Tables 3.7, 3.8 and 3.9. In this case the obtained values are compared (the color and arrows) to the WiMAX girth4 free matrix by shifting obtained in the previous section of this chapter and that is the best result obtained until now, not the original WiMAX code. But for the case of $N = 2304$ the original WiMAX code is already girth4 free so the results are compared with that. As before, this exposition of the simulation values allows also to compare the results between different approaches.

Table 3.7: Summary of results of Section 3.1.2 (Removing girth 4 and girth 6 cycles), $N = 576$, $R = 1/2$

SNR(dB)	Original WiMAX $N = 576$, $R = 1/2$	Girth4 free by shifting Fig: 3.12	Girth4 and 6 free with 2 less submatrices Fig. 3.18	Girth4 and few girth6 Fig. 3.19
0.0	1.36e-01	1.38e-01	↓ 1.33e-01	≈ 1.38e-01
0.5	9.93e-02	1.01e-01	↓ 9.72e-02	↓ 1.00e-01
1.0	5.39e-02	5.39e-02	↓ 5.29e-02	↓ 5.35e-02
1.5	1.72e-02	1.64e-02	↑ 1.69e-02	↓ 1.62e-02
2.0	2.68e-03	2.29e-03	↑ 2.59e-03	↓ 2.26e-03
2.5	2.11e-04	1.38e-04	↑ 1.89e-04	↑ 1.48e-04
3.0	1.55e-05	5.12e-06	↑↑ 1.20e-05	↑ 9.48e-06

The table shows again that none of the approaches can outperform the girth4 free matrix obtained by shifting, even if they are girth6 free.

Table 3.8: Summary of results of Section 3.1.2 (Removing girth 4 and girth 6 cycles), $N = 1440$, $R = 1/2$

SNR(dB)	Original WiMAX $N = 1440$ $R = 1/2$	Girth4 free by shifting Fig: 3.14	Girth4 and 6 free Fig. 3.23
0.0	1.39e-01	1.38e-01	↑ 1.39e-01
0.5	1.03e-01	1.02e-01	↑ 1.04e-01
1.0	5.03e-02	4.85e-02	↑ 4.93e-02
1.5	8.00e-03	7.19e-03	↓ 7.02e-03
2.0	2.31e-04	1.90e-04	↓ 1.63e-04
2.5	2.41e-06	2.32e-06	↓↓ 1.59e-06

The results of this case, as seen in the respective graph, show an important improvement for high SNR values and almost any difference for low SNR values.

Table 3.9: Summary of results of Section 3.1.2 (Removing girth 4 and girth 6 cycles), $N = 2304$, $R = 1/2$

SNR(dB)	Original WiMAX $N = 2304$ $R = 1/2$	Girth4 and 6 free Fig. 3.24
0.0	1.40e-01	≈ 1.40e-01
0.5	1.05e-01	≈ 1.05e-01
1.0	4.85e-02	↓ 4.83e-02
1.5	4.20e-03	↓ 4.12e-03
2.0	3.09e-05	↓ 2.95e-05
2.5	4.38e-07	↑ 4.42e-07

The results show almost no difference with the WiMAX original code.

3.2 Modifications on Base Matrix distribution (creation of new Base Matrices)

Until now, the shift values have been changed and also reset with random values, but the original distribution and place of the full and empty submatrices has not been changed. This section is about creating new Base Matrix distributions and test if the structure can be improved to get better performance. Additionally some tests to try to improve the row weight will be presented. This section will use, as target matrix to outperform, the best BER results until now, the girth4 free WiMAX code matrix got by shifting submatrices (Figure 3.12). The aim of this section is to outperform that matrix, so all the modifications done will try not to create girth4 cycles.

Following the original idea of this field of research that less girth cycles lead to better performance, the idea of the following algorithm is to create Base Matrix distributions with few probabilities of creating girth cycles. It means, if the Base Matrix is the one in Figure 3.25.

$$\begin{bmatrix} -1 & A & -1 & C & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & B & -1 & D & -1 \end{bmatrix}$$

Figure 3.25: Example of girth cycles existence

There are possibilities that A,B,C and D form a girth4 cycle (see the theory explained in the previous section). But if the structure of Base Matrix is like the one in Figure 3.26, there is not any chance of a girth cycle appear, no matter what shift values A,B,C or D are.

$$\begin{bmatrix} -1 & A & -1 & C & -1 \\ -1 & -1 & -1 & D & -1 \\ -1 & B & -1 & -1 & -1 \end{bmatrix}$$

Figure 3.26: Example of girth cycles existence 2

So the next algorithm creates Base Matrices trying to minimize those structural cycles that can lead to girth cycles in the H code matrix. If these structural cycles are minimized

in number, the possibilities of being successful when selecting good shift values that do not create girth cycles in H matrix are much higher. The algorithm can be seen in Figure 3.27.

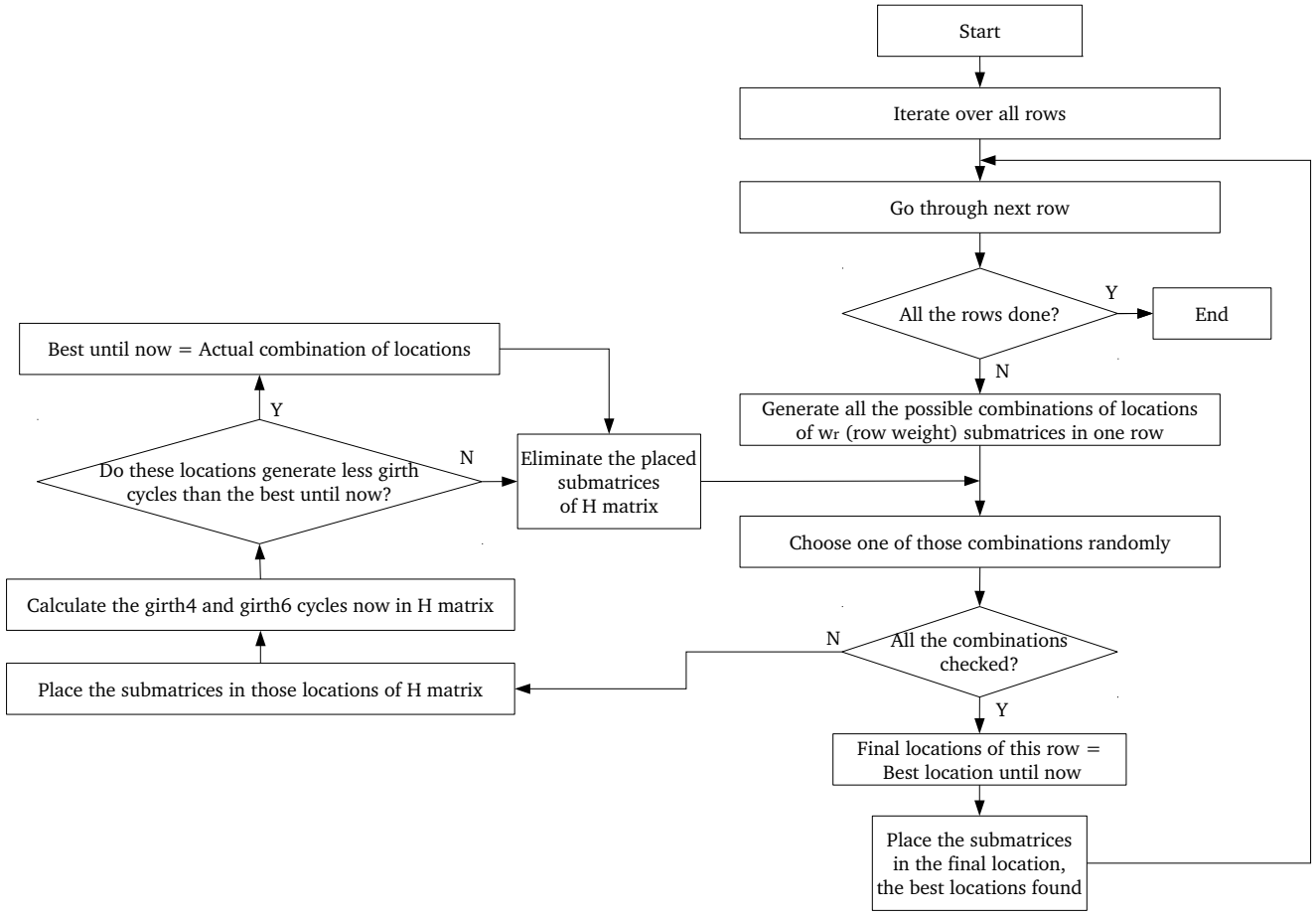


Figure 3.27: Flow chart of the proposed algorithm to minimize girth4 and girth6 possible cycles in the structure of Base Matrix

This algorithm takes a vector of row weights for the left part of the H matrix and in each row introduces the number of non-empty submatrices that the vector indicates for that row.

Once the Base matrix is made with the less potential girth cycles as possible, the algorithm presented in the previous section in Figure 3.17 can work to find shift values that finally do not create girth cycles in H matrix. These algorithms also allows to test the influence of the row weights by giving to it the vector with the desired row weights.

First of all, the row weights of the original WiMAX code for $N = 576$ and $R = 1/2$ are presented in Table 3.10.

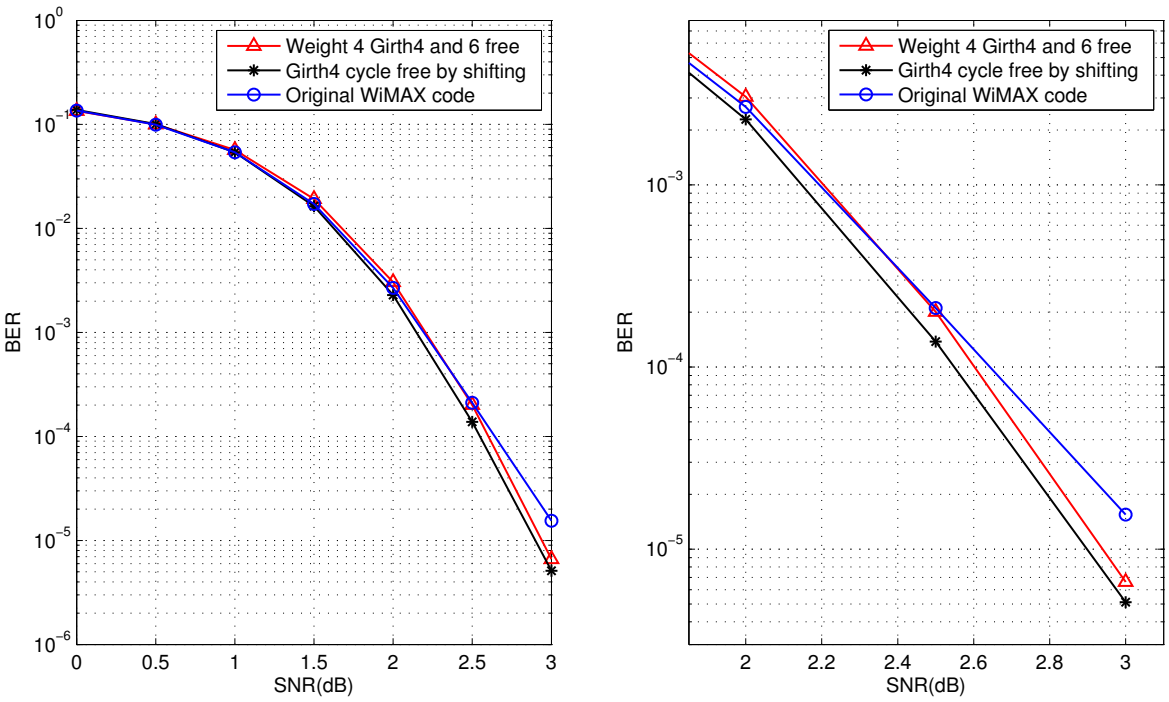
Table 3.10: Row weights of WiMAX original code for $N = 576$ and $R = 1/2$

Row	Row weight of complete WiMAX H matrix	Row weight of left part of WiMAX H matrix
1	6	4
2	7	5
3	7	5
4	6	4
5	6	4
6	7	4
7	6	4
8	6	4
9	7	5
10	6	4
11	6	4
12	6	4

First the influence of the row weights will be tested. To do that, two trials will be done, one with all the rows with 4 weight in the left part (see Table 3.10), and another with 5. The values are selected because the original values are all in the interval [4,5], so with these values, an approach above and below is tested.

So the first trial introduces 4 non-empty submatrices in each row of the left part of the H matrix. Due to introducing these weights, the algorithm can get successful when trying that the desired matrix is girth4 and girth6 free. The results are shown in Figure 3.28.

The results show an improvement over the WiMAX original code but not over the girth 4 free WiMAX matrix, even if this last trial is girth4 and also girth6 free. Besides there is a little deterioration for medium SNR values. It seems the uniform weight 4 is not better than the actual weights of the WiMAX original code. The weights of the original code are 4 on 9 of its 12 rows, so the proposed matrix is different only in the weights of 3 rows. Because of the difference is not big, the results are similar. With the next trial, the difference is bigger, because there are only three rows in the original code with weight of 5, so the proposal code with all the rows with weight of 5 is different in 9 rows, much more different than last one.



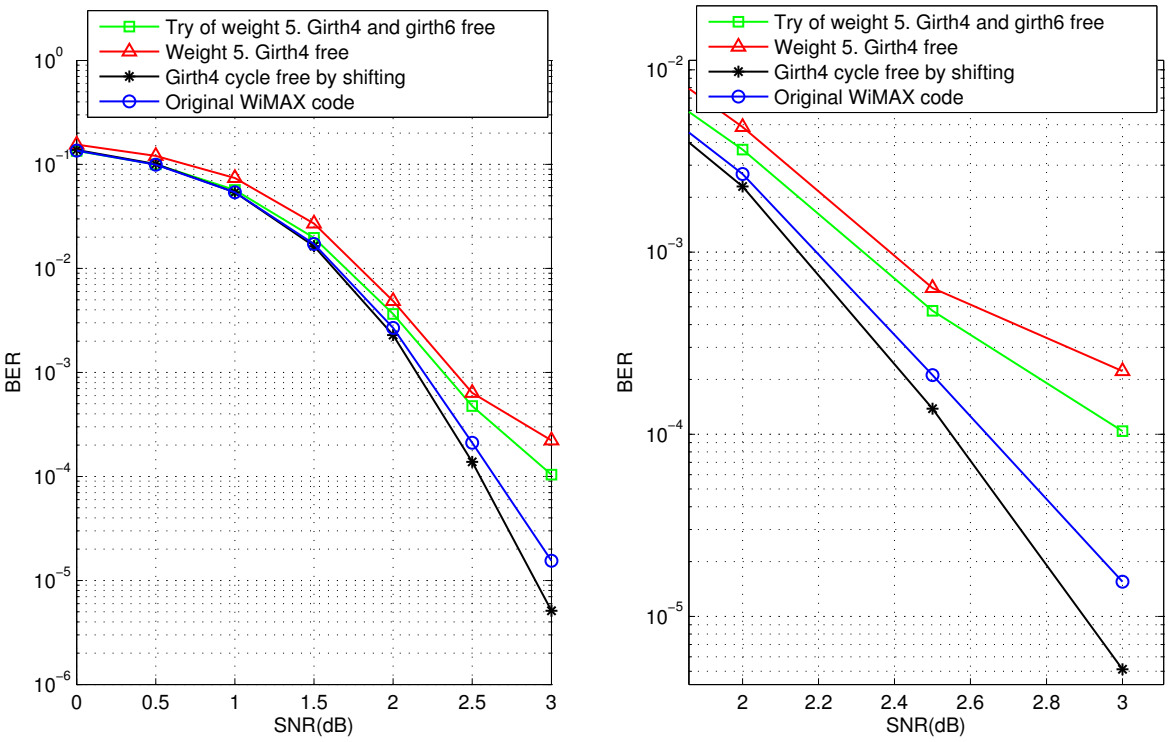
(a) Complete plot

(b) High SNR values zoom

Figure 3.28: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with weight of 4 in the left part. Girth 4 and girth 6 free

For the trial of weight 5 in all the rows, the algorithm is not able to remove all the girth6 cycles and 10 submatrices are filled with empty submatrices. Two cases are presented: the case with 10 less submatrices (so the weight definitely is not 5) but girth4 and girth6 free, and the case with weight 5 but girth6 cycles, so it is only girth4 free. The results of both cases are presented in Figure 3.29.

The results show a high deterioration of the BER performance for both cases, not only over the girth4 WiMAX code but over the WiMAX original matrix too. Between the two cases presented, the one with the real weight of 5 but with girth 6 cycles show worst performance than the one with irregular weights (it has 10 less non-empty submatrices) but girt4 and girth6 free.



(a) Complete plot

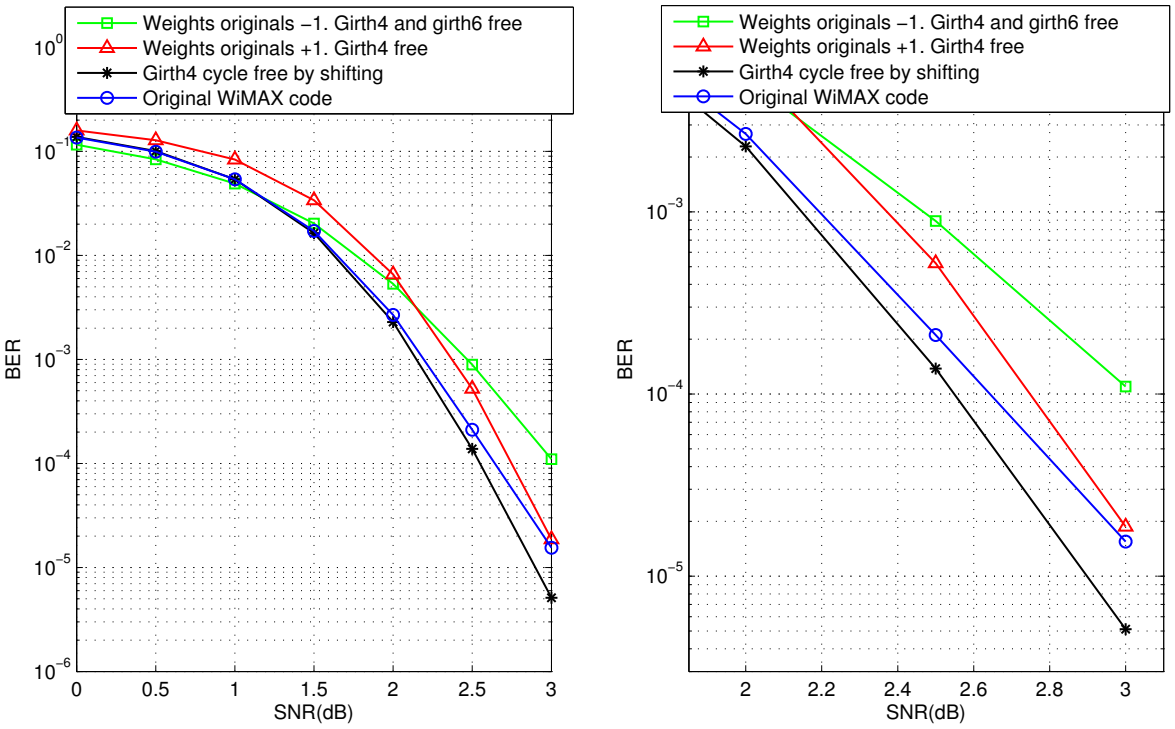
(b) High SNR values zoom

Figure 3.29: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with weight of 5 in the left part. One girth4 free and another girth4 and girth6 free

In the next case, two trials will also be presented. One of them is a code where the row weights have been decreased by 1, so the weights will be 3 for 9 rows and 4 for 3 rows. For this matrix it was possible to remove both girth4 and girth6 cycles. In the other trial, the weights have been increased by 1, so the weights are 5 for 9 rows and 6 for 3 rows. In this trial, it was possible to remove girth4 cycles but not girth6 cycles. The results of this two trials are shown in Figure 3.30.

The results, once again, show a high deterioration on the BER performance for both cases. For the trial of increasing the row weights by 1, at SNR=3 the BER is almost equal to the WiMAX original code, but not for the rest of the SNR values. The trial of decreasing the row weights by 1, it is curious that it outperform both girth4 free WiMAX and WiMAX original codes for low SNR values, but after SNR=1.5 it becomes a performance disaster.

It is curious that the trial where the weights were increased by 1 (so there are lots of rows with weight=5 and some of them with weight=6) has better performance than the one of the previous cases where the weights where all 5. This tells us that maybe the distribution and relation of weights is important, not only the value of the weights.



(a) Complete plot

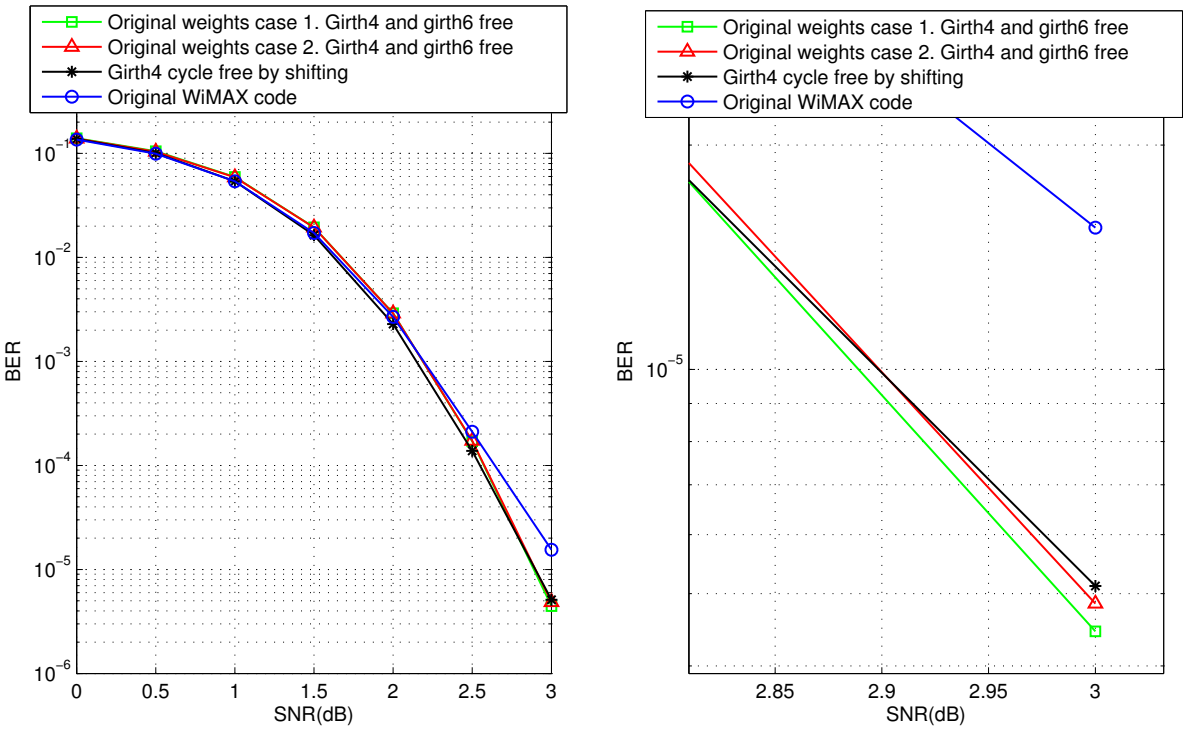
(b) High SNR values zoom

Figure 3.30: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with weight -1 (girth4 and girth6 free) and +1 (girth4 free)

Once that the influence of row weights has been tested, the next objective is to test some different randomly generated Base Matrix distributions for the same weights than the original code. It means, the algorithm will take the same row weights than the WiMAX original code has, but it will distribute the non-empty submatrices to get girth4 and girth6 free codes if possible.

After running the algorithm several times with the original weights, two different girth4 and girth6 free codes are obtained. The results of these codes are shown in Figure 3.31.

The results show that these two cases can get the BER performance of the girth4 WiMAX code for high SNR values, they can even improve the performance a little bit. But for medium SNR values the values are a little bit worst than original code.



(a) Complete plot

(b) High SNR values zoom

Figure 3.31: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with same weights than original but girth4 and girth6 free

In hardware terms, all these modifications done on the weights and the Base Matrix distributions would only require, depending on the case, some barrel shifters more or less and some modifications on the data of the memories that store the behavior of the barrel shifters.

As in previous sections, to finish this one, a summary of the results shown in previous figures is presented in Table 3.11. In this case the obtained values are compared (the color and arrows) again to the WiMAX girth4 free matrix by shifting obtained in the previous section of this chapter and that is the best result obtained until now, not the original WiMAX code. As before this exposition of the simulation values allows also to compare the results between different approaches.

Table 3.11: Summary of results of Section 3.2 (Modifications on Base Matrix distribution (creation of new Base Matrices)), $N = 576$, $R = 1/2$

SNR(dB)	Original WiMAX $N = 576$, $R = 1/2$	Girth4 free by shifting Fig: 3.12	Weight 4 girth4 and girth6 free Fig. 3.28	Try of weight 5 girth4 and girth6 Fig. 3.29
0.0	1.36e-01	1.38e-01	↓ 1.35e-01	↓ 1.35e-01
0.5	9.93e-02	1.01e-01	↓ 1.00e-01	↓ 1.00e-01
1.0	5.39e-02	5.39e-02	↑ 5.69e-02	↑ 5.68e-02
1.5	1.72e-02	1.64e-02	↑ 1.92e-02	↑ 1.97e-02
2.0	2.68e-03	2.29e-03	↑ 3.05e-03	↑ 3.65e-03
2.5	2.11e-04	1.38e-04	↑ 2.01e-04	↑↑ 4.76e-04
3.0	1.55e-05	5.12e-06	↑ 6.61e-06	↑↑ 1.04e-04

SNR(dB)	Weight 5 girth4 free Fig. 3.29	Same weights -1 girth4 and girth6 free Fig. 3.30	Same weights +1 girth4 free Fig. 3.30	Same weights case 1 girth4 and girth6 free Fig. 3.31
0.0	↑ 1.55e-01	↓ 1.16e-01	↑ 1.59e-01	↑ 1.40e-01
0.5	↑ 1.21e-01	↓ 8.39e-02	↑ 1.28e-01	↑ 1.05e-01
1.0	↑ 7.40e-02	↓ 4.90e-02	↑ 8.35e-02	↑ 5.94e-02
1.5	↑ 2.70e-02	↑ 2.03e-02	↑ 3.39e-02	↑ 1.95e-02
2.0	↑ 4.86e-03	↑ 5.33e-03	↑ 6.57e-03	↑ 2.91e-03
2.5	↑↑ 6.35e-04	↑↑ 8.91e-04	↑↑ 5.23e-04	↑ 1.71e-04
3.0	↑↑ 2.22e-04	↑↑ 1.10e-04	↑↑ 1.87e-05	↓ 4.45e-06

As seen in this summary and previously analyzed during this section, some solutions show better performance for low SNR values and one of them shows better performance for high SNR values. Anyway, the improvements are not substantial in any case.

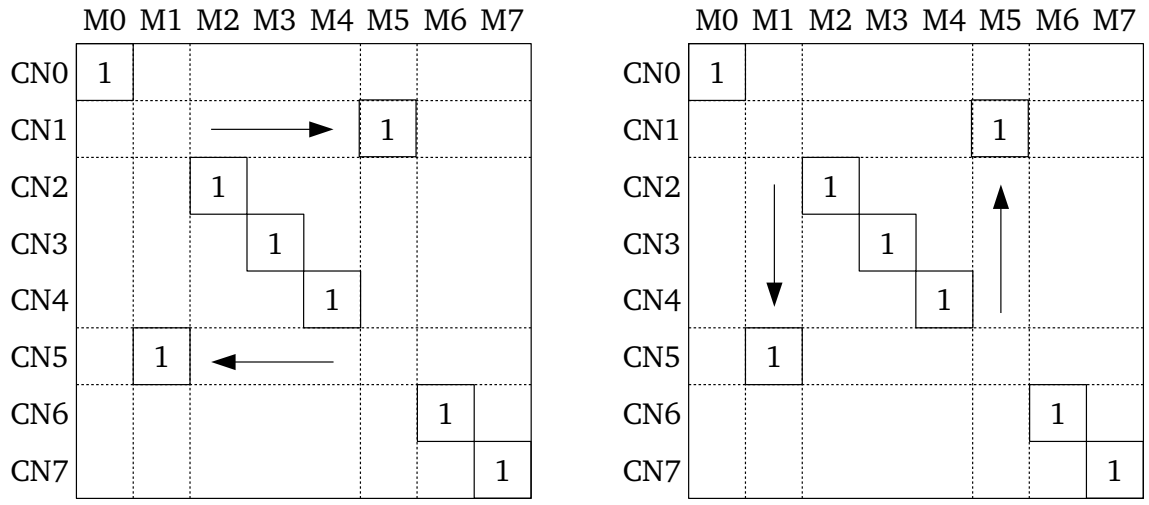
3.3 Modifications on the submatrix distribution

In the Section 2.4 of this work, the motivation to explore some modifications on submatrix distribution was introduced. The idea is to alter the submatrix original structure in a way that does not make the hardware implementation much more complex and so, break the correlation between rows trying to randomize them in some way.

Some different types of modifications will be tested such as row-column exchange, different types of adding ones, adding full or half columns, etc. The modifications will be done over the girth4 free matrix obtained in the Section 3.1.1 and compared with its BER graph showed in Figure 3.12, so these modifications will try to outperform that matrix (the best results until now for $N = 576$ and $R = 1/2$).

3.3.1 Row-Column exchange

Because of the fact that the submatrix is an identity matrix, a row exchange is the same than a column exchange. An example can be shown in Figure 3.32. In Figure 3.32a the columns $M1$ and $M5$ have been exchanged. In Figure 3.32b the rows $CN1$ and $CN5$ have been exchanged. It is easy to see that the result is the same.



(a) Column exchange

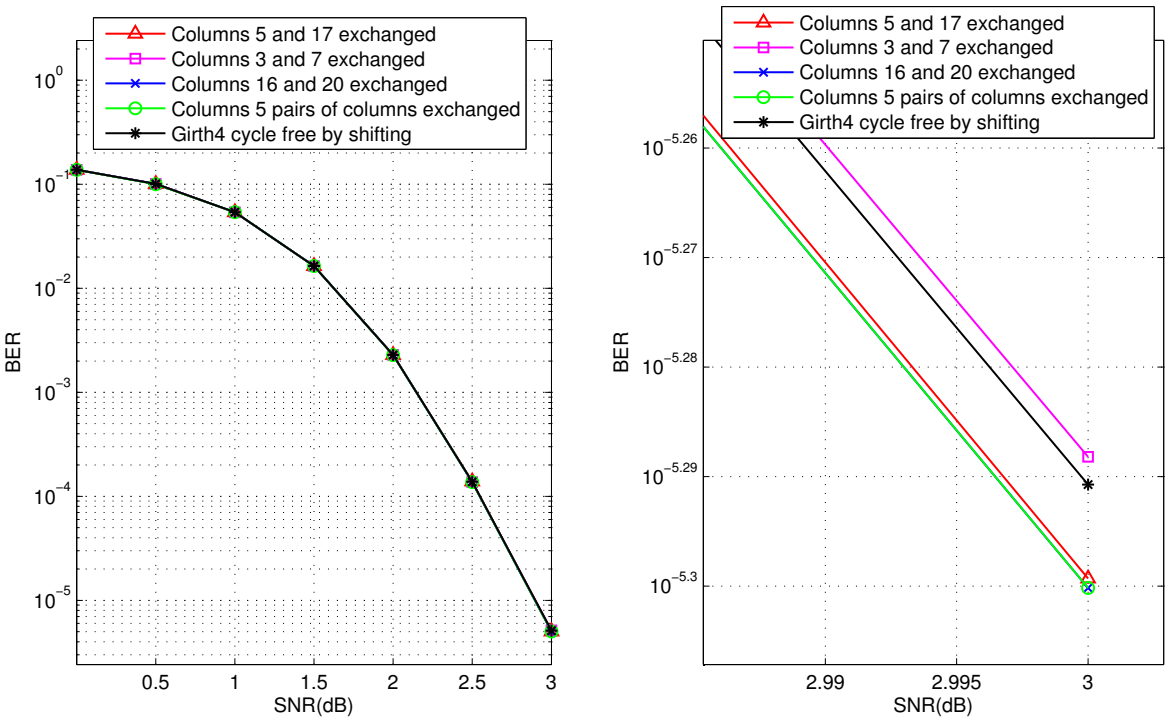
(b) Row exchange

Figure 3.32: Column and row exchange are equivalent in identity matrices

An algorithm to do column exchanges in the submatrices is developed. This algorithm exchanges the same columns of each submatrix. It means, if the exchanged columns are 5 and 12, the columns 5 and 12 are exchanged in all submatrices independently.

The changes of this algorithm would need, in terms of hardware a different redirection of the outputs of the barrel shifter to the check nodes. This redirection would need a change in the actual hardware, but it is not more complex than actual.

The algorithm is applied to the girth4 free WiMAX H matrix, obtained in the previous section, three times, exchanging different pairs of columns in each case. Also a case with five exchanges is done. The BER simulation results can be seen in Figure 3.33.



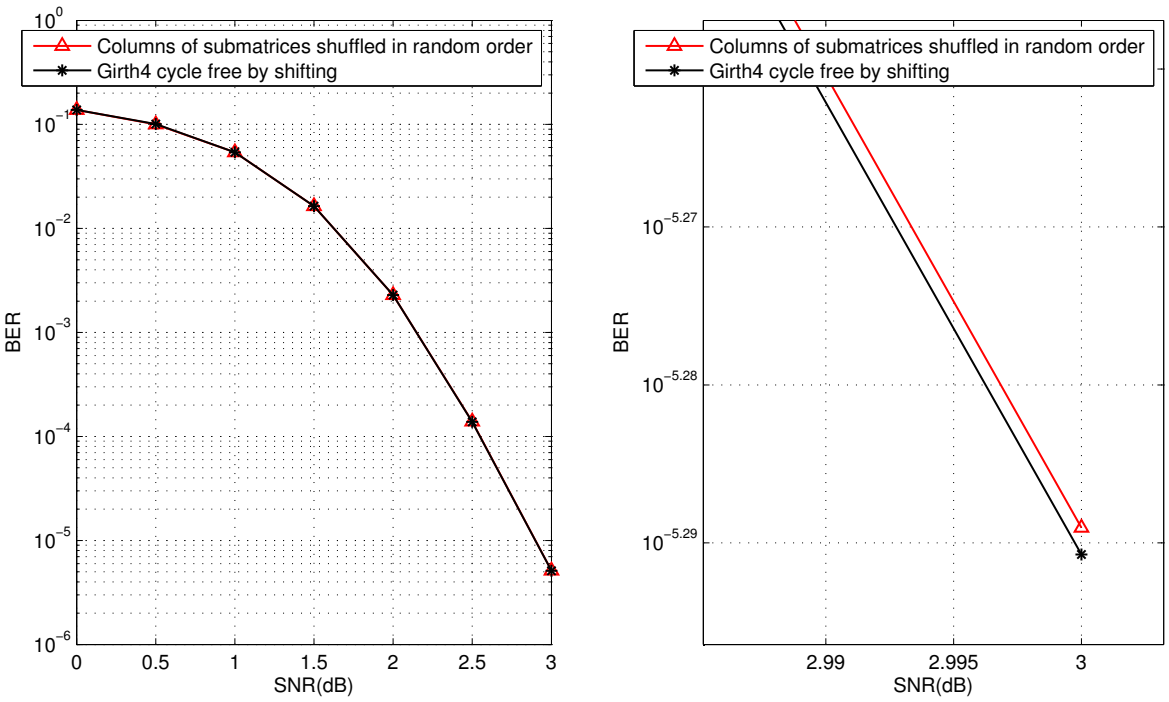
(a) Complete plot

(b) High SNR values zoom

Figure 3.33: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) girth4 free with the same column exchanges for all submatrices

To see any difference in the results a huge zoom is needed so it can be said that the results show no improvement in the BER performance, but not deterioration. The results are too similar to the girth4 WiMAX code. This can be due to the fact that the same exchanges are done in all the submatrices, so this changes do not break in any way the correlation between different row or columns. Besides, because of the type of these modifications, no girth4 cycles are created and so, the results does not show deterioration.

In the next trial, a more general case is tested. In this case, the algorithm shuffles the columns of the submatrix and puts them in a random order. In contrast to the previous case, in this one, the column exchange is not equal for all the submatrices. The column exchange order is common to each column of submatrices, because these share the same barrel shifter, so in terms of hardware, the network that redirects the output of the barrel shifters to the check nodes does not change during the decoding process, and it is simpler. So all the submatrices of a column of submatrices of H matrix have the same column shuffling order. For the others columns of submatrices the order is different. In this way the results are expected to be better because in this case, the correlation between different columns of Base Matrix is broken. The results are shown in Figure 3.34.



(a) Complete plot

(b) High SNR values zoom

Figure 3.34: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) girth4 free with column shuffling common for all the submatrices of a column of Base Matrix

The results are once again, almost the same than girth4 free WiMAX code, so it seems the column exchange is independent to the BER results.

Anyway some final checks will be done in this section. Another last algorithm is developed. This code fractures the submatrices in pieces of a fixed size and shuffles them as well as the previous example did with the columns. An example of this process is shown in Figure 3.35.

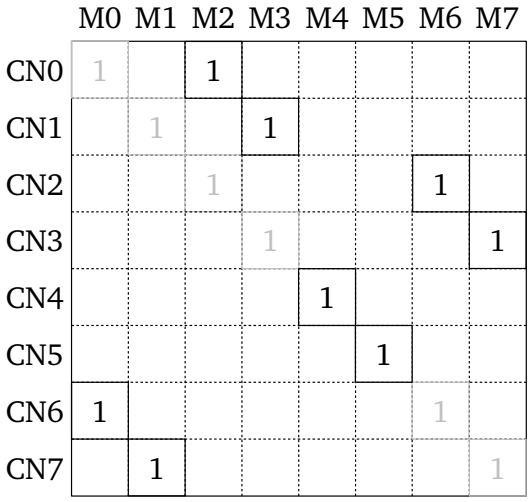
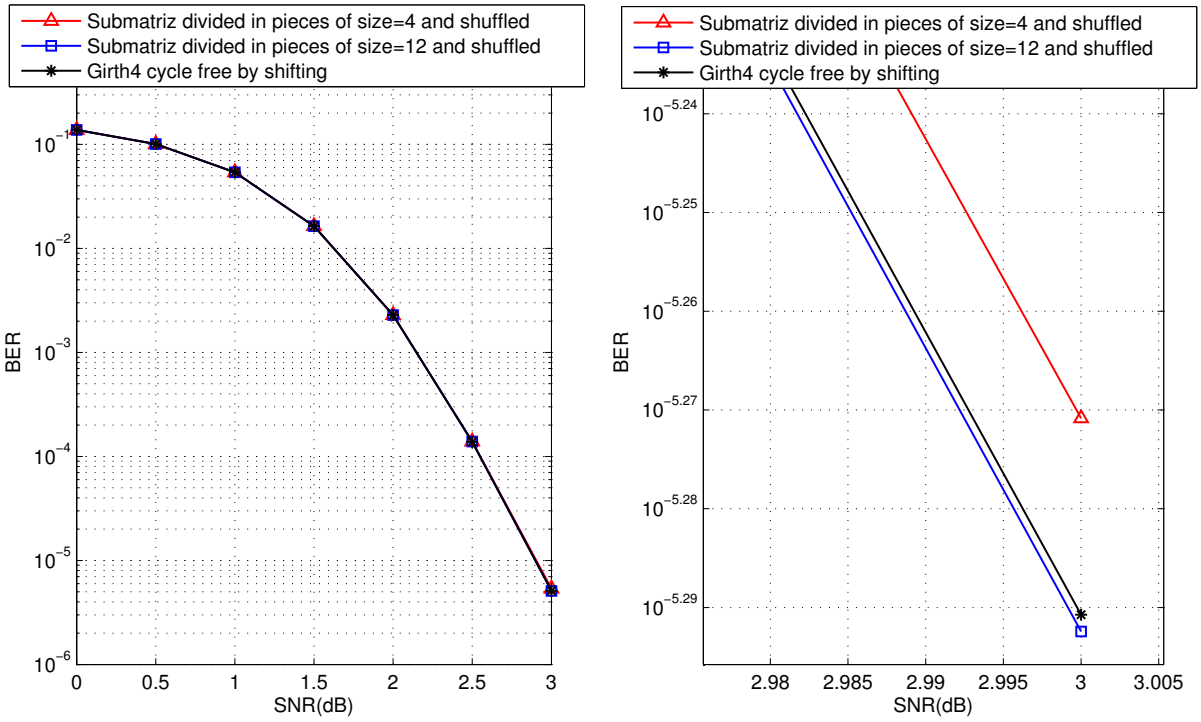


Figure 3.35: Example of submatrix where the original diagonal has been divided in pieces of size=2 and shuffled randomly

This code is run twice over girth4 H matrix, one with a size of pieces of 4 and the other of 12. The BER results are shown in Figure 3.36.

The results, as in the previous cases, do not show any improvement at all. To see the difference between lines a very high zoom is needed. So any hardware change would be justified to get that difference.

With this last trials, this section is concluded.



(a) Complete plot

(b) High SNR values zoom

Figure 3.36: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) girth4 free with each submatrix structure divided in pieces and shuffled. The shuffling is common for all the submatrices of a column of Base Matrix

3.3.2 Adding ones

In this section the possibility of adding 1s to the submatrix in order to alter its structure is introduced. The modification of row weights has been already analyzed in the Section 3.2, but in that case it was made by modifying the whole matrix structure. In this case, the weights are changed inside each one of the non-empty submatrices.

For some of the tests presented in the next pages, the hardware implementation would be too complex and even if the results are good, it would not be a success. But if this is the case, at least it would show to the researchers that this special tpye of submatrix modifications should be deeply studied.

The first of this modifications is to add ones randomly. The algorithm developed adds a 1 in each row of each non-empty submatrix in H. The position of this 1s in each row is a random position that does not create a girth4 cycle in the H matrix. If there is not any position to place an 1 without creating a girth4 cycle in H matrix, not 1s are

added. In this way the row weight is doubled for most rows. An example of a submatrix after adding 1s is shown in Figure 3.37.

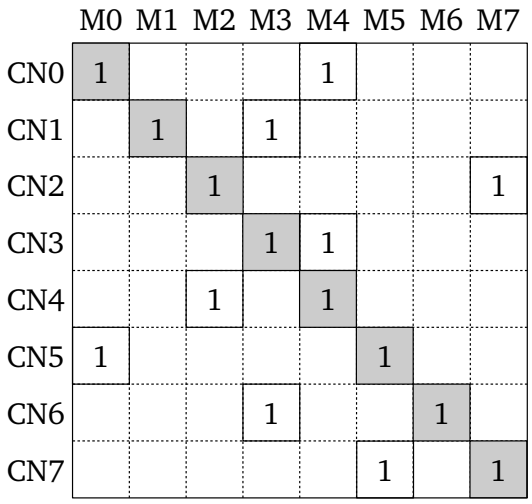
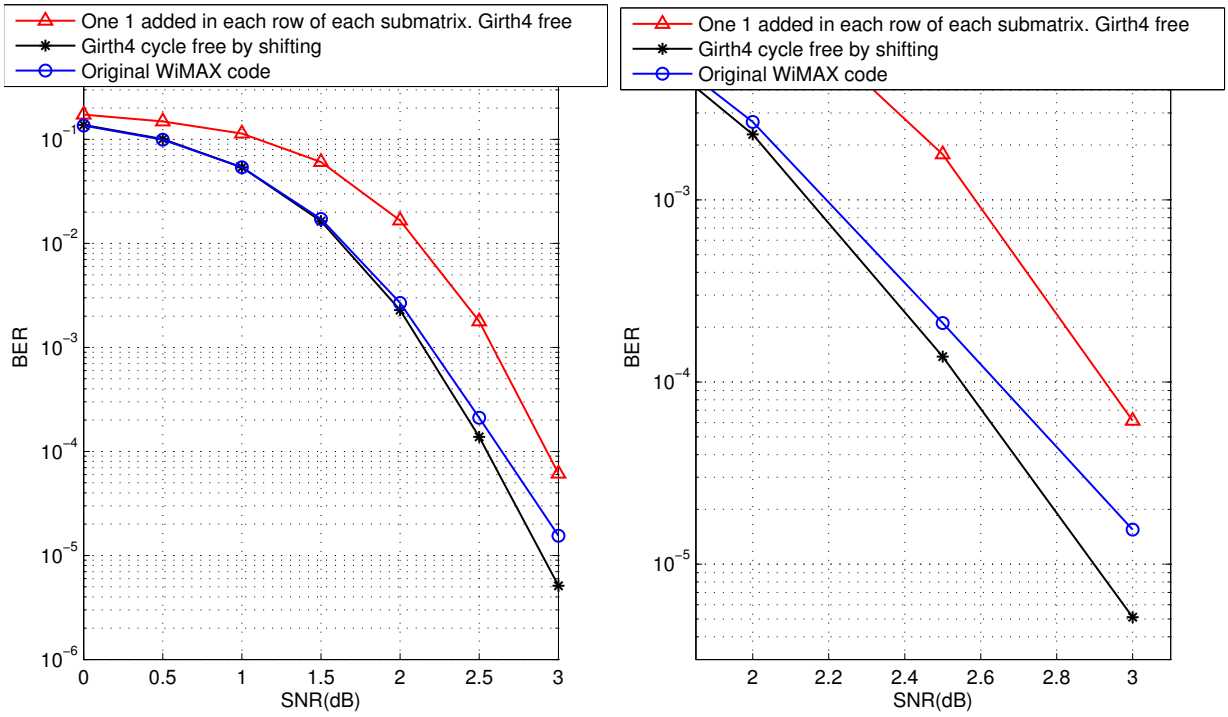


Figure 3.37: Example of submatrix after adding 1s randomly

After applying this algorithm to the girth4 free matrix the BER results obtained are shown in Figure 3.38.

The results show a high deterioration in the BER performance comparing to both girth4 free WiMAX code and original WiMAX code. Besides the hardware complexity of this solution would be high because it would need a big memory of some kilobytes to store the positions of all the ones added.



(a) Complete plot

(b) High SNR values zoom

Figure 3.38: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) girth4 free with 1s added in each row of each non-empty submatrix

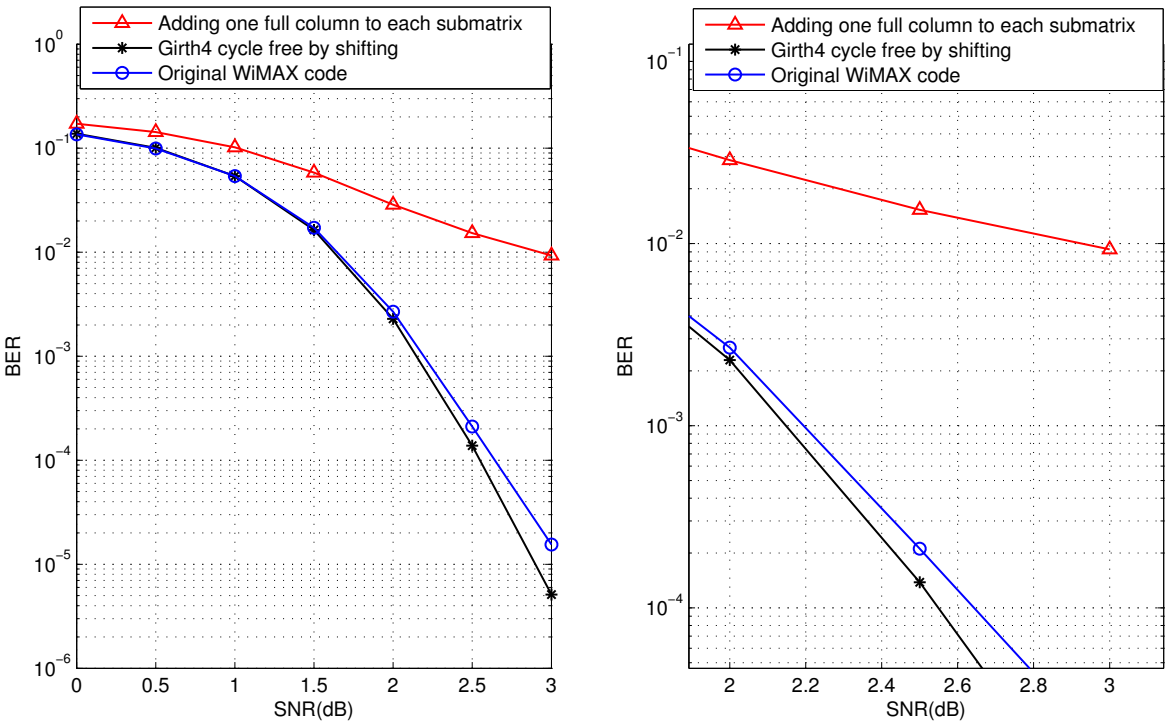
The next solutions are special cases of the last general random positioning of 1s, because in the next cases 1s will be added following some special structures. So if this last simulation did not show any improvement, the next proposals are not expected to do it either.

In the next case, the code developed adds a full column in each submatrix. This full column is placed in a random column. An example of submatrix can be seen in Figure 3.39.

The BER results of this solution can be shown in Figure 3.40. As seen they are much worst than girth4 free WiMAX matrix. The most probably reason seems to be that this solution creates lots of girth4 cycles that are bad for BER performance.

	M0	M1	M2	M3	M4	M5	M6	M7
CN0	1					1		
CN1		1				1		
CN2			1			1		
CN3				1		1		
CN4					1	1		
CN5						1		
CN6						1	1	
CN7						1		1

Figure 3.39: Example of submatrix after adding one full column in a random position



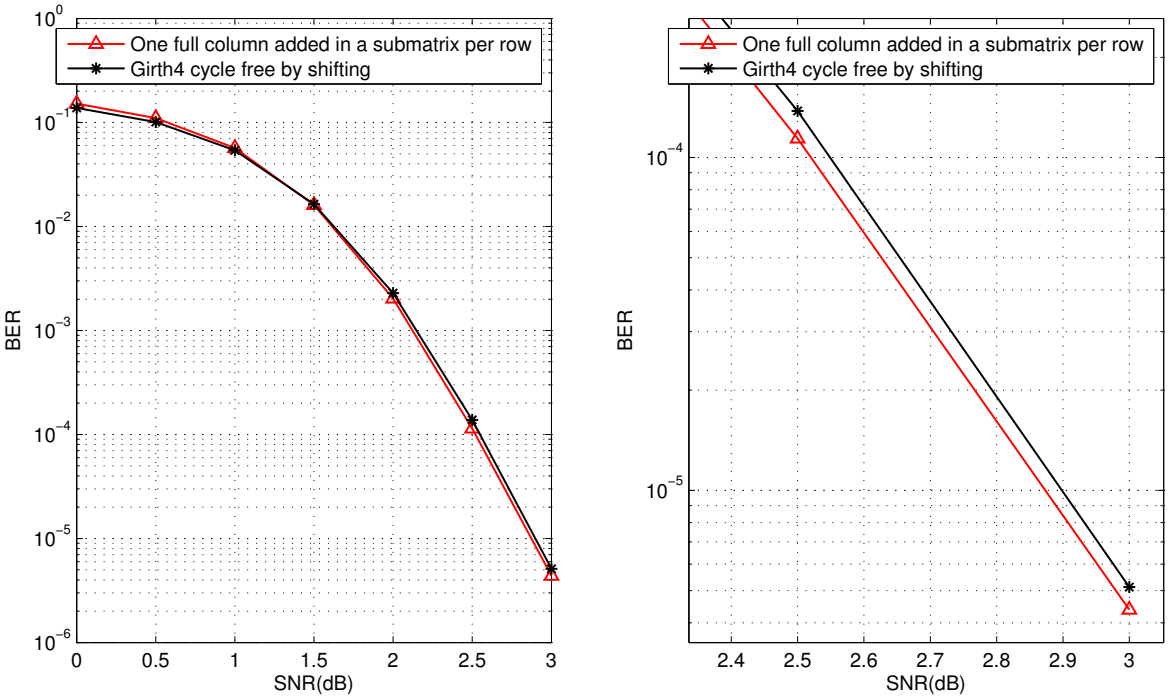
(a) Complete plot

(b) High SNR values zoom

Figure 3.40: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) after adding one full column to each submatrix in H

The next idea is similar to the previous one. A full column is added but not in each submatrix, but only in one submatrix of each row of submatrices. It means, the code developed selects a non-zero submatrix of each row of the left part of Base Matrix and adds a full column in a random position inside that submatrix. Then the code goes to the next row of submatrices in the Base Matrix. The code looks for a position that does not generate girth4 cycles. If there is not any valid position the codes does not add anything, so the result matrix is girth4 free.

The results of this case are shown in Figure 3.41.



(a) Complete plot

(b) High SNR values zoom

Figure 3.41: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) when adding one full column in one submatrix of each row of submatrices

These results show a little improvement for high SNR values and a little deterioration for low SNR values. The hardware cost of this solution would imply a little memory of some bytes to save the position of the columns added and a special network working at the output of the barrel shifters with full columns, that joins a single variable node with lots of check nodes.

The next trial is about adding not full columns, but half columns. The algorithm tries to add two half columns in each row of submatrices. If it can not do it in any case,

without creating girth4 cycles, it does not introduces anything, so the final H matrix is girth4 free. An example of this is shown in Figure 3.42.

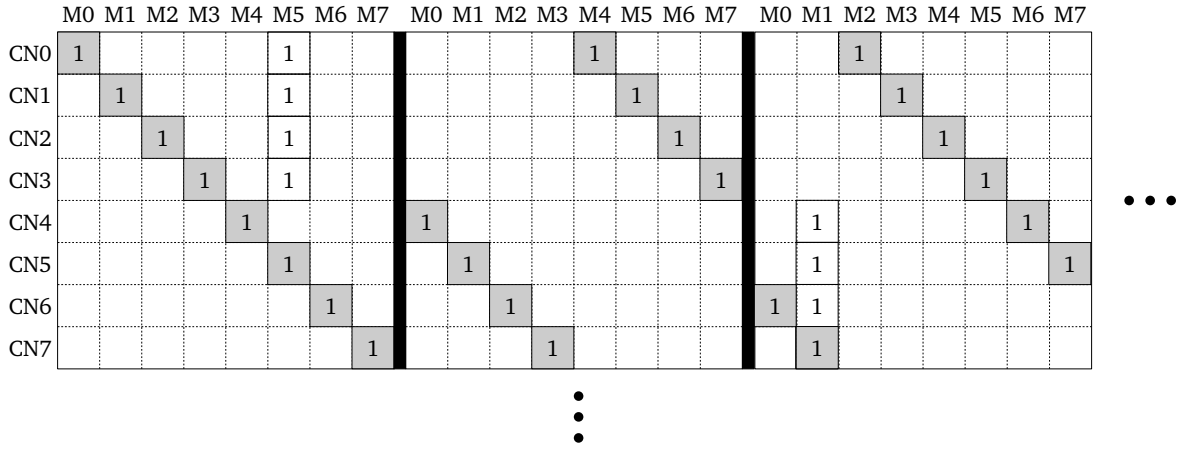
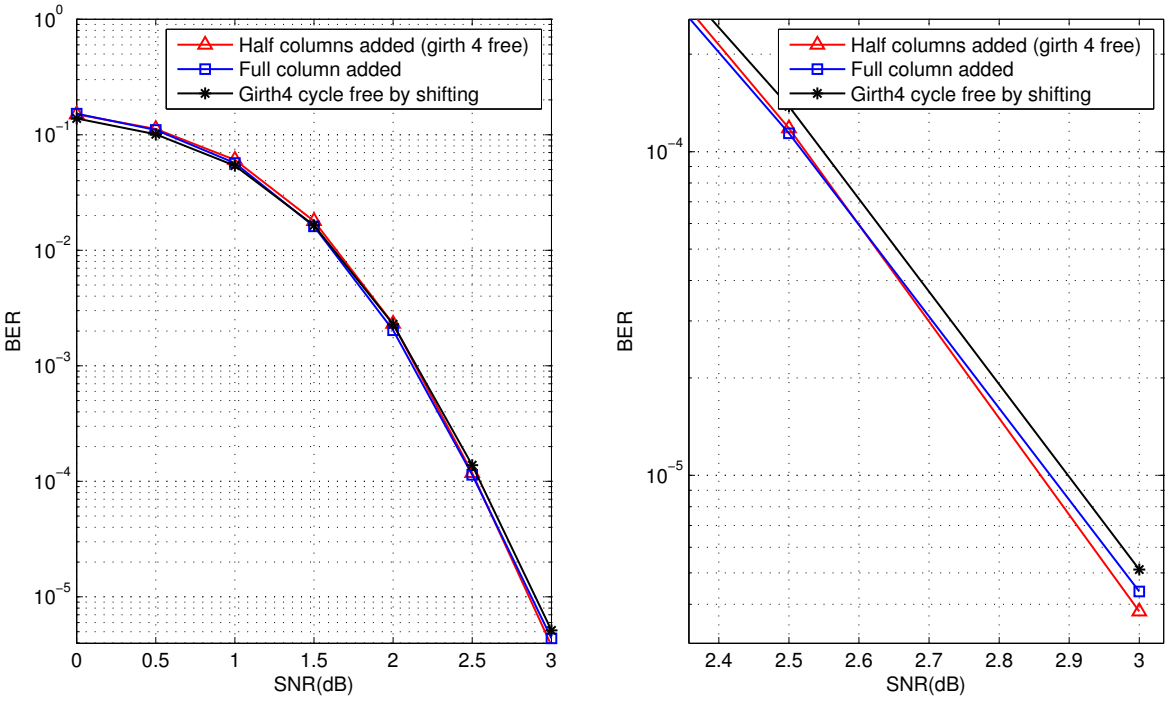


Figure 3.42: Example of a row of submatrices with two half columns added without girth4



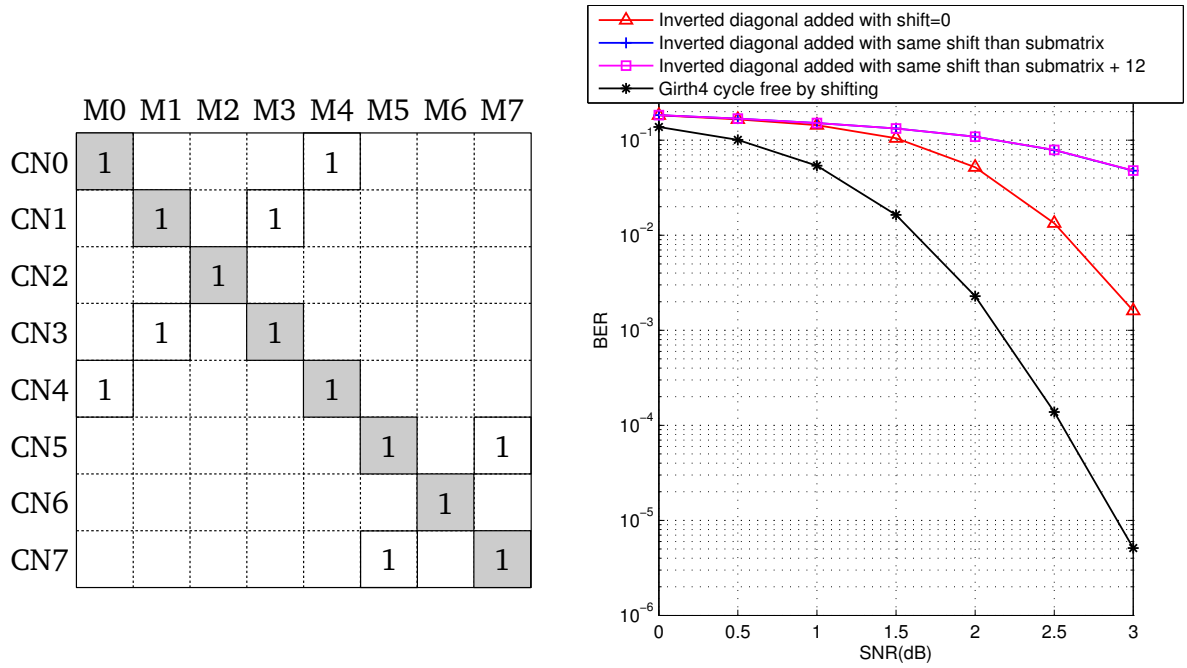
(a) Complete plot

(b) High SNR values zoom

Figure 3.43: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) when adding two halfs of column in each row of submatrices without girth4

The results of this trial can be seen in Figure 3.43. Comparing the results to the previous one of adding full column and also the original WiMAX code girth4 free by shifting, it can be seen this last case, as well as the full column one, outperform the WiMAX girth4 free for high SNR values but not for low values. Besides comparing between last two cases, this last case is only better than full columns case for SNR=3, for previous values the case of full columns is better. But the results are very similar and also similar to the WiMAX girth4 free by shifting. The improvement is too low.

In the following trial the developed code introduces a right circularly shifted inverse diagonal of 1s in each non-empty submatrix. In Figure 3.44a an example of submatrix with an inverse diagonal added can be seen. The inverse diagonal in that image has a circularly right shift value of 5. In Figure 3.44b the BER results of simulation for three different shift values of inverse diagonals are shown.



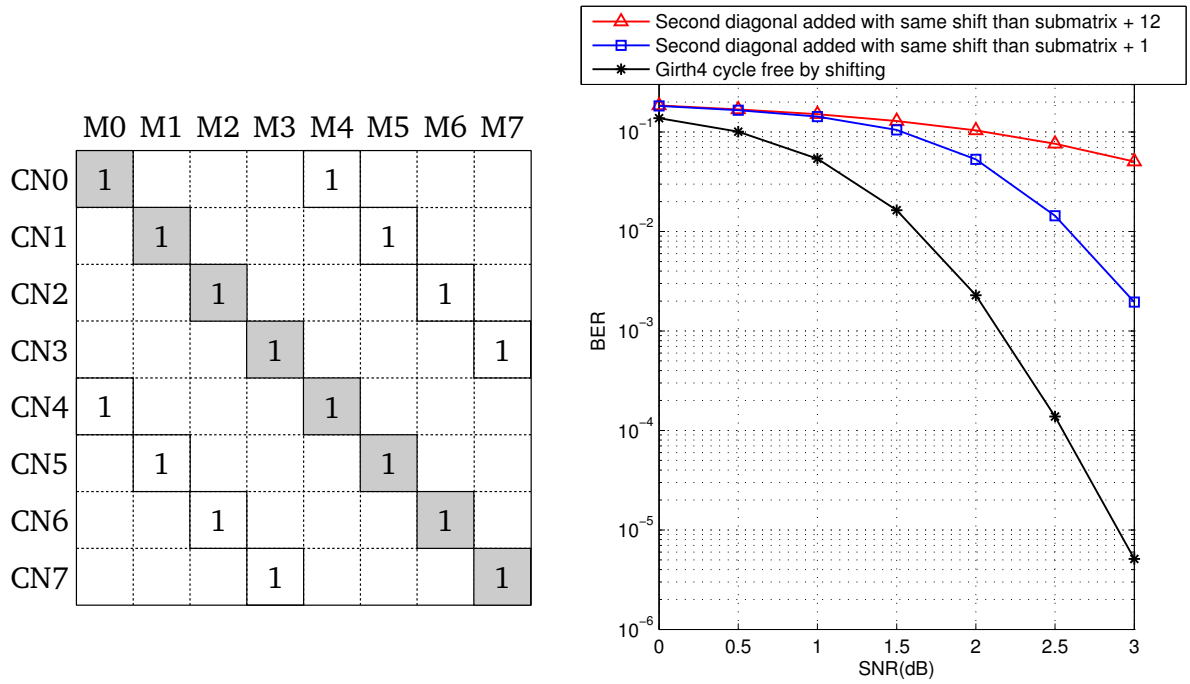
(a) Example of submatrix with inverse diagonal of shift=5 added

(b) Results of some diferents shift values

Figure 3.44: Example of submatrix and BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) when adding one inverse diagonal to each non-empty submatrix

As seen, the results are pretty bad. This may be due to the fact that this inverse diagonals introduce lots of girth4 cycles. Also the fact that row and column weights are very different to the original probably affects.

Another code is developed similar to the previous one but instead of adding an inverse diagonal in each non-empty submatrix, in this code a right circularly shifted normal diagonal is added, so the final structure is a double-diagonal structure. The results can be seen in Figure 3.45b. Some different shift values are tested and with all of them the results are very bad. As well as the previous one, lots of girth4 cycles are added and this can cause the results go bad.



(a) Example of submatrix with double diagonal of shift=4 added

(b) Results of two diferents shift values

Figure 3.45: Example of submatrix and BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) when adding one aditional diagonal to each non-empty submatrix

3.3.3 Special structures

Finally, in this section, two special submatrix structures will be tested in order to alter more the submatrix structure.

The first of this structures is the one shown in Figure 3.46.

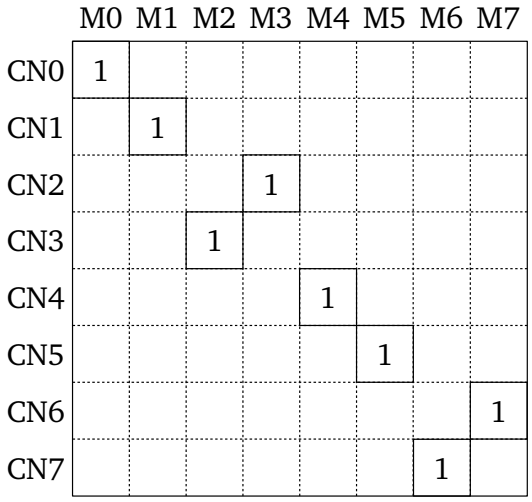
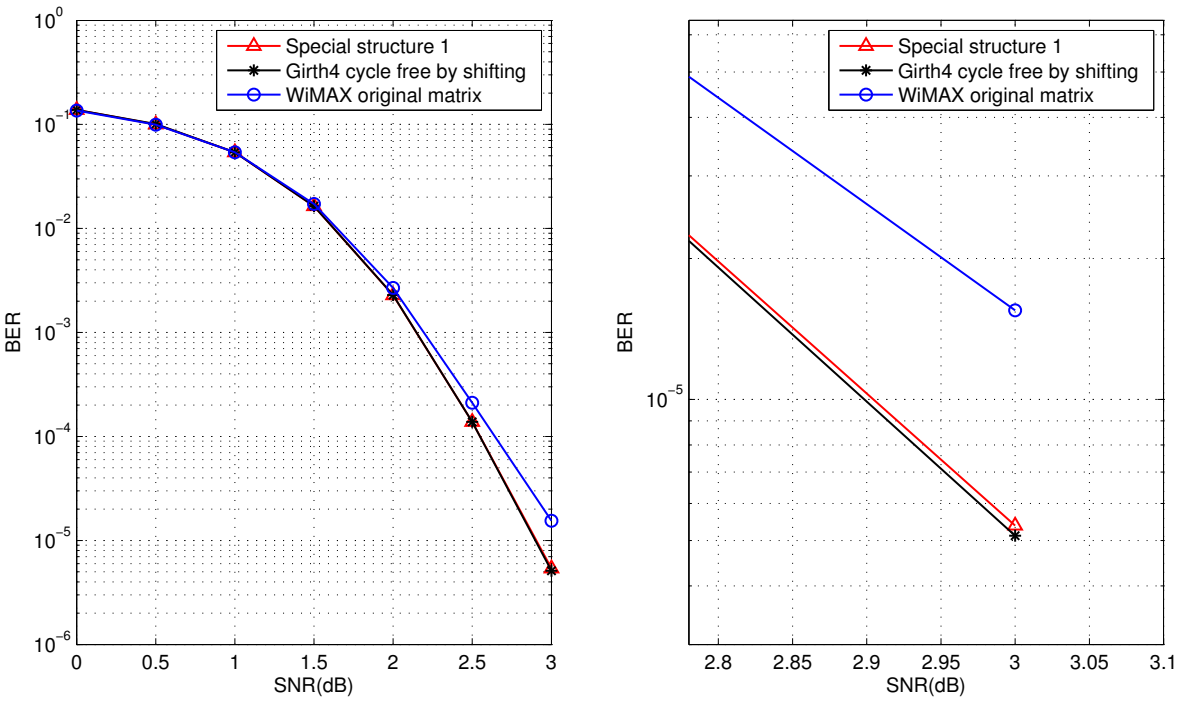


Figure 3.46: Example of submatrix of special structure1

This structure seems a simple column exchange as the ones done in Section 3.3.1 but it is different. When column exchanges were done, they were done in the same way independent of the shift value, it means, if the exchange were between column 4 and 20, it did not mind if the shift value was 0 or 20 for that submatrix, the exchange was done between columns 4 and 20. This structure replaces the identity submatrix so it shifts in the same way as the identity matrix does. It means that the column exchange is different depending on the shift value.

In hardware this would translate to a multiplexors exchange network before the barrel shifter and not after like in previous cases. Changing the way in which the data goes into the barrel shifter can produce this structure and similar ones.

The code developed replaces the identity submatrices of non-empty submatrices with circular right shifts of this structure using the same shift value that the submatrix had previously. After this replacing of submatrices, some girth4 cycles are generated but it was possible to remove them with the algorithm of Figure 3.11. The BER simulation results can be seen in Figure 3.47.



(a) Complete plot

(b) High SNR values zoom

Figure 3.47: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with special structure 1 (Figure 3.46)

Once again, this exchange of columns (even if it depends on the submatrix shift value) does not show any improvement over the girth4 free WiMAX matrix, but not a deterioration, the results are almost equal.

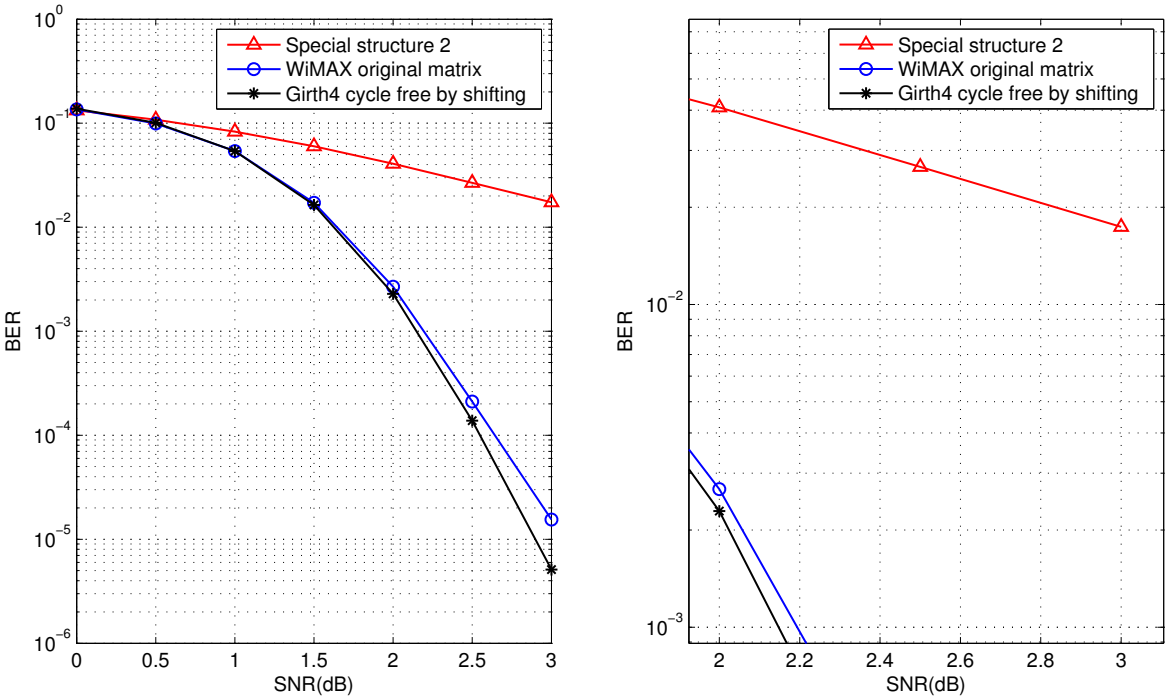
The next trial is similar to previous one. The submatrix structure is changed to the one in Figure 3.48 and it also shifts with the shift value of the submatrix.

Because of the special structure of this case it is not possible to remove all girth4 cycles, so the results are not expected to be better than girth4 free WiMAX submatrix obtained by shifting, few chapters before. The BER results can be seen in Figure 3.49

The results obtained are pretty bad, as said, because of the many girth4 cycles this structure introduces.

	M0	M1	M2	M3	M4	M5	M6	M7
CN0	1							
CN1	1							
CN2			1					
CN3			1					
CN4					1			
CN5					1			
CN6							1	
CN7							1	

Figure 3.48: Example of submatrix of special structure2



(a) Complete plot

(b) High SNR values zoom

Figure 3.49: BER simulation results plots of WiMAX code ($N = 576$, $R = 1/2$) with special structure 2 (Figure 3.48)

Finally, the summary of the last section is presented in Table 3.12. In this case the obtained values are compared (the color and arrows) again to the WiMAX girth4 free matrix by shifting obtained in the first section of this chapter and that is the best result obtained until now, not the original WiMAX code. As before this exposition of the simulation values also allows to compare the results between different approaches.

Table 3.12: Summary of results of Section 3.3 (Modifications on the submatrix distribution), $N = 576$, $R = 1/2$

SNR(dB)	Original WiMAX $N = 576$, $R = 1/2$	Girth4 free by shifting Fig: 3.12	5 pairs of columns exchanged. Girth4 free Fig: 3.33	Columns shuffled in random order. Girth4 free Fig. 3.34
0.0	1.36e-01	1.38e-01	≈ 1.38e-01	≈ 1.38e-01
0.5	9.93e-02	1.01e-01	↓ 1.00e-01	↓ 1.00e-01
1.0	5.39e-02	5.39e-02	≈ 5.39e-02	≈ 5.39e-02
1.5	1.72e-02	1.64e-02	≈ 1.64e-02	≈ 1.64e-02
2.0	2.68e-03	2.29e-03	↓ 2.28e-03	≈ 2.29e-03
2.5	2.11e-04	1.38e-04	↓ 1.37e-04	↑ 1.39e-04
3.0	1.55e-05	5.12e-06	↓ 5.01e-06	↑ 5.14e-06

SNR(dB)	Divided in pieces of size=4 and shuffled. Girth4 free Fig. 3.36	Divided in pieces of size=12 and shuffled. Girth4 free Fig. 3.36	Add 1s in each row into each submatrix. Girth4 Fig. 3.38	Add one full column into each submatrix. free Fig. 3.40
0.0	≈ 1.38e-01	≈ 1.38e-01	↑ 1.73e-01	↑ 1.72e-01
0.5	≈ 1.01e-01	≈ 1.01e-01	↑ 1.49e-01	↑ 1.43e-01
1.0	≈ 5.39e-02	≈ 5.39e-02	↑ 1.14e-01	↑ 1.02e-01
1.5	≈ 1.64e-02	≈ 1.64e-02	↑ 6.06e-02	↑ 5.83e-02
2.0	≈ 2.29e-03	≈ 2.29e-03	↑↑ 1.66e-02	↑↑ 2.87e-02
2.5	↑ 1.39e-04	↑ 1.39e-04	↑↑ 1.78e-03	↑↑ 1.53e-02
3.0	↑ 5.36e-06	↓ 5.10e-06	↑↑ 6.13e-05	↑↑ 9.29e-03

SNR(dB)	Add one full column in each row of subm. Girth4 free Fig 3.41	Add two half columns in each row of sub. Girth4 free Fig 3.43	Add inverted diagonal offset=0 Fig. 3.44b	Add additional diagonal offset=same+1 Fig 3.45b
0.0	↑ 1.52e-01	↑ 1.50e-01	↑ 1.83e-01	↑ 1.83e-01
0.5	↑ 1.10e-01	↑ 1.12e-01	↑ 1.66e-01	↑ 1.66e-01
1.0	↑ 5.68e-02	↑ 6.05e-02	↑ 1.44e-01	↑ 1.43e-01
1.5	↓ 1.61e-02	↑ 1.79e-02	↑ 1.05e-01	↑ 1.05e-01
2.0	↓ 2.03e-03	↑ 2.31e-03	↑↑ 5.21e-02	↑↑ 5.32e-02
2.5	↓ 1.14e-04	↓ 1.18e-04	↑↑ 1.34e-02	↑↑ 1.44e-02
3.0	↓ 4.38e-06	↓ 3.80e-06	↑↑ 1.61e-03	↑↑ 1.95e-03

SNR(dB)	Special structure 1 (Fig. 3.46) Fig. 3.47	Special structure 2 (Fig. 3.48) Fig. 3.49
0.0	≈ 1.38e-01	↓ 1.33e-01
0.5	↓ 1.00e-01	↑ 1.08e-01
1.0	≈ 5.39e-02	↑ 8.31e-02
1.5	≈ 1.64e-02	↑ 6.00e-02
2.0	↓ 2.28e-03	↑↑ 4.08e-02
2.5	≈ 1.38e-04	↑↑ 2.67e-02
3.0	↑ 5.38e-06	↑↑ 1.74e-02

As seen in this summary and previously analyzed during this section, it seems that all the possible modifications done by exchanging columns do not modify the results in a significant way. Most of the other approaches tested get very bad results, and only for some of them like adding full or half columns the simulations get some improvements for high SNR values. For the cases where several similar tests were done (different pairs of columns, offset values...), only the best case has been introduced in this table.

4

Conclusions

In this chapter, the main ideas and conclusions obtained during the study on this work are exposed, focusing on the results obtained during the simulations of Chapter 3. In a second section, the possible future lines of research on this field of study will be introduced, to continue with the work presented in this report.

4.1 Conclusions

Since the final of XX century, mankind is living a huge technology boom. Nowadays almost all families have at home one wireless Internet connection and several wireless devices. Also in Hospitals, Universities, etc. the technology has become necessary. During last 20 years we all have experimented the improvements in these technologies, since in the 90's a good home Internet connection had a speed of 56kbps, nowadays it is possible to get a connection of several Mbps in the cell phone, or more than 100Mbps at home. This progress in communications are due to lots of technology factors such as the insertion of optic fiber, progresses in electronics etc. but one of the factors that leads to a increase in the communication's speed is the improvement of error-correcting codes. In this work, the LDPC used in WiMAX technology have been studied.

LDPC codes are much older than most of technologies used nowadays. They were discovered in the early 60's but they remained unknown until the 90's. The rediscovery of these codes showed they could beat the best error-correcting codes used in those days. Their good performance and their easy hardware implementation tell us that they could be used for most of new communication technologies in the future. So, the study of these codes is justified.

In the Chapter 3 of this work, a wide range of modifications has been tested over the WiMAX original LDPC codes in order to outperform their BER and FER results without making the hardware implementation much more complex.

The idea that says that the higher girth free possible, the best BER performance of the code, has been introduced by lots of researchers in the last few years so in the Section 3.1, the objective of the modifications was exactly that, to remove as many girth cycles as possible.

In Section 3.1.1 some different modifications were done in order to remove girth4 cycles of the WiMAX code and it was shown that by eliminating more girth4 cycles the performance increased. When all the girth4 cycles were removed from the code of $N = 576$ and $R = 1/2$ without altering the code weights, the simulations showed a huge BER improvement over the WiMAX original code. From that moment all the next trials would try to outperform this last good result. The girth4 cycles were removed also for different sizes and rates but the results did not show as good improvements for those cases.

In the Section 3.1.2 some different strategies were tested in order to remove girth6 cycles from the code matrix. This task was much more complex than before because the structured right part of code matrix . For the mainly treated code in this thesis, almost all the girth6 cycles were removed by shifting submatrices (the matrix was already girth4 free), but despite all predictions, these results did not show any improvement over the girth4 free code. The girth6 removal was also carried out over codes of other sizes, showing a good improvement for the size of $N = 1440$ and $R = 1/2$ and a little improvement for the code of $N = 2304$ and $R = 1/2$.

This section was then concluded showing that the removal of girth cycles seems to be important for low girth level and that the improvement is not homogeneous for all sizes and rates.

In the Section 3.2, the Base Matrix structure was altered in order to generate: a) girth free Code Matrices and b) different random Base Matrices distributions. First some trials changing the row weights were done and the results showed that the both cases, with higher row weights than original, and with less row weights than original, caused bad BER results. This results showed that the original code weights were carefully selected, because any change in those weights generate worst results than originals.

In this section, some trials were done conserving the same weights than original code, but changing the non-empty submatrices distribution inside each row of Base Matrix. In this way, it was possible to generate girth4 and girth6 codes of size $N = 576$ and $R = 1/2$ that had not been possible until that moment. The results showed very similar performance than the girth4 free WiMAX code matrix obtained by shifting submatrices. They got a very little improvement for high SNR values but not for medium and low values.

The final Section 3.3 was about altering the submatrix internal distribution in order to randomize it and get better performance. All the trials done in this section by exchanging columns inside the submatrices did not get better performance but neither worst perfromance, these changes seemed to be totally independent of the performance.

In this section also some trials bout introducing more 1s to each submatrix were tested. All the trials showed very bad performance but the trial about introducing full columns and also half columns in each row of submatrices. These cases could outperform the girth4 best code until now, a little bit. This improvement was not enough to make the necessary hardware changes be beneficial.

Some main conclusions could be extracted such as it seems the good design of the weights in the Base Matrix seems to be more important than girth cycles removal, at least for girth levels higher than 4. Good Base Matrices with same row weights than original WiMAX code and also girth6 free were developed and they did not outperform the girth4 free WiMAX code by much. This tells us that the Base Matrix distribution was carefully selected. On the other hand, some researchers studied this before [12] and got some randomly generated Base Matrices that outperformed the original.

About the internal submatrix distribution, the main idea is that is not easy to create a submatrix distribution different than the identity matrix and that is not potential generator of girth4 cycles. Even if the submatrix structure is girth4 free it can generate lots of girth4 cycles when it is replicated in the full code matrix. Only by adding full columns the results were not bad.

Lots of solutions that outperformed the WiMAX original code matrix were found, but almost none that outperformed the girth4 free code obtained by shifting submatrices.

4.2 Future lines of research

After analyzing the results and the conclusions, it seems that the best line of research to continue studying these type of codes would be in the field of generating different Base Matrices, because even if it has been studied here, only the surface has been scratched. This field offers innumerable possibilities that have not been studied in this thesis and that could lead to better performance. The idea of generating random Base Matrices at the same time that caring about the potential possible girth cycles seems interesting and a wide research could be done on it.

About the submatrix distribution, it also offers lots of possibilities, but seeing the cases studied here, it seems that the only cases that show a little improvement in the performance would require several modifications in the hardware implementation. All the rest of the trials do not show improvements or show high deterioration in the performance.

Finally as it is known, the girth removal is important but for this special codes, the structured right part restricts the possibilities a lot, and the research on this field seems not to be easy or fruitful.

A

Annex A. Attached material

This annex pretends to be a guide of reference to understand the attached material. This material consists of the MATLAB programming codes used to modify the matrices in this thesis and also the output files of the simulation processes. The MATLAB codes are listed an briefly explained. This information will cover the explanation of its operation, as well as a list of the input parameters and the output variables.

For the simulation results, more than 100 simulations were carried out during this thesis, but only the most important where exposed. The attached folder contains these important results where each file has its description in its name.

As said before, this chapter is a reference to understand and be able to use the MATLAB codes attached. For each code the main idea of its function will be exposed, and the input and output variables will be presented. All the codes but 1 were developed specifically for this thesis, only *LDPC_girth4a.m* was not. The source of this code is included in its description. More than 50 codes were developed during the period of work on this thesis, here only the more relevant ones are presented. To use them with matrices different of $N = 576$ and $R = 1/2$ some of them need modifications.

- *convert.m*
  This code converts a matrix stored in MATLAB to a file of extension .txt that can be used to generate the .pchk and .gen files needed to run a simulation with the simulator provided by the DET of the *Politecnico di Torino*.

  Input arguments:
  H: Matrix to convert
  nombre_archivo: Char string that will be the name of the file .txt at the output.

  Output arguments:
  Not output arguments in MATLAB. This code generates a .txt file.

- $H = inverse\_convert.m$
  This code takes .txt file with: the first row indicating the rows and columns of the

code and a row for each row of the code indicating the positions with 1s. The output is the MATLAB H matrix of that .txt file.

Input arguments:
nombrefichero: Char string of the complete name of the file .txt to convert to a MATLAB matrix.

Output arguments:
H: Resultant matrix of the conversion.

- *encontrar_girth4_2.m*
  This code finds all the girth4 cycles and stores the top left position in an array.

  Input arguments:
  H: Matrix to detect.

  Output arguments:
  girth: Array of elements that are the top left position of each girth4 cycle in the H matrix.

- *LDPC_girth4a.m*
  This code finds if there is one or more girth4 cycles in the H matrix.

  Input arguments:
  H: Matrix to detect girth4.

  Output arguments:
  out: This variable is equal to 0 if there are not girth4 cycles in H matrix and equal to 1 if there are girth4 cycles.

  Source:
  This code was developed by Yang XIAO, Beijing Jiaotong University (BJTU) on July 22 of 2007.

- *rotatotaldondehaygirth4.m*
  This code takes a code matrix and symmetrically inverts the top left submatrices that form each girth4 cycle.

  Input arguments:
  H: Matrix to operate with.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  $H1$: Resultant matrix of the operation over H matrix.

- *elimina1dondegirth4.m*
  This code takes a code matrix, looks for its girth4 cycles and remove one 1 of each cycle to remove all cycles.

  Input arguments:
  H: Matrix to operate with.

Output arguments:
$H1$: Girth4 free H matrix.

- *rota*1*dondehaygirth*4*_cycle.m*
  This code takes a code matrix and rotates the top left submatrices that form each girth4 cycle until all the girth4 is removed.

  Input arguments:
  H: Matrix to operate with.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  $H1$: Resultant matrix of the operation over H matrix.

- *girth*6*.m*
  This code takes a code matrix and detects if there is girth6 or not. It takes as parameter a girth4 free matrix.

  Input arguments:
  H: Girth4 free matrix to detect girth6.

  Output arguments:
  It prints on the screen "There is girth6" if there are girth6 cycles or "There is not girth6" if there are not.

- *crearhconbasematrix.m*
  This code takes a Base Matrix (only uses the non empty positions for the left part) and creates a H matrix without girth4 and girth6 by introducing submatrices and rotating them. If for any submatrix, any shift value gets the objective, it is filled with an empty submatrix. Some modifications can be done so it only tries to get a girth4 matrix.

  Input arguments:
  $Base\_Matrix$: Base Matrix to operate with.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  H: Resultant code matrix.
  $fails$: Positions of the submatrices that could not be filled without creating girth4 or girth6 cycles.

- *encontrar_girth*6*_adaptadoparacrearhconbasematrix.m*
  This code takes a code matrix and detects the number of rows with girth6 cycles.

  Input arguments:
  H: H code matrix to detect.
  combinaciones: A vector with all the possible combinations of three rows.
  $fila$: Row until the code needs to check the girth6 (This code is used in an iterative algorithm so with this system it takes less).

Output arguments:
numero: Number of rows involved in girth6 cycles in H matrix.

- $Base\_Matrix = busca\_base\_matrix(H, tamano_submatrix).m$
  This code takes a code matrix and gets its Base Matrix of 0s and 1s (only the position of non-empty matrices).

  Input arguments:
  H: H code matrix.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  $Base\_Matrix$: Base Matrix of 0s and 1s of that H matrix.

- *busca_base_matrix_con_indices.m*
  This code takes a code matrix and gets its Base Matrix of shift values.

  Input arguments:
  H: H code matrix.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  $Base\_Matrix$: Base Matrix of shift values of each submatrix.

- *crear_base_buena_a_partir_de_indices_paper.m*
  This code takes a Base Matrix with shift values and try to change them so it gets a girth4 and girth6 free matrix. It uses other codes to detect the position on the Base Matrix that is involved in more cycles and then changes its shift value.

  Input arguments:
  $Base\_Matrix1$: Base Matrix to improve.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  $Base\_Matrix$: Base Matrix improved.

- *encontrar_girth4_adaptadoparacrearhconbasematrix_paper.m*
  This code takes a Base Matrix with shift values and calculate the girth4 cycles in which each element of the Base Matrix is involved.

  Input arguments:
  $Base\_Matrix1$: Base Matrix to detect.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  $matrix\_girth4$: Matrix of the size of Base Matrix with the number of the girth4 cycles in which the element of that position in Base Matrix is involved.

- *encontrar_girth6_adaptadoparacrearhconbasematrix_paper.m*
  This code takes a Base Matrix with shift values and calculates the girth6 cycles in which each element of the Base Matrix is involved.

  Input arguments:
  $Base\_Matrix1$: Base Matrix to detect.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  matrix_girth6: Matrix of the size of Base Matrix with the number of the girth6 cycles in which the element of that position in Base Matrix is involved.

- *crearh_con_base_buena_paper.m*
  This code takes a Base Matrix with shift values and creates its H correspondent matrix.

  Input arguments:
  $Base\_Matrix1$: Base Matrix.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  H: Code matrix correspondent to the input Base Matrix.

- *crear_base_buena_paper.m*
  This code takes a Base Matrix of 0s and 1s (1s in the non-empty submatrices) or a vector of row weights. If the argument is a Base Matrix it generates a Base Matrix of shift values introducing random shift values on the positions of non-empty submatrices. If the argument is a row weights vector, it places the non-empty submatrices in random positions in each row according to the weight of that row stored in the argument vector. Then this code uses *crear_base_buena_a_partir_de_indices_paper.m* to try to remove girth cycles.

  Input arguments:
  $Base\_Matrix\_de\_1s\_o\_vector$: Base Matrix of 0s and 1s or row weights vector.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  $Base\_Matrix2$: Base Matrix improved.

- *generarbasematrixsingirth.m*
  This code takes a row weights vector and generates a Base Matrix of 0s and 1s with those row weights placing the non-empty submatrices strategically with the minimum girth4 and girth6 cycles possible.

  Input arguments:
  vector: Vector of row weights.

  Output arguments:
  $Base\_Matrix$: New Base Matrix of 0s and 1s created.

It also prints the number of rows involved in girth4 and girth6 cycles that still exist in the Base Matrix.

- *encontrar_girth4_adaptadoparagenerarbasematrix.m*
  This code takes an H matrix and calculates the number of rows involved in girth4 cycles.

  Input arguments:
  H: H code matrix to detect.

  Output arguments:
  numero: Number of rows involved in girth4 cycles.

- *swap.m*
  This code takes an H matrix exchanges the columns in the arguments for all the submatrices.

  Input arguments:
  H: Code H matrix to modify.
  *columna*1: Column 1 to exchange.
  *columna*2: Column 2 to exchange.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  $H1$: Modified H code matrix.

- *romperentrozos.m*
  This code takes an H matrix, divides each submatrix in pieces and shuffles these pieces. The shuffling order is equal for all the submatrices in a column of Base Matrix.

  Input arguments:
  H: Code H matrix to modify.
  tamano_submatrix: Size of the submatrix.
  trozos: Number of pieces in which the submatrix is divided.

  Output arguments:
  $H1$: Modified H code matrix.

- *mete*1.*m*
  This code introduces 1s in each row of each submatrix if it does not create girth4 cycles. If it creates girth cycles in all possible positions, it does not introduce anything.

  Input arguments:
  $H1$: Code H matrix to modify.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  $H2$: Modified H code matrix.

- *add_columnas_enteras.m*
  This code introduces one full column of 1s in each submatrix in a random position inside this.

  Input arguments:
  H: Code H matrix to modify.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  $H1$: Modified H code matrix.

- *add_1colxfila_entera.m*
  This code introduces one full column of 1s in a random position inside each row of submatrices if it does not create girth4 cycles. If all possible positions create girth4 cycles no rows are added in that row of submatrices.

  Input arguments:
  H: Code H matrix to modify.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  $H1$: Modified H code matrix.

- *add_mediascolxfila.m*
  This code introduces two separated half columns of 1s in two random positions inside each row of submatrices if it does not create girth4 cycles. If all possible positions for a half column create girth4 cycles that half is not added.

  Input arguments:
  H: Code H matrix to modify.
  tamano_submatrix: Size of the submatrix.

  Output arguments:
  $H1$: Modified H code matrix.

- *meterdiagonalinversa.m*
  This code introduces an additional inverse diagonal in each submatrix. This diagonal is shifted by the value of an argument.

  Input arguments:
  H: Code H matrix to modify.
  tamano_submatrix: Size of the submatrix.
  desplazamiento: The shift value of the inverse diagonal introduced.

  Output arguments:
  $H1$: Modified H code matrix.

- *doble_diagonal.m*
  This code introduces an additional diagonal in each submatrix. This diagonal is shifted by the value of an argument.

Input arguments:
H: Code H matrix to modify.
$tamano\_submatrix$: Size of the submatrix.
$desplazamiento$: The shift value of the inverse diagonal introduced.

Output arguments:
$H1$: Modified H code matrix.

- $forma\_rara2\_inversionespequenas.m$
  This code replaces each submatrix with the special structure of Figure 3.46.

  Input arguments:
  H: Code H matrix to modify.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  $H1$: Modified H code matrix.

- $forma\_rara1\_columnitasde2.m$
  This code replaces each submatrix with the special structure of Figure 3.48.

  Input arguments:
  H: Code H matrix to modify.
  $tamano\_submatrix$: Size of the submatrix.

  Output arguments:
  $H1$: Modified H code matrix.

Bibliography

[1] R.G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, Massachusetts: M.I.T. Press, 1963.

[2] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1", *IEEE International Conference on Communications (ICC)*, vol. 2, pp. 1064-1070, Geneva, May 1993.

[3] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge, UK: Cambridge University Press, 2003.

[4] M. P. C. Fossorier, "Quasi-Cyclic Low Density Parity Check Codes From Circulant Permutation Matrices", *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1788-1793, Aug. 2004.

[5] Gabriel Falcao, Vitor Silva, Jose Marinho and Leonel Sousa,"LDPC Decoders for the WiMAX (IEEE 802.16e)based on Multicore Architectures", *WIMAX New Developments*, Upena D. Dalal and Y. P. Kosta (Ed.), ISBN: 978-953-7619-53-4, InTech, DOI: 10.5772/8265, chap. 6, 2009.

[6] IEEE P802.16e/D12 (2005), *Draft IEEE Standard for Local and Metropolitan Area Networks*, Part 16: *Air Interference for Fixed and Mobile Broadband Wireless Access Systems*, Oct. 2005.

[7] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399-431, March 1999.

[8] D. J. C. MacKay and R.M. Neal, "Near Shannon limit performance of low density parity check codes", *IEEE Electronics Letters*, vol. 33, no. 6, pp. 457-458, March 1997.

[9] R. M. Tanner, D. Sridhara and T. Fuja, "A class of group-structured LDPC codes", ICSTA, Ambleside, U.K., July 2001.

[10] R. M. Tanner, D. Sridhara, A. Sridharan, T. Fuja and D. J. Costello Jr., "LDPC block and convoluational codes based on circulant matrices", *IEEE Transactions on Information Theory*, vol. 50, no. 12, 2004.

[11] D. Sridhara, T. Fuja, and R. M. Tanner, "Low-density parity check codes from permutation matrices", in Proc. *Conf. Information Sciences and Systems*, Baltimore, MD, March 2001.

[12] Sheng Tong, Qinghua Guo, Jiangtao Xi and Yu Yanguang, "Effects of base matrices on iterative decoding performance of irregular QC-LDPC codes", *IEEE 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Dec. 2013.

[13] I. E. Bocharova, B. D. Kudryashov, R. Johannesson, "Combinatorial optimization for improving QC LDPC codes performance", *IEEE International Symposium on Information Theory Proceedings (ISIT)*, July 2013.

[14] E. Boutillon, J. Castura and F. R. Kschischang, "Decoder-First Code Design", *2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, Sep. 2000.

[15] E. Boutillon, J. Castura and F. R. Kschischang, "Decoder-First Code Design", Proc. *2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, Sep. 2000.

[16] Xiaoheng Chen, Shu Lin and V. Akella, "QSN-A Simple Circular-Shift Network for Reconfigurable Quasi-Cyclic LDPC Decoders", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 10, pp. 782-786, Oct. 2010.

[17] Yejun He and Jie Yang, "Construction of QC-LDPC codes with girth larger than eight based on GPU", *IEEE International converence on Wireless Communications and Signal Processing (WCSP)*, Oct. 2012.

[18] Sarah J. Johnson, *Introducing Low-Density Parity-Check Codes*, version 1.1, School of Electrical Engineering and Computer Science, The University of Newcastle, Australia.

Acronyms

LDPC *Low Density Parity Check*

MIT *Massachusetts Institute of Technology*

QC-LDPC *Quasi-Cyclic LDPC*

CPMs *Circulant Permutation Matrices*

WiMAX *Worldwide Interoperability for Microwave Access*

DVB-S2 *Digital Video Broadcasting by Satellite - Second Generation*

IEEE *Institute of Electrical and Electronics Engineers*

BER *Bit Error Rate*

SNR *Signal to Noise Ratio*

BSC *Binary Symmetric Channel*

GF(2) *Galois Field of two elements*

ML *Maximum likelihood*

w_r *row weight*

w_c *column weight*

ASIC *Application-Specific Integrated Circuit*

FEC *Forward Error Correcting*

SPA *Sum-Product Algorithm*

LLR *Log Likelihood Ratio*

VLSI *Very Large Scale Integration*

FER *Frame Error Rate*