Analyse et construction de codes LDPC non-binaires pour des canaux à évanouissement

Matteo Gorgolione

THÈSE

présentée

à l'Université de Cergy Pontoise
École Nationale Supérieure de l'Électronique de ses Applications

pour obtenir le grade de :

Docteur en Science de l'Université de Cergy Pontoise
Spécialité : Sciences et Technologies de l'Information et de la Communication

Par

Matteo GORGOGLIONE

Équipes d'accueil :
Équipe Traitement des Images et du Signal (ETIS) – CNRS UMR 8051
Laboratoire d'Étude et Specification des Systèmes de Communication (LESC) – CEA/LETI

Titre de la thèse

Analyse et construction de codes LDPC non-binaires pour des canaux à évanouissement

Soutenue le xx-xx-2012 devant la commission d'examen composée de :

Prof.	Emmanuel Boutillon	Université Bretagne Sud, Lorient	Examinateur
Dr.	Iryna Andriyanova	ENSEA, Université de Cergy-Pontoise	Examinateur
Prof.	Jean-Pierre Tillich	INRIA Rocquencourt, Le Chesnay	Rapporteur
Prof.	Joseph Boutros	Texas A&M University, Qatar	Rapporteur
Dr.	Valentin Savin	CEA-LETI, MINATEC, Grenoble	Encadrant
Prof.	David Declercq	ENSEA, Université de Cergy Pontoise	Directeur de thèse

Author's publications

Published papers

1. M. Gorgoglione, V. Savin, and D. Declercq, "Optimized puncturing distributions for irregular non-binary LDPC codes", *IEEE International Symposium on Information Theory and Applications* (ISITA), Taichung, Taiwan, October 2010.
2. M. Gorgoglione, V. Savin, and D. Declercq, "Binary Diversity for Non-Binary LDPC codes over the Rayleigh Channel", *IEEE Wireless Communications and Networking Conference* (WCNC), Paris, France, April 2012.
3. M. Gorgoglione, V. Savin, and D. Declercq, "Full diversity NB-LDPC coding with non-binary repetition symbols over the block fading channel", *IEEE International Symposium on Wireless Communication Systems* (ISWCS), Paris, France, August 2012.

Participation to research projects

– The author participated to the research project *"Analyse et optimisation des systèmes d'après 3ème génération"* (APOGEE), supported by the French Research and Innovation Program for Telecommunication.

Abstract

Over the last 15 years, spectacular advances in the analysis and design of graph-based codes and iterative decoding techniques paved the way for the development of error correction systems operating very close to the theoretical Shannon limit. A prominent role has been played by the class of Low Density Parity Check (LDPC) codes, introduced in the early 60's by Gallager's and described latter in terms of sparse bipartite graphs. In the early 2000's, LDPC codes were shown to be capacity approaching codes for a wide range of channel models, which motivated the increased interest of the scientific community and supported the rapid transfer of this technology to the industrial sector. Over the past few years there has been an increased interest in non-binary LDPC codes due to their enhanced correction capacity. Although Gallager already proposed in his seminal work the use of non-binary alphabets (by using modular arithmetic), non-binary LDPC codes defined over finite fields have only been investigated starting with the late 90's. They have been proven to provide better performance than their binary counterparts when the block-length is small to moderate, or when the symbols sent through channel are not binary, which is the case for high-order modulations or for multiple-antennas channels. However, the performance gain comes at a non-negligible cost in the decoding complexity, which may prohibit the use of non-binary LDPC codes in practical systems, especially when the price to pay in decoding complexity is too high for the performance gain that one can get.

This thesis addresses the analysis and design of non-binary LDPC codes for fading channels. The main goal is to demonstrate that besides the gain in the decoding performance, the use of non-binary LDPC codes can bring additional benefits that may offset the extra cost in decoding complexity. "Flexibility" and "diversity" are the two benefits that we demonstrate in this thesis. The "flexibility" is the capacity of a coding system to accommodate multiple coding rates through the use of a unique encoder/decoder pair. The "diversity" of a coding system relates to its capacity to fully exploit the communication channel's heterogeneity.

The first contribution of the thesis is the development of a Density Evolution approximation method, based on the Monte-Carlo simulation of an infinite code. We show that the proposed method provides accurate and precise estimates of non-binary ensemble thresholds, and makes possible the optimization of non-binary codes for a wide range of applications and channel models.

The second contribution of the thesis consists of the analysis and design of ***flexible coding schemes*** through the use of puncturing. We show that the non-binary LDPC codes are more robust to puncturing than their binary counterparts, thanks to the fact that non-binary symbol-nodes can be only partially punctured. For regular codes, we show that the design of puncturing patterns must respect different rules depending on whether the symbol-nodes are of degree 2 or higher. For irregular codes we propose an optimization procedure and we present optimized puncturing distributions for non-binary LDPC codes,

which exhibit a gap to capacity between 0.2 and 0.5dB , for punctured rates varying from 0.5 to 0.9.

The third contribution investigates the non-binary LDPC codes transmitted over a Rayleigh (fast) fading channel, in which different modulated symbols are affected by different fading factors. In case of one-to-one correspondence between modulated and coded symbols, deep fading can make some coded symbols totally unrecoverable, leading to a poor system performance. In order to avoid this phenomenon, ***binary diversity*** can be exploited by using a bit-interleaver module placed between the encoder and the modulator. We propose an optimized interleaving algorithm, inspired from the Progressive Edge-Growth (PEG) method, which ensures maximum girth of the global graph that extends the bipartite graph of the code with a new ensemble of nodes representing the modulated symbols. The optimized interleaver shows a gain with respect to the random interleaver, as far as performance and error detection rates are concerned.

Finally, the fourth contribution consists of a ***flexible coding scheme*** that achieves ***full-diversity*** over the block fading channel. The particularity of our approach is to rely on Root non-binary LDPC codes coupled with multiplicative non-binary codes, so that to easily adapt the coding rate to the number of fading blocks. A simple combining strategy is used at the receiver end before the iterative decoding. As a consequence, the decoding complexity is the same, irrespective of the number of fading blocks, while the proposed technique brings an effective coding gain.

Keywords Shannon capacity, Non-Binary LDPC codes, rate-adaptability, binary diversity, full-diversity coding.

Contents

List of Figures

List of Tables

Glossary

ARQ: Automatic Repeat reQuest
AWGN: Additive White Gaussian Noise

BEC: Binary Erasure Channel
BER: Bit Erasure Rate
BF: Block (Rayleigh) Fading (Channel)
BI-AWGN: Binary Input Additive White Gaussian Noise
BFT: Binary Fourier Transform
BlEC: Block Erasure Channel
BP: Belief-Propagation (decoding)
BSC: Binary Symmetric Channel

DE: Density Evolution

ECC: Error-Correcting codes

FER: Frame Error Rate
FFT: Fast Fourier Transform
FH: frequency hopping

GA-DE: Density Evolution via Gaussian Approximation

IRA: Irregular Repeat-Accumulate (code)

LDPC: Low-Density Parity-Check (code)

MAP: Maximum A Posteriori (decoding)
MC-DE: Density Evolution via Monte-Carlo (simulations)
ML: Maximum Likelihood (decoding)

NB-LDPC: Non-Binary Low Density Parity Check (code)

OFDM: Orthogonal Frequency Division Multiplexing

pdf: probability distribution function
PEG: Progressive Edge-Growth (algorithm)
PSK: Phase-Shift Keying

QAM: Quadrature Amplificatude Modulation

Chapter 1

Introduction

Today we live in a technological world dominated by the use of cell phones, full-connected computers with high-capacity storage supports. Furthermore, the 4G system (the fourth generation of high-speed wireless communication for mobile handset and data terminals) sets the fundations for new horizons of Internet and the possibility to be connected from everywhere. Thus nowadays the challenge is to provide fast, reliable and power efficient communication. The study of efficient methods for having high-speed communication and for protecting the data against errors, due to either the transmission media or the storage devices, is called *coding theory*.

This theory was introduced by Shannon in the middle of the nineteenth century and it is focused on two aspects:

- The *source coding*, which studies how to compress the data efficiently;
- The *channel coding*, which studies how to protect the messages sent over the channel in the presence of disruptive noise (errors).

Our efforts are principally concentrated on the telecommunication problems in which a *source* needs to transmit information to a *receiver* via wireless. The wireless channel is characterized to be disruptive, meaning that part of the information can be lost. Thus, we will focus on the channel coding that aims to find efficient methods for protecting the data transmission. In particular, this dissertation is concerned with the family of Low-Density Parity-Check (LDPC) codes, an important class of Error-Correcting codes (ECC).

This introduction is organized as follows. A general digital system model is presented in Section 1.1 in order to explain the work context. In Section 1.2 we describe briefly the channel coding. The motivations of this work are reported in Section 1.3. Section 1.4 illustrates our contributions together with the outline of the rest of the thesis.

1.1 Digital Communication Systems

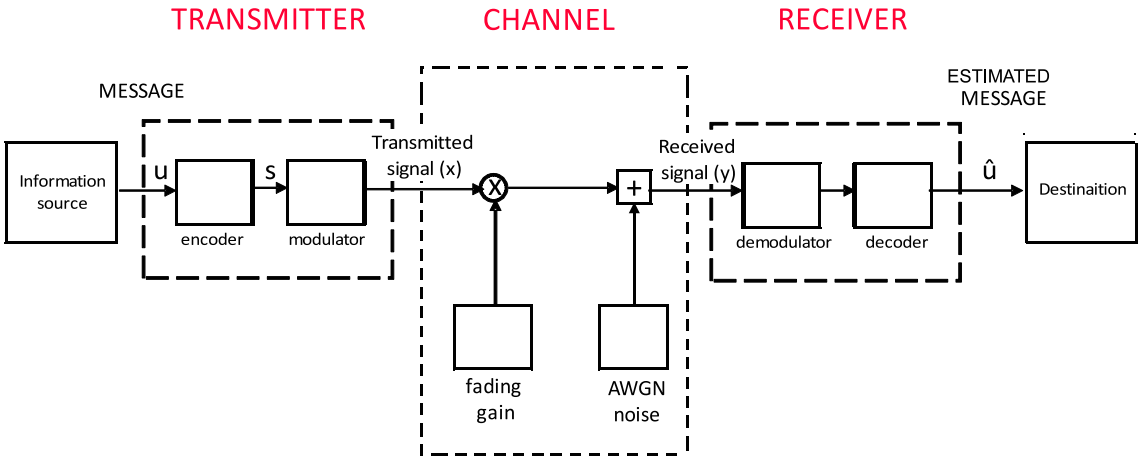

Figure 1.1: A simply communication model

Any communication system aims to transmit information messages from a source to a destination in a reliable manner. The nature of the information is not relevant from the coding point of view: it could be the exchanged data between two computers, a telephonic conversation between two people or even the reading and writing processes in storage supports like CDs, flash disk and so on.

A general model of a digital communication system is drawn in Figure 1.1. The information messages crosses several steps (represented by the different blocks) between their generation and the destination utilization. In the following we will describe briefly these steps. The communication model consists on three principal parts:

1. a transmitter,
2. a channel,
3. a receiver.

Transmitter. The transmitter is composed by a source [1] that gives a succession of binary or non-binary digits [2]. The following block (with dashed lines) works these messages out in order to be sent. The main requests for a transmitter are to protect and to modulate the data. Hence, this block is composed by two sub-blocks:

1. a channel encoder,
2. a modulator.

The channel encoder introduces, in a controlled manner, some redundancy in the messages that can be used at the receiver to overcome the noise effects encountered in the transmission of the signal through the channel. It receives in input the *information* message u and outputs an *encoded message* s, whose length is greater than that of u. The

1. In this dissertation we will treat only discrete sources: a time-continuous source can be quantized in order to make it discrete.

2. This sequence is generally the output of the *source encoding* that converts the source output in binary sequence and compress this sequence in order to eliminate the redundancy of the source.

goal of the channel coding is to minimize the redundancy part, yet allowing reliable transmission: it tries to maximize the coding rate, defined as the ratio between the number of information symbols and the number of encoded symbols. The rate is regulated by the *noisy channel coding* theorem introduced by Claude Shannon [66]: he stated that reliable transmission is possible if and only if the rate is upper bounded by the *channel capacity*. Obviously, the way in which the redundancy is generated is important and this is one of the core issues in *coding theory*.

The modulator transforms the coded messages in waveform capable to be physically transmitted in the channel.

Channel. The channel could be, for instance, the air for wireless transmissions, an optical cable for a wired communications, or a storage medium in case of memory supports.

The transmitted signal, which passes through the channel, can be corrupted by background noise, due to the thermal atmospheric noise, which is summed to the transmitted signal x. This is modeled by the Additive White Gaussian Noise (AWGN) channel.

Other disturbing phenomenons occur in the transmission so that the signal experiences attenuation, delay and phase shift. For example the signal can be undergone ionospheric reflection and refraction, or reflections caused by objects or the earth ground: this induces a multiple signal reception that disturbs the communication. The *fading channel* is the model used in wireless communications for a mathematical description of the channel. There exists several fading models but in this work we will principally treat the Rayleigh model [51]. More details about the channel models are reported in the next chapter.

Receiver. The receiver carries out the inverse operations seen in the transmitter. In this section the signal is demodulated and decoded in order to estimate the original messages. There are several kinds of decoding algorithms in the literature. In general the decoder receives information about the sent messages (beliefs, probabilities, ...) and it aims to find the sent messages of the transmitter. The hope at the destination is that the estimated message, despite the encountered noise and disturbs, is the same as the sent one.

1.2 Channel Codes

1.2.1 Coding in a Noisy Channel

A linear block code represents an important class of error-correcting codes. In such a code, the redundant symbols are generated as linear combinations of the information symbols.

Consider a binary code defined over $\mathbb{F}_2$, where $\mathbb{F}_2$ is the finite field with 2 elements. A linear binary code $\mathcal{C}(N, K)$ [3] is a K-dimensional linear subspace of the vector space $\mathbb{F}_2^N$. This subspace is generated by a basis $\{g_1, \ldots, g_K\} \subseteq \mathbb{F}_2^N$, whose elements are placed as rows of the matrix:

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{bmatrix}$$

3. $N \geq K$

If $u = [u_1, u_2, \ldots, u_K]$ is an information message, the corresponding encoded message $s = [s_1, s_2, \ldots, s_N]$, which is also referred to as *codeword*, is defined by:

$$s = uG$$

For this reason G is called *generator matrix* of the code $\mathcal{C}(N, K)$.

The *codebook* is composed of all the 2^K codewords, and it is in bijective correspondence with the set of information messages. This helps the decoder to decide the transmitted data according to the received signal.

A linear block code can equivalently be specified by using a *parity check matrix* H, whose rows generate the orthogonal complement of $\mathcal{C}(N; K)$. This means that for any codeword $s \in \mathcal{C}(N; K)$:

$$Hs^T = \underline{0} \rightarrow H(u\,G)^T = H\,G^T u^T = \underline{0},$$

where $\underline{0}$ is the all-zero vector. This happens iff $H\,G^T = \underline{0}$ and the matrix H is chosen of maximal rank.

Example 1.1. *Consider the following $\mathcal{C}(7;4)$ linear binary code (or the Hamming(7,4)-code [27]). This code is composed by an information part of $K = 4$ bits and a redundant part of $M = 3$ bits. Therefore, the coding rate is $r = 4/7$. Let $u = (u_0, u_1, u_2, u_3)$ be an information message and $s = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$ be the corresponding codedword of length $N = 7$ bits. It is defined by the following linear combinations:*

$$s = (u_0 + u_1 + u_3, u_0 + u_2 + u_3, u_1 + u_2 + u_3, u_0, u_1, u_2, u_3)$$

The redundant part is also called parity part and a parity bit is given by the sum modulo 2 (XOR) of the information bits according to the corresponding equation. This can be also reported as a matrix computation $s = u\,G$, where:

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

This example represents a systematic code*, meaning that the information bits are embedded into the transmitted coded word. For this reason G is referred to as* systematic generator matrix*. The generator matrix can take the form of $G = [P^T | I_{4\times4}]$.*
A parity check matrix of this code takes the form of $H = [I_{3\times3} | P]$:

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

and it is possible to verify that $H\,G^T = \underline{0}$.

Hamming weight and the minimum distance. The *Hamming weight* [27] $w(c)$ of a vector c is the number of non-zero entries (coordinates) of c.

The *Hamming distance* $d_H(c, c')$ between two vectors c and c' is the number of coordinates at which the two vectors are different. For binary vector, $d_H(c, c')$ is also equal to the Hamming weight of their sum modulo 2 (*i.e.,* the xor operation):

$$d_H(c, c') = w(c \oplus c')$$

The minimum (Hamming) distance $d_{min}(\mathcal{C})$, or simply d_{min}, of a code $\mathcal{C}$ is defined by:

$$d_{min} = \min_{\substack{c, c' \in \mathcal{C} \\ c \neq c'}} d_H(c, c')$$

For linear codes, we also have:

$$d_{min} = \min_{c \in \mathcal{C}} w(c)$$

An equivalent definition of the minimum distance d_{min} comes from the parity-check matrix H: d_{min} is the smallest number of the linear dependent columns of H. We observe that the code in the Example 1.1 has Hamming distance $d_{min} = 3$.

In the classical coding theory, the minimum distance is an important parameter for determining the error-detection and error-correction capabilities of a linear code, as far as the Maximum-Likelihood decoding is considered (see next section).

1.2.2 Channel Decoding

In this section we present the principles of the main decoding systems. Given a channel output y, the objective of the decoder is to estimate the information vector $\hat{u}$ that have been sent over the channel (after the encoding process). For simplicity, we will consider only the case of binary codes.

Maximum Likelihood (ML). This decoder minimizes the *word error probability*, and outputs the most likely information word $\hat{u}$ according to the received sequence y:

$$\hat{u} = \operatorname*{argmax}_{u \in \mathbb{F}_2^K} \Pr(\, u \,|\, y \,) = \operatorname*{argmax}_{u \in \mathbb{F}_2^K} \Pr(\, s(u) \,|\, y \,),$$

where $s(u)$ denotes the codeword obtained by encoding the information vector u. Equivalently, it is possible to choose first the codeword $\hat{s}$ that maximizes the a posteriori probability $\Pr(s|y)$:

$$\hat{s} = \operatorname*{argmax}_{s \in \mathcal{C}} \Pr(\, s \,|\, y \,),$$

and then take $\hat{u}$ to be the corresponding information word.

Maximum A Posteriori (MAP). This decoder minimizes the *bit error probability* and makes a separate decision for each position within the information word. Precisely, for each position $i = 1, \ldots, K$ the decoder outputs an estimation $\hat{u}_i \in \mathbb{F}_2$ of the i-th information bit that maximizes the a posteriori probability $\Pr(u_i|y)$:

$$\hat{u}_i = \operatorname*{argmax}_{u_i \in \mathbb{F}_2} \Pr(\, u_i \,|\, y \,) = \operatorname*{argmax}_{u_i \in \mathbb{F}_2} \sum_{\substack{u' \in \mathbb{F}_2^K \\ u'_i = u_i}} \Pr(\, u' \,|\, y \,) = \operatorname*{argmax}_{u_i \in \mathbb{F}_2} \sum_{\substack{u' \in \mathbb{F}_2^K \\ u'_i = u_i}} \Pr(\, s(u') \,|\, y \,),$$

where $s(u')$ denotes the codeword obtained by encoding the information vector u'.

Note that it is also possible to make a decision for each position within the transmitted codeword:

$$\hat{s}_i = \operatorname*{argmax}_{s_i \in \mathbb{F}_2} \Pr(\, s_i \,|\, y \,) = \operatorname*{argmax}_{s_i \in \mathbb{F}_2} \sum_{\substack{s' \in \mathcal{C} \\ s'_i = s_i}} \Pr(\, s' \,|\, y \,)$$

However, in this case the sequence $\hat{s} = (\hat{s}_1, \ldots, \hat{s}_N)$ needs not be a codeword, hence, it needs no have a corresponding information vector $\hat{u}$! Yet, if the code is systematic, $\hat{u}$ can be determined by taking the values at the appropriate positions in $\hat{s}$. We also note that if $\hat{s}$ is a codeword, it is necessarily the same as the one provided by the ML decoder.

In order to minimize the *bit* error probability, the decoder must choose a codeword $s \in \mathcal{C}$ that contains the bit s_i (at the i^{th} position) such that s must be the closest codeword to the channel output y. Put differently, the choice of the i^{th} bit must maximize the a posteriori probability $\Pr(s_i|y)$:

$$\hat{s}_i = \underset{s_i \in \mathcal{C}}{\operatorname{argmax}} \Pr(\, s_i \,|\, y \,) = \underset{\substack{s' \in \mathcal{C} \\ s'_i = s_i}}{\operatorname{argmax}} \Pr(\, s' \,|\, y \,)$$

where $\hat{s}_i$, with $i = 1, \ldots, N$, is the estimated bit. We have to observe that the estimated word $\hat{s} = (\hat{s}_1, \ldots, \hat{s}_N)$ is not necessarily a codeword.

The ML and MAP decoders are rather theoretic decoders, which are in general not implementable in practice, because the decoding complexity increases exponentially with K. There exist however some notable exceptions:

- Over erasure channels (*e.g.* the Binary Erasure Channel) the ML and MAP decoder are equivalent, and they consist of solving a linear system whose unknown variables correspond to the erased bits. This can be implemented by using Gaussian Elimination, whose complexity scales as K^3. However, this complexity is still prohibitive for large values of K.
- The Viterbi algorithm [20, 73] provides a practical implementation of the ML decoding for error correcting codes defined on trellises (principally convolutional codes) with relatively small constraint length value.
- The BCJR algorithm [2] provides a practical implementation of the MAP decoding for error correcting codes defined on trellises (principally convolutional codes) with relatively small constraint length value.

Although ML and MAP decoders are optimal with respect to *word* and *bit* error rates, respectively, their complexity prohibit the use of long codes. Yet, from the Shannon theory, we know that the channel capacity can be achieved only in the asymptotic limit of the code length. To deal with long codes, the idea is to perform MAP estimations locally, and to propagate these local estimations by means of some iterative "message-passing" procedure. This leads to the Belief-Propagation algorithm, which has demonstrated empirical success in numerous applications including coding theory (decoding of LDPC and turbo codes), free energy approximation in thermodynamics, and satisfiability in mathematical logic.

Belief Propagation (BP). The BP algorithm operates on a factor graph representing the parity check matrix of the code. This is a bipartite graph containing variable-nodes corresponding to coded bits (columns of the parity-check matrix) and factors (or check-nodes) corresponding to parity-check equations (rows of the parity-check matrix). Edges connect variables and the factors in which they appear, and correspond to the non-zero entries of the parity check matrix.

The BP algorithm works by passing real valued functions, called *messages*, along the edges of the graph, in both directions. Each message passed on an edge expresses the probability that the incident variable-node is equal to 0 or 1. These probabilities are

iteratively updated by using Bayesian inference. In this way, an a posteriori (marginal) probability is computed for each variable node of the graph or, equivalently, for each bit of the transmitted codeword. If the factor graph is cycle free, it can be shown that after some number of iterations, these probabilities are exactly the same as those computed within the MAP decoding. Interestingly, the same algorithm can be used for graphs with cycles. In this case, the BP decoding can be seen as an "approximate solution" for the MAP decoding. Yet, if the factor graph does not contain small cycles, it turns out that this approximation is good enough to be useful! In order to avoid small cycles, the parity-check matrix must be sparse, that is, it must contain only few non-zero entries. This is the main characteristic of the *Low-Density Parity-Check* (LDPC) codes, introduced by Gallager in the early '60s. LDPC codes can be decoded with BP and for well designed codes, the resulting performance is near the Shannon limit. More details about LDPC codes and BP decoding, as well as other low-complexity message-passing decoding algorithms, are given in Chapter 3.

1.3 Motivations

Shannon's theory. The coding theory starts with the seminal work of Claude E. Shannon on the mathematical theory of communication in 1948 [66]. He proved that over a noisy channel, the error rate can be reduced to any desired level as long as the information transfer rate is less than the capacity of the channel. For this reason, this theoretical maximum information rate is called Shannon limit. Shannon formulated his theorems but he did not indicated the way to construct optimum codes. Prior to 1990, the best constructions were serial concatenated codes based on an outer Reed-Solomon [57] (or BCH [6, 29]) error correction code combined with an inner Viterbi-decoded short constraint length convolutional code, also known as RSV codes. In 1993, the Turbo codes [4] were the first practical codes to closely approach the channel capacity, using an iterative Belief-Propagation decoder. The success of Turbo codes also brought to light a family of iteratively decodable codes invented 30 years earlier, namely the family of LDPC codes.

A brief history of the LDPC codes. Low Density Parity Check (LDPC) codes are a class of linear error correcting codes defined by sparse parity-check matrices. One of their most attractive features is that they can be decoded in linear time (with respect to their block-length), by using iterative message-passing algorithms. Nevertheless, such decoding algorithms were considered "impractical to implement" when LDPC codes were invented by Gallager in early 60's [22, 23]. This explains why they have been somehow "neglected" for more that three decades, and "rediscovered" in the late 90's by MacKay [46], after the power of iterative decoding techniques had been also confirmed by the discovery of Turbo-codes [4]. Yet, during that time, several important research papers dealt with LDPC codes, improving our knowledge on graph-based codes and iterative decoding techniques. It worth mentioning here the work of Tanner [70], who described LDPC codes in terms of sparse bipartite graphs and proposed a more general construction of graph-based linear codes (nowadays, these codes are often referred to as *Generalized LDPC* codes – GLDPC for short).

A large body of knowledge has been also acquired starting with the late 90's – early 2000's, especially techniques for the analysis of iterative decoding algorithms [12, 18, 43, 59], as well as techniques developed for code construction and optimization [21, 32, 39]. Gallager's efforts were finally recompensed ten years ago when Richardson et. all demon-

strated that LDPC codes are capacity approaching codes [11, 60]: for a wide range of channel models, practical constructions exist, which yield families of LDPC codes having a noise threshold [4] very close (or even arbitrarily close, for the BEC) to the channel capacity. This motivated the increased interest of the scientific community and supported the rapid transfer of this technology to the industrial sector. Nowadays, LDPC codes are included in many communication standards, as for instance:

- G.hn/G.9960 (ITU-T Standard for networking over power lines, phone lines and coaxial cable),
- 802.3an (10 Giga-bit/s Ethernet over Twisted pair),
- CMMB (China Multimedia Mobile Broadcasting),
- DVB-S2 / DVB-T2 / DVB-C2 (Digital video broadcasting, 2nd Generation),
- DMB-T/H (Digital video broadcasting),
- WiMAX (IEEE 802.16e standard for microwave communications),
- IEEE 802.11n-2009 (Wi-Fi standard).

Non-binary LDPC codes. Although Gallager already proposed in his seminal work the use of non-binary alphabets (by using modular arithmetic), non-binary LDPC codes defined over finite fields have only been investigated by Davey in his Ph.D dissertation starting with the late 90's [15].

The primary motivation for considering LDPC codes defined over non-binary alphabets it that they have better performance with respect to their binary counterparts: it can be demonstrated that LDPC codes can approach the channel capacity for shorter blocklengths when the codes are defined over high order alphabets. Another important fact is that when the LDPC codes are defined over an alphabet with order greater than or equal to the modulation order, the decoder is initialized with uncorrelated messages, which helps the decoder to be closer to the MAP decoding than in the binary case. Several works proved that NB-LDPC codes have excellent performance in several contexts. As a matter of fact, non-binary LDPC codes are now recognized as a potential competitor to binary coded solutions, especially when the codeword length is small to medium [32, 54], or when the order of the symbols sent through channel are not binary [3], which is the case for high-order modulations or for multiple-antennas channels [52].

However, the performance gain comes at a non-negligible cost in the decoding complexity, which may prohibit the use of non-binary LDPC codes in practical systems, especially when the price to pay in decoding complexity is too high for the performance gain that one can get. At the cost of a small performance degradation, several low-complexity decoding algorithms have been proposed in the literature, such as the Extended-Min-Sum decoding [16] or the Min-Max decoding [65].

The main goal of this thesis is to demonstrate that besides the gain in the decoding performance, the use of non-binary LDPC codes can bring additional benefits that may offset the extra cost in decoding complexity. "Flexibility" and "diversity" are the two benefits that we demonstrate in our work. The "flexibility" is the capacity of a coding system to accommodate multiple coding rates through the use of a unique encoder/decoder pair. The "diversity" of a coding system relates to its capacity to fully exploit the communication channel's heterogeneity.

4. The noise threshold is defined as the maximum channel noise up to which the probability of lost information can be made as small as desired, by using an arbitrarily-length code of the given family.

1.4 Thesis Outline & Contributions

This section presents the thesis outline. Chapters 2 and 3 briefly review the necessary background on information and coding theory, while Chapters 4–7 contain the main contributions of this thesis.

Chapter 2 - Channel Models and Information-Theoretic Aspects

In this chapter we trace the evolution of Shannon's theory of information and give the most important definitions, as the concepts of mutual information and channel capacity. Afterward, we compute the capacity for two of the most frequently encountered channels: the Additive White Gaussian Noise (AWGN) and the Rayleigh channels. We also propose approximate formulas that allow an accurate and fast approximation of the capacity of AWGN and Rayleigh channels with higher order complex modulations. Finally, we describe the block Rayleigh fading channel, which is a non-ergodic channel model with zero capacity. For this class of channels, we define and compute the *outage probability*: a lower bound that allows one to estimate the transmission quality when a long enough code is transmitted.

Chapter 3 - Low-Density Parity-Check Codes

This chapter gives a brief introduction to LDPC codes, with a particular emphasis on non-binary LDPC codes defined over Galois (finite) fields. Besides the Belief-Propagation decoding, we also present some lower complexity iterative message-passing algorithms, *e.g.* the Extended Min-Sum and the Min-Max decoding algorithms.

The asymptotic analysis of LDPC code ensembles, in terms of *density evolution* (DE), is also briefly reviewed in this chapter. We discuss the DE for both binary and non-binary LDPC codes: it worth mentioning here that for binary LDPC codes the exact DE can be derived for a large class of channel models, while for non-binary LDPC codes the exact DE is only known for the Binary Erasure Channel.

Part of this chapter is also dedicated to the design and optimization of finite-length LDPC codes. In particular we describe the Progressive Edge-Growth (PEG) algorithm and the optimization of the code components thanks to the properties of the binary image of the parity-check matrix. We note that a slightly different version of the PEG algorithm is proposed, which allows the construction of *doubly-irregular* LDPC codes, that is codes having irregular distributions on both variable and check-nodes. Finally, the last part of the chapter is concerned with encoding issues for LDPC codes: we discuss the encoding complexity and present some constructions that allow linear-time encoding.

Chapter 4 - Monte-Carlo Estimation of the Asymptotic Performance

This chapter is concerned with the asymptotic analysis of non-binary LDPC codes. This analysis is based on the Density Evolution (DE) method, aimed at recursively determining the probability density of messages passed throughout the iterative Belief Propagation decoding process. It allows deriving a *threshold* value, separating the region where reliable transmission is possible from that where it is not.

However, for non-binary LDPC codes, exact DE equations are only known when the transmission takes place over the Binary Erasure Channel (BEC). In this chapter, we propose a density evolution approximation method, by using Monte-Carlo simulation of an infinite code. The main idea is to observe a finite number of messages, while resampling

the graph interleaver (that is, the way that variable and check-nodes are connected) at each decoding iteration, according to the irregularity profile of the LDPC code [5]. We show that Density Evolution computed with Monte-Carlo simulations provides accurate (very close) and precise (small variance) estimates of non-binary LDPC thresholds. To this end, we compare the estimated thresholds with their exact values in case of binary LDPC codes over the AWGN channel and in case of non-binary LDPC codes over the BEC.

Finally, we also briefly present the *differential evolution* algorithm: a genetic optimization algorithm that optimizes a problem by iteratively trying to improve a *candidate solution* with regard to a given *measure of quality* [69]. Taking as "candidate solutions" different irregularity profiles of NB-LDPC codes, and as "measure of quality" the corresponding threshold value (estimated by the proposed Monte-Carlo-DE method), one can optimize NB-LDPC codes for a wide range of applications and channel models.

Chapter 5 - Punctured NB-LDPC codes over AWGN Channel

This chapter is concerned with rate-adaptability solutions for non-binary LDPC codes. Rate adaptation requires codes of different rates, which can be efficiently obtained by using one low rate mother code and puncture it to get higher rates. The advantage of puncturing is that the same decoder can be used regardless the puncturing pattern: according to the channel conditions, the transmission system adapts by just changing the puncturing pattern at the transmitter, and the depuncturing pattern at the receiver.

We show that the non-binary LDPC codes are more robust to puncturing than their binary counterparts, thanks to the fact that non-binary symbol-nodes can be only partially punctured. In particular, we design non-uniform bit-wise puncturing distributions for non-binary LDPC codes. Puncturing distribution are defined in terms of fractions $\{f_{d,k}\}_{d,k}$ of degree-d symbol-nodes with exactly k punctured bits per (non-binary) symbol.

For regular codes, we show that the design of puncturing patterns must respect different rules depending on whether the symbol-nodes are of degree 2 or higher. For irregular codes, we optimize puncturing distributions by minimizing the decoding threshold of the punctured LDPC code, the threshold being computed with a Monte-Carlo implementation of Density Evolution (Chapter 4). We present optimized puncturing distributions for non-binary LDPC codes with small maximum degree, which exhibit a gap between 0.2 and 0.5 dB to the channel capacity, for punctured rates varying from 0.5 to 0.9.

Chapter 6 - Binary Diversity over Rayleigh Channel

Mobile communications can present a multipath phenomenon so that the magnitudes of the received signals from the different paths have a Rayleigh distribution. This chapter investigates the non-binary LDPC codes transmitted over a Rayleigh fast fading channel, in which different modulated symbols are affected by different fading factors.

We propose the use of a binary interleaver module, placed between the encoder and the modulator, in order to mitigate the fading effects. The non-binary nature of the LDPC codes yields an effective coding gain: the bit-interleaver spreads the deep fading effects within different transmitted symbols and lowers the error probability toward the AWGN limit. Therefore, we analyse the performance of several bit-interleaving strategies applied to NB-LDPC codes over the Rayleigh fading channel.

5. Note that the edge labels (corresponding to the non-zero entries of the non-binary parity-check matrix) are also resampled at each iteration.

Moreover, we propose an optimized interleaving algorithm, inspired from the Progressive Edge-Growth (PEG) method, which ensures maximum girth of the global graph that extends the bipartite graph of the code with a new ensemble of nodes representing the modulated symbols. The optimized interleaver shows a gain with respect to the random interleaver, as far as performance and error detection rates are concerned.

Chapter 7 - Full-Diversity NB-LDPC Codes over Block-Fading Channel

The block-fading channel was introduced in order to model channels involving slow time-frequency hopping (used in wireless cellular networks), or multicarrier modulation using orthogonal frequency division multiplexing (OFDM). More generally, this simplified model proves to be very useful in code designs for slow-varying fading environments.

In the last chapter of this thesis we propose a flexible coding scheme that achieves full-diversity over the block-fading channel. The particularity of our approach is to rely on non-binary LDPC codes coupled with multiplicative non-binary codes, so that to easily adapt the coding rate to the number of fading blocks. A simple combining strategy is used at the receiver end before the iterative decoding. As a consequence, the decoding complexity is the same, irrespective of the number of fading blocks, while the proposed technique brings an effective coding gain. The performance of the proposed coding scheme over the Rayleigh block fading channel is evaluated by Monte-Carlo simulation, and we show that it performs close to the outage probability, for a large range of coding rates.

Chapter 2

Channel Models and Information-Theoretic Aspects

This chapter is dedicated to the Shannon's theory, which is the basis of the modern communication systems.

Shannon developed a mathematical model based on probability theory and statistics for quantifying the information that passes through a communication channel. The information is measured either in terms of *entropy*, the information in a random variable, or of *mutual information*, the amount of information shared by two random variables. This mathematical model is known as the *information theory*.

Though the information theory concerns many other sub-fields as for example the source coding [1] or the information-theoretic security [2] aspects, we focus our attention on the coding theory, that studies how to have reliable transmissions over noisy channels. In order to set up an error-free transmission, the transmitter needs to add redundancy to the information messages, so that the receiver can be helped to decode the original messages. The aim of this theory is to determine the minimum quantity of redundancy to use in order to guarantee the reliability of the transmission in noisy channels.

Shannon proved that for any communication channel, it is possible to communicate in a reliable manner, accepting an arbitrary small error probability, if and only if the information rate is below a maximum rate, called *channel capacity*. Put differently, the capacity of a channel is the maximum information rate expressed in units of information per channel use, which can be achieved with arbitrary small error probability.

The chapter is organized as follows. In Section 2.1 we define the notion of information. In Section 2.2 we introduce some ordinary channel models and the concept of channel capacity. In the following Sections 2.3 and 2.4 we compute the capacity for two channel models: the Additive White Gaussian Noise channel and the Rayleigh channel. Finally, in the Section 2.5 we compute the outage probability for block-fading channels.

1. How to compress data without losing information.
2. The cryptosystems.

2.1 Notations and Basic Definitions

The following notation is used throughout this dissertation:

1) X is a random variable. $\mathbb{X} = \{x_1, \dots x_n\}$ is the set of values that X can take.
2) Y is a random variable. $\mathbb{Y} = \{y_1, \dots y_m\}$ is the set of values that Y can take.
3) $n = |\mathbb{X}|$ and $m = |\mathbb{Y}|$ represent the cardinality of $\mathbb{X}$ and $\mathbb{Y}$, respectively.

We define the following probability functions:

4) p_X is the probability distribution of X given by $p_X(x) = \Pr(X = x)$, $\forall x \in \mathbb{X}$.
5) p_Y is the probability distribution of Y given by $p_Y(y) = \Pr(Y = y)$, $\forall y \in \mathbb{Y}$.
6) $p_{X,Y}$ is the joint probability distribution of $X \times Y$ given by $p_{X,Y}(x, y) = \Pr(X = x, Y = y)$, $\forall (x, y) \in \mathbb{X} \times \mathbb{Y}$.
7) $p_{X|Y}$ is the conditioned probability distribution of X knowing Y given by:

$$p_{X\,|\,Y}(x\,|\,y) = \Pr(X = x\,|\,Y = y) = \frac{\Pr(X = x\,,\, Y = y)}{\Pr(Y = y)}$$

When no confusion is possible, the above probability distributions will be simply denoted by $p(x)$, $p(y)$, $p(x, y)$, and $p(x \mid y)$.

Information. The *information* associated with an event $x \in \mathbb{X}$ of probability $p(x)$ is the non-negative quantity:

$$I(x) = \log_2 \frac{1}{p(x)} > 0 \tag{2.1}$$

This function, intuitively, means that the information given by an event increases as the probability of the event is small.

Entropy. The information of the random variable X, called commonly *entropy*, is the average information over all the events $x \in \mathbb{X}$:

$$H(X) = E[I(x)] = \sum_{x \in \mathbb{X}} p(x) \log_2 \frac{1}{p(x)} \tag{2.2}$$

The entropy $H(x) \in [0, \log_2(n)]$ measures the uncertainty of the random variable X.

Conditional Entropy. The entropy of X conditional on a particular event $Y = y$ is defined by:

$$H(X \mid y) = \sum_{x \in \mathbb{X}} p(x \mid y) \log_2 \frac{1}{p(x \mid y)} \tag{2.3}$$

The definition of the entropy of X conditional on Y is obtained by averaging over all the events $y \in \mathbb{Y}$. By using the Bayes' rule we obtain:

$$\begin{aligned} H(X \mid Y) &= \sum_{y \in \mathbb{Y}} p(y) \sum_{x \in \mathbb{X}} p(x \mid y) \log_2 \frac{1}{p(x \mid y)} \\ &= \sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} p(x, y) \log_2 \frac{p(y)}{p(x, y)} \end{aligned} \tag{2.4}$$

Mutual Information. The mutual information of X and Y is defined by:

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= \sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} p(x, y) \log_2 \frac{p(x, y)}{p(x) p(y)} \end{aligned}$$

Useful Properties.

(P1) $0 \leq H(X) \leq \log_2(n)$

(P1.a) $H(X) = 0 \Leftrightarrow \exists x_i \in \mathbb{X}$ such that $p(x_i) = 1$ and $p(x_j) = 0\ \forall x_j \neq x_i$

(P1.b) $H(X) = \log_2(n) \Leftrightarrow p(x_1) = \cdots = p_(x_n) = 1/n$ all the symbols are equally likely.

(P2) $H(X,Y) \leq H(X) + H(Y)$

(P2.a) if X and Y are independent, $H(X,Y) = H(X) + H(Y)$

(P3) $H(X \mid Y) \leq H(X)$

(P4) $0 \leq I(X;Y) \leq H(X)$

(P5)
$$\begin{aligned} I(X;Y) &= I(Y;X) \\ &= H(X) - H(X \mid Y) \\ &= H(Y) - H(Y \mid X) \\ &= H(X) + H(Y) - H(X,Y) \\ &= H(X,Y) - H(X \mid Y) - H(Y \mid X) \end{aligned}$$

This digression can be analysed graphically thanks to the Venn diagram of Figure 2.1. The mutual information is given by the intersection of the two ensembles.

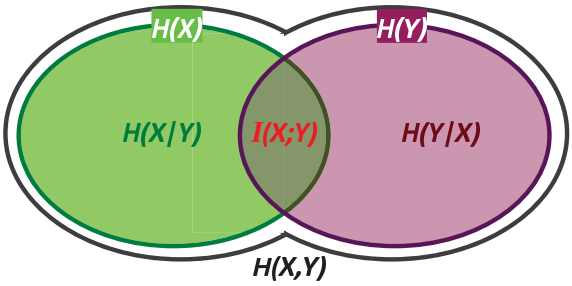

Figure 2.1: $H(X)$, $H(Y)$, joint $H(X,Y)$, conditional $H(X \mid Y)$ and $H(X \mid Y)$, and mutual information $I(X;Y)$

The Meaning.

(M1) **Logarithmic functions.** By definition, the information is non-negative (2.1) and the information carried by independent random variables is the sum of the respective information amounts (P2.a); this justifies the use of logarithmic functions as information-theoretic measures.

(M2) **Logarithmic base.** The logarithmic base corresponds to the choice of a unit for measuring the information. Generally, the base 2 will be used, in which case the resulting units are expressed in terms of *bits*. However, the use of a different base might be more convenient in some cases; if so, the use of a logarithmic base different from 2 will be clearly indicated in the text. In case that the natural basis is used, the information will be measured in *nats*. The above measures can be defined using any logarithmic base $b \neq 2$. In this case, they will be denoted by $H_b(X)$, $H_b(X \mid Y)$, $I_b(X;Y)$. The above properties remain valid for any logarithmic base $b \neq 2$.

(M3) **Entropy.** $H(X)$ is usually interpreted as a measure of the uncertainty associated with the random variable X. If $H(X) = 0$ there is no uncertainty (P1.a), while the case of maximum entropy $H(X) = \log_2(n)$ corresponds to the maximum uncertainty (all the events are equiprobable – see (P1.b)).

(M4) **Conditioned entropy.** $H(X \mid Y)$ does not increase the entropy of X because Y introduces a condition that increases the certainty of X (P3).

(M5) **Typical sequences.** The entropy can also be used to determine the *probability of a typical long sequence*. Precisely, consider a long sequence of T symbols obtained by T successive, independent realizations of the random variable X. By the law of large numbers, for $T \gg 0$, this sequence will contain with high probability about $p(x_1)T$ occurrences of x_1, $p(x_2)T$ occurrences of x_2, etc. Such a sequence is called *typical*. The typical sequences form a subset of $\mathbb{X}^T$, the set of all the possible sequences of length T, whose total probability goes to one as T goes to infinity (this is still a consequence of the law of large numbers). Typical sequences are equally likely and their probabilities, denoted by p, is given by:

$$p = p(x_1)^{p(x_1)T} p(x_2)^{p(x_2)T} \cdots p(x_n)^{p(x_n)T}$$

Hence:

$$\log_2 p = T \sum_{x \in \mathbb{X}} p(x) \log_2 p(x) = -H(X)T \Rightarrow p = 2^{-H(X)T} = n^{-\frac{H(X)}{\log_2 n} T}$$

Put differently, among the n^T possible sequences in $\mathbb{X}^T$, there are $n^{\frac{H(X)}{\log_2 n} T}$ typical sequences, each one of which has probability $p = n^{-\frac{H(X)}{\log_2 n} T}$. Moreover, when T goes to infinity, the output of T successive, independent realizations of X is, with probability going to 1, a typical sequence of length T.

Remark 2.1. *Recall that* $n = |x|$. *Then* $H_n(X) = \dfrac{H(X)}{\log_2 n} = -\sum_{x \in \mathbb{X}} p(x) \log_n p(x)$ *is the entropy on* X *in base* n *and verifies* $0 \leq H_n(X) \leq 1$.

(M6) **Mutual Information.** Intuitively, mutual information measures the information that X and Y share: it measures how much knowing one of these variables reduces the uncertainty about the other. For example, if X and Y are independent, then knowing X does not give any information about Y and vice versa, so $I(X;Y) = 0$. At the other extreme, if $X = Y$ then knowing X determines the value of Y and vice versa. As a result, we have $I(X;Y) = H(X)$, which is the uncertainty contained in X alone.
Mutual information quantifies the dependence between the joint distribution of X and Y and what the joint distribution would be if X and Y were independent. Mutual information is a measure of dependence in the following sense: $I(X;Y) = 0$ if and only if X and Y are independent random variables.

(M7) **Continuous Random Variables.** Same definitions remain valid for continuous random variables, by replacing summation $\sum_{x \in \mathbb{X}} [\cdots]$ with integration $\int_{\mathbb{X}} [\cdots] \, \mathrm{d}x$.

2.2 Capacity of Noisy Channels

Using the notation and the definitions from the previous section, we can now characterize the capacity of *noisy channels*. Throughout this dissertation we limit ourselves to the case of memoryless channels, for which the channel output depends only on the current channel input. A noisy channel consists of:

- an *input alphabet* $\mathbb{X}$,
- an *output alphabet* $\mathbb{Y}$,
- a set of *transaction probabilities* $p(y \mid x)$

The value $p(y \mid x)$ represents the probability of the channel output being $y \in \mathbb{Y}$, given that the input symbol is $x \in \mathbb{X}$. Hence, the following must hold:

$$\sum_{y \in \mathbb{Y}} p(y \mid x) = 1, \quad \forall x \in \mathbb{X}$$

Now, let X be a random variable (a source) with values in $\mathbb{X}$ and probability distribution p_X. By fitting X to the channel input, the channel output defines a random variable Y on $\mathbb{Y}$, whose probability distribution p_Y is given by:

$$p(y) = \sum_{x \in \mathbb{X}} p(x) p(y \mid x)$$

2.2.1 Channel Models

1) **Binary Input Additive Gaussian Noise Channel.**
The BI-AWGN channel consists of:
- a binary input alphabet $\mathbb{X} = \{-1, +1\}$,
- an output alphabet $\mathbb{Y} = \mathbb{R}$.

The output Y is the sum of the binary input X and of the noise Z:

$$Y = X + Z,$$

where the noise $Z \sim \mathcal{N}_{\mathbb{R}}(0, \sigma^2)$ is distributed as a real-valued Gaussian random variable with zero mean and variance σ^2:

$$p_Z(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^2}{2\sigma^2}}$$

2) **Additive Gaussian Noise Channel with complex signaling set $\mathbb{X}$.**
This channel model consists of:
- an input alphabet, which is a finite subset $\mathbb{X} \subseteq \mathbb{C}$, such that $\sum_{x \in \mathbb{X}} |x|^2 = 1$,
- an output alphabet $\mathbb{Y} = \mathbb{C}$.

This model of channel is similar to the previous one except that the input alphabet consists of the set of complex constellation points of some fixed modulation. Note that, we always assume that $\mathbb{X}$ is normalized, meaning that $\sum_{x \in \mathbb{X}} |x|^2 = 1$. The output Y is related to the input X by the formula:

$$Y = X + Z,$$

where the noise $Z \sim \mathcal{N}_{\mathbb{C}}(0, \sigma^2)$ is a complex-valued Gaussian distributed random variable with mean 0 and variance σ^2. We denote by

$$p_Z(z) = \frac{1}{\pi\sigma^2} e^{-\frac{|z|^2}{\sigma^2}}$$

the probability density function of Z, where $|z|^2 = \Re(z)^2 + \Im(z)^2$.
$\Re(Z)$ and $\Im(Z)$ are real-valued, Gaussian distributed, independent random variables with mean 0 and variance $\frac{\sigma^2}{2} = \left(\frac{\sigma}{\sqrt{2}}\right)^2$.

3) **Binary Symmetric Channel (BSC).**

The BSC model consists of:
– a binary input alphabet $\mathbb{X} = \{-1, +1\}$,
– a binary output alphabet $\mathbb{Y} = \{-1, +1\}$.
The transaction probabilities $\Pr(Y = -1 \mid X = +1) = \Pr(Y = +1 \mid X = -1) = \epsilon$ (see Figure 2.2).

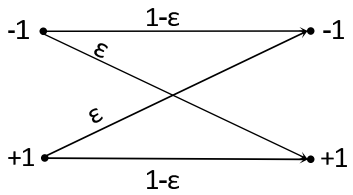

Figure 2.2: Binary Symmetric Channel with transaction probability ϵ

Remark 2.2. *The BSC can be seen as an BI-AWGN channel with binary input and hard decision mechanism that filters the channel output according to its sign. In this case, the hard decision mechanism transforms the BI-AWGN channel with noise variance σ^2 in a BSC channel with error probability $Q(1/\sigma^2)$, where Q denotes the Q-function defined by $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} e^{-\frac{u^2}{2}} du$.*

4) **Binary Erasure Channel (BEC).**

The BEC model consists of:
– a binary input alphabet $\mathbb{X} = \{-1, +1\}$,
– an output alphabet $\mathbb{Y} = \{-1, +1, E\}$,
where E represents an erasure.

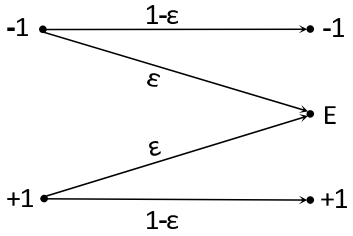

Figure 2.3: Binary Erasure Channel with transaction probability ϵ

This model is presented in Figure 2.3 and it is characterized by the transaction probabilities $\Pr(y = E \mid x = -1) = \Pr(y = E \mid x = +1) = \epsilon$ and $\Pr(y = -1 \mid x = -1) = \Pr(y = +1 \mid x = +1) = 1 - \epsilon$.

This model ensures that the channel output is certainly correct in case that $y \in \{-1, +1\}$, while it does not provide any information about the input if $y = E$, in which case the input has the same possibility of being -1 or $+1$.

5) **Rayleigh Fading Channel with complex signaling set $\mathbb{X}$.**

This channel model consists of:
– an input alphabet, which is a finite subset $\mathbb{X} \subseteq \mathbb{C}$, such that $\sum_{x \in \mathbb{X}} |x|^2 = 1$,
– An output alphabet $\mathbb{Y} = \mathbb{C}$.

The input alphabet $\mathbb{X} \subseteq \mathbb{C}$ consists of the set of complex constellation points of some fixed modulation. We also assume that $\mathbb{X}$ is normalized. Input X and output Y are related by the formula:

$$Y = GX + Z$$

where $G \sim \mathcal{N}_{\mathbb{C}}(0,1)$ is a circular complex normal distributed random variable with mean 0 and variance 1, and $Z \sim \mathcal{N}_{\mathbb{C}}(0,\sigma^2)$ is a circular complex normal distributed random variable with mean 0 and variance σ^2. We denote by

$$p_Z(z) = \frac{1}{\pi\sigma^2} e^{-\frac{|z|^2}{\sigma^2}} \tag{2.5}$$

the probability density functions of Z and

$$p_G(g) = \frac{1}{\pi} e^{-|g|^2} \tag{2.6}$$

the probability density functions of G.

$\Re(G)$ and $\Im(G)$ are real, Gaussian distributed, independent random variables with mean 0 and variance $\frac{1}{2}$. In particular, the envelop of the channel response

$$|G| = \sqrt{\Re^2(G) + \Im^2(G)}$$

is distributed according to the Rayleigh distribution with parameter $\frac{1}{\sqrt{2}}$.

6) **Block Rayleigh Fading Channel.**

The alphabets' definition of this channel model is the same as of the previous one:
– an input alphabet, which is a finite subset $\mathbb{X} \subseteq \mathbb{C}$, such that $\sum_{x \in \mathbb{X}} |x|^2 = 1$,
– An output alphabet $\mathbb{Y} = \mathbb{C}$.

Again, the input alphabet $\mathbb{X} \subseteq \mathbb{C}$ consists of the set of complex constellation points of some fixed modulation. In the Rayleigh channel model the multiplicative coefficients change for each symbol: in the block Rayleigh channel model, the fading is constant for a given symbol period.

Precisely, the block-fading channel is composed by n_f blocks, each block consists of L complex symbols $x \in \mathbb{X}$. Each block is distinguished by a fading gain coefficient $g_j \in \{g_1, \ldots, g_{n_f}\}$. We assume that the fading gain coefficients are identically and independently Rayleigh distributed from one block to another according to (2.6).

2.2.2 Channel Capacity

The channel capacity, denoted by C, is defined as:

$$C = \max_{p_X} I(X;Y)$$

where maximization is taken over all possible probability distributions X on $\mathbb{X}$.

Remind that $I(X;Y) = H(X) - H(X|Y)$. $H(X \mid Y)$ may be interpreted as the lost part of the information $H(X)$ due to the channel noise. Indeed, in case of noiseless channels, $H(X \mid Y) = 0$ meaning that the received symbol unambiguously determines the transmitted one. On the contrary, for channels with a very high level of noise the received symbols tend to be independent from the transmitted symbols, then $H(X \mid Y) \simeq H(X)$ (the two ensembles, H(X) and H(Y), are almost disjoint).

Generally, the input symbols of the channel are considered to be equally likely. Therefore, we define:

$$C_u = I(X_u, Y),$$

where X_u is a random variable uniformly distributed over $\mathbb{X}$; that is, $p_{X_u}(x) = 1/n$, $\forall x \in \mathbb{X}$, where $n = |\mathbb{X}|$ is the number of symbols of the channel input alphabet $\mathbb{X}$. It will be referred as the *uniform-input capacity*. C_u can be expressed as *bits per channel use* or *symbols per channel use*, according to the logarithmic base used for the computation of the mutual information:

- If the base-2 logarithm is used for the mutual information, then the capacity is expressed in terms of *bits per channel use*. Since $I(X;Y) = H(X) - H(X \mid Y) \leq H(X) \leq \log_2 n$, where n is the number of symbols of the input alphabet $\mathbb{X}$, it follows that $C \leq \log_2 n$.
- If the base-n logarithm is used, then the capacity is expressed in terms of *symbols per channel use*. In this case $C = \max_{p_X} I_n(X;Y) \leq 1$.

Note also that for most channels $C = C_u$; meaning that the maximum, in the above definition of the capacity, is obtained for the uniform distribution on the channel input alphabet $\mathbb{X}$.

When expressed as *symbols per channel use*, the uniform-input capacity can be written as:

$$C_u = I_n(X_u, Y) = H_n(X_u) - H_n(X_u \mid Y) = 1 - H_n(X_u \mid Y)$$

Channel codes. In order to reliable communicate over a noisy channel, the idea is to add some redundancy to the information data. This redundant data is used at the receiver in order to decode the information data. The quantity of redundancy is measured by the *coding rate* $r \in [0, 1]$, defined as the ratio between the number of information symbols and the number of coded symbols.

Noisy-channel coding theorem [weak version]. Consider a channel whose uniform-input capacity, expressed as *symbols per channel use*, is equal to $C_u \in [0, 1]$. Moreover, assume that a code with coding rate $r \in [0, 1]$ is used to transmit over the channel.

(1) If $r \leq C_u$ there exists a coding system such that the transmission has an arbitrary small frequency of errors, assuming that the code length can be made arbitrarily large.

(2) If $r > C_u$, there is no method of encoding which gives an arbitrarily small frequency of errors.

Proof [sketch of]. Let T denote the number of coded symbols and assume that $T \gg 0$. A coded message of length T will have $r\,T$ information symbols. Hence, the total number of coded messages of length T is given by n^{rT}.

On the other hand, an output sequence of length T could be produced, with high probability, by about $n^{H_n(X_u|Y)T}$ input sequences.

Now, we can transmit with arbitrary small error probability iff for any output sequence, only one coded message is among the $n^{H_n(X_u|Y)T}$ input sequences that could produced the observed output. Therefore the number of input sequences of length T must be at least $n^{rT} n^{H_n(X|Y)T}$. We obtain:

$$n^T \geq n^{rT} n^{H_n(X_u|Y)T} \Leftrightarrow 1 \geq r + H_n(X_u \mid Y) \Leftrightarrow C_u \geq r$$

□

Remark 2.3. *In case that the channel capacity is expressed in terms of bits per channel use, we have* $C_u \in [0, \log_2 n]$. *In this case the channel capacity is the maximum* information rate *that allows reliable communication, where the information rate* $R = r \log_2 n$ *is the number of information bits per channel use.*

2.3 Capacity of Gaussian Channels

In this section we compute the capacity of the BI-AWGN and AWGN channels introduced in the previous section. We also derive approximated formulas for these capacities.

Throughout this dissertation when we need to specify the complex signal set for an exact formulation we use the QAM constellation and x will denote a modulated symbol. Each complex symbol consist of $log_2 n$ bits. This notation includes also the case of BPSK modulation that could be eventually referred to as 2-QAM.

2.3.1 Capacity of the BI-AWGN Channel

It can be proved that the capacity C is equal to the uniform-input capacity C_u [66]. If we consider an uniform channel input X, that is $p_X(-1) = p_X(+1) = 1/2$, then the probability density function of Y is given by:

$$\begin{aligned} p_Y(y) &= \frac{1}{2\sqrt{2\pi\sigma^2}} \left(e^{-\frac{(y-1)^2}{2\sigma^2}} + e^{-\frac{(y+1)^2}{2\sigma^2}} \right) \\ &= \frac{1}{2\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}} \left(1 + e^{-\frac{2y}{\sigma^2}} \right) \\ &= \frac{1}{2\sqrt{2\pi\sigma^2}} e^{-\frac{(y+1)^2}{2\sigma^2}} \left(1 + e^{\frac{2y}{\sigma^2}} \right) \end{aligned}$$

We have:

$$\begin{aligned} C = C_u &= H(X_u) - H(X_u \mid Y) \\ &= 1 - H(X_u \mid Y) \end{aligned}$$

Conditional entropy $H(X_u \mid Y)$.

$$\begin{aligned} H(X_u \mid Y) &= \int_{\mathbb{R}} p(y) H(X_u \mid y) \, \mathrm{d}y \\ &= \int_{\mathbb{R}} p(y) \left(\sum_{x=\pm 1} p(x \mid y) \log_2 \frac{1}{p(x \mid y)} \right) \, \mathrm{d}y \\ &= \int_{\mathbb{R}} p(y) \left(\frac{\log_2 \left(1 + e^{-\frac{2y}{\sigma^2}} \right)}{1 + e^{-\frac{2y}{\sigma^2}}} + \frac{\log_2 \left(1 + e^{\frac{2y}{\sigma^2}} \right)}{1 + e^{\frac{2y}{\sigma^2}}} \right) \, \mathrm{d}y \\ &= \frac{1}{2\sqrt{2\pi\sigma^2}} \left[\int_{\mathbb{R}} e^{-\frac{(y-1)^2}{2\sigma^2}} \log_2 \left(1 + e^{-\frac{2y}{\sigma^2}} \right) \, \mathrm{d}y + \int_{\mathbb{R}} e^{-\frac{(y+1)^2}{2\sigma^2}} \log_2 \left(1 + e^{\frac{2y}{\sigma^2}} \right) \, \mathrm{d}y \right] \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{\mathbb{R}} e^{-\frac{(y-1)^2}{2\sigma^2}} \log_2 \left(1 + e^{-\frac{2y}{\sigma^2}} \right) \, \mathrm{d}y \end{aligned}$$

We get:

$$\boxed{C_{\text{BI-AWGN}} = 1 - \frac{1}{\sqrt{2\pi\sigma^2}} \int_{\mathbb{R}} e^{-\frac{(y-1)^2}{2\sigma^2}} \log_2 \left(1 + e^{-\frac{2y}{\sigma^2}} \right) \, \mathrm{d}y \quad \text{(bits per channel use)}}$$

2.3.2 Capacity of the AWGN Channel with Complex Signaling Set $\mathbb{X}$

In this section we compute the uniform-input capacity C_u, expressed as *symbols per channel use* of the AWGN channel with a complex signaling set $\mathbb{X}$.

If we consider an uniform channel input X, that is $p_X(x) = 1/n$, $\forall x \in \mathbb{X}$ (where $n = |\mathbb{X}|$), then the probability density function of Y is given by:

$$p_Y(y) = \frac{1}{n\pi\sigma^2} \sum_{x \in \mathbb{X}} e^{-\frac{|y-x|^2}{\sigma^2}}$$

We have:

$$\begin{aligned} C_u &= H_n(X_u) - H_n(X_u \mid Y) \\ &= 1 - H_n(X_u \mid Y) \end{aligned}$$

Conditional entropy $H_n(X_u \mid Y)$.

$$\begin{aligned} H_n(X_u \mid Y) &= \int_{\mathbb{C}} p(y) H_n(X_u \mid y) \, \mathrm{d}y \\ &= \int_{\mathbb{C}} p(y) \left(\sum_{x \in \mathbb{X}} p(x \mid y) \log_n \frac{1}{p(x \mid y)} \right) \mathrm{d}y \\ &= \frac{1}{n\pi\sigma^2} \int_{\mathbb{C}} \sum_{x \in \mathbb{X}} e^{-\frac{|y-x|^2}{\sigma^2}} \log_n \frac{s(y)}{e^{-\frac{|y-x|^2}{\sigma^2}}} \, \mathrm{d}y, \text{ where } s(y) = \sum_{x' \in \mathbb{X}} e^{-\frac{|y-x'|^2}{\sigma^2}} \\ &= \frac{1}{n\pi\sigma^2} \sum_{x \in \mathbb{X}} \int_{\mathbb{C}} e^{-\frac{|y-x|^2}{\sigma^2}} \log_n \left(\sum_{x' \in \mathbb{X}} e^{\frac{|y-x|^2 - |y-x'|^2}{\sigma^2}} \right) \mathrm{d}y \\ &= \frac{1}{n\pi\sigma^2} \sum_{x \in \mathbb{X}} \int_{\mathbb{C}} e^{-\frac{|y|^2}{\sigma^2}} \log_n \left(\sum_{x' \in \mathbb{X}} e^{\frac{|y|^2 - |y+x-x'|^2}{\sigma^2}} \right) \mathrm{d}y \\ &= \frac{1}{n\pi} \sum_{x \in \mathbb{X}} \int_{\mathbb{C}} e^{-|y|^2} \log_n \left(\sum_{x' \in \mathbb{X}} e^{|y|^2 - \left|y + \frac{x-x'}{\sigma}\right|^2} \right) \mathrm{d}y \\ &= \frac{1}{n\pi} \sum_{x \in \mathbb{X}} \left[\int_{\mathbb{C}} e^{-|y|^2} |y|^2 \log_n(e) \, \mathrm{d}y + \int_{\mathbb{C}} e^{-|y|^2} \log_n \left(\sum_{x' \in \mathbb{X}} e^{-\left|y + \frac{x-x'}{\sigma}\right|^2} \right) \mathrm{d}y \right] \\ &= \frac{1}{n\pi} \sum_{x \in \mathbb{X}} \left[\log_n(e)\pi + \int_{\mathbb{C}} e^{-|y|^2} \log_n \left(\sum_{x' \in \mathbb{X}} e^{-\left|y + \frac{x-x'}{\sigma}\right|^2} \right) \mathrm{d}y \right] \\ &= \log_n(e) + \frac{1}{n\pi} \sum_{x \in \mathbb{X}} \int_{\mathbb{C}} e^{-|y|^2} \log_n \left(\sum_{x' \in \mathbb{X}} e^{-\left|y + \frac{x-x'}{\sigma}\right|^2} \right) \mathrm{d}y \end{aligned}$$

We get:

$$C_{\mathrm{AWGN},\mathbb{X}} = 1 - \log_n(e) - \frac{1}{n\pi} \int_{\mathbb{C}} e^{-|y|^2} \left(\sum_{x \in \mathbb{X}} \log_n \left[\sum_{x' \in \mathbb{X}} e^{-\left|y + \frac{x-x'}{\sigma}\right|^2} \right] \right) \mathrm{d}y$$

(symbols per channel use)

Remark 2.4. *To express the capacity* $C_{AWGN,\mathbb{X}}$ *in terms of* bits per channel use, *the above value must be multiplied by* $\log_2 n$.

The (uniform-input) capacities of the AWGN channel for various QAM modulations are shown in Figure 2.4. In this figure, the complex signaling set $\mathbb{X}$ is the set of constellation points of BPSK, QPSK, 8-PSK and 16-QAM modulations. The abscissa represents the Signal to Noise Ratio (SNR) in dB. When the size of the QAM constellation goes to infinity, the capacity curves approach a *limiting curve* (brown, dotted curve), which corresponds to the capacity of the AWGN channel without any modulation constraint, that is $\mathbb{X} = \mathbb{C}$. It can be shown that this (unconstraint) capacity can be computed by the following formula:

$$C_{\text{AWGN}} = \log_2\left(1 + \frac{1}{\sigma^2}\right) \quad \text{(bits per channel use)}$$

From another point of view, this limit can be seen as a signal transmitted with a constellation of infinite points.

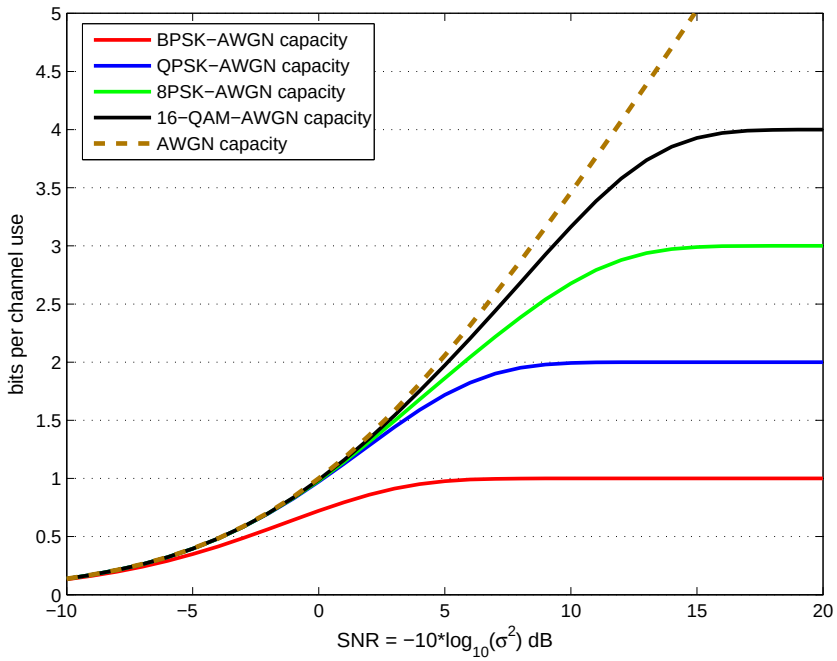

Figure 2.4: Capacity (*bits per channel use*) of the AWGN channel for various QAM modulations

2.3.3 Capacity of AWGN Channel with Limited Bandwidth.

More generally, consider a communication (rather than information theoretic) channel with bandwidth B, which corrupts the transmitted signal with additive white Gaussian noise (AWGN). Indeed, in real case, the received signal is limited within a band B such that the unlimited noise spectrum can be filter and restricted in a limited bandwidth. The Shannon-Hartley theorem states that the channel capacity is given by [28]:

$$C = B\log_2\left(1 + \frac{S}{N}\right) = B\log_2\left(1 + \frac{S}{N_0 B}\right)$$

where

- C is the channel capacity, in *bits per second*
- B is the bandwidth of the channel in Hertz
- S is the average signal power
- N is the total noise power
- $\frac{N_0}{2}$ is the noise power spectral density, hence $N = N_0 B$

Remark: the ratio $\frac{S}{N}$ is the linear SNR value (not in decibels)

We know that the capacity in *bits per channel use* is given by

$$\log_2\left(1+\frac{1}{\sigma^2}\right) = \log_2\left(1+\frac{S}{N}\right)$$

The capacity in *bits per second* is obtained by multiplying the above value by the bandwidth; that is $C = B\log_2\left(1+\frac{S}{N}\right)$.

Ultimate Shannon limit. Now, suppose we are sending binary digits at a transmission rate equal to the channel capacity, that is C bits per second. Since the average signal power is S and the bit duration is $1/C$ seconds, it follows that the average energy per bit is $E_b = S/C$. Replacing in the above formula, we get:

$$C = B\log_2\left(1+\frac{E_b C}{N_0 B}\right) \Leftrightarrow \frac{E_b}{N_0} = \frac{2^{\frac{C}{B}}-1}{\frac{C}{B}}$$

Since the function $f(x) = \dfrac{2^x-1}{x}$ is increasing and $\lim\limits_{x\to 0} f(x) = \ln 2$ it follows that:

$$\frac{E_b}{N_0} \geq \ln 2 = 0.6931, \text{ linear value, or equivalently } \frac{E_b}{N_0} \geq -1.5917 \text{ dB, in decibels.}$$

Hence, below this value there is no error-free communication at any information rate. This is called the *ultimate Shannon limit*.

2.3.4 Approximate Formulas for the Capacity of the AWGN Channel with $2^{\mathtt{m}}$-QAM Modulation

For $2^{\mathtt{m}}$-QAM constellation, the $C_{\text{AWGN},2^{\mathtt{m}}\text{-QAM}}$ formulas from the above section involve a quite complicated integral over the complex plane, which can be very expensive in terms of computation time. The goal of this section is to give quick-to-compute approximate formulas for the capacity of the AWGN channel with $2^{\mathtt{m}}$-QAM modulation, for $\mathtt{m} = 1, 2, 3, 4, 6$.

We use the following formula to approximate the capacity:

$$\tilde{C}_{\text{AWGN},2^{\mathtt{m}}\text{-QAM}} = \mathtt{m}\nu \frac{\log(1+s)}{\log\left(2^{\mathtt{m}\nu} + \gamma_1 s^{\nu_1} + \gamma_2 s^{\nu_2}\right)}, \quad \text{(bits per channel use)}$$

$$\text{where } s = \frac{1}{\sigma^2} \text{ and } \nu = \max(\nu_1, \nu_2)$$

parameters $\gamma_1, \gamma_2, \nu_1, \nu_2$ are shown in Table 2.1.

Parameters $\gamma_1, \gamma_2, \nu_1, \nu_2$ were determined, for each $\mathtt{m}$ value, by curve fitting. For a given $\mathtt{m}$ value, we searched for parameters $\gamma_1, \gamma_2, \nu_1, \nu_2$ that minimize the Root Mean Square (RMS) error between the exact capacity $C_{\text{AWGN},2^{\mathtt{m}}\text{-QAM}}$ and its approximate value $\tilde{C}_{\text{AWGN},2^{\mathtt{m}}\text{-QAM}}$. The minimization has been performed by using the Differential Evolution algorithm [69].

The parameters $\gamma_1, \gamma_2, \nu_1, \nu_2$ are given in Table 2.1; it is also shown the value of the corresponding RMS. Exact vs. approximate capacity curves are shown in Figure 2.5.

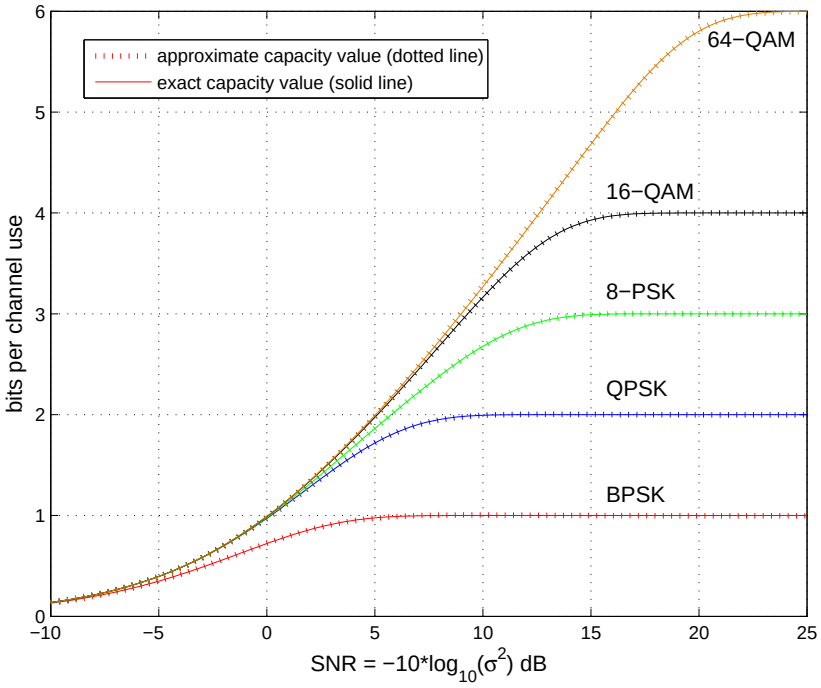

Figure 2.5: Approximate and exact capacity values, in *bits per channel use*, for the AWGN channel with various QAM modulations

Table 2.1: Parameters $\gamma_1, \gamma_2, \nu_1, \nu_2$, according to the value of $\mathtt{m}$

$\mathtt{m} = 1$	$\gamma_1 = 1.6734$ $\nu_1 = 0.7697$	$\gamma_2 = 1.0372$ $\nu_2 = 1.9756$	RMS $= 0.0023$
$\mathtt{m} = 2$	$\gamma_1 = 1.0770$ $\nu_1 = 1.1873$	$\gamma_2 = 1.0169$ $\nu_2 = 2.1925$	RMS $= 0.0023$
$\mathtt{m} = 3$	$\gamma_1 = 32.9172 \times 10^2$ $\nu_1 = 1.7436$	$\gamma_2 = 1.0410$ $\nu_2 = 4.6064$	RMS $= 0.0013$
$\mathtt{m} = 4$	$\gamma_1 = 74.5002 \times 10^2$ $\nu_1 = 0.9178$	$\gamma_2 = 1.0221$ $\nu_2 = 3.8186$	RMS $= 0.0030$
$\mathtt{m} = 6$	$\gamma_1 = 1.0147$ $\nu_1 = 3.9641$	$\gamma_2 = 702.0165 \times 10^4$ $\nu_2 = 0.4636$	RMS $= 0.0099$

2.4 Capacity of Rayleigh Channels

Our purpose is to compute the uniform-input capacity of the Rayleigh channel. Hence, we consider an uniform channel input X, that is $p_X(x) = 1/n,\ \forall x \in \mathbb{X}$, (where $n = |\mathbb{X}|$) and the probability density function of Y is given by:

$$p_Y(y) = \frac{1}{n\pi^2\sigma^2} \int_{\mathbb{C}} e^{-|g|^2} \sum_{x\in\mathbb{X}} e^{-\frac{|y-gx|^2}{\sigma^2}} \,\mathrm{d}g$$

There are two possible conditions for computing this capacity: 1) the availability of perfect Channel State Information (CSI) to the receiver or, on the contrary, 2) the receiver is not aware of channel realizations.

2.4.1 Perfect Knowledge of the Channel at the Receiver

In this section we assume that the receiver has perfect knowledge of the channel. As the channel changes at each transmitted symbol, this is a not realistic assumption but just an information-theoretic one and the general case will be study in the next section.

The perfect knowledge of the channel means that the receiver knows both the values of Y and G; hence, the channel capacity is given by:

$$C_u = H(Y,G) - H(Y,G \mid X_u)$$

We have:

$$\begin{aligned} C_u &= H(Y,G) - H(Y,G \mid X_u) \\ &= \Big(H(G) + H(Y \mid G)\Big) - \Big(H(G \mid X_u) + H(Y \mid G, X_u)\Big) \\ &= H(Y \mid G) - H(Y \mid G, X_u) \qquad \text{[knowing that } H(G) = H(G \mid X_u)\text{, as } G \text{ and } X_u \text{ are independent]} \\ &= \int_{\mathbb{C}} p_G(g) \Big(H(Y \mid G = g) - H(Y \mid X_u, G = g)\Big) \,\mathrm{d}g \end{aligned}$$

Now, for a given g, let $Y' = \frac{Y}{g}$ and $Z' = \frac{Z}{g}$. We obtain $Y' = X + Z'$, where $Z \sim \mathcal{N}_{\mathbb{C}}\left(0, \frac{\sigma^2}{|g|^2}\right)$. Hence, $H(Y \mid G = g) - H(Y \mid X_u, G = g)$ is just the capacity of the AWGN channel, with noise variance $\frac{\sigma^2}{|g|^2}$. Therefore, for a fixed $2^{\mathtt{m}}$-QAM modulation, the capacity is given by:

$$\boxed{\begin{aligned} C_{\text{RAYLEIGH, KNOWN-}G,\mathbb{X}}(\sigma^2) &= \frac{1}{\pi} \int_{\mathbb{C}} e^{-|g|^2} C_{\text{AWGN},2^{\mathtt{m}}\text{-QAM}}\left(\frac{\sigma^2}{|g|^2}\right) \mathrm{d}g \\ &= 2\int_0^{+\infty} r e^{-r^2} C_{\text{AWGN},2^{\mathtt{m}}\text{-QAM}}\left(\frac{\sigma^2}{r^2}\right) \mathrm{d}r \end{aligned}}$$

On the other hand, if we do not consider any constraint with respect to the modulation

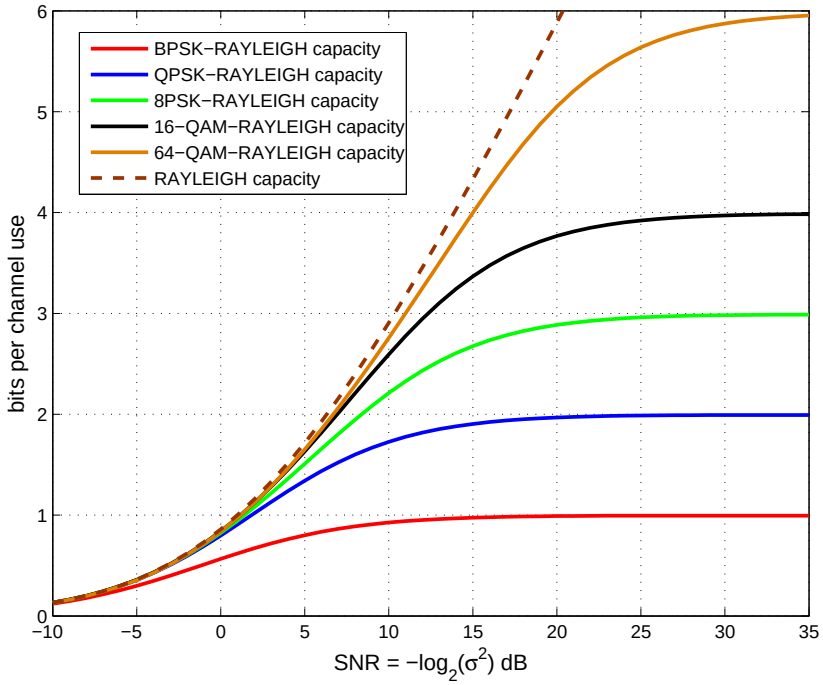

Figure 2.6: Capacity of the Rayleigh channel, in *bits per channel use*, for various QAM modulations. Perfect knowledge of the channel is assumed at the receiver.

(that is, we let the constellation size go to infinity), we obtain the following capacity:

$$\begin{aligned} C_{\text{RAYLEIGH, KNOWN-}G}(\sigma^2) &= \frac{1}{\pi}\int_{\mathbb{C}} e^{-|g|^2} C_{\text{AWGN}}\left(\frac{\sigma^2}{|g|^2}\right) \mathrm{d}g \\ &= \frac{1}{\pi}\int_{\mathbb{C}} e^{-|g|^2} \log_2\left(1+\frac{|g|^2}{\sigma^2}\right) \mathrm{d}g \\ &= 2\int_0^{+\infty} r e^{-r^2} \log_2\left(1+\frac{r^2}{\sigma^2}\right) \mathrm{d}r \end{aligned}$$

The capacity of the Rayleigh channel, assuming perfect knowledge of the channel at the receiver, is plotted at Figure 2.6, where the complex signaling set $\mathbb{X}$ is the set of constellation points of BPSK, QPSK, 8PSK, 16-QAM and 64-QAM.

2.4.2 No Knowledge of the Channel at the Receiver

In case that the receiver has no knowledge of the channel, the capacity is given by:

$$\begin{aligned} C_u &= H(Y) - H(Y \mid X_u) \\ &= \Big(H(Y,G) - H(G \mid Y)\Big) - \Big(H(Y,G \mid X_u) - H(G \mid Y, X_u)\Big) \\ &= \Big(H(Y,G) - H(Y,G \mid X_u)\Big) - \Big(H(G \mid Y) - H(G \mid Y, X_u)\Big) \end{aligned}$$

The first term of the above subtraction is just the capacity when the receiver has perfect knowledge of the channel. The second is the capacity of finding G given Y. We can write:

$$C_{\text{RAYLEIGH, UNKNOWN-}G}(\sigma^2) = C_{\text{RAYLEIGH, KNOWN-}G}(\sigma^2) - \Big(H(G \mid Y) - H(G \mid Y, X_u)\Big)$$

Remark 2.5. *The Rayleigh channel is an example of* fast-fading channel*, where the codeword length spans over a great number of coherence periods of the channel. Hence, one can average over independent channel fades by coding over a large number of coherence time intervals. As a consequence, we have seen that it is possible to achieve a reliable rate of communication of*

$$E\left[\log_2\left(1+\frac{|g|^2}{\sigma^2}\right)\right] = \frac{1}{\pi}\int_{\mathbb{C}} e^{-|g|^2}\log_2\left(1+\frac{|g|^2}{\sigma^2}\right)\,dg$$

which gives the capacity of the fast-fading Rayleigh channel.

2.5 Outage Probability for Block-Fading Channels

In this section we deal with *slow-fading channels*, where the coherence time is greater than the latency requirement. We will consider a model with $n_f > 1$ fading blocks, where each block is composed by L complex symbols. In particular, the fading is flat and constant on each block and independent and identically Rayleigh distributed on different blocks [5, 51] (see also Section 2.2.1).

We will also assume that the receiver has perfect knowledge of the channel. The channel parameters can be estimated by transmitting training-sequences at the beginning of each transmission. This assumption is strong since the fading gains cannot be estimated due to the unexpected variations of channel; nevertheless it can be justified when the fading varies slowly with time as in this channel case.

Let us first look to the case when the transmitted symbols span one single coherence period of the channel (that is $n_f = 1$), then the maximum rate of reliable communication supported by the channel is $\log_2\left(1+\frac{|g|^2}{\sigma^2}\right)$, and it depends on the random channel gain $|g|^2$. If the transmitter encodes data at rate R bits per channel use, there is a probability that the decoding error probability cannot be made arbitrarily small, given by:

$$P_{\text{out}} = \Pr\left[\log_2\left(1+\frac{|g|^2}{\sigma^2}\right) < R\right],$$

in which case the system is said to be in *outage*. With a non-zero probability that the channel is in deep fade, the capacity of the slow-fading channel in strict sense is zero. However, it is possible to determine the largest value of R such that the *outage probability* P_{out} is less than ε. This value is known as the *ε-outage capacity*.

More in general, for $n_f > 1$ and a given channel gain vector $\mathbf{g} = \left[g_1, \ldots, g_{n_f}\right]$, the maximum rate of reliable communication supported by the channel is:

$$I(\mathbf{g}, \sigma^2) = \frac{1}{n_f}\sum_{i=1}^{n_f}\log_2\left(1+\frac{|g_i|^2}{\sigma^2}\right)$$

If $g_1, \ldots, g_{n_f}$ are the channel gains experienced by the transmitted codeword, the above value is also referred to as *instantaneous mutual information* between the input and the output of the BF channel (or *instantaneous capacity*). If the transmission rate is greater

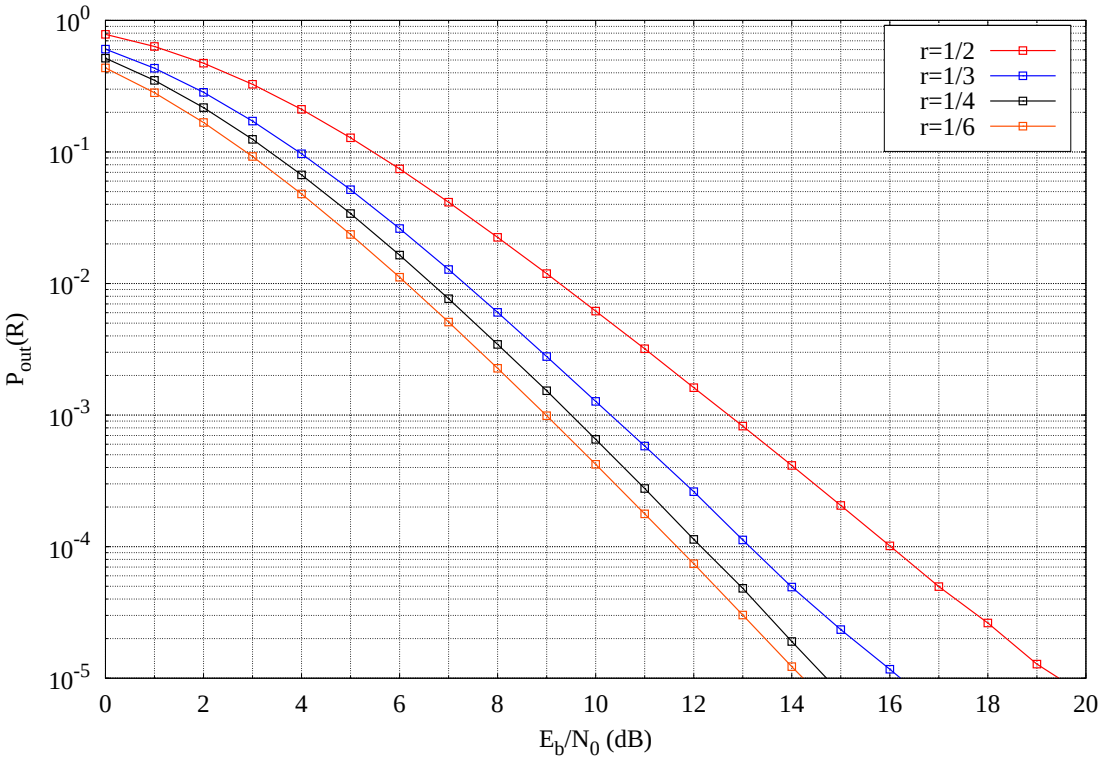

Figure 2.7: Outage probability for $n_f = 4$ and QPSK modulation

than the instantaneous capacity, the system is said to be in outage. The outage probability is defined as:

$$\boxed{P_{\text{out}}(R, \sigma^2) = \Pr\left[\frac{1}{n_f}\sum_{i=1}^{n_f} \log_2\left(1 + \frac{|g_i|^2}{\sigma^2}\right) < R\right]}$$

For a fixed complex signaling set $\mathbb{X}$, the outage probability can be computed by:

$$\boxed{P_{\text{out},\mathbb{X}}(R, \sigma^2) = \Pr\left[\frac{1}{n_f}\sum_{i=1}^{n_f} C_{\text{AWGN},\mathbb{X}}\left(\frac{\sigma^2}{|g|^2}\right) < R\right]}$$

The ε-outage capacity is defined as:

$$\boxed{C_{\varepsilon\text{-out}}(\sigma^2) = \sup\{R \mid P_{\text{out}}(R, \sigma^2) < \varepsilon\}}$$

In Figure 2.7 we have drawn the outage probability as a function of the noise variance (expressed in $\frac{E_b}{N_0}$) for different coding rates r, for a block-fading channel with $n_f = 4$ and QPSK modulation[3]. We add progressively redundancy, passing from a rate $r = 1/2$ symbol per channel use to a rate $r = 1/6$ symbols per channel use. We can observe an increasing *diversity gain* of these curves, corresponding to a slope more and more steeper when r

3. In order to obtain the corresponding rate R in term of bits per channel use we need to multiple r times 2.

decreases from 1/2 to 1/6. Also, we observe that when $r = 1/6$ the outage probability curve remains parallel to the curve $r = 1/4$. More in general, all the curves corresponding to coding rates $r > 1/n_f$ will stay parallel to the curve corresponding to $r = 1/n_f$. This concept is relative to the *full-diversity* coding for block-fading channel and we will give more details in Chapter 7.

Chapter 3

Low-Density Parity-Check Codes

Low-Density Parity-Check codes are a class of linear correcting-codes with near-capacity performance on a large class of transmission and storage channels. They have been introduced in 1960 by Gallagher in his PhD dissertation [23] but ignored because of technological limit at that time for implementing iterative decoders. With the works of Tanner [70], who introduced a graphical representation of the codes, and of MacKay [46] in 1996, these codes started to attract much more attention.

In the modern approach of error correcting codes, LDPC codes are playing a very important role due to their low decoding complexity. The decoding is based on an iterative exchange of messages until a codeword is found or a maximum number of iterations is reached. LDPC codes shows asymptotic performance approaching the theoretical Shannon limit [60]. The asymptotic analysis relies on tracking the evolution of the probability density of exchanged messages and allows one to determine the worse channel condition for which reliable transmission is possible (assuming that the code-length goes to infinity).

With the evolution of the technology, new families of LDPC codes defined over non-binary alphabets have been introduced. These codes exhibit better performance for short to medium code lengths [13], without sacrificing the nearly optimal asymptotic performance. The performance gain at short length comes at an additional cost in the decoding complexity. However, this additional cost can be reduced by using low-complexity algorithms, as proposed e.g. in [16, 65].

In this chapter we introduce the keynotes relative to the design of good LDPC codes, the encoding/decoding algorithms and the analysis of code performance. The efficiency of LDPC codes is evaluated in terms of error rates and the complexity of the encoding and decoding systems.

The chapter is organized as follows. LDPC codes are defined in Section 3.1, then several iterative message-passing algorithms are presented in Section 3.2. The asymptotic analysis and code optimization are discussed in Section 3.3. In Section 3.4 we describe the construction and the finite-length optimization of NB-LDPC codes. Finally, the Section 3.5 discusses the principal encoding systems.

3.1 LDPC Codes

In this dissertation we focus on non-binary LDPC codes. Gallager [23] already introduced LDPC codes defined over non-binary alphabets and he proposed a probabilistic iterative decoding. The first generalization of the decoding algorithm for LDPC codes defined over Galois fields has been presented in [13].

Many alphabet definitions have been explored since the first introduction of the NB-LDPC codes by Gallager. In [68] codes over rings and groups have been designed for coded modulation. In [24] Goupil *et al.* have introduced a wide class of LDPC codes, including definition over finite fields, rings, groups and non-linear codes; moreover a low-complexity encoding/decoding system has been developed. LDPC codes built over finite fields show some limitations in the choice of the non-binary coefficients of the parity-check matrix: codes defined over the General Linear groups demonstrate a slightly improved performance with respect to codes built over finite fields [10].

In the sequel we first define NB-LDPC codes over more general alphabets, then, we focus on LDPC codes defined over finite fields $\mathbb{F}_q$, where q is a power of 2.

3.1.1 Definition

First definition. We consider a non-binary alphabet $\mathcal{A}$ with $q = 2^p$ elements. For practical reasons, we consider $\mathcal{A}$ endowed with a vector space structure over $\mathbb{F}_2 = \{0, 1\}$, and we fix once for all an isomorphism:

$$\mathcal{A} \xrightarrow{\sim} \mathbb{F}_2^p \tag{3.1}$$

Elements of $\mathcal{A}$ are referred to as *symbols*. For any symbol $s \in \mathcal{A}$, its image $(s_1, \ldots, s_p) \in \mathbb{F}_2^p$ under the above isomorphism is called the *binary image* of s.

A NB-LDPC code is defined as the kernel of a sparse matrix $H \in \mathbf{M}_{M\times N}(\mathrm{End}(\mathcal{A}))$, where $\mathrm{End}(\mathcal{A})$ denotes the algebra of endomorphisms of $\mathcal{A}$. The matrix H is referred to as the *parity-check matrix*, and the sparsity condition means that only few entries of H are non-zero. Thus, a codeword $S = \{s_1, \ldots, s_N\}$ is the solution of the linear system $H \cdot S^T = \mathbf{0}$:

$$\sum_{n=1}^{N} h_{m,n}(s_n) = 0, \, \forall m = 1, \ldots, M$$

Using the above isomorphism $\mathcal{A} \tilde{\to} \mathbb{F}_2^p$, we can also identify $\mathrm{End}(\mathcal{A}) \backsimeq \mathbf{M}_{p\times p}(\mathbb{F}_2)$, the algebra of $p \times p$ matrices with coefficients in $\mathbb{F}_2 = \{0, 1\}$.

In this dissertation we mainly focus on a particular case, namely the case of NB-LDPC codes defined over *Galois fields*. Let $\mathbb{F}_q$ denotes the (finite) Galois field with $q = 2^p$ elements. Then $\mathbb{F}_q$ is endowed with a vector space structure over $\mathbb{F}_2$, and we fix an isomorphism:

$$\mathbb{F}_q \xrightarrow{\sim} \mathbb{F}_2^p \tag{3.2}$$

Moreover, by using the internal field multiplication, each symbol $h \in \mathbb{F}_q$ defines an endomorphism of $\mathbb{F}_q$, which maps $s \mapsto h \cdot s$, $s \in \mathbb{F}_q$. Hence, we obtain:

$$\mathbb{F}_q \hookrightarrow \mathrm{End}(\mathbb{F}_q) \simeq \mathrm{End}(\mathbb{F}_2^p) = \mathbf{M}_{p\times p}(\mathbb{F}_2) \tag{3.3}$$

Therefore, a non-binary LDPC code is said to be *defined over the Galois Field* $\mathbb{F}_q$ if each entry of H corresponds to an element of $\mathbb{F}_q$. A binary LDPC code is a particular case, for which the alphabet is simply the binary field $\mathbb{F}_2 = \{0, 1\}$.

The Galois field $\mathbb{F}_q$ can be defined as the quotient field of the ring of polynomials $\mathbb{F}_2[\alpha]$ by a primitive polynomial $P(\alpha)$ of degree p . Therefore, the field elements can be denoted by $\{0, \alpha^0, \alpha^1, \ldots, \alpha^{(q-2)}\}$, since in this case α is a primitive element of the Galois field.

Alternative definition. A NB-LDPC code is defined as the kernel of a sparse parity-check matrix $H = (h_{m,n})_{m,n} \in \mathbf{M}_{M,N}(\mathbb{F}_q)$, Thus, a codeword $S = (s_1, \ldots, s_n, \ldots, s_N)$ is the solution of the linear system $H \cdot S^T = \mathbf{0}$:

$$\sum_{n=1}^{N} h_{m,n}\, s_n = 0 \ , \forall m = 1, \ldots, M \tag{3.4}$$

where $h_{m,n}$ is the entry corresponding to the m^{th} row and n^{th} column of H. The matrix H is referred to as the *parity-check matrix*. In order to respect the low density condition for these codes, the associated parity check matrices must contain few entries.

Binary representation. Using the isomorphism $\mathbb{F}_q \tilde{\to} \mathbb{F}_2^p$ (3.2), each symbol $s_n \in \mathbb{F}_q$ corresponds to a binary vector $(s_{n,1}, \ldots, s_{n,p}) \in \mathbb{F}_2^p$ referred to as its binary image. As a consequence each NB-codeword $S = \{s_n\}_{n=1,\ldots,N}$ with $s_n \in \mathbb{F}_q$ can be transformed into a binary codeword $S_b = \{s_{n,i}\}_{n=1,\ldots,N;i=1,\ldots,p}$, with $s_{n,i} \in \mathbb{F}_2$.

Moreover, by using Eq. (3.3) each entry h of H correspond to a $p \times p$ binary matrix, such that the multiplication of a symbol s by h corresponds to the multiplication of a binary image of s by the matrix associated with h. The binary parity-check matrix H_b is obtained by replacing each entry $h_{m,n}$ of H by the associated binary $p \times p$ matrix. Hence, we have:

$$H \cdot S^T = 0 \Longleftrightarrow H_b \cdot S_b^T = 0$$

Coding rate. Let $\mathcal{C}$ be the non-binary code defined by the parity-check matrix H. The non-binary dimension of $\mathcal{C}$ is given by $K = N - \mathrm{rank}(H)$, and the coding rate is defined as:

$$r = \frac{K}{N} = 1 - \frac{\mathrm{rank(H)}}{N}$$

The designed coding rate of $\mathcal{C}$ is by definition

$$r' = 1 - \frac{M}{N}$$

Note that when H is full rank, which is generally the observed case, then $r = r'$.

3.1.2 Tanner Graph

In his work [70], Tanner studied LDPC codes and their representation using the associated bipartite graph (from now on it will be simply referred as *Tanner graph*).

The Tanner graph is associated with the parity-check matrix H. It will be denoted by $\mathcal{H}$ and it is composed by:

- N symbol-nodes, corresponding to the N columns of H,
- M constraint-nodes, corresponding to the M rows of H.

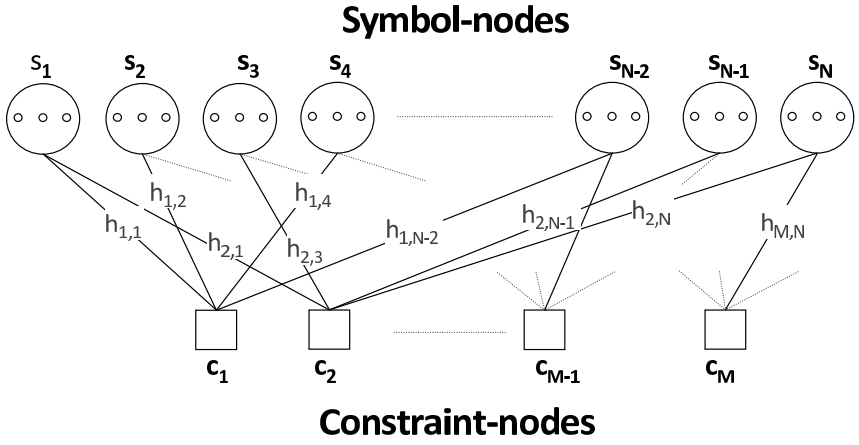

Figure 3.1: Tanner Graph of a NB-LDPC codes

A symbol-node n and a constraint-node m are connected by an edge iff the corresponding entry $h_{m,n}$ of H is different from 0. In this case, we further assume that the edge is labeled by the $h_{m,n}$ value.

Equivalently, the N symbol-nodes corresponds to the N coded symbols $\{s_n\}_{n=1,\dots,N}$, while the M constraint-nodes, together with the incident labeled edges, correspond to the linear equation between these nodes. A representation of the Tanner graph is drawn in Figure 3.1. This Tanner graph represents a LDPC codes defined over $\mathbb{F}_8$ (with $p = 3$ bits). In the binary case, the symbol-nodes are also referred to as bit-nodes, while the constraint-nodes are also referred to as *check*-nodes (in the following check and constraint are indistinctly used).

Two nodes are said to be *neighbours* if they are connected by an edge. The degree of a node is defined as the number of incoming edges.

Example 3.1. *Figure 3.2 is an example of Tanner graph for a non-binary LDPC code with* 3 *constraint-nodes and* 6 *symbol-nodes. The non-zero coefficients of the parity-check matrix*

$$\boldsymbol{H}= \begin{bmatrix} 6 & 2 & 0 & 5 & 0 & 0 \\ 0 & 1 & 4 & 0 & 7 & 0 \\ 3 & 0 & 5 & 0 & 0 & 5 \end{bmatrix}$$

are elements of Galois Field $\mathbb{F}_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$ *represented in the labeled edges of the graph This code has coding rate* $r = 1/2$.

The following notation will be used throughout this dissertation:

1) $\mathcal{C}(N, K)$: a code with parameters K and N.
2) Symbol-nodes.
   a) $n \in \{1, \dots, N\}$: symbol-node
   b) $\mathcal{H}(n)$: the set of neighbour check-nodes of symbol-node n.
   c) $d_{\mathrm{s}}(n) = |\mathcal{H}(n)|$: degree of symbol-node n.
   d) $d_{\mathrm{s,max}} = \max_{n=1,\dots,N} |\mathcal{H}(n)|$: maximum symbol-node degree.

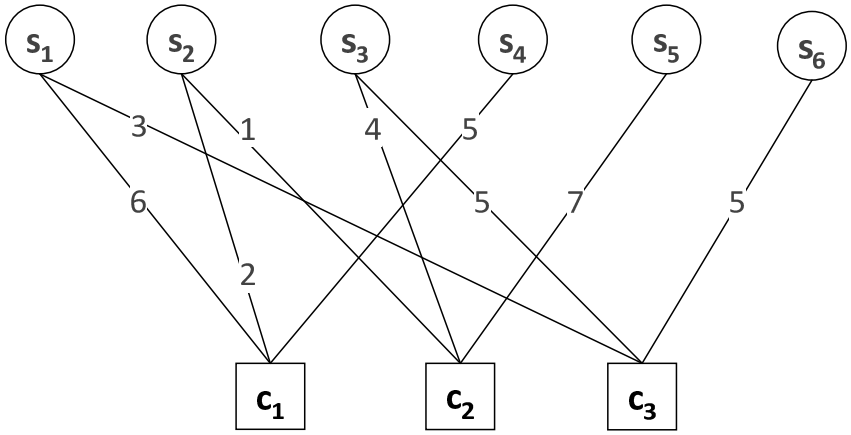

Figure 3.2: Tanner Graph of a NB-LDPC codes of Example 3.1.

e) $d_{\text{s,avg}} = \sum_{n=1,\dots,N} \frac{|\mathcal{H}(n)|}{N}$: average symbol-node degree.

3) Constraint-nodes.

a) $m \in \{1, \dots, M\}$: constraint-node

b) $\mathcal{H}(m)$: the set of neighbour symbol-nodes of constraint-node m.

c) $d_{\text{c}}(m) = |\mathcal{H}(m)|$: degree of constraint-node m.

d) $d_{\text{c,max}} = \max_{m=1,\dots,M} |\mathcal{H}(m)|$: maximum constraint-node degree.

e) $d_{\text{c,avg}} = \sum_{m=1,\dots,M} \frac{|\mathcal{H}(m)|}{M}$: average constraint-node degree.

4) $\texttt{conf}_m = \left\{ [a_n]_{n\in\mathcal{H}(m)} \in \mathbb{F}_q^{d_{\text{c}}(m)} | \sum_{n\in\mathcal{H}(m)} h_{m,n} a_n = 0 \right\}$, the set of *local configurations*. It is the set of symbols $a_n \in \mathbb{F}_q$ verifying the m^{th} constraint-node.

5) $\texttt{conf}_m(n, a) = \{[a_{n'}]_{n'\in\mathcal{H}(m)} \in \texttt{conf}_m | a_n = a\}$ is the set of local configurations for the check-node m, verifying $a_n = a \in \mathbb{F}_q$.

Any codeword $S \in \mathcal{C}(N, K)$ satisfies all the constraint-nodes, therefore the equation (3.4) can be rewritten by considering only the non-zero elements $h_{m,n}$:

$$\sum_{n\in\mathcal{H}(m)} h_{m,n} s_n = 0, \forall m = \{1, \dots, M\} \tag{3.5}$$

A code with a constant number of incoming edges for both type of nodes is referred to as *regular* LDPC code, on the contrary case the code is referred to as *irregular* LDPC code.

In case of regular codes, we denote by $d_s = d_{\text{s}}(n)$, $\forall n = 1, \dots, N$ and $d_c = d_{\text{c}}(m)$, $\forall m = 1, \dots, M$, and we say that the code is (d_s, d_c)-regular. Since the number of incoming edges for the symbol-nodes must be equal to the number of incoming edges of constraint-nodes, we have:

$$Md_c = Nd_s \tag{3.6}$$

As a consequence, the designed coding rate is given by:

$$r = 1 - \frac{M}{N} = 1 - \frac{d_s}{d_c} \tag{3.7}$$

Note also that in case of irregular codes a similar assertion holds by replacing d_s by $d_{\mathrm{s,avg}}$ and d_c by $d_{\mathrm{c,avg}}$. More details will be given in paragraph 3.3.1.

3.2 Iterative Decoding

Non-binary LDPC codes can be decoded by iterative algorithms that pass messages along the edges of the bipartite graph (in both directions). Each message is a probability distribution over $\mathbb{F}_q$, giving the probabilities of the incident symbol-node being equal to each of its q possible values. These probability distributions are iteratively updated by Bayesian rules, until a codeword has been found or a maximum number of iterations has been reached.

The decoding bottleneck is represented by the check-node computation, observing a number of operations per check-node dominated by $\mathcal{O}(q^2)$ [13]. This complexity limited the implementation of decoder for larger order, i.e. $q > 16$, so that alternative decoding schemes have been studied. Indeed, equivalent more robust log-domain decoding algorithms have been introduced in order to reduce the decoding complexity [77]. Successively, the complexity order was reduced by the introduction of Fourier transform [15, 59] in BP decoding, yielding to a computational complexity more suitable for alphabets with $q \geq 64$.

In [67] the authors investigated of Q-ary LDPC codes over Magnetic Recording Channels and they present an implementation of a remarkable reduced complexity Log-FFT Belief Propagation algorithm in which they combined operations in real and log domains.

Several decoding algorithms have been also proposed, allow one to trade off between performance and complexity. In [74] the authors studied the performance of an ultimate Extended Min-Sum decoder with complexity $\mathcal{O}(q' \log_2 q')$ by truncating the number of messages from q to q'. In [65] the Min-Max decoder decreases the complexity by reducing the number of *useful* symbols involved in the check-node processing step.

In the following we will first describe the BP algorithm, then we will give some insights of several low-complexity decoding algorithms.

3.2.1 Belief-Propagation Algorithm

We consider that a codeword S is sent through a noisy channel and let Y denote the channel output.

The Belief Propagation (BP) decoding [76] is a message-passing algorithm that exchanges messages in both directions, along the graph's edges. Messages are updated iteratively until a codeword has been found or a maximum number of iterations has been reached. Each message is a probability vector of size q, giving the probabilities of the incident symbol-node of being equal to each of its possible values.

Notation.

(N1) $\mathbf{P}_n = [P_n(a)]_{a \in \mathbb{F}_q}$ where $P_n(a) = \Pr(s_n = a \mid Y)$, is the probability vector at symbol-node n conditioned on the channel output. These probability vectors constitute the decoder input.

(N2) $\mathbf{P}_{m\to n} = [P_{m\to n}(a)]_{a\in\mathbb{F}_q}$ is the message passing from constrain-node m to symbol-node n. $P_{m\to n}(a)$ is the message concerning symbol $a \in \mathbb{F}_q$.

(N3) $\mathbf{P}_{n\to m} = [P_{n\to m}(a)]_{a\in\mathbb{F}_q}$ is the message passing from symbol-node n to constraint-node m. $P_{n\to m}(a)$ is the message concerning symbol $a \in \mathbb{F}_q$.

(N4) $\tilde{\mathbf{P}}_n = \left[\tilde{P}_n(a)\right]_{a\in\mathbb{F}_q}$ is the a posteriori probability vector at symbol-node n.

Each symbol-node estimation is computed from the associated a posteriori probability, more precisely $\hat{s}_n = \underset{a\in\mathbb{F}_q}{\operatorname{argmax}} \tilde{P}_n(a)$. The goal of the decoding is to detect a codeword $\hat{S} = (\hat{s}_1, \ldots, \hat{s}_N) \in \mathcal{C}(N; K)$ after a finite number of iterations. If the estimated codeword $\hat{S} = S$ then the decoding is successful.

Initialization. The decoder input consists of N probability vectors $\mathbf{P}_n$:

$$\mathbf{P}_n = \left[\Pr\left(s_n = a \mid Y\right)\right]_{a\in\mathbb{F}_q}, n = 1, \ldots, N$$

Their computation depends on the channel output Y, and will be determined for different channel models in Subsection 3.2.4. At this step, symbol-to-constraint messages are initialized by $\mathbf{P}_{n\to m} = \mathbf{P}_n$, $\forall n = 1, \ldots, N$.

Three Steps Iteration.

(S1) *check-node processing.* In this step each constraint-node computes messages to be sent to its neighbour symbol-nodes. Consider a constraint node m and let us denote its degree by d_m. Let us number the symbol-nodes of the neighbourhood of m as $\{n_1, n_2, \ldots, n_d, n\} = \mathcal{H}(m)$, where $d = d_m - 1$ and n corresponds to the destination symbol-node. m computes a message for the symbol-node n from all the received messages from symbol-nodes n_1, n_2, $\ldots$, n_d. The same operation has to be iterated for all the other symbol-nodes n_1, n_2, $\ldots$, n_d that become in turn the symbol-node n.

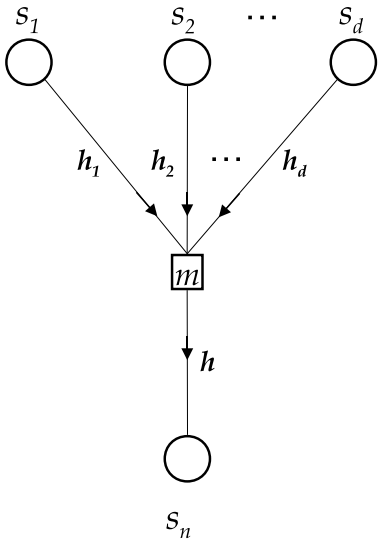

Figure 3.3: Check-node Processing

Figure 3.3 presents this configuration, in which the arrows from $n_1, \ldots, n_d$ represent the incoming messages in the processing node m.

As said above, the output message consists of a probability vector $[P_{m\to n}(a)]_{a\in\mathbb{F}_q}$. Let us focus on a single element $P_{m\to n}(a)$ of this vector, which gives the probability of s_n being equal to a, conditioned on the incoming messages at constraint-nodes m. Consider for the instant there exists only one configuration such that $h_1a_1+\cdots+h_da_d+ha=0$, where $a_1,\ldots,a_d$ are the values of symbol-nodes $n_1,\ldots,n_d$; then:

$$P(s_n=a)=\prod_{i=1}^{d}P(s_{n_i}=a_i)=\prod_{i=1}^{d}P_{n_i\to m}(a_i)$$

Since there exist more than one configuration that satisfies the constraint-node m, the probability $P(s_n=a)$ is given by averaging on all the local configuration in $\mathtt{conf}_m(n,a)$. This can be formulated as follows:

$$\begin{aligned}P_{m\to n}(a) &= \sum_{\substack{a_1,\ldots,a_d\in\mathbb{F}_q^d\\ h_1a_1+\cdots+h_da_d+ha=0}}\ \prod_{i=1}^{d}P_{n_i\to m}(a_i)\\ &= \sum_{[a_{n'}]_{n'\in\mathcal{H}(m)}\in\mathtt{conf}_m(n,a)}\ \prod_{n'\in\mathcal{H}(m)\backslash n}P_{n'\to m}(a_{n'})\end{aligned} \tag{3.8}$$

This operation is effectively a convolution of pdfs in $\mathbb{F}_q$, denoted by the symbol $\circledast$, then it can be written more shortly as:

$$\mathbf{P}_{m\to n}=\mathop{\circledast}_{n'\in\mathcal{H}(m)\backslash n}\mathbf{P}_{n'\to m} \tag{3.9}$$

(S2) *symbol-node processing.*
In this step each symbol-node computes messages to be sent to the neighbour constraint-nodes. Consider a constraint-node m that represents the destination node and let $\mathcal{H}(n)\backslash m$ represent the neighbourhood of a symbol-node n without the destination node. The output message is a probability vector $[P_{n\to m}(a)]_{a\in\mathbb{F}_q}$. Let us focus on a single element of this vector: the probability $P(s_n=a)$ is the product of the incoming messages that affirm that $s_n=a$ with probability $P_{m'\to n}(a)$, $\forall\, m'\in\mathcal{H}(n)\backslash m$.
In Figure 3.4 we have drawn the incoming messages in the symbol-node n that processes the probabilities and it has as output the message for m. We can write:

$$P_{n\to m}(a)=\gamma P_n(a)\prod_{m'\in\mathcal{H}(n)\backslash m}P_{m'\to n}(a), \tag{3.10}$$

where γ is a normalization constant such that $\sum_{a\in\mathbb{F}_q}P_{m\to n}(a)=1$.
The same operation is iterated for all the other constraint-nodes m' that become in turn the destination node m.

(S3) *a posteriori processing.*
Consider a symbol-node n, we are interested in estimating its value $\hat{s}_n$. For this reason we compute the a posteriori probability that the symbol-node value is a from all the incoming contributions of symbol-node n.

$$\tilde{P}_n(a)=\gamma P_n(a)\prod_{m\in\mathcal{H}(n)}P_{m\to n}(a), \tag{3.11}$$

where γ is a normalization constant such that $\sum_{a\in\mathbb{F}_q}P_{m\to n}(a)=1$.

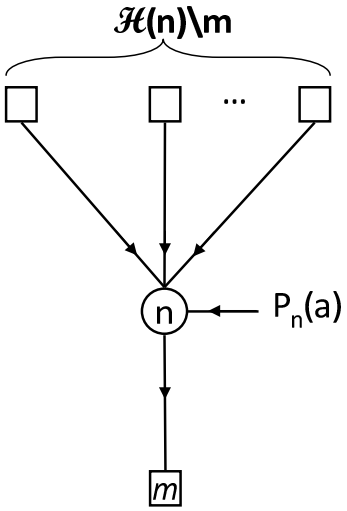

Figure 3.4: Symbol-node Processing

Note that the exchanged messages represent *extrinsic* information, as they are computed from all the available information, except the message received from the destination node.

The complexity of the BP algorithm scales as the square of the cardinality of the alphabet, due to the number of operations required by the step (S1). The BP decoding is also referred as Sum-Product algorithm (SPA) because the above steps involve principally sums and products of probabilities.

The increasing complexity restricts the use of a maximum field order $q = 16$. In his Phd dissertation, Davey [15] introduced a decoding scheme based on the Fast Fourier Transform (FFT) for computing (S1). It will be discussed in the next section.

3.2.2 Binary Fourier Transform Based BP Decoding

We first introduce the Binary Fourier Transform (BFT) operating in $\mathbb{F}_q$. Let $\mathfrak{F}(\mathbf{P})$ be the BFT of the vector $\mathbf{P} = [P_n(a)]_{a\in\mathbb{F}_q}$. The Fourier Transform $\mathfrak{F}(\mathbf{P})$ is simply the product of $\mathbf{P}$ with the operator matrix $\mathcal{F}_q$, called Hadamard matrix:

$$\mathfrak{F}(\mathbf{P}) = \mathbf{P} \cdot \mathcal{F}_q$$

where $\mathcal{F}_q$ is the $q \times q$ Hadamard matrix obtained recursively as:

$$\mathcal{F}_q = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathcal{F}_{q-1} & \mathcal{F}_{q-1} \\ \mathcal{F}_{q-1} & -\mathcal{F}_{q-1} \end{bmatrix}$$

The matrix initialization is given by the Hadamard matrix of order 2.

$$\mathcal{F}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

For this reason, the Binary Fourier Transform is also referred as Hadamard Transform.
Then, one can simplify the operations in (S1) by expressing the probability functions into the Fourier domain. Indeed, by applying this method, the convolution of Eq. (3.9) becomes a simple product and it allows to reduce drastically the operational complexity. Then, nothing that $\mathfrak{F}^{-1} = \mathfrak{F}$, we can rewrite the step (S1):

(S1)$_{\mathfrak{F}}$ *frequency-domain check-node processing.*

$$\mathbf{P}_{m\to n} = \mathfrak{F}\left(\prod_{n'\in\mathcal{H}(m)\backslash n} \mathfrak{F}\left(P_{n'\to m}(a_{n'})\right)\right) \tag{3.12}$$

Since the Binary Fourier Transform $\mathfrak{F}$ can be implemented by using the Fast Fourier Transform (FFT) algorithm, the complexity of step (S1)$_{\mathfrak{F}}$ scales now as pq. This complexity reduction of the Sum-Product decoding allows decoding LDPC codes defined over higher order alphabets (i.e. $q > 16$).

3.2.3 Other Message-Passing Algorithms

Decoding Algorithms are often implemented in log-domain in order to reduce the complexity and the computation instability. The logarithms allow to transform the products into sums, so that the algorithm implementations result to be low-complex and they need less memory. Moreover, the log-domain algorithms have better numerical stability since the obtained messages are already normalized.

Approximate algorithms of the SPA have been proposed in the literature with good compromise between performance and complexity. The Sum-Product in step (S1) is replaced by low-complex operations and the vector products are replaced by term-by-term summations. In the following we describe a general form of low-complexity decoding. According to its rules we obtain several decoding implementations.

Min-Norm Decoding. In the decoding algorithm with lower complexity, the check-node processing results simpler due to the fact that the Sum-Product is replaced by a Min-Norm operations. In fact, the check-node processing is less complex because the operations are sums and comparisons, instead of sums and products.

For a symbol-node n connected to a constraint-node m, the *belief* that $s_n = a \in \mathbb{F}_q$ is evaluated from the beliefs of all other symbol-nodes connected to m that present the best, *i.e.,* as the most likely, configuration.

The possible configurations for the symbol-node n are evaluated in term of *marginal distance* of the symbol-nodes $\{s_1,\ldots,s_d\} \in \mathcal{H}(m)\backslash n$ from the symbols $\{a_1,\ldots,a_d\} \in \mathtt{conf}_m(n,a)$, with $a, a_1,\ldots,a_d \in \mathbb{F}_q$. The configuration $\mathtt{conf}_m(n,a)$ that presents the minimum distance gives the belief of s_n of being each of the q elements:

$$\begin{aligned}
P_{m\to n}(a) &= \min_{\substack{a_1,\ldots,a_d\in\mathbb{F}_q^d \\ h_1a_1+\cdots+h_da_d+ha=0}} \mathtt{dist}(\{s_1,\ldots,s_d\},\{a_1,\ldots,a_d\}) \\
P_{m\to n}(a) &= \min_{\substack{a_1,\ldots,a_d\in\mathbb{F}_q^d \\ h_1a_1+\cdots+h_da_d+ha=0}} \|\mathtt{dist}(s_1,a_1),\ldots,\mathtt{dist}(s_d,a_d)\|_p \\
&= \min_{[a_{n'}]_{n'\in\mathcal{H}(m)}\in\mathtt{conf}_m(n,a)} \left(\sqrt[p]{\sum_{n'\in\mathcal{H}(m)\backslash n}(P_{n'\to m}(a_{n'}))^p}\right)
\end{aligned} \tag{3.13}$$

The p-norm is used here to evaluate the *marginal distance*. Let us describe now the min-norm decoding [65].

Initialization.

$$\mathbf{P}_n = \left[\ln \left(\frac{\Pr\left(s_n = a_n \mid Y\right)}{Pr\left(s_n = a \mid Y\right)} \right) \right]_{a \in \mathbb{F}_q}$$

where $a_n = \underset{a \in \mathbb{F}_q}{\operatorname{argmax}} \Pr\left(s_n = a \mid Y\right)$ is the most likely symbol for the symbol-node n.

Three Steps-Iteration.

(S1) *check-node processing*

According to the used p value in this step, different types of algorithm can be derived.

– 1-norm : **(modified) Extended Min-Sum decoding.**

$$P_{m \to n}(a) = \min_{[a_{n'}]_{n' \in \mathcal{H}(m)} \in \mathtt{conf}_m(n,a)} \left(\sum_{n' \in \mathcal{H}(m) \setminus n} P_{n' \to m}(a_{n'}) \right) \tag{3.14}$$

– 2-norm : **Euclidian decoding.**

$$P_{m \to n}(a_n) = \min_{[a_{n'}]_{n' \in \mathcal{H}(m)} \in \mathtt{conf}_m(n,a)} \left(\sqrt{\sum_{n' \in \mathcal{H}(m) \setminus n} \left(P_{n' \to m}(a_{n'})\right)^2} \right) \tag{3.15}$$

– ∞-norm : **Min-Max decoding.**

$$P_{m \to n}(a_n) = \min_{[a_{n'}]_{n' \in \mathcal{H}(m)} \in \mathtt{conf}_m(n,a)} \left(\max_{n' \in \mathcal{H}(m) \setminus n} P_{n' \to m}(a_{n'}) \right) \tag{3.16}$$

(S2) *symbol-node processing*

$$\begin{aligned} P'_{n \to m}(a) &= P_n(a) \sum_{m' \in \mathcal{H}(n) \setminus m} P_{m' \to n}(a) \\ P' &= \min_{a \in \mathbb{F}_q} \left[P'_{n \to m}(a) \right] \\ P_{n \to m}(a) &= P'_{n \to m}(a) - P' \end{aligned}$$

(S3) *a posteriori processing*

$$\tilde{P}_n(a) = P_n(a) + \sum_{m \in \mathcal{H}(n)} P_{m \to n}(a) \tag{3.17}$$

The novelty of the min-norm decoding scheme is that the exchanged messages could be seen as metrics for measuring the most likely symbols. The messages are Log-Likelihood Ratio with respect to the most likely symbols $s_n = a_n \in \mathbb{F}_q$, which presents the minimum ratio P'.

The 1-norm is an *equivalent* version of the Extended Min-Sum [16] in which the Log-Likelihood Ratio are originally computed with respect to a fixed symbol.

The maximum norm may seem excessive but it could be observed that the computed distances is prevalently dominated by the maximum values; moreover, in practice cases, by using the maximum norm the check-node processing is even more accurate. This can be explained by the fact that the messages in min-max decoding have the minimum incertitude. The complexity of these decoders could be further reduced by decreasing the number of involved symbols in the computation; for more details [65].

3.2.4 Initialization of the Decoder According the $\mathcal{M}$-ary Constellation

Code alphabet. We remind that in this dissertation we define the LDPC codes over a finite field $\mathbb{F}_q$ with q elements. Each element (or symbol) is composed by p bits. The total number of coded symbols has been denoted as N, whereas the total number of coded bits will be denoted by n_b.

Modulation. Let $\mathbb{X}$ be the set of complex symbols *i.e.,* in the $\mathcal{M}$-QAM constellation, where $\mathcal{M} = 2^{\mathtt{m}}$. The k^{th} transmitted modulated symbol $x_k \in \mathbb{X}$ is composed by $\mathsf{m} = \log_2 \mathcal{M}$ coded bits; let N be the number of modulated symbols (note that one may have $\mathsf{N} \neq N$).

Channel output. Assume that the received signal is:

$$y_k = x_k + z_k \tag{3.18}$$

where $k \in \{1, \ldots, \mathsf{N}\}$ and z_k is a Additive White Gaussian Noise distributed with mean 0 and variance σ^2. We denote by:

$$p_Z(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^2}{2\sigma^2}}$$

the probability density function of the random variable Z.

Decoder input. At the receiver side, the soft demapper computes the probability vectors $\mathbf{P}_n = [P_n(a)]_{a \in \mathbb{F}_q}$, for $n = \{1, \ldots, N\}$, where $P_n(a) = \Pr(s_n = a \mid Y)$ is the probability of the transmitted symbol s_n being equal to a conditioned on the channel output Y.

Demapping. Two different instances can occur, for which a symbol or a bit demapping can be adopted. The former can be used only when each coded symbol is mapped directly in one QAM symbol. The latter is specifically applied when one uses a bit-mapping, such that the bits of the same coded symbol can be mapped in different QAM symbols. The last case is typically used in presence of an interleaver (we will deal with this subject in Chapter 6). In the following we analyze these types of demapping.

1) **Symbol-demapping.** This demapping is used if there exists a one-to-one correspondence between the complex signaling set and the alphabet definition. It occurs when the order of the constellation $\mathbb{X}$ is equal to the order of the alphabet $\mathbb{F}_q$, meaning that the number of bits of the modulated symbols is equal to the number of bits of the coded symbols ($\mathtt{m} = p$). In this case the probability vectors are directly derived from the demapped symbols. For this, another requirement is that no interleaver must be placed between the encoder and the modulator. Therefore, the modulated symbols x_k can be referred equivalently to as x_n, y_n will refer to the n^{th} received complex symbol of the channel.
   Let μ be a bijective function mapping the code field $\mathbb{F}_q$ into the complex signaling set $\mathbb{X}$:
   $$\mu : \mathbb{F}_q \xrightarrow{\sim} \mathbb{X} \tag{3.19}$$
   Then a codeword $(s_1, \ldots, s_N)$ is mapped into a vector of complex symbols $(x_1, \ldots, x_N)$, where $x_n = \mu(s_n)$.
   The probability of the transmitted symbol s_n being equal to a conditioned on the received complex symbol y_n is given by:
   $$P_n(a) = \Pr(s_n = a \mid y_n)$$

We can also rewrite it :

$$P_n(a) = \gamma \exp(-|y_n - \mu(a)|^2), \tag{3.20}$$

where γ is a normalization constant such that $\sum_{a \in \mathbb{F}_q} P_n(a) = 1$.

Remark 3.1. *In this case the number of complex symbols* N *is equal to the number of the coded symbols* N.

2) Bit-demapping. This demapping is used when there does not exist any correspondence between the complex signaling set and the alphabet definition. It occurs when the order of the constellation and the coding alphabet size are different, or when a de-interleaver is placed before the decoder (in presence of an interleaving system). Since the bit-demapping degrades the performance, one should be avoid it if a symbol-demapping could be used.

In the following we consider the probability computation of one symbol s_n, with $n = 1, \ldots, N$. In order to compute the probability of s_n, the demapper needs to compute the probabilities of its binary components of being 0 or 1 according to the channel output.

Transmitter side.

1) **Binary image.** The binary image $(s_{n,1}, \ldots, s_{n,p})$ of s_n is obtained according to the isomporphism (3.2):

$$\mathbb{F}_q \xrightarrow{\sim} \mathbb{F}_2^p$$

2) **Bit-mapping.** Let μ be a bijective function mapping $\mathbb{F}_2^{\mathtt{m}}$ into the complex signaling set $\mathbb{X}$:

$$\mu : \mathbb{F}_2^{\mathtt{m}} \xrightarrow{\sim} \mathbb{X}$$

such that it maps a vector of $\mathtt{m}$ bits, taken from the binary image of the codeword, into a QAM-symbol x_k:

$$\mathbf{b} = (b_1, \ldots, b_{\mathtt{m}}) \mapsto \mu(b_1, \ldots, b_{\mathtt{m}}) \in \mathbb{X}$$

Let $s_{n,i}$ be a transmitted bit and t be its position within the vector of consecutive bits $(b_1, \ldots, b_{\mathtt{m}})$ that is mapped into a complex symbol $x_k = \mu(b_1, \ldots, b_{\mathtt{m}})$, this means that $b_t = s_{n,i}$.

Receiver side.

2') **Bit-level demapping.** This is the reverse operation of the bit-mapping and it outputs the probability of each bit of being 0 or 1. Hence, the goal of the binary demapping is to compute:

$$\Pr(s_{n,i} = 0 \mid y_k) \quad \text{and} \quad \Pr(s_{n,i} = 1 \mid y_k)$$

where y_k is the received complex symbol according to (3.18). We have:

$$\begin{aligned} \Pr(s_{n,i} = 0) &= \sum_{\substack{(b_1,\ldots,b_t,\ldots b_{\mathtt{m}}) \\ b_t = 0}} \Pr(\mathbf{b}|y_k) \qquad (3.21) \\ &= \gamma \sum_{\substack{(b_1,\ldots,b_t,\ldots b_{\mathtt{m}}) \\ b_t = 0}} \exp(-|y_k - \mu(\mathbf{b})|^2) \end{aligned}$$

It follows for the other case:

$$\Pr(s_{n,i} = 1) = \gamma \sum_{\substack{(b_1,\ldots,b_t,\ldots b_{\mathtt{m}}) \\ b_t = 1}} \exp(-|y_k - \mu(\mathbf{b})|^2) \qquad (3.22)$$

where γ is a normalization factor such that $\Pr(s_{n,i} = 0) + \Pr(s_{n,i} = 1) = 1$. This operation is referred to as bit-marginalization.

1') **Probability of the symbol s_n.** Let $(a_1, \ldots, a_p)$ be the binary image of the element $a \in \mathbb{F}_q$. From the bit to symbol-demapping we have:

$$P_n(a) = \Pr(s_n = a) = \gamma \prod_{i=1}^{p} \Pr(s_{n,i} = a_i) \qquad (3.23)$$

where γ is a normalization factor such that $\sum_{a \in \mathbb{F}_q} \Pr(s_n = a) = 1$

Then, the probability $P_n(a)$ is one of the elements of $\mathbf{P}_n, \forall n = 1, \ldots, N$. These probabilities (that could also be expressed in log or LLR domain, according to the decoding algorithm) represent the input of the decoder.

Remark 3.2. *If there is no matching between coded and modulated symbols, the marginalization performed by the bit-demapper induces a marginalization loss. In the Section 6.5 (page 109) we will show that the performance loss can be greater than* $0.5dB$*. For a more general discussion we refer to [17].*

3.3 Asymptotic Optimization

Although the study of non-binary LDPC codes is mainly motivated by their performance at short to medium lengths, their asymptotic analysis proves also to be very useful. In particular, it allows the understanding of topology independent phenomena or, stated differently, it allows distinguishing between events due to code family parameters, and those due to a particular code realization.

3.3.1 Irregularity Profile

The irregularity profile identifies a specified ensemble of LDPC codes, independently from the used alphabet and the codelength. We have already denoted the *degree of a node* as the number of its incident edges, and that we referred to symbol-node degree as $d_{\mathrm{s}}(n)$ and to constraint-node degree as $d_{\mathrm{c}}(m)$ (see the notation at page 36).

Irregular LDPC codes represent a general class of LDPC codes because they have non-constant symbol and check degrees. The irregularity profile can be polynomially represented by check-node and symbol-node degree distributions [62].

In Eq. (3.24) we express the symbol-node distribution $\lambda(x)$ and the check-node distribution $\rho(x)$ from the *edge perspective*:

$$\lambda(x) = \sum_{d=1}^{d_{\mathrm{s,max}}} \lambda_d x^{d-1} \qquad\qquad \rho(x) = \sum_{d=1}^{d_{\mathrm{c,max}}} \rho_d x^{d-1} \tag{3.24}$$

where λ_d is the fraction of edges connected to symbol-nodes of degree d and, similarly, ρ_d is the fraction of edges connected to check-nodes of degree d.

Another way to express the irregularity of the code is by using the symbol-node distribution $L(x)$ and the check-node distribution $R(x)$ from the *node perspective*:

$$L(x) = \sum_{d=1}^{d_{\mathrm{s,max}}} l_d x^{d} \qquad\qquad R(x) = \sum_{d=1}^{d_{\mathrm{c,max}}} r_d x^{d} \tag{3.25}$$

where l_d is the fraction of symbol-nodes of degree d, and, similarly, r_d is the fraction of check-nodes of degree d. These two perspectives are related each other by the following equations:

$$\lambda(x) = \frac{L'(x)}{L'(1)} \qquad\qquad \rho(x) = \frac{R'(x)}{R'(1)} \tag{3.26}$$

$$L(x) = \frac{\int_0^x \lambda(z)dz}{\int_0^1 \lambda(z)dz} \qquad\qquad R(x) = \frac{\int_0^x \rho(z)dz}{\int_0^1 \rho(z)dz} \tag{3.27}$$

We note that $L'(1) = R'(1)$ is the number of graph edges, while $\int_0^1 \rho(z)dz = M$ and $\int_0^1 \lambda(z)dz = N$.

For the regular LDPC codes we have already computed the rate by using the node degrees in Section 3.1.2. More generally, the rate is associated with the average degrees $d_{\mathrm{s,avg}}$ and $d_{\mathrm{c,avg}}$:

$$r = 1 - \frac{d_{\mathrm{s,avg}}}{d_{\mathrm{c,avg}}} = 1 - \frac{L'(x)}{R'(x)} = 1 - \frac{\int_0^1 \rho(x)}{\int_0^1 \lambda(x)} \tag{3.28}$$

Example 3.2. *Consider as example: the Hamming code already introduced in 1.1 (page 4) represented by the matrix H:*

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

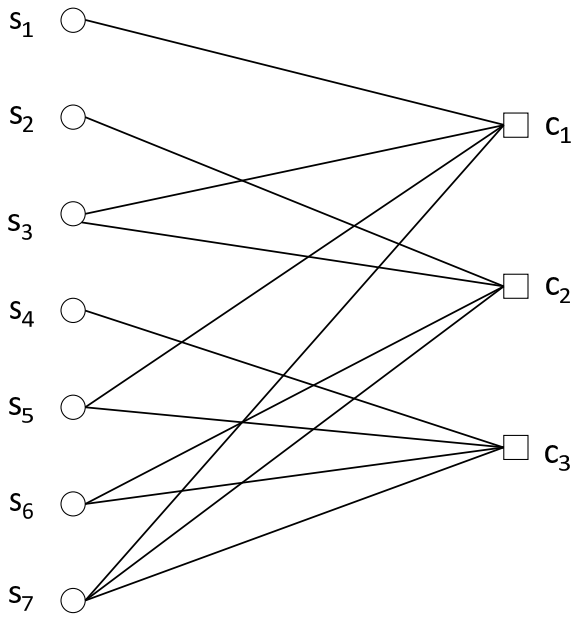

Figure 3.5: Tanner graph associated to the (7,4) Hamming code of Example 3.2

We can associate to the parity-check matrix H a graph $\mathcal{H}$ of Figure 3.5. The degree distributions of that code are the following:

$$\lambda(x) = \frac{3}{12}x^0 + \frac{6}{12}x^1 + \frac{3}{12}x^2 \qquad \rho(x) = \frac{12}{12}x^3$$

$$L(x) = \frac{3}{7}x^1 + \frac{3}{7}x^2 + \frac{1}{7}x^3 \qquad R(x) = \frac{3}{3}x^4$$

It easy to verify the above equations and that the rate is $r = 4/7$.

3.3.2 Density Evolution

Let $\mathcal{E}(\lambda, \rho)$ be the ensemble of all the LDPC codes with edge-perspective degree distribution polynomials $\lambda(x)$ and $\rho(x)$. The asymptotic performance of the ensemble $\mathcal{E}(\lambda, \rho)$ is referred to as threshold, and corresponds to the worst channel condition that allows transmission with an arbitrary small error probability when the code length tends to infinity. Put differently, given a LDPC code from an ensemble $\mathcal{E}(\lambda, \rho)$ with a sufficiently large blocklength N, the coded transmission is considered to be reliable iff the channel parameter is smaller than the computed threshold value. Hence, the decoding is successful after sufficiently many iterations with an arbitrary small probability.

The computation of the threshold is based on tracking the probability density of the messages passed throughout the message-passing iterative decoder. This is known as *density evolution* method and it has been introduced in [59]; this method is extensible to a very broad class of transmission channels and iterative decoding algorithms. In the following we describe the principles of the density evolution computation.

Definition 3.1. Cycles. *Given a graph $\mathcal{H}$, a cycle of length g is a path of g edges which closes back on itself. The girth, denoted by g_{min}, is the length of the shortest cycle of the graph.*

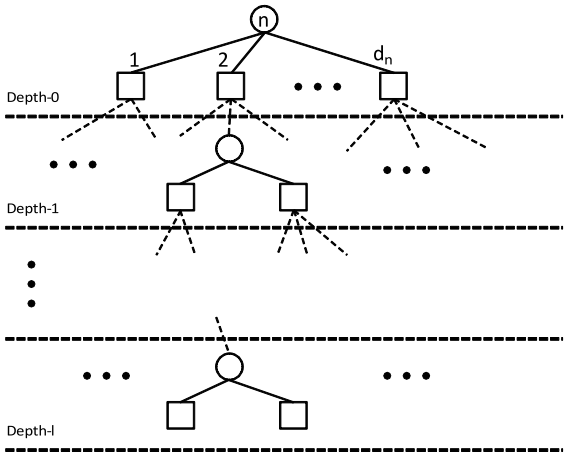

Figure 3.6: Representation of $\mathcal{N}_n^{(l)}$

For a bipartite graph, the girth is necessarily an even number and depends on the degree distribution as well the codeword length N.

The set of all the nodes connected to n by a path of length less than or equal to $2l+1$, referred to as the local neighbourhood of depth l of n, will be denoted by $\mathcal{N}_n^{(l)}$.

Definition 3.2. Tree-like assumption (or independence assumption). *This assumption states that for any symbol-node n, its local neighbourhood of depth l, $\mathcal{N}_n^{(l)}$, is a* tree *(a tree is a graphs without cycles, see Figure 3.6).*

Clearly, for a graph with girth $g_{\min}$, the tree-like assumption is valid for any $l \leq g_{\min}/2$. The tree-like assumption is important because it ensures that the messages passed along the graph edges, at any decoding iteration $\leq l$, are *statistically independent*. The following theorem [59] states that the tree-like assumption is *almost surely* verified in the asymptotic limit of the code-length.

Theorem 3.1. Convergence to cycle-free case. *For any $l \geq 0$, $\mathcal{N}_n^{(l)}$ is cycle-free with probability approaching one in the limit of infinite code blocklength N.*

Density Evolution for binary LDPC codes. Consider whichever channel model depending on a parameter, such that the channel conditions worsen when it increases, *i.e.,* this parameter may be the noise variance σ^2 for the AWGN channel or the erasure probability ϵ for the BEC channel. The threshold of the ensemble $\mathcal{E}(\lambda, \rho)$ is defined as the worst channel condition (supremum value) that allows transmission with an arbitrary small error probability, assuming that the transmitted data is encoded with an arbitrary length code of $\mathcal{E}(\lambda, \rho)$.

In the following we consider the case of AWGN channel. The decoder stops after a fixed maximum number of iterations, thus usually we are interested in the capacity of such decoder after l rounds. For a given decoder and an ensemble of codes $\mathcal{E}(\lambda, \rho)$, this capacity is expressed as the error probability of exchanged messages at iteration l, denoted by $P_e^{(l)}(\lambda, \rho, \sigma)$. We also note that for symmetric channels [59], this error probability is independent on the transmitted codeword and, for a sake of simplicity, one may assume that the all-zero codeword is transmitted.

The density evolution analysis, when applied to a cycle-free graph at iteration l, allows computing $P_e^{(l)}(\lambda, \rho, \sigma)$. This computation (not detailed here) is possible thanks to statistically independence of the messages passed along the graph edges. Moreover, according to Theorem 3.1, $P_e^{(l)}(\lambda, \rho, \sigma)$ gives the error probability of exchanged messages, in the asymptotic limit of the code-length and averaging over all codes in $\mathcal{E}(\lambda, \rho)$. Interestingly, this error probability depends only on the channel parameter σ, and the degree distribution polynomials λ and ρ.

Finally, for the given ensemble $\mathcal{E}(\lambda, \rho)$, the message-passing decoder is successful iff:

$$\lim_{l \to +\infty} P_e^{(l)}(\lambda, \rho, \sigma) = 0$$

and the asymptotic threshold of the ensemble $\mathcal{E}(\lambda, \rho)$ is defined as the supremum value of the channel parameter σ (worst channel condition) that allows successful decoding:

$$\sigma_{\text{th}}\left(\mathcal{E}(\lambda, \rho)\right) = \sup\left\{\sigma \mid \lim_{l \to +\infty} P_e^{(l)}(\lambda, \rho, \sigma) = 0\right\}$$

The analysis of binary LDPC code is quite long and complicated to describe in this chapter. The reader can find more details by reading the paper of Richardson and Urbanke [59]. The exact density evolution method has as main disadvantage the prohibitive computational resources needed for the numerical evaluation of message densities.

Exact Density Evolution for non-binary LDPC codes. Unfortunately, for the non-binary case, exact density evolution is only manageable for the Binary Erasure Channel [56, 64]. In [64] the asymptotic performance of NB-LDPC ensembles has been derived, by considering both the irregularity profile of the graph and the probability distribution of the edges label $h_{m,n}$. Finally, we note that over the AWGN channel, an analysis of the density evolution under Gaussian approximation for non-binary LDPC codes has been done in [43].

3.3.3 Density Evolution via Gaussian Approximation for the Binary Case

In the case of more general memoryless channels, like the binary-input AWGN channel, density evolution can be approximated by modeling the density of the messages, e.g. assuming that they are Gaussian distributed [12, 41, 42]. This method allows an easier and faster analysis at the cost of a less precision with respect to the exact density evolution.

Under the Gaussian approximation, exchanged messages are assumed to be symmetric Gaussian distributed. Consequently, the density evolution only have to track the mean of the exchanged message. This simplification yields to a closed form formula for the computation of the threshold that, with acceptable accuracy, approaches the real threshold. More precisely, for the Gaussian Approximation analysis, the following assumption is made:

Gaussian Distribution Assumption (GD) Messages received at every node at every iteration are independent and identically distributed (i.i.d), with symmetric Gaussian distribution of the form:

$$f(x) = \frac{1}{\sqrt{4\pi\mu}} e^{-\frac{(x-\mu)^2}{4\mu}}$$

where the parameter μ is the mean.

Theorem 3.2. *[41]*
Consider an ensemble $\mathcal{E}(\lambda, \rho)$. Let $P_e^{(0)}(\sigma) = Q\left(\frac{1}{\sigma}\right)$, be the error probability at iteration 0 (decoder input). Then under the GD assumption, the error probability $P_e^{(l+1)}$ of the BP decoder at iteration l, can be recursively obtained as follows:

$$P_e^{(l)}(\sigma) = \sum_{i=2}^{d_s} \lambda_i Q\left(\sqrt{\frac{1}{\sigma^2} + (i-1)\left[Q^{-1}\left(\frac{1-\rho\left(1-2P_e^{(l-1)}(\sigma)\right)}{2}\right)\right]^2}\right)$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\left(-\frac{t^2}{2}\right)} dt$.

This formula allows to recursively compute $P_e^l(\sigma)$ starting from $P_e^0(\sigma)$.

3.3.4 Optimization of Degree Distributions

The asymptotic analysis is also aimed at finding LDPC codes with the best thresholds. The threshold value depends on the *irregularity profile* (*i.e.* degree distribution polynomials) that determines the ensemble of codes.

There are different methods for optimizing irregularity profile of the LDPC codes [11, 60]. In general, the performance of LDPC codes in noisy channels is more sensitive to the bit-node degree distribution. Consequently, it is preferable to have irregular codes with optimized λ distribution and an uniform ρ distribution. Precisely, assuming that $\lambda(x)$ is given, $\rho(x)$ is chosen to be a polynomial concentrated on two successive degrees:

$$\rho(x) = \rho x^w + (1-\rho)x^{w+1}$$

with $w \geq 2$ and $0 < \rho \leq 1$ are determined according to given λ and target coding rate. Therefore, the problem of optimizing the irregularity profile (λ, ρ) reduces to optimizing the degree distribution polynomial λ only. According to the channel model and density evolution equations, this optimization problem can be solved by linear optimization or genetic optimization algorithms.

In this thesis, we used the Differential Evolution algorithm (a genetic optimization algorithm) to optimize the symbol-node degree distribution polynomial λ for non-binary LDPC codes. This algorithm is described in the next chapter, when we discuss numerical methods for approximating the density evolution of non-binary LDPC codes.

3.4 LDPC Construction

The construction of good LDPC codes passes through the design of the parity-check matrix with prescribed properties. The design approaches follow various criteria as for example the near-capacity performance and efficient encoding and decoding.

Consider the parity-check matrix H. It is preferable to have a higher column weight of H thus the symbol-nodes can better evaluate the outgoing messages, on the other hand, lower row weights of H yields to a more precise computation in check-node processing. Intuitively this trade off could be managed thanks to the flexibility given by the irregular LDPC codes. Indeed, irregular LDPC have demonstrated improved performance with respect to the regular codes [44].

Fundamental issue for the LDPC codes performance is the presence of cycles in the Tanner graphs. Roughly speaking, with a message passing type algorithm, cycles lead to auto-confirmation of exchanged messages, and then degrade the decoder performance. In practice, it is impossible to avoid cycles, but one may try to construct graphs with girths as large as possible, so that to maximize the uncorrelation of messages.

The construction of good LDPC codes have been a subject of intensive research in the years. The design of LDPC codes is done starting by the sparse parity-check matrix H. Tanner proposed a general recursively construction of long error-correcting codes and he determined a lower bound on the minimum distance that grows exponentially with the girth of the code [70]. In [9] the bit-filling algorithm has been introduced: the authors present an algorithm that adds columns to a previously constructed parity-check matrix obtaining high girths.

There exist several propositions of LDPC construction in the literature. LDPC codes can be randomly built once the weights for each row and column of H have been fixed. However, this strategy does not avoid short cycles. MacKay's construction attempt to keep the minimum number of short cycles [46]. In particular, his method avoid the *overlap* between more than two entries in two column, so that to ensure at least the absence of cycle of length 4.

The Progressive Edge-Growth (PEG) proposed in [30, 32] is a general method that progressively connects nodes in order to construct graphs with large girths. Regular and irregular LDPC codes have been successfully realized and the simulation results demonstrate that using PEG for short blocklength codes improves the code performance.

The edge label choice represents an another degree of freedom in the design of non-binary LDPC code. One can obtain NB-LDPC by designing a binary parity-check matrix and replacing the 1's by symbols of the finite field. The non-zero elements can be selected randomly [32] or not completely random: in [46] these entries are chosen according to the marginal entropy principle. In [54] LDPC codes defined over $\mathbb{F}_q$ have been designed by using the algebraic properties of the binary image representation of the parity-check matrix. The optimization proceeds by maximizing the minimum distance per row, in such a way to make more reliable the exchanged messages along the graph edges.

3.4.1 Progressive Edge-Growth Construction

The Progressive Edge-Growth is an algorithm for constructing bipartite graphs with large girths. It is a suboptimal algorithm because the underlying graph grows edge by edge in a best-effort manner. The idea is to maximize the local girth each time a new edge is added by connecting each symbol-node to the most *distant* check-node. Once given the number of symbol-nodes N and the number of check-nodes M, and fixed the degree distribution pair (λ, ρ), the algorithm starts by placing connections between symbol and check nodes.

Each time a new connection is defined, the current check-node degree is updated. Consider a constraint-node m, let $d_{\mathrm{c}}(m)$ be its expected degree [1], whereas d_m is its (current) degree and $\overline{d}_m = d_{\mathrm{c}}(m) - d_m$ is its (current) complementary degree. The expected degree [2] of a symbol-node n is denoted by $d_{\mathrm{s}}(n)$.

1. According to the constraint-node degrees distribution.
2. According to the symbol-node degrees distribution.

The algorithm maintains a set A of the *available* check-nodes, that is constraint-nodes that did not reach their expected degree: $\mathsf{A} = \{m \mid \overline{d}_m > 0\}$. Each time a a check-node reaches its expected degree, it is removed from A. For each symbol-node n the algorithm starts by connecting it to a first check-node with the maximum complementary degree. Then, for each of all the other connections of n, it expands the whole graph starting with n, by considering all the connections (edges) previously established in the graph. This expansion allows determining the distance [3] from all the other nodes to n. Nodes that are not connected to n by any path are considered as being at infinite distance from n. Finally, when the expansion is completed, the algorithm chooses an available check-node from A, which is as maximum distance from n. If there are several check-node in A at maximum distance from n, a check-node with the maximum complementary degree is chosen, meaning that the selection gives preference to check-nodes with more "free" (unestablished) edges. Note also that the graph expansion may be stopped if all the available check-nodes from the set A have appeared in the expanded graph. Moreover, one can decide to expand the graph till the bottom or to stop the expansion to a depth of expansion referred to as $l_{\max}$.

Let us explain the algorithm with a pseudo-code.

Progressive Edge-Growth Algorithm

Define $\mathsf{A} = \{1, \ldots, M\}$ (the set of all the check nodes)
for $n = 1$ to N (loop over all the symbol-nodes)
  for $d = 1$ to $d_{\mathrm{s}}(n)$ (loop over the degree of the considered symbol)

    if $d = 1$ (first connection)
      Choose $m \in A$ with maximum complementary degree.
    else
      Expand the graph rooted by n, until depth $l_{\max}$, and let $\mathcal{N}_n^{(\max)}$ denote the expanded graph. Two cases occur:
      (1) $\mathsf{A} \not\subseteq \mathcal{N}_n^{(\max)}$. In this case, choose $m \in A \setminus \mathcal{N}_n^{(\max)}$ with maximum complementary degree.
      (2) $\mathsf{A} \subseteq \mathcal{N}_n^{(\max)}$. In this case, choose $m \in A$ at maximum distance from n and with maximum complementary degree.
    end

    – Make the connection (n, m).
    – Increase d_m: $d_m = d_m + 1$
    – Update A: if $d_m = d_{\mathrm{c}}(m) \Rightarrow \mathsf{A} = \mathsf{A} \setminus \{m\}$
  end
end

Remarks.

(R1) **Original version of PEG algorithm.** In its original version [32], the PEG algorithm does not receive in input the check-node degrees distribution, but only the symbol-node degrees distribution. It selects the check-nodes at maximum distance

3. The distance between two nodes is the length of the shortest path connecting the two nodes.

from n and having the minimum current degree. Consequently, it constructs a graph in which the check-node degrees are concentrated around an average values.

Instead, in the implemented PEG, as we have just seen, we can eventually choose a desired check-node distribution by fixing the fractions of check-nodes with a certain degree, according to the chosen polynomial. Moreover, the algorithm selects the check-nodes with maximum complementary degree in order to give priority to the check-nodes with the greatest degree.

(R2) **Edge fraction distributions.** Since the size of the designed LDPC code is finite, the PEG algorithm can only assure an approximation of the edge fractions obtained as rounding off the desired degree distributions. Moreover, if we associate the PEG algorithm to the asymptotic analysis results, one have to know that the larger the degrees used, the larger the blocklength is necessary in order to approach the predicted threshold.

(R3) **Bounds on the girth and minimum distance.** Lower and upper bounds on the girth and on the minimum distance have been formulated, in general for regular LDPC codes [30]. These bounds depends exclusively on the number of check-nodes, and the node degrees d_s and d_c.

Moreover, the minimum distance of regular LDPC codes, with $d_s \geq 3$, increases linearly with the codelength N. Some bounds of the minimum distance for a general Tanner graph have been computed in [71]; in [30, 31] some bounds of the minimum distance have been computed for PEG-designed LDPC codes.

(R4) **Maximum Depth** In the above description of the PEG algorithm, we stop the graph expansion at a maximum depth l_{max}. This reduces the diameter of the graph when the number of decoding iterations is fixed to be smaller than l_{max} and allows to a complexity reduction of the PEG algorithm. This approach can be adopted for long-length and low-rate codes when the minimum distance is not a critical issue. Otherwise, one may decide to exclude this condition from the algorithm.

3.4.2 Optimization of the Edge Labels

Before discussing the optimization procedure, let us first discuss some of the parameters the overall performance of the LDPC code depends on.

Minimum distance. The minimum (*Hamming*) distance d_{min} of a block code is the minimum number of digits in which any two distinct binary codewords differ. Thanks to the linearity property of the code, d_{min} is also equal to the minimum codeword weight. It follows that d_{min} is also equal to the minimum number of linearly dependent columns of the parity-check matrix H (since each linearly dependent subset of d columns corresponds to a codeword of weight d).

Stopping sets. For a binary LDPC code, a stopping set $\mathcal{S}$ is a sub-set of bit-nodes such that all the neighbours of $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice [18].

The stopping set concept is fundamental in finite-length analysis of LDPC codes over the BEC. Let us briefly explain the reason. Over the BEC (page 17), the belief-propagation (BP) decoding translates into a simple technique of recovering the erased bits, by iteratively searching for check-nodes with only one erased neighbor bit-node. Indeed, if a check-node is connected to only one erased bit-node, the value of the latter can be recovered as the XOR of all the other neighbor bit-nodes. This value is then injected into the decoder, and the

decoding continues by searching for some other check-node having a single erased neighbor bit-node. In this case, we do not have to specify any maxim number of decoding iterations. The decoding will stop by itself if all the bits have been recovered, or when it "gets stuck" because any check-node is connected to at least two erased bit-nodes. The latter situation corresponds to the above definition of a stopping set. Hence, the BP decoding is successful iff the set of erased bits does not contain any stopping set. Otherwise, the decoding fails, and the set of bits that cannot be recovered is the maximum stopping set contained in the set of erased bits.

The size of the minimum stopping set is called *minimum stopping distance*. Clearly stopping sets necessarily contain cycles. Therefore, the minimum stopping distance is lower bounded by the girth of the code. Put differently, codes with small minimum stopping distance also have small girths. The converse needs node be true: for codes with any symbol node degree $d_{\mathrm{s}}(n) \geq 3$, the minimum stopping distance grows exponentially with the girth [49, 50]. However, best codes (*i.e.* with asymptotic threshold close to the channel capacity) generally have a large fraction of degree-2 bit-nodes, and thus may contain small stopping sets. Small stopping sets clearly determine the performance of the code in the error-floor region over the BEC, but they also impact, at a different degree, the error-floor performance over more general channel models, as explained below.

Trapping sets. The code performance in the error-floor region can be analyzed in terms of *trapping sets* [58]. An (a, b)-trapping set is a set $\mathcal{T}$ of a bit-nodes, having exactly b neighbor check-nodes that are connected to $\mathcal{T}$ an odd number of times. Informally, a trapping set is a set of variable nodes that is not well connected to the rest of the graph, in a way that causes the decoder trouble. The most problematic cases are those corresponding to small b values. Clearly, a stopping set $\mathcal{S}$ if a particular example of $(|\mathcal{S}|, 0)$ trapping set.

Local optimization of the component codes. In [54] an exhaustive method has been proposed in order to optimize regular (2,d_c)-LDPC codes. The optimization is divided in two parts, one for the waterfall region and one for the error floor, both optimization ideas are based on the minimum distance of the binary image of the parity-check matrix H. The optimization takes principally into account of maximizing the minimum distance, reducing the short cycles and mitigating the effects of stopping sets. Let us summarize the key concepts.

A *binary component codes* is a subcode $\mathcal{C}_m$ obtained from the m^{th} parity-check equation. Let $H_{m,n}$ be the transpose binary image of the entry $h_{m,n}$, then:

$$\sum_{n \in \mathcal{H}(m)} H_{m,n} \mathbf{s}_n^T = \mathbf{0}$$

where $\mathbf{s}_n^T$ is the transpose binary image of the symbol s_n and $\mathbf{0}$ is the zero vector.

Ideas of optimization

1) **Improving the waterfall region.** The careful selection of good coefficients can reduce the error probability in the waterfall region. The idea is to maximize d_{min} by locally maximize the minimum distance for each component code $\mathcal{C}_m$. Another issue is also to minimize the multiplicity of codewords with Hamming weight $w_H = d_{min}$. A good minimum distance d_{min} renders the exchanged messages more distinguishable and reliable.

2) **Improving the error floor.**
   1) There is a relation between d_{min} and cycles. Let H_b^l be the binary image of the coefficients involved in a cycle l. If H_b^l is full rank the cycle does not provide any codeword, meaning that the considered cycle is not involved in the equivalent minimum distance. Therefore, one needs to choose the coefficients in order to have full-rank binary parity-check matrices H_b^l, at least for the shortest cycles. This yields lower error floors.
   2) Stopping set are related to the cycles, then if short cycles are avoided, small stopping sets are reduced. For improving the error floor performance, one has also to maximize the minimum stopping distance of the code.

Optimization of the component codes. In [17], the authors generalize the approach of [54]. The idea is to concentrate the mutual information of the extrinsic messages (propagated in a message-passing decoding) around their average value. The idea is that the exchanged messages are more reliable if they statistically behave equally. In order to do that the following algorithm has been proposed:

$\forall\, m = 1, \ldots, M$
- Consider the m^{th} constraint-node and let $d_c = d_c(m)$ be its degree. Consider the set of non-zero values $\mathbf{h} = \{h_1, \ldots, h_{d_c}\} \in \mathbb{F}_q$
- Consider the d_c binary subcodes $\mathcal{C}_m(t)$ formed from the combination of the $d_c - 1$ values of $\mathbf{h}$ except h_t.
- Choose $\mathbf{h}$ such that:

$$\mathbf{h} = \underset{\{h_1,\ldots,h_{d_c}\}}{\operatorname{argmax}} \left(\sum_{t=1}^{d_c} d_{min}\left(\mathcal{C}_m(t)\right) \right)$$

  constrained to $\mid d_{min}(\mathcal{C}_m(t)) - (\mathcal{C}_m(t')) \mid \leq 1$

From this optimization, the following sets of coefficients are obtained for $d_c = 4$:

For codes defined over $\mathbb{F}_{64} = \mathbb{F}_2[\alpha]/(\alpha^6 + \alpha + 1)$:

$$(\alpha^0, \alpha^9, \alpha^{26}, \alpha^{46}) \qquad (\alpha^0, \alpha^{17}, \alpha^{26}, \alpha^{43})$$
$$(\alpha^0, \alpha^{17}, \alpha^{37}, \alpha^{54}) \qquad (\alpha^0, \alpha^{20}, \alpha^{27}, \alpha^{46})$$

For codes defined over $\mathbb{F}_{256} = \mathbb{F}_2[\alpha]/(\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1)$:

$$(\alpha^0, \alpha^8, \alpha^{173}, \alpha^{183}) \qquad (\alpha^0, \alpha^{10}, \alpha^{82}, \alpha^{90})$$
$$(\alpha^0, \alpha^{72}, \alpha^{80}, \alpha^{245}) \qquad (\alpha^0, \alpha^{165}, \alpha^{175}, \alpha^{247})$$

3.5 Encoding

As any linear block code, LDPC codes can be encoded by using the generator matrix that can be obtained from the parity-check matrix of the code by Gausian elimination. Yet, this standard method results to have complexity $\mathcal{O}(N^2)$, which becomes prohibitive for large block size N.

In [61] an efficient encoding system has been presented. It allows to encode LDPC codes by exploiting the sparseness of the parity-check matrix. The encoding complexity scales as $g^2 N$, where g is generally a small value, called the *gap of the code*.

Other linear-time encoding systems have been presented in the literature. The Irregular Repeat Accumulate (IRA) codes [19, 34] combines characteristics of both LDPC codes and turbo codes. The IRA's encoding presents simple structure and linear complexity. Theses codes also demonstrate near-capacity performance.

In the following we explain how to encode a non-binary LDPC code, by using either the generator or the parity-check matrix. We explain the encoding for systematic codes, for which it is also possible to permute the positions of the information and parity symbols on condition that at the receiver side they take again the original positions.

3.5.1 Encoding using the Generator Matrix

The binary image of any NB-LDPC code is a linear binary code $\mathcal{C}(\mathtt{n}_b, \mathtt{k}_b)$ where $\mathtt{k}_b = Kp$ and $\mathtt{n}_b = Np$. The subscript "b" indicates the binary vectors or values. This code is obtained as a $\mathtt{k}_b$-dimensional linear subspace of vector space $\mathbb{F}_2^{\mathtt{n}_b}$. The generator basis $\{g_1, \ldots, g_{\mathtt{k}_b}\}$ can be placed as rows of the matrix G, so that each binary codeword $s_b = u_b G$, where $u_b = [u_1, \ldots, u_{\mathtt{k}_b}]$ is the information vector and $s_b = [s_1 \ldots, s_{\mathtt{n}_b}] = [s_{1,1} \ldots, s_{1,p}, \ldots s_{N,1}, s_{N,p}]$ is encoded vector. For this, G is called *generator matrix* of the code $\mathcal{C}(N, K)$. Thus the transmitted codeword is grouped in symbols of p bits according to the alphabet definition. By deepening the definition given in 3.1.1 at page 34:

$$H_b s_b^T = 0 = H G^T u_b^T = 0 \Longleftrightarrow H_b G^T = 0$$

G can be obtained by putting the sparse matrix H_b in the form $[P^T I_{\mathtt{m}_b \times \mathtt{m}_b}]$ via Gaussian elimination ($\mathtt{m}_b = \mathtt{n}_b - \mathtt{k}_b$). Then $G = [I_{\mathtt{k}_b \times \mathtt{k}_b} P]$, where P is a matrix of dimension $\mathtt{k}_b \times \mathtt{m}_b$; I indicates the identity matrices, in which the subscripted indexes give the matrix dimension. The sequence of symbols S is obtained according to the isomporphism (3.2) by grouping the vector s_b per p bits.

The generator matrix taken in this form, has the property that the information symbols are embedded in the resulting codeword. These kind of codes are named *systematic codes* and they are composed by two vectors:

$$S = (u_1, \ldots, u_K, p_1, \ldots, p_M)$$

the first part of encoded vector corresponds to the source vector $u = \{u_1, \ldots, u_K\}$, while the second part corresponds to the parity vector $p = \{p_1, \ldots, p_M\}$.

Regrettably, even if the parity-check matrix is sparse, the matrix P needs not be equally sparse. As a consequence, the encoding complexity grows as $\mathcal{O}(N^2)$ and it could be prohibitive if the blocklength gets very high.

3.5.2 Encoding using the Parity Check Matrix

In [61] Richardson and Urbanke proposed systematic LDPC codes with a low-complexity encoding algorithm. The efficiency of this encoder derives from the sparseness of the parity-check matrix H.

Triangular Encoder. The main idea is to work the source vector out in order to iteratively obtain the parity symbols. Assume that the matrix H is in the form $[B, T]$ where T is a triangular matrix with a full-filled diagonal as shown in the Figure 3.7. The first part

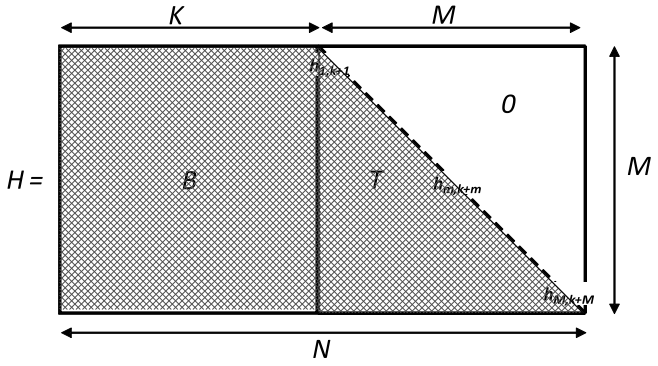

Figure 3.7: Equivalent parity-check matrix in lower triangular form

of the codeword is filled with the systematic part u. The parity part is determined using the *forward-substitution*, more precisely:

$$p_m = \begin{cases} h_{m,m+K}^{-1} \left(\sum_{i=1}^{K} h_{m,i} s_i \right) & \text{if } m = 1 \\ h_{m,m+K}^{-1} \left(\sum_{i=1}^{K} h_{m,i} s_i + \sum_{i=1}^{m-1} h_{m,\,i+K}\; p_i \right) & \text{if } 1 < m \leq M \end{cases} \tag{3.29}$$

Example 3.3. *This example can be useful for understanding this encoder type. Consider a NB-LDPC code defined by the parity-check matrix*

$$H = \left[\begin{array}{ccccc|ccccc} h_{1,1} & 0 & 0 & h_{1,4} & 0 & h_{1,6} & 0 & 0 & 0 & 0 \\ 0 & 0 & h_{2,3} & 0 & h_{2,5} & 0 & h_{2,7} & 0 & 0 & 0 \\ 0 & h_{3,2} & 0 & 0 & h_{3,5} & 0 & 0 & h_{3,8} & 0 & 0 \\ h_{4,1} & 0 & h_{4,3} & 0 & 0 & h_{4,6} & 0 & h_{4,8} & h_{4,9} & 0 \\ 0 & h_{5,2} & 0 & h_{5,4} & 0 & 0 & h_{5,7} & 0 & 0 & h_{5,10} \end{array}\right] \tag{3.30}$$

and by the graph in the Figure 3.8. The matrix H can be split in two sub-matrices: the matrix $B_{5\times5}$, on the left side, and the triangular matrix $T_{5\times5}$, on the right side. The encoding procedure starts by computing the first parity symbol $\mathrm{s}_6 = h_{1,6}^{-1}(h_{1,1}\mathrm{s}_1 \oplus h_{1,4}\mathrm{s}_4)$. Then, moving down on the diagonal, the remaining parity symbols are computed as linear combinations of the information symbols and the already computed parity symbols.

The graphical representation of the encoding procedure is illustrated in Figure 3.8.

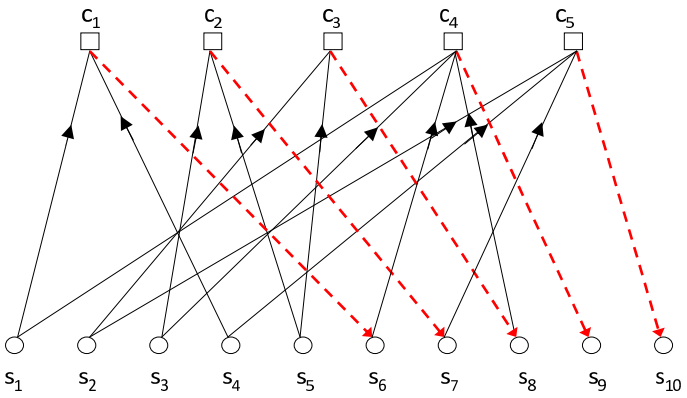

Figure 3.8: Tanner graph associated to the (10,5) code of the Example 3.3

Chapter 4

Monte-Carlo Estimation of the Asymptotic Performance

The asymptotic analysis is fundamental for the design of LDPC codes. In the previous chapter, we have described the Density Evolution technique. This technique allows to exactly compute the worst channel condition for which reliable transmission can be achieved by using a sufficiently long code for an ensemble of LDPC codes. However the exact computation is known only in some cases and for some channel models and sometimes at the cost of hard computational complexity and time. Some approximated methods have been proposed, e.g. the density evolution under Gaussian approximation, which can be applied for the AWGN channel model, for binary as well as for non-binary LDPC codes. This approximated method presents lower complexity but the approximated values may not be so close to real thresholds.

In this chapter we introduce an alternative method for the asymptotic analysis, based on the Monte-Carlo simulation of the decoding of infinite codes. Given a degree distribution pair associated to an ensemble of codes, this technique permits to estimate its threshold according to the used channel and the chosen decoding algorithm. The exact density evolution method provides thresholds only in two cases: for binary LDPC codes and for the non-binary LDPC codes transmitted over the BEC. Hence, we can compare these exact thresholds with the corresponding thresholds obtained from the Monte-Carlo Density-Evolution (MC-DE) method and we observe that this method is precise and accurate. The density evolution via Monte-Carlo simulation proves also to be a low-complexity and fast-enough method so that it can be used for LDPC code optimization.

Furthermore, in this chapter we present the Differential Evolution [69], a heuristic method used in order to optimize non-linear multi-dimension functions in many domains. This optimization method aims to solve a problem by iteratively improving the candidate solution even if the found solution can not be guaranteed to be the optimal one. In our case, the Differential Evolution can be coupled with the Monte-Carlo method, in order to optimize the degree distributions of symbol-nodes as well as the fractions of non-binary coefficients according to the alphabet definition of the code.

The chapter is organized as follows. In Section 4.1 we give the context of the work and we motivate the use of MC method. In Section 4.2 we describe the asymptotic analysis by using Monte-Carlo simulations. In Section 4.3.1 we compare the simulation results with the exact thresholds obtained via the density evolution for binary codes in AWGN channel. MC-approximated thresholds for non-binary codes in binary erasure channel are given in the Section 4.3.2; these results have been compared with the exact thresholds obtained by the density method of [56]. Section 4.3.3 presents the Monte-Carlo simulation results for non-binary codes for the AWGN channel, these results have been compared with the approximated results obtained by the density evolution under Gaussian approximation of [43]. In Section 4.4 we present an optimization method via the differential evolution. Finally, in Section 4.5, we conclude the chapter.

4.1 Motivations

Density Evolution method. As reported in the Section 3.3.2 (page 48), the Density Evolution method allows to asymptotically analyse the performance of an ensemble of LDPC codes transmitted over physically degraded channels and for a given decoding algorithm. In particular, this technique allows to determine the threshold probability of a given ensemble of LDPC codes corresponding the worst channel condition of a channel that allows a reliable transmission, assuming that the codelength goes to infinity. This method tracks the probability distribution function of the exchanged messages and recursively evaluates the error probability at a given decoding iteration for a certain channel condition.

Richardson and Urbanke provided two important theorems [59] for ensuring the correctness of this method:

1) The *concentration* theorem states that the performance of a randomly chosen LDPC code from the ensemble converges exponentially fast with the codelength to the ensemble average performance.

2) The *convergence to the cycle-free case* theorem states that the exchanged messages are considered to be independent because we assume that, at a fixed round, the decoding works in a tree-like.

We have already said that the computation of the exact threshold is quite hard and long to be reported in this thesis and the reader may refer to [59]. However, this analysis is valid for a large family of channels and decoding systems.

Unfortunately the Density Evolution method depends on the used alphabet in code definition. The exact computation is limited to the binary LDPC case study transmitted over memoryless symmetric channels. For the non-binary LDPC decoding, the exchanged messages are probability vectors with the same dimension of the alphabet size. Therefore, the non-binary decoding system as well as the density evolution result more complex.

As a consequence there are not any exact formulation of the density evolution except for a simple case represented by the binary erasure channel (BEC). In non-binary case, the BEC is the only exception for which we can exactly compute the threshold [56]. Rathi and Urbanke derived density evolution equations for the codes defined over the the General Linear Group $\mathrm{GL}_p(\mathbb{F}_2)$ [1]. A simplification comes from the fact that the message vectors of the same dimension have the same probabilities. As remarkable result, it has been proven that the threshold does not have a monotonous increasing with the alphabet order. We also note that the codes defined over the Galois Fields $\mathbb{F}_q$, with $q = 2^p$, may be seen as a particular case of codes defined over $\mathrm{GL}_p(\mathbb{F}_2)$ because the binary images of the field elements are also invertible. As a matter of fact, the derived equations for $\mathrm{GL}_p(\mathbb{F}_2)$ are also valid for codes defined over $\mathbb{F}_q$, if $p \leq 3$. Conversely, the thresholds will depend on the chosen primitive elements when the field order $p > 3$.

A similar method is proposed in [64]. The density evolution analysis is conducted taking into account also the labels of graph edges, which respect a certain distribution on the Galois Field $\mathbb{F}_q$; it results that the method is even more complicated due to the addition of another constraint parameter and because the messages of same dimension do

1. A non-binary LDPC codes is said to be defined over $\mathrm{GL}_p(\mathbb{F}_2)$ if each entry of the parity-check matrix is either 0 or an invertible $p \times p$ matrix (hence, belongs to $GL_p(\mathbb{F}_2)$).

not necessarily have the same probabilities. However, it has been pointed out that slight performance improvements are observed by using non-uniform edge label distributions.

As the Density Evolution converges slowly to the code thresholds and it is applicable only to some cases, faster methods have been introduced in order to find nearby thresholds for larger class of channels.

Density Evolution under Gaussian Approximation for NB-LDPC codes. In [43] Li *et al.* introduced Density Evolution under the Gaussian approximation assumption for the non-binary LDPC codes. This technique reduces the complexity at the cost of loss in the accuracy of the thresholds estimation. In fact, under this assumption, which states that the exchanged messages are normally distributed, it is possible to track only the expectations of the messages instead of the complete densities. In such a manner the size of the messages and then the whole complexity is greatly reduced. However, it has been proven that the messages after the constraint-node processing fail to satisfy the Gaussian assumption. This causes an inevitable loss of the accuracy in the evolution of the message expectations. As a consequence, the estimated threshold value may be away from the exact threshold. This work is an extension of [12, 42] done for the binary LDPC codes.

Monte-Carlo Density Evolution simulations. The density evolution via Monte-Carlo simulations (MC-DE) provides an alternative method for the density evolution estimation. In [14] it was pointed out that asymptotic thresholds could be estimated by simulating the messages exchanged within the iterative decoding of an *infinite* code of a given ensemble. The idea of simulating an infinite code is to render the exchanged messages statistically independent in order to respect the independence assumption.

The threshold estimation could be sufficient for the optimization purpose or in order to evaluate an ensemble of LDPC codes. It depends on the confidence one puts in this method. The goal of our work is to prove that the MC technique is a trusted method for estimating the code thresholds. In order to evaluate the accuracy of the MC method, we have compared the simulation results with different exact code thresholds, which are obtained by using the techniques proposed in the literature. We summarize it in the following.

1) **Binary case.** In Section 4.3.1 we present threshold estimations by Monte-Carlo simulations for the binary LDPC codes and we compare these thresholds with the exact thresholds computed by the using the exact Density Evolution equations [59].
2) **Non-binary case and BEC.** In order to evaluate the accuracy of the MC method in non-binary case, we have reported in Section 4.3.2 the analysis proposed in [53], in which the author compared the estimated thresholds by Monte-Carlo simulations and the exact thresholds [56] over the BEC.
3) **Non-binary case and AWGN channel.** In case of NB-LDPC codes transmitted over the AWGN channel we have already said that we can not compare the estimated thresholds with the exact ones. For this reason in Section 4.3.3 we present the simulation results of the MC-DE and we compare these thresholds with the thresholds obtained via Gaussian approximation [43].

At the end of these analysis we conclude that this method:
- highlights the convergence rapidity towards the threshold as the number of outgoing messages increases.
- is precise and accurate, especially when the size of the alphabet increases.

– is low-complex and fast-enough to allow code optimization

We deduce that the MC-DE method can be successfully applied for the non-binary case transmitted over a broad class of channels.

4.2 Description

In this section we describe the Monte-Carlo approximation of the Density Evolution of NB-LDPC codes. Consider an ensemble $\mathcal{E}(\lambda, \rho)$ of non-binary LDPC codes defined over an alphabet $\mathbb{F}_q$ with edge-perspective degree distribution λ and ρ.

We explain first the general idea. We remind that, for the AWGN channel, the threshold is represented by the noise standard deviation of the additive Gaussian noise. The considered channel degrades as the noise standard deviation increases. We simulate the transmission by generating directly the noisy input of the decoder. The decoder is the Belief-Propagation (Sum-Product) decoder, introduced in Section 3.2.1 (page 38), over random designed graphs according to the degree distribution pair (λ, ρ). Let us see the details of MC-DE method.

For symmetric channels we can assume that the all-zero codeword is transmitted [59]: this will simplify the algorithm. First we have to ensure that exchanged messages remain uncorrelated for a large number of iterations so that the independence assumption is respected. Since the density evolution is observed on a large but finite number of exchanged messages, we proceed as described below in order to decorrelate them.

Each iteration consists of two half iterations, one for the symbol-node processing and another one for the constraint-node processing. At each half iteration E messages pass over E outgoing edges from either symbol or constraint nodes.

We have illustrated an overview of the MC-DE method in Figure 4.2, and will consider this schema as a reference for the operations that we will describe in the following. In this figure we have represented the exchange of E messages over E edges as input of a box, which can not be considered as an interleaver because is not a bijective mapping. The degree distributions (edge perspective) are $\lambda(x) = \frac{1}{3}x + \frac{2}{3}x^3$ and $\rho(x) = x^5$. The designed code rate $r = 1/2$ according to (3.28). The outgoing symbol and constraint-node messages are respectively denoted as $\mathbf{P}_{n\to *}$ and $\mathbf{P}_{m\to *}$ and they are treated independently of the nodes toward they are sent. For the same reason, the edge labels are denoted by $h_{i,d}$, where d is an index depending on the node degree. In the following we describe the two half iterations.

– Consider for the instant that we have to compute an outgoing message from a symbol-node. For each symbol-node n we sample the node degree d_n according to the $\lambda(x)$. Then, $d_n - 1$ messages are randomly chosen from the E check-node outgoing messages $\mathbf{P}_{m\to *}$ produced at the previous half iteration. These messages are considered as being received by the node n on the $d_n - 1$ incoming edges. The $d_n - 1$ edge labels are also uniformly sampled from $\mathbb{F}_q^*$.
  Besides, in order to compute outgoing messages from symbol-nodes, we also need the knowledge of the a priori information corresponding to the channel output. In this case, for each new outgoing symbol-node message, the channel output is also resampled according to the channel model. The a priori information corresponding to the channel output is denoted by $\mathbf{P}_n$. The outgoing message is then computed according to the Eq. (3.10).
– In the same manner we operate for the outgoing messages from the constraint-node.

For each constraint-node m we sample the node degree d_m according to the $\rho(x)$. Then, $d_m - 1$ messages are randomly chosen from the E symbol-node outgoing messages $\mathbf{P}_{n\to *}$ produced at the previous half iteration. These messages are considered as being received by the node m on the $d_m - 1$ incoming edges. The $d_m - 1$ edge labels are also uniformly sampled from $\mathbb{F}_q^*$. The outgoing message is then computed according to the Eq. (3.8).

The decoding is successful iff the obtained codeword is formed by all zero symbols, or equivalently if the $\tilde{\mathbf{P}}_n = [1, 0, \ldots, 0]\ \forall n = 1, \ldots, E$. We iterate this procedure by changing σ according to a dichotomic search algorithm. The estimation is done by tightening the bounds in which the estimated threshold is included. Let $[\sigma_{\text{inf}}, \sigma_{\text{sup}}]$ be the interval in which the estimated value should be. Then, at each decoding iteration two cases can occur:

1) the decoding is successful then the algorithm increases the lower bound: $\sigma_{\text{inf}} = \sigma$,
2) conversely, the decoding in unsuccessful then the algorithm decreases the upper bound : $\sigma_{\text{sup}} = \sigma$.

Then the new value of σ is set to $\sigma = \frac{\sigma_{\text{inf}} + \sigma_{\text{sup}}}{2}$. This goes ahead till $\sigma_{\text{sup}} - \sigma_{\text{inf}} < \epsilon$, where $\epsilon > 0$ is an arbitrarily small value. In this way, at each round, the threshold is chosen from a smaller interval in order to increase the precision. The first time one can use for example $\sigma_{\text{inf}} \ll \sigma$ and $\sigma_{\text{sup}} \gg \sigma$. In Figure 4.1 we show an example of the dichotomic search.

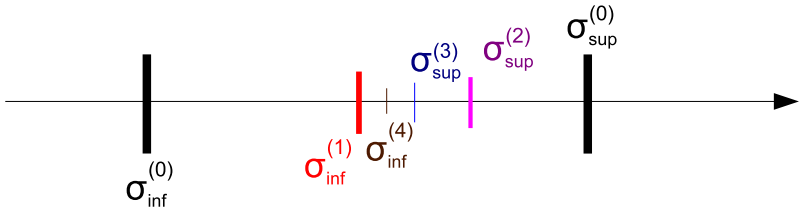

Figure 4.1: Example of the evolution of the dichotomic algorithm

Remark 4.1. *The number of the total incoming edges is greater than the number E of messages produced at each half iteration, then the incoming messages into the nodes are necessarily repeated. Indeed, the number of incoming edges for symbol-nodes is equal to $(d_{s,avg} - 1)E \geq E$; the number of incoming edges for constraint-nodes is equal to $(d_{c,avg} - 1)E > E$.*

Remark 4.2. *The exchanged messages are decorrelated thanks to three factors that change at each iteration:*

1) the resampled connections between symbol and check-nodes,
2) the resampled channel output and
3) the resampled edge labels, so that, especially for higher order finite fields, the exchanged messages result further uncorrelated.

In the following, we report the pseudo-coded steps of the Monte-Carlo algorithm.

Notation.

(N1) E = number of exchanged messages

(N2) R_{MAX} = maximum number of estimations. It needs for having a good statistic about the threshold estimation (e.g. $R_{\text{MAX}} = 100$).

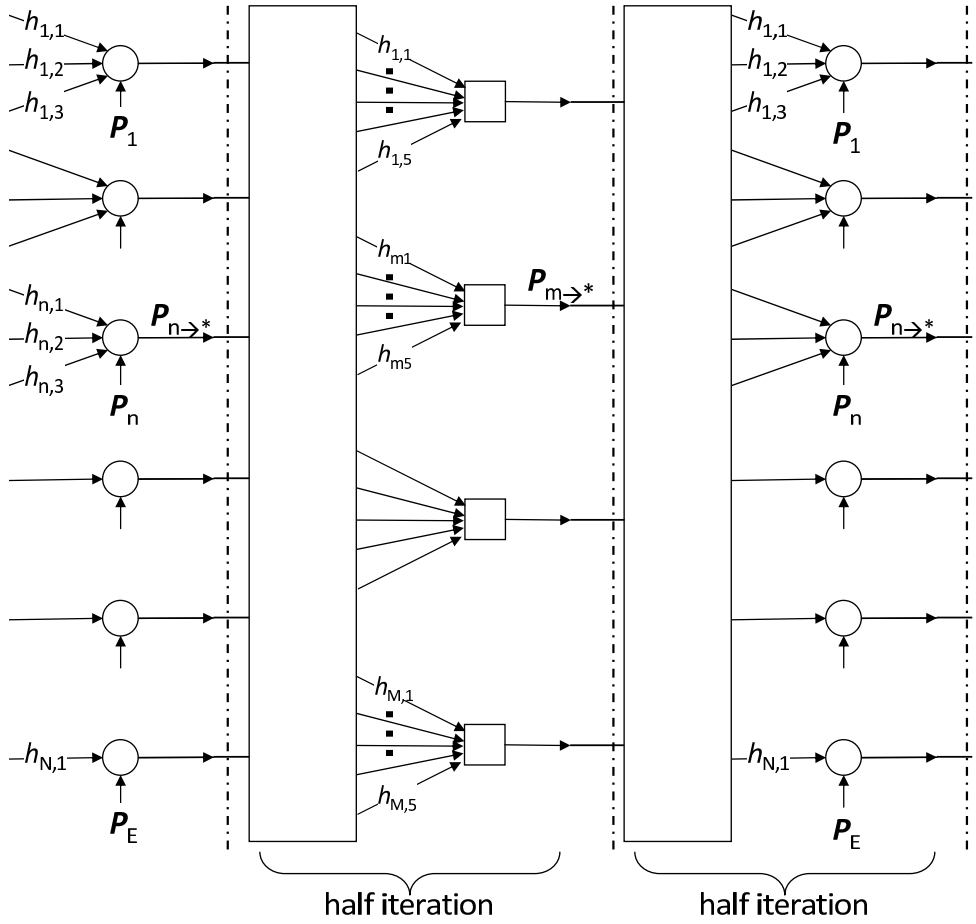

Figure 4.2: Density evolution approximation by Monte-Carlo simulation

(N3) I_{MAX} = maximum number of decoding iterations.
There exists a tradeoff between the speed of convergence (given by a smaller I_{MAX}) and the accuracy of the estimation (given by a greater I_{MAX}). From the analysis in the next section we choose $I_{\mathrm{MAX}} = 200$ or 500.

(N4) σ = estimated threshold (standard deviation of AWGN Channel)

(N5) σ_{inf} = threshold inferior bound

(N6) σ_{sup} = threshold superior bound

(N7) σ_{avg} = average threshold

(N8) ϵ = threshold precision (e.g. $1E-7$)

Initialization.

- $\mathbf{P}_{m\to *} = \left[\frac{1}{q}, \ldots, \frac{1}{q}\right] \ \forall m = 1, \ldots, E.$
- $\sigma_{\mathrm{avg}} = 0$
- $\sigma = \frac{\sigma_{\mathrm{inf}} + \sigma_{\mathrm{sup}}}{2}$

Execution.

for #thresholds = 1 to R_{MAX}

do

for #iterations = 1 to I_{MAX}

First half iteration.

1) Set the symbol-node degrees according to $\lambda(x)$ distribution.
2) Random choose of the incoming messages among the E messages $\mathbf{P}_{m\to *}$ produced by the check-node processing of the previous half iteration.
3) Sample the edge labels according to $\xi(x)$ distribution [2] on $\mathbb{F}_q^*$.
4) Sample the channel output $\{\mathbf{P}_n\}_{n=1,\ldots,N} = [y_{n,1}\frac{2}{\sigma^2}, \ldots, y_{n,p}\frac{2}{\sigma^2}]$.
5) Compute the messages $\mathbf{P}_{n\to *}$ according to the Eq. (3.10) of the BP algorithm.

Second half iteration.

1) Set the constraint-node degrees according to $\rho(x)$ distribution.
2) Random choose of the incoming messages among the E messages $\mathbf{P}_{n\to *}$ produced by the symbol-node processing of the previous half iteration.
3) Sample the edge labels according to $\xi(x)$ distribution on $\mathbb{F}_q^*$.
4) Compute the messages $\mathbf{P}_{m\to *}$ according to Eq. (3.8) of the BP algorithm.

End.

Compute the a posteriori probabilities according to Eq.(3.11) of the BP algorithm.

if $\tilde{\mathbf{P}}_n = [1, 0, \ldots, 0] \quad \forall n = 1, \ldots, E \Longrightarrow \{\hat{s}_1, \ldots \hat{s}_N\} = \{0, \ldots, 0\} \Longrightarrow \sigma_{\mathrm{inf}} = \sigma$

else $\tilde{\mathbf{P}}_n \neq [1, 0, \ldots, 0] \quad \forall n = 1, \ldots, E \Longrightarrow \{\hat{s}_1, \ldots \hat{s}_N\} \neq \{0, \ldots, 0\} \Longrightarrow \sigma_{\mathrm{sup}} = \sigma$

Set the new threshold: $\sigma = \dfrac{\sigma_{\mathrm{sup}} + \sigma_{\mathrm{inf}}}{2}$

While - Stop Criteria: $(\sigma_{\mathrm{sup}} - \sigma_{\mathrm{inf}}) < \epsilon$ (check if σ has the requested precision)

$$\sigma_{\mathrm{avg}} = \frac{[\sigma_{\mathrm{avg}}(\#\mathrm{thresholds} - 1)] + \sigma}{\#\mathrm{thresholds}}$$

End.

Finally, the approximated threshold value is equal to the average value $\sigma = \sigma_{\mathrm{avg}}$ obtained after R_{MAX} estimations.

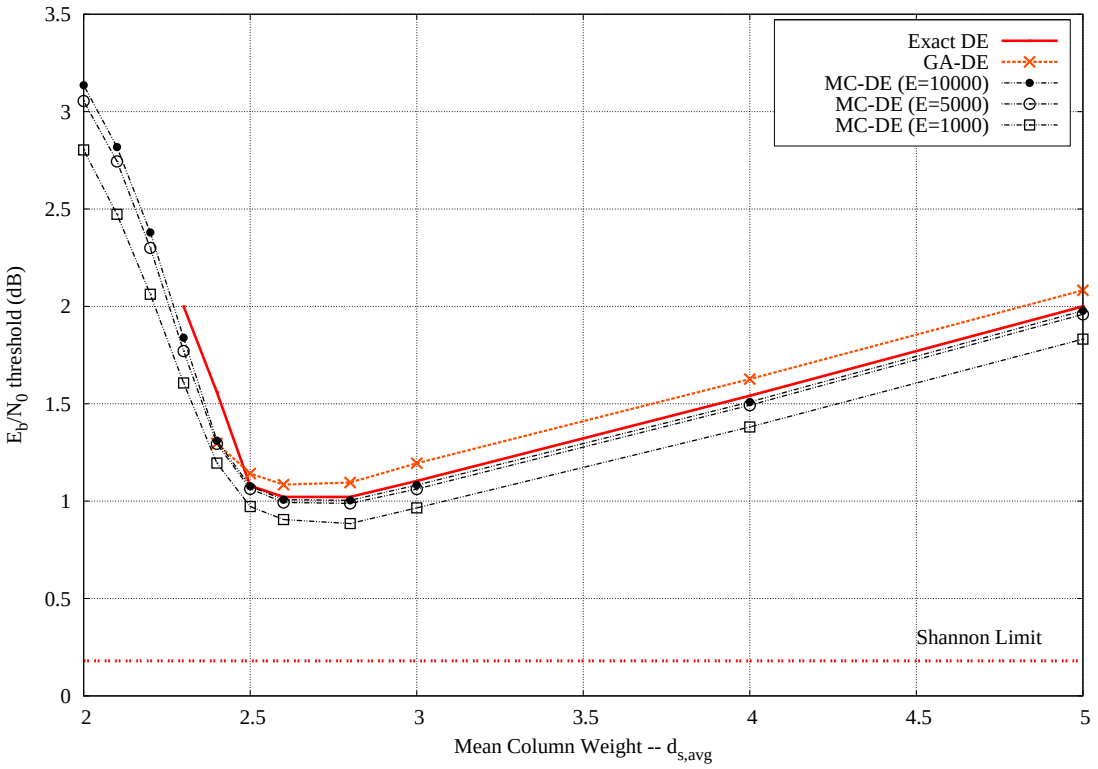

Figure 4.3: Thresholds for *semi-regular* binary LDPC codes with rate $r = 1/2$ — Comparison of the curves obtained with MC-DE, GA-DE and Exact DE

4.3 Monte-Carlo Simulation Results

The goal of this section is to demonstrate that the Monte-Carlo is an accurate and precise method for the estimation of the DE thresholds. This section is motivated by the fact that we would like to extend this method also to the non-binary case transmitted in the AWGN channel, for which the asymptotic code performance are unknown. In the following σ^* will denote the noise standard deviation that one desires to estimate *i.e.,* the exact threshold, whereas σ is the estimated threshold value.

4.3.1 Threshold Comparison for the Binary LDPC Codes

We start with the analysis of binary LDPC codes with regular or *semi-regular* profile. For these codes we have analyzed the obtained thresholds according to some variations of parameters, as the number of outgoing edges and the maximum number of decoding iterations. Then we have pursued with the analysis of irregular LDPC codes that, in general, outperform the regular codes. In the following, since we will deal with only binary LDPC codes, the symbol-nodes are simply referred to as bit-nodes.

Effect of the number of the outgoing edges E. In Figure 4.3 we present the results of Monte-Carlo simulations. The thresholds on the ordinate are reported as logarithmic values in terms of $\frac{E_b}{N_0}$ [3]. Hence, the best thresholds correspond to the lowest point of these

3. $\sigma^2 = 2r\frac{E_b}{N_0}$, if we assume a transmission in 2-QAM.

curves.

We have simulated binary codes with different code lengths over the AWGN channel. The design coding rate is $r = 0.5$. In particular, in order to conduct comparisons with the results of the binary codes of [43], the simulated codes are *semi-regular*, meaning that the bit-node degrees take only on two consecutive values. So that, in this section we have referred to LDPC codes that have concentrated degree distribution pair ($L(x)$,$R(x)$) around their average degree pair ($d_{\text{s,avg}}$, $d_{\text{c,avg}}$). We can refer $d_{\text{s,avg}}$ as the weight column average of the parity-check matrix; let $d = \lfloor d_{\text{s,avg}} \rfloor$ be the the nearest integer below $d_{\text{s,avg}}$, then:

$$\begin{cases} d_{\text{s,avg}} = \lambda_d d + \lambda_{d+1}(d+1) \\ \lambda_d + \lambda_{d+1} = 1 \end{cases} \tag{4.1}$$

where λ_d and λ_{d+1} are the fractions associated to the degree d and $d+1$. The ρ distribution is derived by knowing the symbol-node degree distribution and the rate according to the formula given in the previous chapter Eq. (3.28).

We have conducted comparisons with the exact thresholds and the approximated thresholds. We have plotted the thresholds as function of $d_{\text{s,avg}}$ that varies from 2 to 5 for several code lengths. The GA-DE threshold results are obtained from the column "$q = 2$" of the Table II of [43].

Consider first the MC-DE results. An unexpected phenomenon is that the curves approach the real threshold values from down to the top as the number of the considered outgoing edges increases. Surprisingly, with a smaller number of exchanged messages (e.g $E = 1000$), the estimated threshold is better than the estimated thresholds obtained by simulating a greater number of messages (e.g. $E = 10000$), which should induces more independence of the messages. However, the values closest to the real thresholds correspond to the simulations with an increasing number of messages.

In Table 4.1 we detail the values of the Figure 4.3. For understanding the accuracy of the estimated values we have computed the Root Mean Square Error (RMSE). The RMSE obtained with the Gaussian approximation evaluation is ∼0.1dB , whereas already

$d_{\text{s,avg}}$	Exact-DE $\frac{E_b}{N_0}$	GA-DE $\frac{E_b}{N_0}$	Monte-Carlo Density Evolution $\frac{E_b}{N_0}$				
			E = 5000	E = 10000	E = 20000	E = 100000	E = 200000
2	-	-	3.0545	3.1360	3.2247	3.3168	3.3418
2.1	-	-	2.7441	2.8181	2.8953	2.9922	3.0359
2.2	-	-	2.3002	2.3796	2.4658	2.5690	2.6049
2.3	2.0003	-	1.7708	1.8390	1.8588	1.9895	2.0300
2.4	1.5592	1.2969	1.2945	1.3102	1.3314	1.3577	1.5592
2.5	1.0788	1.1420	1.0631	1.0764	1.0901	1.0924	1.0943
2.6	1.0217	1.0847	0.9934	1.0075	1.0168	1.0261	1.0290
2.8	1.0216	1.0956	0.9883	1.0042	1.0082	1.0223	1.0235
3	1.1036	1.1947	1.0624	1.0825	1.0944	1.1015	1.1038
4	1.5411	1.6268	1.4927	1.5091	1.5249	1.5366	1.5379
5	2.0000	2.0827	1.9594	1.9778	1.9890	2.0037	2.0070
RMSE		0.0991	0.0811	0.0571	0.0500	0.0038	0.0024

Table 4.1: Thresholds for *semi-regular* binary LDPC codes with rate $r = 1/2$ — Analysis of the number of exchanged messages in Monte-Carlo simulations

the MC-DE method with $E = 10000$ halves this result. In particular, closer and closer results have been obtained as the number E increases. We have simulated a maximum number of $E = 200000$ exchanged messages — even if this choice is not practicable because of the simulation time — it demonstrates that this method permits to reduce the error as far we are willing to spend in resources. Regrettably, the accuracy of this method does not become linearly better with the value E. Nevertheless, we can reach the same standard requirement of the Gaussian Approximation method by simulating the MC with E less than 5000 exchanged messages.

We can observe that the thresholds depend on the used distribution and the best threshold is obtained for $d_{\text{s,avg}} = 2.8$ (with a node-perspective bit-node degree distribution $L(x) = 0.8x^2 + 0.2x^3$).

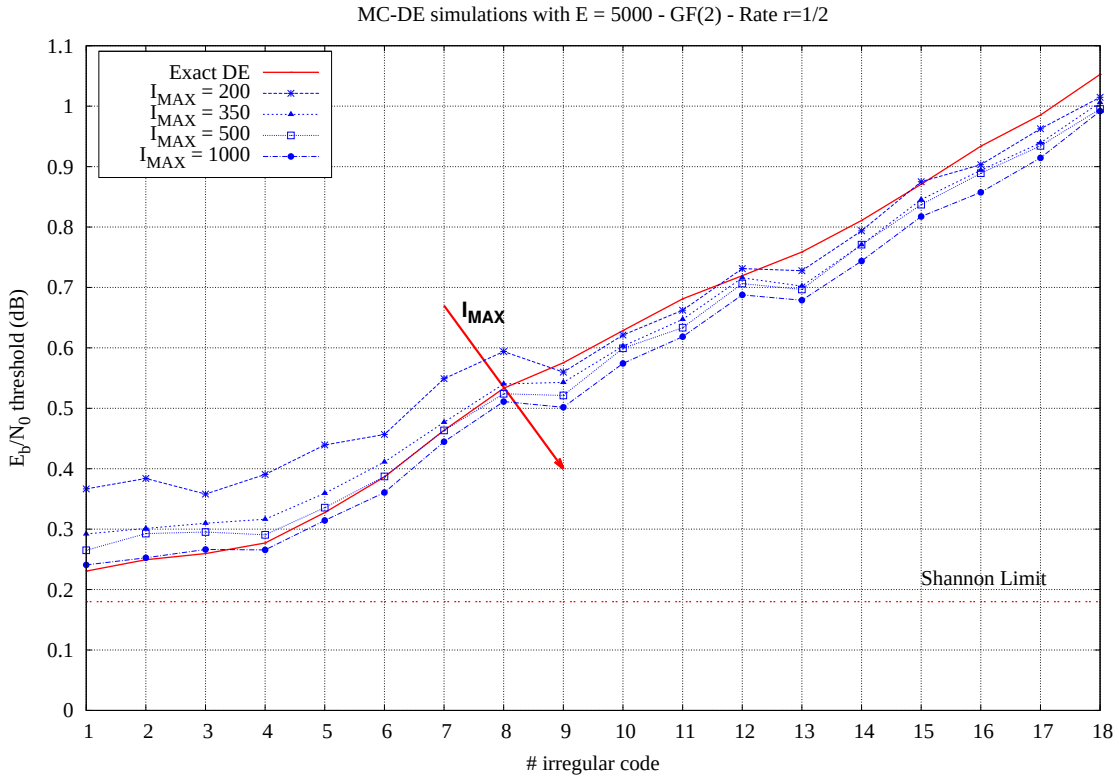

Figure 4.4: Thresholds for binary irregular LDPC codes with rate $r = 1/2$ — Analysis of the maximum number of iterations

Effect of the maximum number of iterations. We have simulated the Density Evolution via Monte-Carlo for 18 binary codes with irregular profiles and rate $r = 1/2$. At each half iteration of the Monte-Carlo simulations, $E = 5000$ outgoing messages are observed. In Figure 4.4 it has been plotted the average values of MC-DE thresholds (in an increasing order) with different number of maximum decoding iterations I_{MAX} varying from 200 up to 1000.

The positive effect of increasing the maximum number of iterations is to lower the threshold values. However, we can observe an inverse trend for each curve from a cut-off point: they pass to stay up to go below the real threshold curve (in red). In general, one can accept a common maximum number of iteration I_{MAX} equal to 500. However, for codes

that demonstrate thresholds 0.4dB away from the Shannon limit, a maximum number of iteration I_{MAX} equal to 200 can be sufficient. Also for this reason, we have used this last value in the evaluations done in Section 4.3.3.

In Table 4.2 we have written the codes used for this analysis. We have obtained these distributions from a software implemented on-line [1]. For our purpose we have limited the maximum symbol-node degree $d_{\mathrm{s,max}}$ to 50. The first code has a threshold equal to 0.2305dB (away only 0.0435dB from the Shannon capacity).

code number	edge perspective degree distribution $\lambda(x)$ - Rate $r = 1/2$
1	$0.1904x^2 + 0.1778x^3 + 0.1066x^6 + 0.0911x^7 + 0.0701x^{14} +$ $0.1007x^{15} + 0.2633x^{50}$
2	$0.2123x^2 + 0.1977x^3 + 0.0660x^6 + 0.1688x^7 + 0.0305x^{15} + 0.1044x^{16} +$ $0.0225x^{17} + 0.0014x^{28} + 0.0471x^{37} + 0.1240x^{38} + 0.0161x^{42} + 0.0092x^{43}$
3	$0.2098x^2 + 0.2108x^3 + 0.0106x^4 + 0.0198x^5 + 0.0357x^6 +$ $0.1066x^8 + 0.1338x^9 + 0.2729x^{30}$
4	$0.2005x^2 + 0.1866x^3 + 0.1038x^6 + 0.0841x^7 + 0.0569x^8 + 0.3682x^{30}$
5	$0.2521x^2 + 0.2201x^3 + 0.0000x^4 + 0.0611x^5 + 0.1434x^6 +$ $0.0003x^7 + 0.3228x^{15}$
6	$0.2725x^2 + 0.2376x^3 + 0.0704x^4 + 0.4195x^{10}$
7	$0.2956x^2 + 0.2482x^3 + 0.2952x^6 + 0.1610x^{15}$
8	$0.2959x^2 + 0.3531x^3 + 0.3510x^8$
9	$0.2949x^2 + 0.1970x^3 + 0.5081x^7$
10	$0.4838x^3 + 0.0002x^{13} + 0.2186x^{15} + 0.0001x^{49} + 0.2972x^{50}$
11	$0.3380x^2 + 0.1288x^3 + 0.5332x^5$
12	$0.6420x^3 + 0.2893x^{16} + 0.0687x^{17}$
13	$0.2591x^2 + 0.1336x^3 + 0.6073x^8$
14	$0.3500x^2 + 0.1500x^3 + 0.5000x^4$
15	$0.3667x^2 + 0.4070x^3 + 0.0481x^4 + 0.1783x^5$
16	$0.2168x^2 + 0.6556x^3 + 0.1275x^{20}$
17	$0.8572x^3 + 0.1417x^{12} + 0.0011x^{14}$
18	$0.7500x^3 + 0.2500x^7$

Table 4.2: Edge-Perspective degree distributions of the codes used in Figure 4.4

Comparison of the threshold estimations. We have estimated the thresholds of several binary LDPC codes with Monte-Carlo method and choosing the right parameters according to the above analysis. Indeed, we have given a hint about the number of exchanged messages over the outgoing edges, here we give an deeper analysis of the accuracy irrespective of those parameters.

Table 4.3 presents a comparison of thresholds obtained by Monte-Carlo Density-Evolution (MC-DE) and those obtained by exact density evolution, for irregular binary LDPC codes with rate $r = 1/2$ over the AWGN channel. Thresholds are shown both in terms of noise variance and corresponding $\frac{E_{\mathrm{b}}}{N_0}$ value. In Figure 4.5 we represented the values of Table 4.3 and we can observe that the Monte-Carlo simulation results are closer to the exact thresholds than the approximated results obtained by using the Gaussian Approximation computation.

$d_{\mathrm{s,max}}$	Exact-DE		GA-DE		MC-DE		σ_{dev}
	σ^*	E_b/N_0^*	σ	E_b/N_0	σ	E_b/N_0	$\times 10^3$
4	0.911	0.808	0.904	0.876	0.916	0.757	1.2
5	0.919	0.730	0.911	0.806	0.923	0.691	1.1
6	0.930	0.627	0.907	0.774	0.931	0.622	1.0
7	0.942	0.515	0.922	0.699	0.935	0.577	1.7
8	0.950	0.448	0.938	0.557	0.947	0.468	1.5
9	0.954	0.409	0.940	0.533	0.954	0.411	1.1
10	0.956	0.393	0.942	0.523	0.955	0.403	1.2
11	0.957	0.380	0.942	0.518	0.957	0.377	1.1
12	0.958	0.373	0.942	0.513	0.958	0.367	0.8
15	0.962	0.335	0.944	0.503	0.962	0.340	1.6
20	0.965	0.310	0.946	0.482	0.963	0.325	0.7
30	0.969	0.273	0.946	0.469	0.965	0.299	1.6
50	0.972	0.248	0.952	0.428	0.971	0.260	0.8
RMSE			0.0168	0.1459	0.0031	0.0272	

Table 4.3: Real and approximated thresholds for binary irregular LDPC codes with maximum symbol-node degree $d_{\mathrm{s,max}}$

The maximum bit-node degree is reported in the left column; the corresponding degree distribution polynomials $\lambda(x)$ and $\rho(x)$ can be found in [60], Tables I and II. For comparison purposes, we have also included threshold values computed by using the Gaussian-Approximation (GA-DE) proposed in [12]. At each half-iteration of the MC-DE simulation, $E = 10000$ outgoing messages are computed, and a maximum number of $I_{\mathrm{MAX}} = 500$ iterations are executed.

For each pair of degree distributions (λ, ρ), several threshold estimations have been performed; the mean value is reported in the MC-DE/σ column, and the standard deviation ($\times\ 10^3$) is reported in the last column. As it can be observed, the standard deviation σ_{dev} is between 0.0008 and 0.0017; hence we conclude that the proposed method is precise. Furthermore, the Root Mean Square Error (RMSE) of estimated thresholds (reported on the last row) is equal to 0.0031, which proves that the proposed method is also accurate.

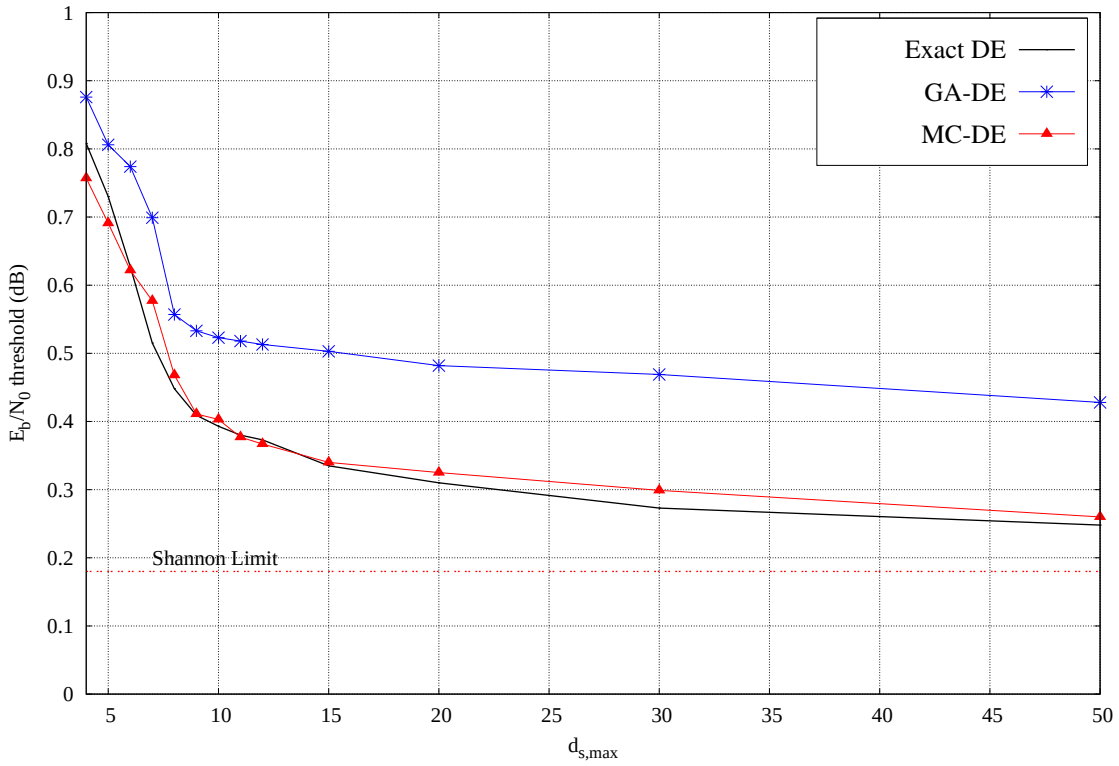

Figure 4.5: Real and approximated thresholds for binary irregular LDPC codes of Table 4.3

4.3.2 Threshold Comparison for the NB-LDPC Codes over the BEC

In this section we deal with the asymptotic analysis of the LDPC codes over the binary erasure channel. We compare the approximated thresholds of [53] and the exact thresholds obtained with the technique of [56]. We analyze two non-binary cases, one for the regular codes and the other one for the irregular codes. Note that the σ parameter in this case corresponds to the channel erasure probability (we use the same symbol σ to keep uniform notation throughout the chapter; the E_b/N_0 is also shown as a basis for comparison on a logarithmic scale).

Regular LDPC codes. We consider regular (2,4)-LDPC codes with coding rate $r = 1/2$ defined over $\mathbb{F}_q$, with $q = 2, 4, 8, 16, 32$, and 64. The estimated thresholds are presented in Table 4.4.

Alphabet	Exact-DE		MC-DE		σ_{dev}
	σ^*	E_b/N_0^*	σ	E_b/N_0	$\times 10^4$
$\mathbb{F}_2$	0.3333	9.5432	0.3379	9.4242	12.4
$\mathbb{F}_4$	0.4096	7.7528	0.4082	7.7825	8.7
$\mathbb{F}_8$	0.4506	6.9241	0.4507	6.9222	4.0
$\mathbb{F}_{16}$	0.4680	6.5951	0.4682	6.5913	1.9
$\mathbb{F}_{32}$	0.4742	6.4807	0.4744	6.4771	1.7
$\mathbb{F}_{64}$	0.4746	6.4734	0.4744	6.4771	1.7
RMSE			0.0020	0.0502	

Table 4.4: Real and approximated thresholds for regular (2,4)-LDPC defined over $\mathbb{F}_2$ $\mathbb{F}_4$, $\mathbb{F}_8$, $\mathbb{F}_{16}$, $\mathbb{F}_{32}$ and $\mathbb{F}_{64}$, and transmitted over the BEC

Irregular LDPC codes. In this part we give the results of 4 optimized LDPC codes of rate 1/2. Each of the codes has been optimized for an alphabet definition. The edge-perspective degree distributions are reported in the Table 4.6.

The simulation results are shown in Table 4.5.

Alphabet	Exact-DE		MC-DE		σ_{dev}
	σ^*	E_b/N_0^*	σ	E_b/N_0	$\times 10^4$
$\mathbb{F}_2$	0.4955	6.0991	0.4938	6.1289	5.2
$\mathbb{F}_4$	0.4926	6.1501	0.4892	6.2102	10.1
$\mathbb{F}_8$	0.4931	6.1412	0.4920	6.1606	2.6
$\mathbb{F}_{16}$	0.4945	6.1166	0.4939	6.1272	2.0
RMSE			0.0020	0.0353	

Table 4.5: Real and approximated thresholds for irregular LDPC defined over $\mathbb{F}_2$ $\mathbb{F}_4$, $\mathbb{F}_8$ and $\mathbb{F}_{16}$, and transmitted over the BEC

We can observe that, in both cases, the estimated thresholds are very close to the exact ones. It can be also observed that the standard deviation σ_{dev} of the estimated thresholds becomes even smaller for increasing alphabet order (in these tables $\sigma_{\text{dev}} \times 10^{4!}$):

the standard deviation is between 0.00017 and 0.0012 that demonstrates that this method is very precise. We conclude that the MC-DE is also more precise compared to the binary case. Finally, the last row of these tables report the RMSE = 0.0020, in both cases. This confirms that this method is very accurate.

This section reinforces what we said before: the estimation performance improves as the alphabet size increases, for both the regular and the irregular cases. Therefore, we can trust the MC-DE method can be successfully applied to the non-binary LDPC transmitted over the AWGN channel.

Alphabet	edge-perspective degree distributions - Rate $r = 1/2$
$\mathbb{F}_2$	$\lambda(x) = 0.26328x + 0.1802x2 + 0.27x^6 + 0.28649x^{29}$ $\rho(x) = 0.63407x^7 + 0.36593x^8$
$\mathbb{F}_4$	$\lambda(x) = 0.457504x + 0.278481x^3 + 0.001328x^{10} + 0.262453x^{11}$ $\rho(x) = 0.729x^5 + 0.271x^6$
$\mathbb{F}_8$	$\lambda(x) = 0.548406x + 0.055368x^2 + 0.201158x^4 + 0.193247x^{11}$ $\rho(x) = 0.244x^4 + 0.756x^5$
$\mathbb{F}_{16}$	$\lambda(x) = 0.596x + 0.186x^4 + 0.071x^7 + 0.147x^{17}$ $\rho(x) = 0.284x^4 + 0.716x^5$

Table 4.6: Edge-Perspective degree distributions of the codes used in Table 4.5

4.3.3 Threshold Estimation of NB-LDPC Codes over the BI-AWGN Channel

In this section we show the asymptotic analysis of ensembles of codes transmitted under binary input AWGN channel.

As said, the exact threshold computation have not been accomplished because of the impracticability of finding a computing method for NB-LDPC codes transmitted over the AWGN channel. Therefore, in this case we are not able to evaluate how precise and accurate the approximated thresholds would be without a basis for a comparison. However we can analyse the Monte-Carlo simulation results with respect to the results of the Density Evolution under Gaussian Approximation. We have simulated ensembles of semi-regular codes with rate $r = 1/2$ in order to compare them with the approximated thresholds reported in [43], Table II.

The Monte-Carlo simulations have as parameters $E = 20000$ outgoing messages that are observed at each half iteration and a maximum number of decoding iterations I_{MAX} equals to 200. In Section 4.3.1 we have studied the case of the maximum number of iterations in the binary case and we accept $I_{\mathrm{MAX}} = 200$ for thresholds 0.4dB away from the Shannon limit. Also in the case on non-binary LDPC we have chosen $I_{\mathrm{MAX}} = 200$: indeed the decoding of non-binary LDPC codes is less penalized from a smaller maximum number of iterations. We have justified this by simulating codes with $d_{\mathrm{s,avg}} = 2.2$ and two different $I_{\mathrm{MAX}} = 200$ and 500. The difference between the two thresholds is only 0.01dB as reported in Table 4.7. Then, the choice of $I_{\mathrm{MAX}} = 200$ makes the simulation faster with an acceptable precision.

Figure 4.6 draws the Monte-Carlo simulation results. In this figure, we observe that the thresholds values are plotted as function of the alphabet size q and the average symbol-node degree $d_{\mathrm{s,avg}}$. For all the q values, the best results are provided by $d_{\mathrm{s,avg}} \lesssim 3$ and

$d_{\text{s,avg}}$	MC-DE – $E = 20000$ – GF(64)					
	$I_{\text{MAX}} = 200$			$I_{\text{MAX}} = 500$		
	σ	$\sigma \times 10^3$	$\frac{E_b}{N_0}$	σ	$\sigma \times 10^3$	$\frac{E_b}{N_0}$
2.2	0.938	0.4	0.553	0.939	0.3	0.542

Table 4.7: Comparison of two different Monte-Carlo thresholds for $d_{\text{s,avg}} = 2.2$ with $I_{\text{MAX}} = 200$ and 500

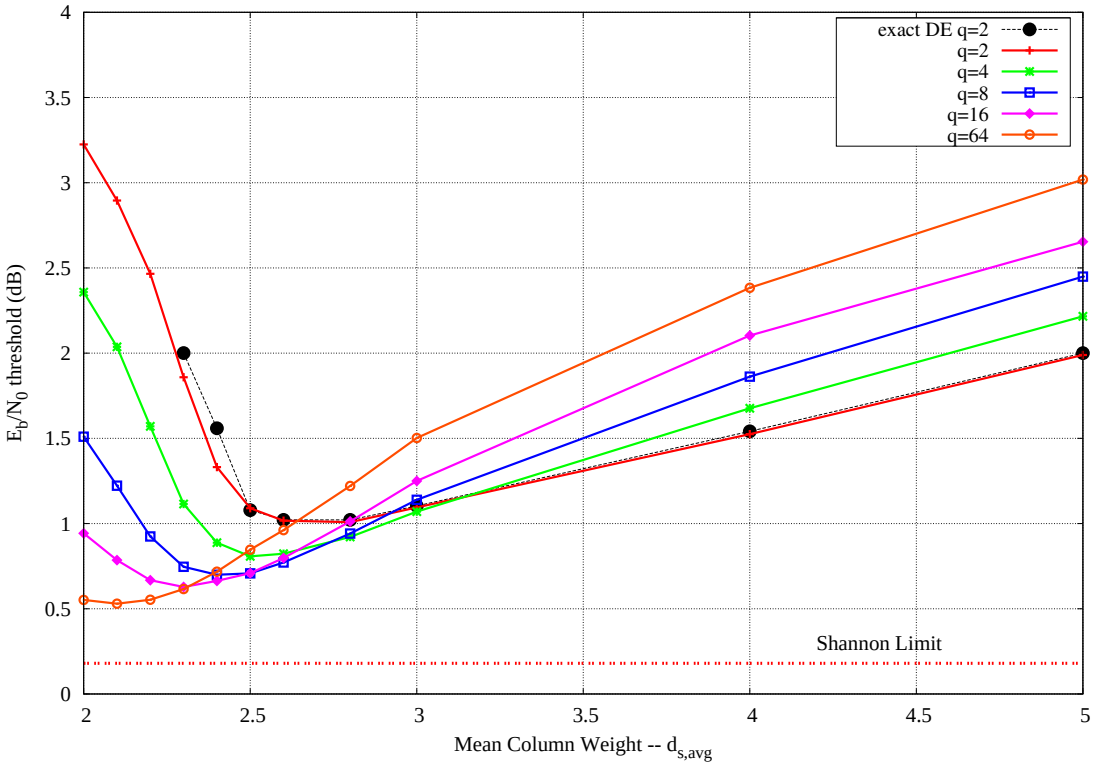

Figure 4.6: Threshold results obtained via MC-DE for NB-LDPC codes with rate $R = 1/2$

increasing q. For each alphabet size, the value $d_{\text{s,avg}}$ for which it has been observed the best threshold decreases from $d_{\text{s,avg}} = 2.8$ for the binary case till $d_{\text{s,avg}} = 2.1$ for the ensemble of codes defined over $\mathbb{F}_{64}$. We have also observed that for $d_{\text{s,avg}} > 3$ the better results correspond to smaller value of q.

$d_{\mathrm{s,avg}}$	$\mathbb{F}_2$ $\frac{E_\mathrm{b}}{N_0}$ [dB]		$\mathbb{F}_4$ $\frac{E_\mathrm{b}}{N_0}$ [dB]		$\mathbb{F}_8$ $\frac{E_\mathrm{b}}{N_0}$ [dB]		$\mathbb{F}_{16}$ $\frac{E_\mathrm{b}}{N_0}$ [dB]		$\mathbb{F}_{64}$ $\frac{E_\mathrm{b}}{N_0}$ [dB]	
	GA-DE	MC-DE	GA-DE	MC-DE	GA-DE	MC-DE	GA-DE	MC-DE	GA-DE	MC-DE
2	-	3.2247	-	2.3583	-	1.5097	0.8096	0.9428	0.2422	0.5518
2.1	-	2.8953	-	2.0371	-	1.2223	0.5745	0.7854	0.2003	0.5301
2.2	-	2.4658	-	1.5705	0.7668	0.9243	0.4300	0.6678	0.2511	0.5531
2.3	-	1.8588	1.0171	1.1146	0.6224	0.7464	0.3999	0.6280	0.3582	0.6157
2.4	1.2969	1.3314	0.8689	0.8876	0.5819	0.6994	0.4437	0.6637	0.4996	0.7174
2.5	1.1420	1.0901	0.8173	0.8077	0.6070	0.7077	0.5301	0.7091	0.6650	0.8469
2.6	1.0847	1.0168	0.8211	0.8230	0.6631	0.7714	0.6210	0.7979	0.7801	0.9612
2.8	1.0956	1.0082	0.9161	0.9215	0.8335	0.9413	0.8431	1.0125	1.0479	1.2209
3	1.1947	1.0944	1.0769	1.0699	1.0464	1.1393	1.0956	1.2502	1.3455	1.5021
4	1.6268	1.5249	1.6741	1.6763	1.7768	1.8619	1.9273	2.2257	2.2477	2.3833
5	2.0827	1.9890	2.2117	2.2159	2.3700	2.4489	2.5487	2.6537	2.8799	3.0187

Table 4.8: Thresholds for *semi-regular* NB-LDPC codes — Density Evolution under Gaussian Approximation and Density Evolution via Monte-Carlo simulations

In Table 4.8 we report the threshold results for $\mathbb{F}_2$, $\mathbb{F}_4$, $\mathbb{F}_8$, $\mathbb{F}_{16}$ and $\mathbb{F}_{64}$. For each alphabet size there are two columns, one for the DE under Gaussian Approximation (GA) and the other one for the DE under Monte-Carlo simulations (MC). Concerning the MC-DE we have omitted to write the standard deviations σ_{dev} for a matter of space, however, we can affirm that these values are all less than 0.002.

We can observe that the approximated thresholds obtained with the two methods diverge as the alphabet size increases.

In conclusion of this analysis, it can be noticed that the performance of a code approaches the Shannon limit as the alphabet size increases. This is the same conclusion of the authors in [43], but the main difference is that we think that the threshold for $\mathbb{F}_{64}$ is no so close to the Shannon limit as predicted by the GA-DE method.

For a rate $r = 1/2$ and with an increasing alphabet size, we can consider that the regular (2,4)-LDPC codes present the best performance. These empirical results have been proven also in [13, 63], where it has been pointed out that when the size of the alphabet grows, best decoding thresholds are obtained for average density of edges closer and closer to $d_s = 2$.

Practically, for NB-LDPC codes defined over Galois fields $\mathbb{F}_q$ with $q > 64$, best codes, both asymptotically and at finite lengths, are ultra-sparse codes, as demonstrated also in our simulation results.

Remark 4.3. *The distributions of edge labels over $\mathbb{F}_q^*$ can also be optimized in order to have better code thresholds. However this improvement seems to be minimal [64]; therefore in this work we have considered that the edge labels are uniformly distributed over $\mathbb{F}_q^*$.*

4.4 Optimization with the Differential Evolution

In this section we introduced the *Differential Evolution* technique, originally due to Storn and Price [69] and belonging to the class of genetic algorithms. The Differential Evolution is a heuristic method for optimizing multidimensional real-valued functions, which are not necessarily continuous or differentiable. Moreover, this technique does not use the gradient of the function as required by classic optimization methods, thus it can be used on a very large optimization problems that are difficult or absolutely not analytically solvable. Conversely, it can provide approximated solutions and it has not be proven the convergence to an unique solution.

This technique is used in some search algorithms or in many fields that need optimization in which the underlying problems result complicated and the mathematical knowledge are not sufficient. For instance, it is used in genetic optimization algorithm with great success; in general the efficiency varies according to the application field.

In code optimization, this technique can be for example coupled with the Monte-Carlo method in order to optimize the node degree distribution or the entry values of the parity-check matrix. Moreover, in the Chapter 5 we will use this technique for optimizing the puncturing distribution; it emerges a straightforward improving in the performance of the optimized punctured codes.

4.4.1 Description

This method creates candidate solutions, drawn from a predefined population, by combining existing solutions in a linear map. It keeps the best solution according to the eval-

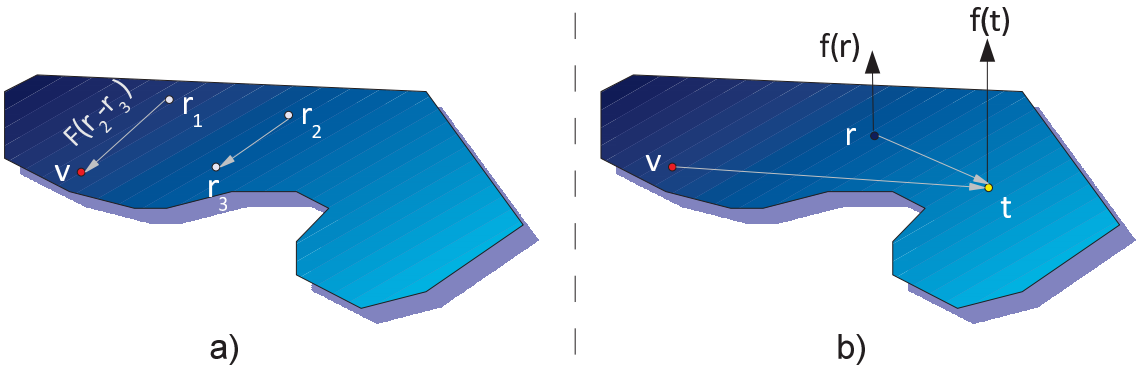

Figure 4.7: Differential Evolution

uation of the cost function. In this way the objective function is treated as a black-box, which provides only the quality measure of the candidate solution.

Initial population. Let NP be the number of population elements, referred to as *members* (each member is a vector). The initial population is randomly generated and it will be denoted as the set $\mathcal{P} = \{r_1, \ldots, r_{NP}\}$.

Population evolution. A *mutant element* is generated by combining three pseudo-random elements denoted $\{r_1, r_2, r_3\}$, with the condition that they are all different. The choice of the three elements depends on some strategies. Let v be the mutant element, then:

$$v = r_1 + F(r_3 - r_2)$$

where F is a constant real value within the range $[0; 2]$. This operation is drawn in Figure 4.7 a); the blue plan represents the population set $\mathcal{P}$ (with some abuse of representation, this plan represents a multi-dimensional space, instead of a 2-D space, as normal).

Afterwards v is recombined with a *crossover* element, denoted as r, in order to form a *trial* vector t: each component of t takes values from v or from r according to a crossover probability $p_{cr} \in [0; 1)$.

Consider, for example, that each member is a 5-components' vector, thus

$$v = [v_1, v_2, v_3, v_4, v_5] \text{ and } r = [r_1, r_2, r_3, r_4, r_5]$$

Let $p_i \in [0; 1]$ be a drawn random value associated to each i^{th} component and consider a fixed real value $p_{cr} \in [0; 1)$ corresponding to a crossover:

$$t_i = \begin{cases} r_i \text{ if } p_i < p_{cr} \\ v_i \text{ if } p_i \geq p_{cr} \end{cases} \tag{4.2}$$

As result of this operation we can have, for instance, $t = [r_1, v_2, v_3, r_4, v_5]$, if $p_1, p_4 < p_{cr}$ and $p_2, p_3, p_5 \geq p_{cr}$.

t is evaluated by the objective function $f(\cdot)$: if t has a better result, it replaces r in the new population, otherwise r is kept. In this manner, at each iteration the population evolves toward a better one. In Figure 4.7 b) we show this last operation, in which the evaluation of the two elements is represented by vertical arrows

4.4.2 Optimization of LDPC Codes

In our purpose of optimization, the Differential Evolution can be coupled with the Monte-Carlo simulations that evaluates the trial elements. We proceed by iteratively finding the symbol-node degree distribution that assures good performance in terms of estimated threshold (the check-degree distribution is computed according to the Eq. (3.28)).

Given the desired rate, the members are vectors $(\lambda_d)_d$ determined by the coefficients of the edge-perspective degree distribution polynomial; the size of the vector depends on the maximum degree of the $\lambda(x)$ polynomial. For example we have obtained good distribution polynomials for codes of rate $r = 1/2$ and defined over $\mathbb{F}_{16}$. We report the optimized distributions in the following (this code will be also used in the next chapter):

$$\begin{aligned} \lambda(x) &= 0.5376x + 0.1678x^2 + 0.1360x^4 + 0.1586x^9 \\ \rho(x) &= 0.5169x^4 + 0.4831x^5 \end{aligned}$$

The corresponding threshold E_b/N_0 for this ensemble of codes is equal to 0.3638dB . Figure 4.8 shows the performance of a code from this ensemble with 4000 source bits and designed by using the Progressive Edge-Growth (PEG) algorithm [32].

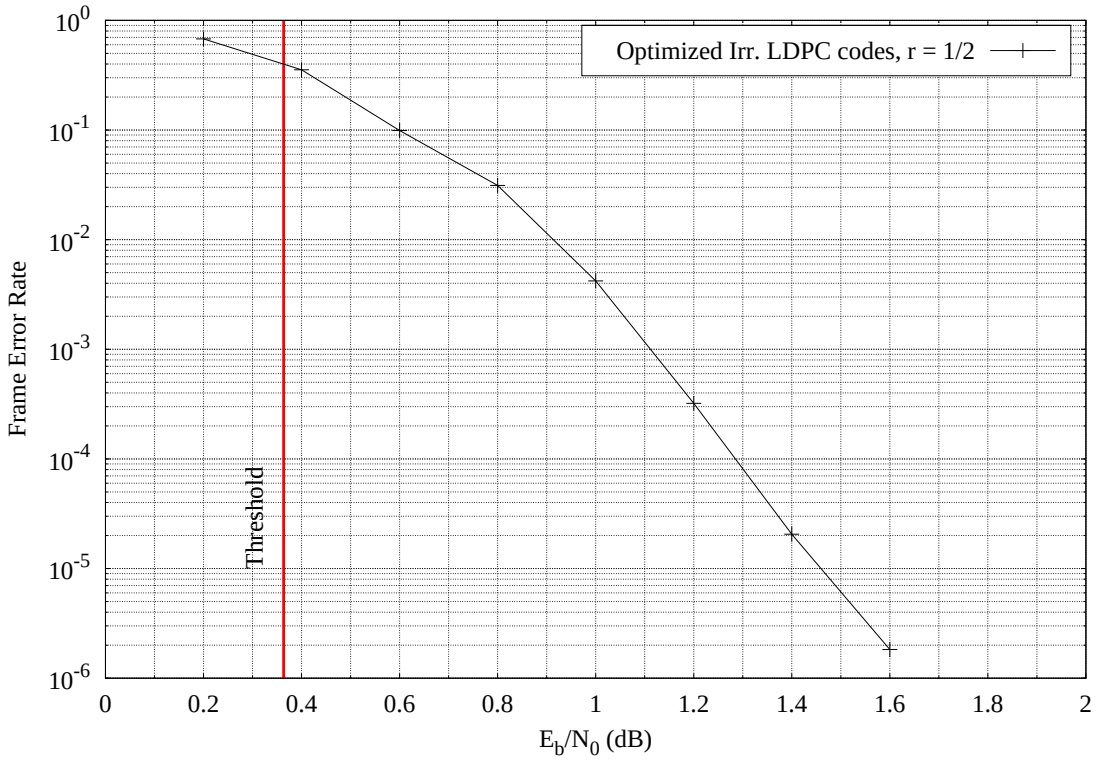

Figure 4.8: NB-LDPC code optimized via Differential Evolution technique

In the same manner we can optimize the edge label distribution polynomial $\xi(x)$ and the population is formed by vector of $q-1$ components. Moreover, this method will also be employed for optimizing the puncturing distributions in the next chapter.

4.5 Conclusions

The asymptotic analysis allows one to compute the threshold and then to find ensembles of LDPC codes with good performance. Under the independence assumption, for which the exchanged messages are considered independent, one can estimate the worst channel condition at which it is possible to have reliable data transmission. This can be done via the Density Evolution method that tracks the evolution of the densities of exchanged messages between the nodes. However, the knowledge of this threshold can be recursively obtained only for binary LDPC codes and NB-LDPC codes transmitted over the binary erasure channel. The Density Evolution using the Gaussian approximation have been proposed in order to simplify the threshold computation; this allows one to reduce the complexity of the density evolution if we accept approximated results.

The motivation for the present work comes from the observation that the Monte-Carlo simulations for binary LDPC codes offer better results with respect to the Gaussian approximation. In non-binary case we can compare the results of MC simulation with the exact thresholds only if we consider the binary erasure channel for which we know the exact threshold computation. We are not able to evaluate the method precision when the codes are transmitted over the AWGN channel because it does not exist an exact computation and then we can not compare the MC simulation results with the exact thresholds. Yet, the use of non-binary alphabets should play an important role in making the exchanged messages more uncorrelated, so that the method should be more precise for non-binary LDPC codes.

We reported several results for an extensive study. First of all, we have compared the GA and the MC methods with the exact Density Evolution in the binary case. We have demonstrated that the MC-DE shows thresholds more closer to the real thresholds with respect to the GA-DE, even for a small number of observed outgoing messages. Secondly, we have analyzed the different implementation parameters of the MC simulations (e.g. the number of exchanged messages and the maximum number of decoding iterations) in order to evaluate the right tradeoff between accuracy and fastness of the algorithm. We have shown that the Monte-Carlo method outperforms the Density Evolution under Gaussian approximation. These results allow:

1) to extend the asymptotic analysis to a large class of channel models and for non-binary codes,
2) to estimate with great precision the exact thresholds at least with the purpose of knowing the general behavior of code ensembles and, finally,
3) to quickly obtain the thresholds due to the rapid convergence of this method.

We have described the Differential Evolution that is a genetic method used for optimization problems. This method has the major advantage that it treats the cost function as a "black-box" and it can be used for a broad class of optimization problems. As drawback it does not assure that the found solutions are absolutely the best ones. When this technique is coupled with the MC-DE method, it can be used in order to find good ensembles of codes.

The MC-DE method will also be used in the next chapter, where we deal with rate-adaptability issues. Indeed, the MC-DE method will be used in order to evaluate the thresholds of the punctured NB-LDPC codes, whereas the Differential Evolution will be used for optimizing the puncturing distributions.

Chapter 5

Punctured NB-LDPC Codes over AWGN Channels

Generally, when the channel allows to transmit with small error probability the transmission is more reliable and a *light* encoding can be envisaged, otherwise, when the channel behaviour worsen due to the increasing noise, the data transmission requires a more robust encoding. Moreover, this phenomena could vary rapidly and without prediction. For this reason we are interested in analyzing the capability of the LDPC codes to flexibly adapt their rates to the channel conditions. The rate adaptability is done by puncturing a mother code [1]. The advantage of such a method is that the same encoding and decoding systems can be used regardless of the coding rate of the transmission.

The rate adaptability for binary LDPC codes consists in choosing the bit-nodes to be punctured. In case of NB-LDPC codes we have an extra degree of freedom as the symbol-nodes can be punctured either completely or only partially (*i.e.,* only certain bits are punctured within a given symbol-node). As a consequence, NB-LDPC codes are more robust to the puncturing operations than their binary counterparts.

In this chapter, we analyse the optimization of the distribution of the punctured bits according to the degree distribution of symbol-nodes. The punctured codes are analyzed with the Density Evolution via Monte-Carlo simulation method. We demonstrate that these codes have good asymptotic performance, as well as they exhibit good performance at short to medium lengths.

The chapter is organized as follows. After introducing and motivating the problem in Section 5.1, we describe the puncturing distribution in Section 5.2. In Section 5.3 we analyse the puncturing distributions for regular and semi-regular LDPC codes. The optimization of puncturing distributions for irregular codes is addressed in Section 5.4. Finally, with Section 5.5 we conclude the chapter.

1. Therefore, the chosen mother code has a smaller coding rate than the punctured codes.

5.1 Motivations

When coded bits are transmitted over time-varying channels, as for wireless communication systems, the rate of the forward error correction code has to be adapted to the quality of the channel, so that to achieve both high throughput and reliability.

The rate adaptation requires codes of different rates, which can be efficiently obtained by using one low rate mother code and puncture it in order to get higher rates. The advantage of puncturing is that a single encoder/decoder pair can be used regardless of the puncturing pattern: according to the channel conditions, the transmission system adapts itself by just changing the puncturing pattern at the transmitter, and the depuncturing pattern at the receiver. This technique allows for a high flexibility systems and a fast adaptation of the transmission data.

The transmission of punctured codewords can be seen as a particular case of the binary erasure channel in which the punctured bits represent the erasures in the channel. At the receiver side, the decoder receives in input only the noisy information corresponding to the unpunctured bits. It should be clear that the puncturing process is completely transparent for the decoder. In effect, the decoder input corresponding to a bit is the probability of being 0 or 1; in case of a punctured bit this probability is in both cases – 0 or 1 – equals to 1/2, or equivalently the corresponding LLR is zero.

Recovering process. In the binary case, Ha *et al.* [25] gave a puncturing method based on the irregularity profile of the binary LDPC code and they showed that good capacity-approaching codes exist. They demonstrated that the puncturing patterns can be optimized so that *rate-compatible* codes results in a small loss of the asymptotic performance. They modified the Gaussian approximation of [12] in order to include the messages of the punctured bit-nodes into the computation. In a second time they optimized the puncturing via linear programming taking into account the asymptotic analysis results.

They have also proposed an optimized puncturing method for finite-length codes with regular or irregular profiles [26]. Their idea of optimization is that the punctured bit-nodes are chosen from the ones that have the greatest probability to be recovered. More precisely, these bit-nodes are selected from those that have more *reliable* neighbor check-nodes. The reliability of a check-node depends on the number of unpuctured bit-nodes in its neighborhood. The process of receiving a non-zero message from a neighbor check-node is referred to as *recovery* (as in the decoding process over the BEC). A punctured bit-node that is connected to check-nodes whose incoming messages are from unpuctured bit-nodes is referred to as one-step-recoverable (1-SR) bit-node[2]. In general, k-SR bit-nodes will be recovered at the k^{th} decoding iteration and they are used to recover the $(k+1)$-SR bit-nodes. The idea is then to concentrate the punctured bits in the lower recoverable steps in order to improve the reliability of the exchanged messages.

For non-binary codes, Klinc et al. [37] proposed a design of puncturing patterns based on symbol recovery levels, similar to the design proposed in [26]. Given a target punctured rate, the algorithm proposed in [37] computes a puncturing pattern that minimizes the number of recovery steps for the punctured bits. They noticed that by puncturing many bits in symbols with high levels of recoverability has negative effect on the performance of the codes. The algorithm proceeds in two steps:

1) it maximizes the number of symbol-nodes that can be recovered and

2. This is a little abuse of terminology because we remind that we transmit over AWGN channel. Then, a bit-node is not recovered necessarily after k iterations. It would be valid only for the BEC case.

2) it punctures uniformly the bits for the lower recoverability-step symbol-nodes.

Their approaches is thought for finite length puncturing design, but does not give insights regarding the optimization of the puncturing distribution for ensembles of NB-LDPC codes. In our work we optimize puncturing distributions for NB-LDPC ensembles, according to the symbol-node degree distribution of the ensemble. Moreover, we analyze the code performance for both the finite and the infinite blocklengths.

Asymptotic and finite-length analysis. We have mentioned the asymptotic performance of an ensemble of codes in the previous chapters. The Density Evolution allows to exactly compute the *threshold* of code ensembles. The exact density evolution in non-binary case is only manageable for the binary erasure channel. In the case of more general memoryless channels, like the binary-input AWGN channel, density evolution must be approximated using numerical techniques, e.g. the Monte-Carlo simulations. In the previous chapter, we have shown that the Density Evolution of non-binary LDPC codes can be tightly approximated by using fast Monte-Carlo approximation of the density of the messages exchanged during the iterative decoding of an infinite code. This method allows for obtaining very close estimates of thresholds of non-binary ensembles, making possible the optimization of non-binary codes for a wide range of applications and channel models. In particular, it can be successfully applied for optimizing puncturing distributions for rate-compatible non-binary LDPC codes.

The infinite code analysis is very important for understanding the way of puncturing the bits, regardless of the particular graph realization. In particular, we are interested in optimizing the puncturing distribution for an ensemble $\mathcal{E}(\lambda, \rho)$ of NB-LDPC codes; afterwards we simulate short-medium codes with degree pair (λ, ρ) and puncturing patterns that comply with the optimized puncturing distributions.

Advantages of non-binary LDPC codes. In the binary case, the puncturing problem is restricted to choose the punctured bit-nodes. When we puncture a non-binary LDPC code we have another degree of freedom given by the possibilities of choosing the number of bits to puncture for each symbol. Then, we will demonstrate that a carefully choice of puncturing patterns leads to an effective performance gain.

But non-binary codes have also another interesting property: by increasing the size of the alphabet, good codes tend to be sparser. Consequently, asymptotic optimization can be performed so that to obtain (asymptotically) good ensembles of codes having low node degrees. This, in turn, positively affects the performance of "finite-length" codes from the optimized ensemble, as low node degrees yield faster convergence toward the asymptotic limit. As said in the previous chapter, for an alphabet $\mathbb{F}_q$ with $q \geq 64$ the best codes are ultra-sparse, meaning that the symbol-node degrees are concentrated in $d_s = 2$. We will see that symbol-nodes of degree 2 play an important role in the puncturing process.

Puncturing distribution. In this work we focus on exploring the rate-adaptability behaviour of non-binary LDPC codes. We propose a method that adapts the transmission rate according to a *puncturing pattern*. The puncturing pattern identifies which bits will be transmitted and which bits will be punctured. The choice of the punctured pattern is done according to a *puncturing distribution* that gives the fractions of punctured bits according to the degrees of symbol-nodes. In the binary case, the puncturing distribution is defined by fractions $\{f_d\}_d$ of punctured bit-nodes of degree d. For non-binary codes, the puncturing distribution can be defined in terms of fractions $\{f_{d,k}\}_{d,k}$ of degree-d symbol-nodes with exactly k punctured bits per symbol. This will be addressed in Section 5.2.

There are two ways for puncturing the symbol-nodes, we can have *cluster-puncturing*, in which the punctured-bits are concentrated in a small number of symbol-nodes, or we can have a *spread-puncturing*, in which the punctured bits are dispersed among the greatest number of symbol-nodes.

Note that our approach to the optimization problem of the puncturing distributions is complementary to the finite-length approach of [37]. Indeed, for an optimized puncturing distribution, a puncturing pattern (satisfying the specified distribution) can be constructed by using the method of [37] (or a slightly modified version of it) [3].

Analysis and optimization of the puncturing distribution. The analysis of the puncturing problem will be divided in three parts according to the kind of code.

1) **Regular NB-LDPC codes.** In 5.3.1 we will firstly analyse via the MC-DE how to puncture the bits for regular codes. The main problem for regular codes is the choice of a clustering or a spreading puncturing. We will see that it is better to spread the punctured bits for regular (2,4)-LDPC codes, whereas a clustering puncturing results better for regular (d_s,d_c)-LDPC codes with $d_s \geq 3$. It seems logic because if we puncture completely a symbol-node of degree 2 it does not receive information from the channel and it forwards the only incoming message. Instead, it is preferred to cluster the punctured bits in less symbol-nodes with degree greater than 2 because they are more *robust* and they can be recovered more easily.
2) **Semi-regular NB-LDPC codes.** Afterward, we will analyse the semi-regular codes, in which the symbol-node degrees are concentrated around a value. Our idea, based on the results obtained in the regular code analysis, was to spread the punctured bits for 2-degree symbol-nodes and to cluster the punctured bits for symbol-nodes with degrees greater than 2. However, this approach, shown in 5.3.2, is not the best. In this case, it is better to spread all the punctured bits on symbol-nodes with lower degree.
3) **Irregular NB-LDPC codes.** Concerning the irregular LDPC codes, we will optimize the puncturing distributions by using the genetic algorithm illustrated in Section 4.4. In Section 5.4 we have coupled the Differential Evolution with the Monte-Carlo simulations in order to search good puncturing distributions. For non-binary LDPC codes over $\mathbb{F}_{16}$, with maximum symbol-node degree equal to 10, the optimized punctured codes exhibit a gap to capacity between 0.2 and 0.5dB, for punctured rates varying from 0.5 to 0.9.

5.2 Puncturing Distributions for Non-Binary LDPC Codes

Let a non-binary LDPC code be used over a binary-input channel: a non-binary codeword is mapped into its binary images by using Eq.(3.2), then transmitted over the channel. Hence, even if non-binary codes operate at the symbol level, non-binary codewords still can be punctured at the bit-level. A coded symbol can be either completely or partially punctured, according to whether all the bits of its binary image are punctured or not. In order to design good puncturing patterns we have to answer the following questions:

3. As a parallel, consider the construction of bipartite Tanner graphs: for given node-degree distributions, we can advantageously use the Progressive Edge-Growth (PEG) algorithm, so as to maximize the girth of the constructed graph; however, if the PEG algorithm is not constrained with a given degree distribution, the algorithm will tend to construct a regular graph, which does not provide the best performance.

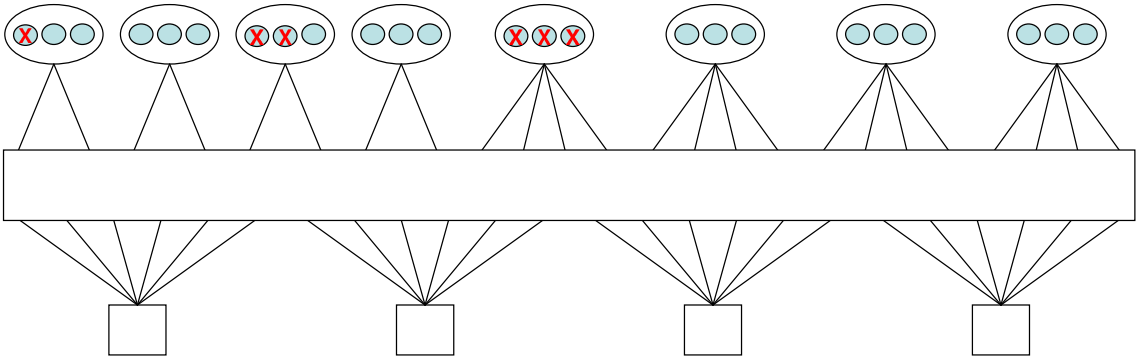

Figure 5.1: Bit level puncturing pattern for non-binary LDPC codes

1) Assuming that the code is regular, should puncturing bits be clustered or spread over the symbol-nodes? Clustering yields a reduced number of completely punctured symbols, which receive no information from the channel. On the contrary, spreading results into an increased number of partially punctured symbols, which still benefit from some partial information from the channel.
2) In case of irregular codes, how should punctured bits be chosen with respect to symbol-node degrees?

Puncturing distributions. We will define the puncturing distribution according to $L(x)$, the degree distribution polynomial for symbol-nodes from the node-perspective. We remind that the fraction of symbol-nodes of degree d is obtained by (see page 46):

$$L_d = \frac{\lambda_d}{d \int_0^1 \lambda(x)\,\mathrm{d}x}$$

Let $f_{d,k}$ denote the fraction of degree-d symbol-nodes with k punctured bits. For any degree d, we note $f_d = (f_{d,0}, f_{d,1}, \ldots, f_{d,p})$. Note that $\sum_{k=0}^{p} f_{d,k} = 1$, as any symbol-node contains either $k = 0$ (unpunctured), or $k = 1$, ..., or $k = p$ punctured bits. Let $f_{d,k}L_d$ be the fraction of symbols-node of degree d with k punctured bits. The overall fraction of punctured bits is:

$$f = \frac{1}{p} \sum_{d=1}^{d_{\mathrm{s,max}}} \sum_{k=0}^{p} k f_{d,k} L_d \tag{5.1}$$

The corresponding punctured rate, that will be referred to as r_p, is given by:

$$r_p = \frac{r}{1-f} \tag{5.2}$$

This is illustrated in the Figure 5.1 for an LDPC code over $\mathbb{F}_8$ and degree distribution polynomials:

$$L(x) = \frac{1}{2}x^2 + \frac{1}{2}x^4 \quad \text{and} \quad R(x) = x^6$$

The designed code rate is $r = 1/2$. In the example, each symbol has 3 bits represented as blue points, while the red crosses mark the punctured bits. In this case the fractions of punctured bits are: $f_{2,1} = f_{2,2} = 1/4$, and $f_{4,3} = 1/4$.
Thus $f = 0.25$ and the designed punctured rate is $r_p = 2/3$.

5.3 Analysis of Puncturing Distributions

In this section we analyse different puncturing distributions for regular and semi-regular non-binary LDPC codes over the BIAWGN channel. We have already defined the semi-regular LDPC codes in the Section 4.3.1 at page 69. What we want to know is how adapt the puncturing distributions according to the degree of symbol-nodes.

5.3.1 Regular (2,d_c)-LDPC Codes

First of all, we consider three ensembles of regular LDPC codes with rate 1/2, and (symbol, constraint)-node degrees equal to $(2, 4)$, $(3, 6)$, or $(4, 8)$. A fraction $f \in [0, 0.25]$ of coded bits are punctured, which corresponds to a punctured rate varying from $r_p = 1/2$ (no puncturing) to $r_p = 2/3$. For each fraction of punctured bits, two puncturing distributions have been compared:

1) *clustering distribution*: punctured bits are clustered on a fraction f of completely punctured symbols-nodes; it could happen that the number of punctured bits is not a multiple of p, in this case one symbol will be partially punctured [4].
2) *spreading distribution*: if $f < \frac{1}{p}$ punctured bits are spread over a fraction pf of symbol-nodes, each one with one single punctured bit. Otherwise, all the symbol-nodes are punctured, each symbol-node containing either $\lfloor pf \rfloor$ or $\lceil pf \rceil$ punctured bits.

Spreading and clustering puncturing for different punctured rates. We have simulated the Density-Evolution via Monte-Carlo with 10000 outgoing messages and 200 maximum decoding iterations. Corresponding thresholds for codes over $\mathbb{F}_8$ and $\mathbb{F}_{16}$ are shown respectively in Figure 5.2 and Figure 5.3. We note the different behaviour of these distributions, depending on the symbol-node degree: the spreading distribution provides better performance for the regular $(2, 4)$-code, but it is outperformed by the clustering distribution for the other two codes with higher symbol-node degrees.

4. This has a negligible effect on the performance.

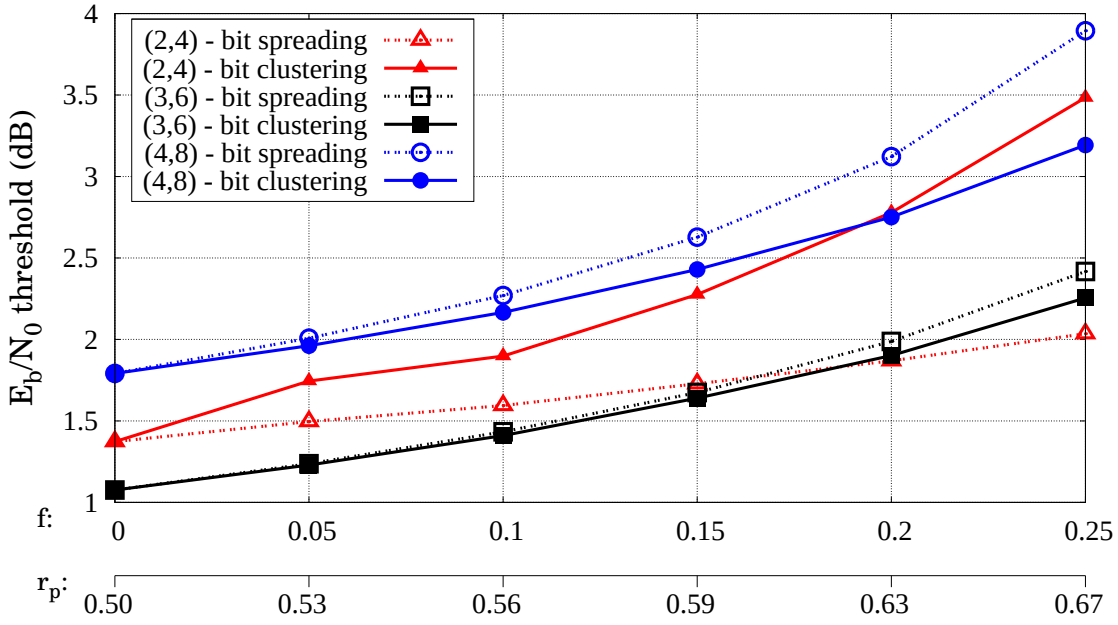

Figure 5.2: Spreading vs. clustering distribution; regular LDPC codes over $\mathbb{F}_8$

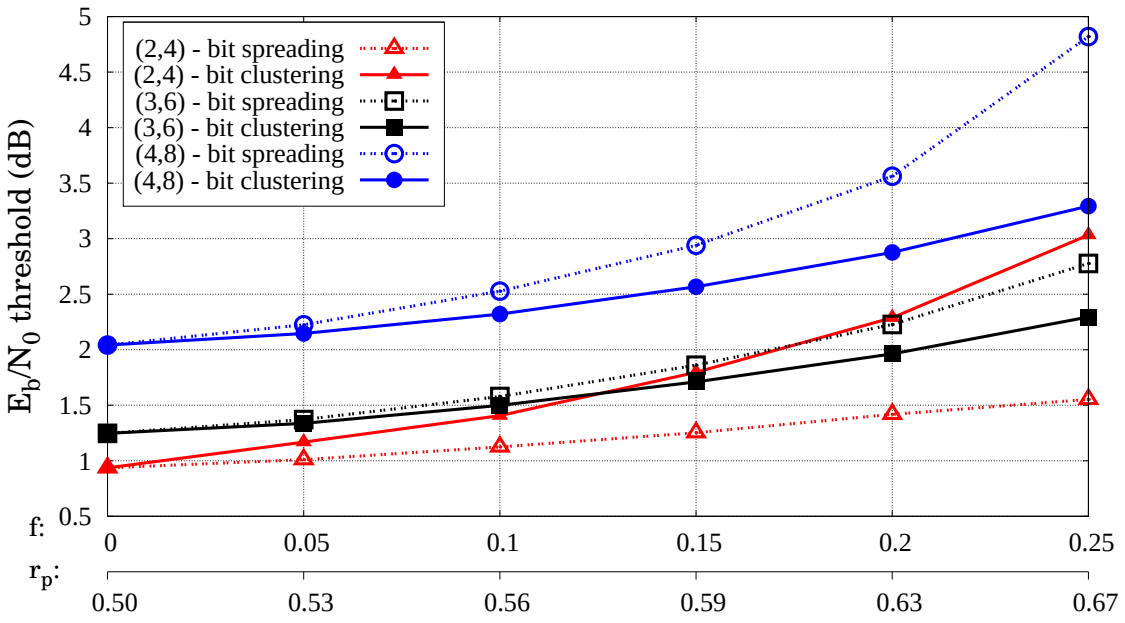

Figure 5.3: Spreading vs. clustering distribution; regular LDPC codes over $\mathbb{F}_{16}$

We can also observe that the gap of performance between cluster-spread puncturing increases with the coding rate. Consider first the threshold of the punctured $(2,4)$ codes with rate $r_p = 2/3$. The spread puncturing shows a difference of 1.5dB with respect to the code in which a clustered puncturing is applied. For the two other codes an increasing of the gap is observed with the rate between the curves of the spread and the clustered methods: observing the last point at rate $r_p = 2/3$, for the $(3,6)$ LDPC code we observe a gap of $\sim$0.3dB , whereas for $(4,8)$ LDPC code this gap is up to 0.7dB .

The intuition behind these results goes as follows. An outgoing message from a degree-2 symbol-node depends on the information coming from the channel and the unique incoming extrinsic message. Thus, if a degree-2 symbol-node is completely punctured, it has no channel information, and the outgoing message is equal to the incoming one. This causes a lack of diversity in the iterative message-passing, which results a degraded decoding performance. Conversely for the codes with $d_s > 2$ it is better to concentrate the punctured bits on the minimum number of symbols because they can easily be recovered from the $d_s - 1$ extrinsic messages.

Mixed puncturing distributions. Intermediary puncturing distributions can be obtained by mixing the above clustering and spreading distributions. As illustrated at Figure 5.4 for regular codes over $\mathbb{F}_4$: the fraction of punctured bits is $f = 0.25$, corresponding to a punctured rate $r_p = 2/3$, and it is decomposed as $f = f_1 + f_2$ [5], where f_1 is the fraction of bits that are punctured in a spreading manner (in this case, 1 bit per symbol), and f_2 is the fraction of bits that are punctured in a clustering manner (2 bits per symbol). Again, we can observe that the spreading distribution provides better performance for the regular $(2,4)$-code. While for the other codes the performance are quite the same, probably due to the used alphabet.

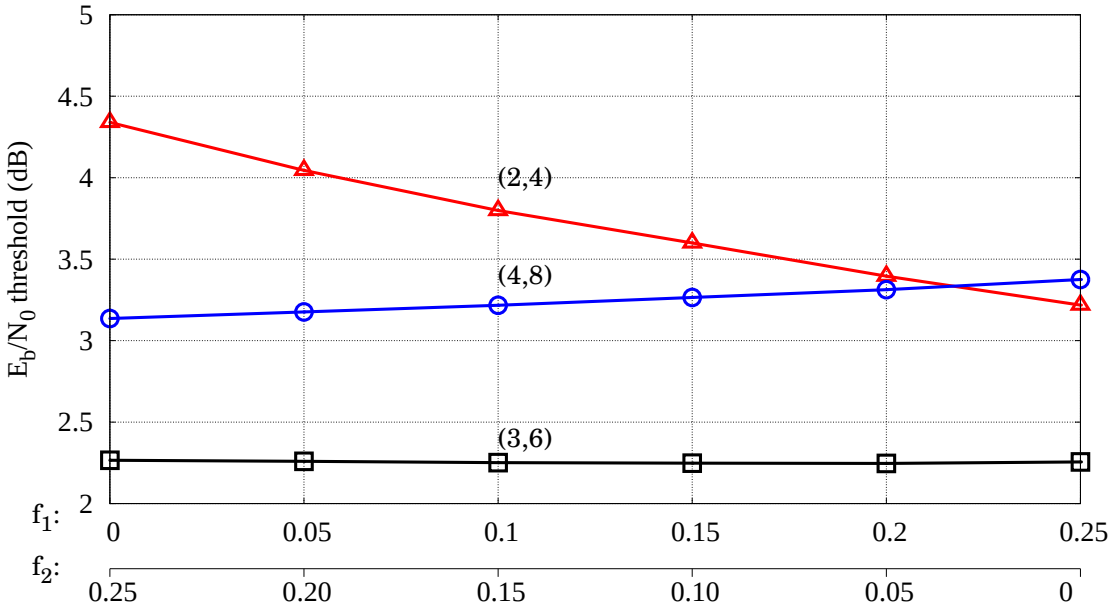

Figure 5.4: Intermediary puncturing distributions for regular codes over $\mathbb{F}_4$

5. We omit the the degree index d since the codes are regular.

Similar results are also shown at Figure 5.5 for codes over $\mathbb{F}_{16}$, that is 4 bits each symbol. Let the *clusterization degree* be the number of punctured bits per symbols. For a clusterization degree $k \in \{1, 2, 3, 4\}$ (on the abscissa), a fraction $\frac{1}{k}$ of symbol-nodes are punctured, by punching k bits per symbol and keeping the fraction $f = 0.25$. It means that we can puncture a single bit (which corresponds to 1 over 4 bits) of all the symbol-nodes. Equivalently, for example, we can puncture 2 bits of the 50% of symbol-nodes that keeps $f = 0.25$. Once more, we can observe that spreading is suitable for symbol-nodes of degree-2, while clustering is more appropriate for symbol-nodes of degree ≥ 3.

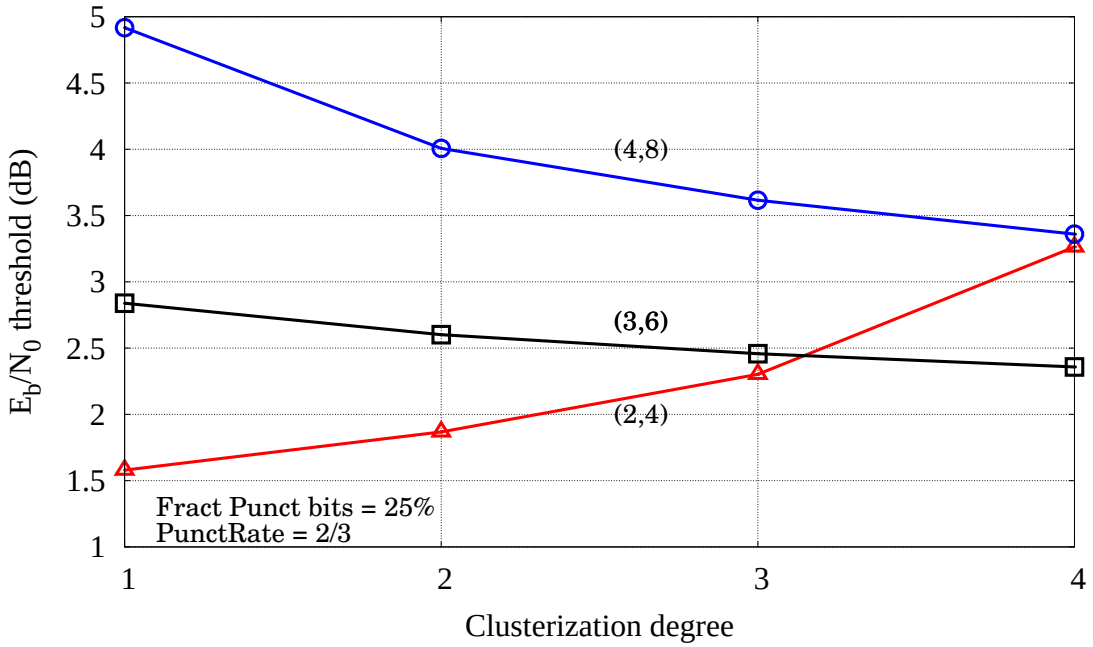

Figure 5.5: Various clusterization degrees for regular codes over $\mathbb{F}_{16}$

5.3.2 Semi-Regular LDPC Codes

We investigate now different puncturing distributions for semi-regular LDPC codes, with node-perspective degree distribution polynomials $L(x) = \frac{1}{2}x^2 + \frac{1}{2}x^4$ and $R(x) = x^6$, corresponding to a designed coding rate $r = 1/2$. The same fraction $f = 0.25$ of coded bits are punctured, so as to obtain a punctured rate $r_p = 2/3$. The fraction is decomposed as $f = f_2 + f_4$ [6], where f_2 is the fraction of bits that are punctured on degree-2 symbol-nodes, and f_4 is the fraction of bits that are punctured on degree-4 symbol-nodes.

Numerical results are shown at Figure 5.6 and Figure 5.7: for solid curves, punctured bits are either clustered (full squares) or spread over the symbol-nodes (empty squares), for both symbol-nodes of degree 2 and 4. For the dashed curve, punctured bits are spread over symbol-nodes of degree 2, and clustered over symbol-nodes of degree 4. In view of previous results for regular codes, we was expecting this last distribution to provide the best performance. Surprisingly, it performs in between the clustering and the spreading distribution. What we observe is that in case there are punctured symbols of degree 4 ($f_4 > 0$), the best performance is given by the clustering distribution. However, since all plots correspond to the same punctured rate, the best puncturing distribution in each figure is given by the lowest plot. In both cases this is the distribution that spreads all the punctured bits over symbol-nodes of degree 2 ($f_2 = 0.25$, $f_4 = 0$, empty square) [7].

6. Also in this case, for the sake of the simplicity, we omit the degree index d. It should be $f = f_{2,2} + f_{4,4}$.
7. However, the clustering distribution with $f_2 = 0.2$ and $f_4 = 0.05$ yields almost the same performance.

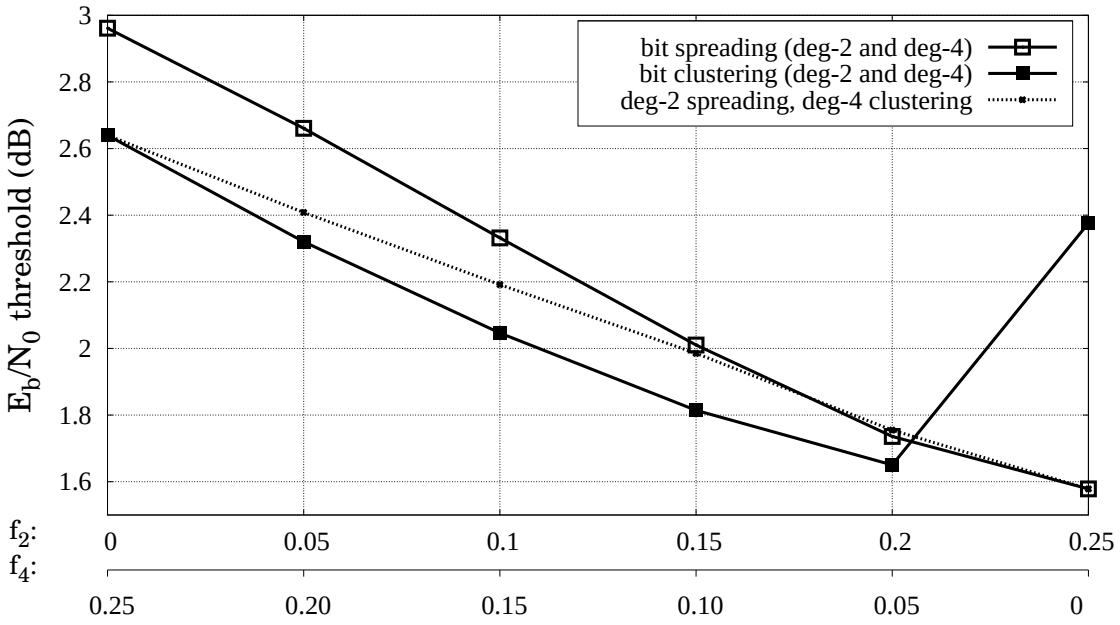

Figure 5.6: Low vs. high symbol-degree, and spreading vs. clustering puncturing distributions for irregular LDPC codes over $\mathbb{F}_8$

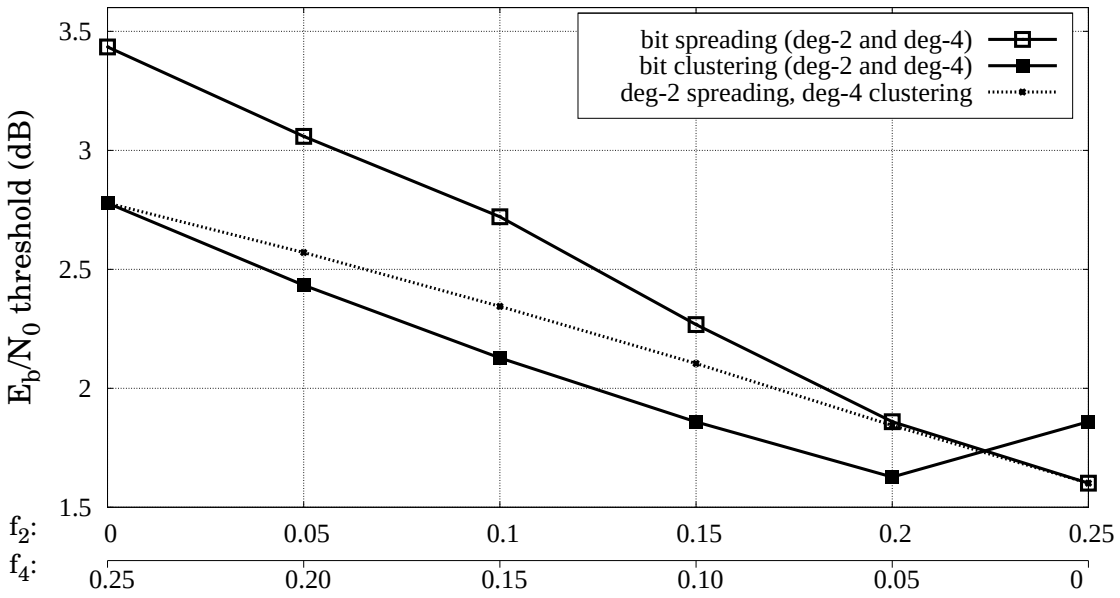

Figure 5.7: Low vs. high symbol-degree, and spreading vs. clustering puncturing distributions for irregular LDPC codes over $\mathbb{F}_{16}$

In the Figure 5.8 we have illustrated the puncturing problem in the binary case. We have concentrate the punctured bits from the symbol-nodes of degree 4 (left side of the figure) to symbol-nodes of degree 2 (right side of the figure). We observe that the best choice of puncturing corresponds to a balanced concentration between the degree 2 and 4.

We conclude saying that the results for NB-LDPC codes represent an important dissimilarity compared to the binary case, in which choosing all the punctured bits on degree-2 bit-nodes proves to be catastrophic. In the non-binary case, spreading punctured bits over degree-2 symbol nodes yields good performance, *provided that the fraction of punctured bits is not too high*. The next section will also provide evidence on this fact.

Finally, in this section we answered to the first question concerning the fact if it would be better a clustered or a scattered puncturing distribution. Indeed, we have demonstrated that the regular (2,4)-LDPC codes transmitted over $\mathbb{F}_q$ (with $q \geq 16$) demonstrate the best performance when we spread the punctured bits among the greatest number of symbol-nodes. This result is fundamental also in light of the considerations about the code defined over $\mathbb{F}_q$ with $q \geq 64$ for which the ultra-sparse codes represent the best codes.

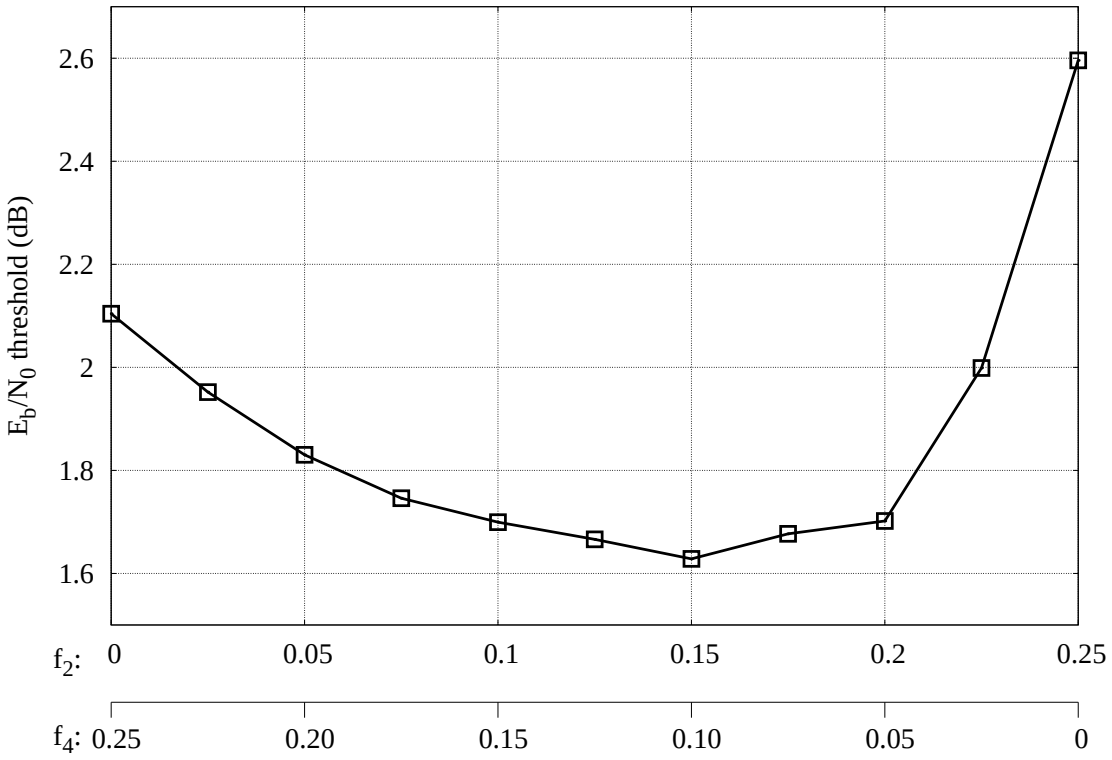

Figure 5.8: Spreading vs. clustering distribution; regular LDPC codes over $\mathbb{F}_2$ – Binary case

5.4 Optimization of Puncturing Distributions

In this section we present optimized puncturing distributions for an irregular code over $\mathbb{F}_{16}$ with rate 1/2, and punctured code rates from 0.55 to 0.9. Optimization has been performed by using the Differential Evolution algorithm [69]. First of all we searched for good degree distribution pairs λ and ρ of rate 1/2, with maximum symbol degree equal to 10. The optimized distributions and the corresponding E_b/N_0 threshold are given below:

$$\begin{aligned} \lambda(x) &= 0.5376x + 0.1678x^2 + 0.1360x^4 + 0.1586x^9 \\ \rho(x) &= 0.5169x^4 + 0.4831x^5 \\ E_b/N_0 &= 0.3638\text{dB} \end{aligned}$$

Next, we searched for good puncturing distributions for punctured rates:

$$r_p \in \{0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90\}$$

Optimized distributions $\{f_{d,k}\}_{\substack{d=2,3,5,10 \\ k=0,1,2,3,4}}$ for different punctured rates are shown in Table 5.1. Also indicated are $\bar{k}_d$, the average number of punctured bits per symbol-node of degree d, and f_k, the average fraction of symbols with k punctured bits. As an equivalent representation, the corresponding *clusterization distributions* are shown at Figure 5.9: they consist of fractions of *punctured symbols* of degree-d, with k punctured bits per symbol, which are given by $\frac{f_{d,k}}{1-f_{d,0}}$, for $d = 2, 3, 5, 10$, and $k = 1, \ldots, 4$. These distributions seem to be random, and properties observed for regular codes shall not apply is this case.

Thresholds of optimized punctured distributions are shown at Figure 5.10, in terms of E_b/N_0. Are also plotted the theoretical Shannon limit (capacity), and thresholds of distributions spreading punctured bits over degree-2 symbol-nodes. The gap between optimized distributions and capacity vary between 0.18 db for $r_p = 0.5$ (unpunctured code) and 0.52 dB for $r_p = 0.9$. We also note that the degree-2-spreading distribution yields very good performance up to punctured rate $r_p = 0.7$. However, such a puncturing distribution proves to be catastrophic for punctured rates $r_p \geq 0.85$

Finally, Figure 5.11 presents the Frame Error Rate performance of optimized distributions for finite code lengths. All the codes have binary dimension (number of source bits) equal to 4000 bits (1000 $\mathbb{F}_{16}$-symbols). The mother code with rate 1/2 has been constructed by using the Progressive Edge-Growth (PEG) algorithm [32], and punctured patterns have been randomly chosen according to the optimized distribution. It is very likely that the performance can be further improved by using an optimized design of the puncturing patterns (in terms of symbol-recoverability steps, cf. discussion in the introduction), especially for high punctured rates.

Table 5.1: Optimized distribution, $r_p \in \{0.55, \ldots, 0.90\}$

$r_p = 0.55$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.8924	0.0043	0.0248	0.0620	0.0165	0.3059
$d=3$	0.8204	0.0675	0.0789	0.0048	0.0284	0.3533
$d=5$	0.6899	0.1001	0.1024	0.1031	0.0044	0.6320
$d=10$	0.6205	0.1034	0.1029	0.0843	0.0888	0.9175
f_k	0.8547	0.0253	0.0422	0.0573	0.0205	

$r_p = 0.60$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.7132	0.1211	0.1149	0.0083	0.0425	0.5456
$d=3$	0.5953	0.1536	0.0161	0.0743	0.1607	1.0514
$d=5$	0.7414	0.0009	0.1822	0.0479	0.0275	0.6190
$d=10$	0.4608	0.1218	0.1187	0.1109	0.1878	1.4429
f_k	0.6865	0.1172	0.1050	0.0257	0.0656	

$r_p = 0.65$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.6246	0.1663	0.0585	0.1143	0.0363	0.7713
$d=3$	0.5456	0.0471	0.1591	0.0217	0.2265	1.3364
$d=5$	0.6088	0.0232	0.1711	0.1517	0.0453	1.0015
$d=10$	0.3256	0.0504	0.2285	0.1868	0.2087	1.9026
f_k	0.5985	0.1326	0.0894	0.1061	0.0734	

$r_p = 0.70$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.3687	0.2371	0.2618	0.1002	0.0322	1.1901
$d=3$	0.6182	0.0034	0.3144	0.0016	0.0624	0.8867
$d=5$	0.6248	0.2787	0.0401	0.0281	0.0283	0.5563
$d=10$	0.2035	0.0174	0.3128	0.2573	0.2090	2.2510
f_k	0.4185	0.1952	0.2556	0.0866	0.0441	

$r_p = 0.75$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.3954	0.2453	0.2044	0.1041	0.0508	1.1696
$d=3$	0.2778	0.1177	0.1648	0.1521	0.2877	2.0542
$d=5$	0.5561	0.0855	0.2053	0.0868	0.0664	1.0219
$d=10$	0.2054	0.1053	0.2386	0.2850	0.1658	2.1006
f_k	0.3812	0.2081	0.1999	0.1179	0.0929	

$r_p = 0.80$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.2026	0.3865	0.1047	0.2757	0.0308	1.5448
$d=3$	0.3616	0.3680	0.0876	0.1129	0.0699	1.1615
$d=5$	0.5783	0.0038	0.0435	0.3546	0.0197	1.2335
$d=10$	0.1998	0.0060	0.3841	0.0230	0.3870	2.3913
f_k	0.2545	0.3390	0.1096	0.24585	0.0510	

$r_p = 0.85$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.1590	0.4312	0.0180	0.3855	0.0063	1.6491
$d=3$	0.5644	0.0813	0.0671	0.0049	0.2823	1.3595
$d=5$	0.1730	0.1040	0.3498	0.3663	0.0069	1.9300
$d=10$	0.0259	0.4386	0.0176	0.4036	0.1144	2.1421
f_k	0.2159	0.3541	0.0500	0.3270	0.0530	

$r_p = 0.90$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$\bar{k}_d$
$d=2$	0.0960	0.4187	0.1077	0.2857	0.0919	1.8589
$d=3$	0.6543	0.0070	0.0779	0.1035	0.1572	1.1023
$d=5$	0.1304	0.3957	0.1314	0.2905	0.0521	1.7384
$d=10$	0.0413	0.0132	0.2822	0.3780	0.2854	2.8530
f_k	0.1811	0.3369	0.1125	0.2623	0.1072	

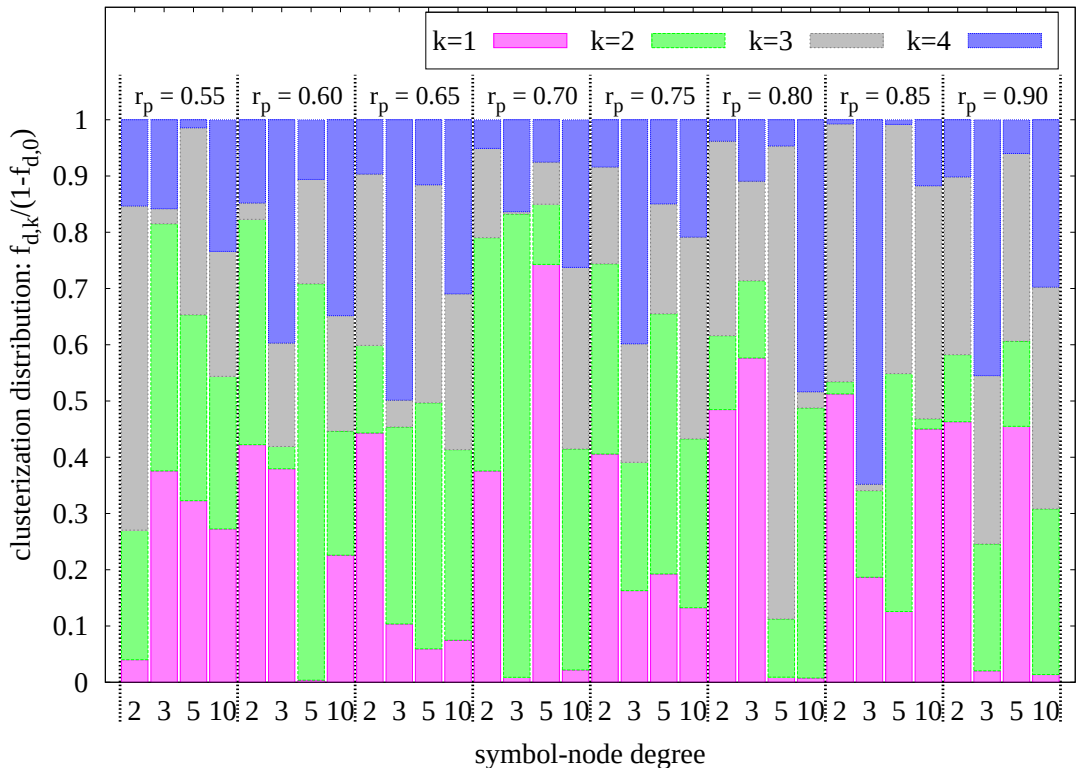

Figure 5.9: Clusterization distributions for $r_p \in \{0.55, \ldots, 0.90\}$

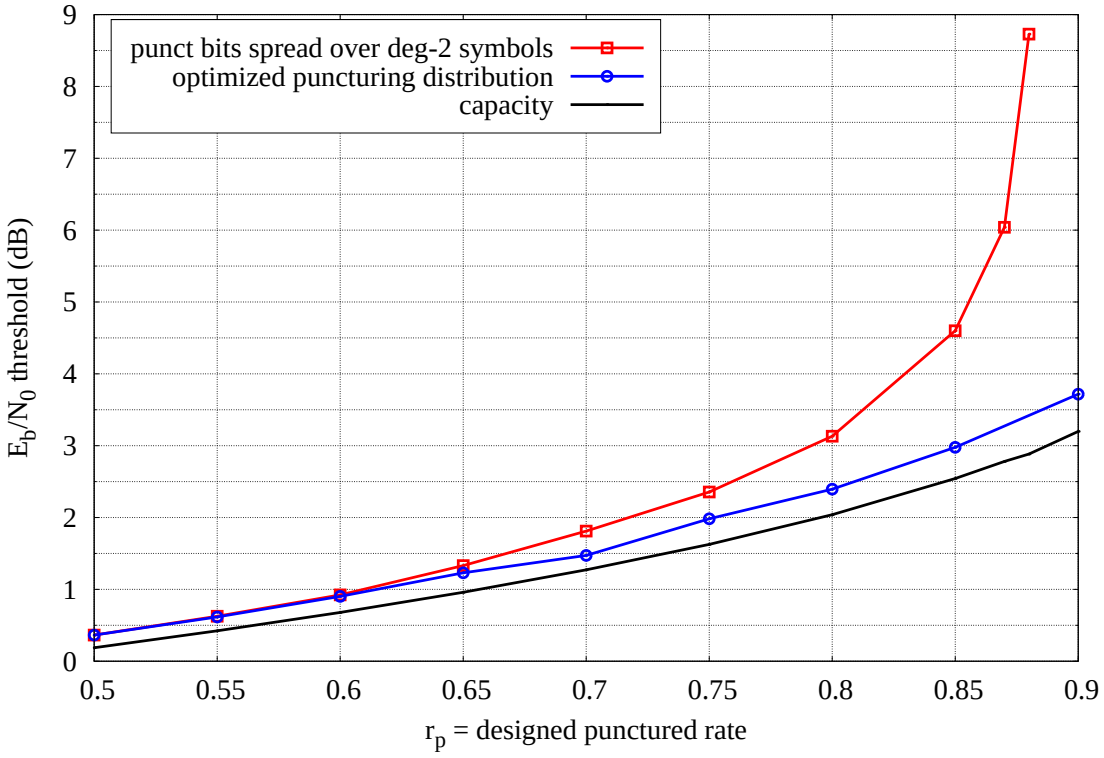

Figure 5.10: Optimized puncturing distributions for regular codes over $\mathbb{F}_{16}$

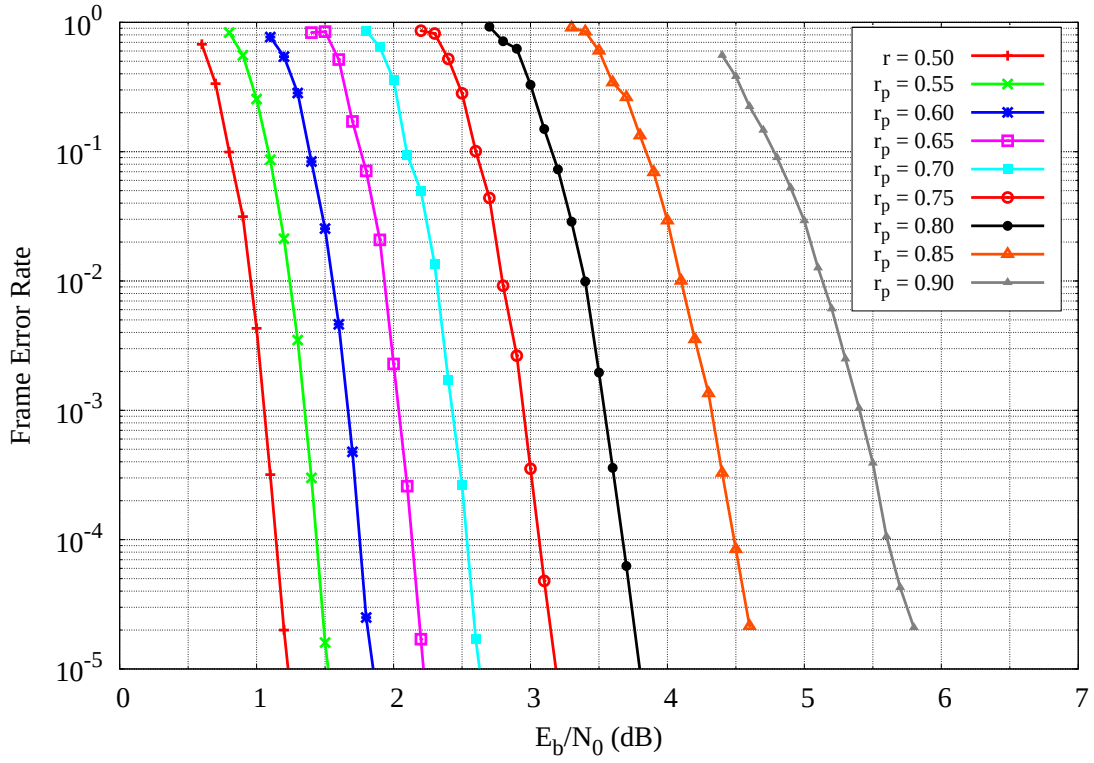

Figure 5.11: Frame error rate, optimized distributions, $K = 4000$ bits

5.5 Conclusions

In this chapter rate-adaptability solutions for non-binary LDPC codes have been investigated. This issue is mainly motivated when the channel meets some special requirements for physical layer due principally to the channel behaviour and one needs to rapidly adapt the rate. At the transmitter side, the system can adapt the rate by simply changing the puncturing pattern. Thus, this method allows one to keep the same encoding/decoding regardless of the coding rate of the transmission. In this work we have focused on the punctured NB-LDPC codes.

Puncturing distributions for regular and irregular codes have been analysed by using simulated density evolution thresholds over the AWGN channel. Puncturing pattern have been analysed with respect to bit clustering/scattering and symbol degrees distribution polynomial $L(x)$. We have demonstrated that we can have an incremental puncturing, while maintaining the gap to the capacity on a wide range of punctured rates.

For regular codes, we showed that the design of puncturing patterns must respect different rules depending on the symbol-node degree: punctured bits must be spread over degree-2 symbol-nodes, while they must be clustered on symbol-nodes of higher degrees.

If the number of punctured bits is relatively small, spreading punctured bits over degree-2 symbol-nodes yields also good performance for irregular codes. However, such a puncturing distribution could be catastrophic for higher punctured rates, in which case optimized puncturing distributions are indispensable. In order to find good puncturing profiles we used the MC-DE method introduced in the previous chapter, to approximate the asymptotic performance of the punctured codes. Finally, we presented optimized puncturing distributions for non-binary LDPC codes with small maximum degree, which exhibit a gap to capacity between 0.2 and 0.5 dB, for punctured rates varying from 0.5 to 0.9.

Chapter 6

Binary Diversity For NB-LDPC Codes over Rayleigh Channels

The AWGN channel is a classical model that is used for studying the free space model. Other disturbing phenomenons due to external interference can affect the transmitted signal in a more significant way. It is the case of radio channels in multipath environments that can be modeled by a Rayleigh distributed fading [51]. The introduction of the interleaving, placed between the encoding and the modulator, can mitigate the fading effects by providing at the receiver side a collection of uncorrelated signal samples. In fact, without interleaving, the bits of each coded symbol are affected by a unique fading coefficient; the interleaver spreads the bits with the same fading coefficients within different transmitted symbols. As a consequence, it allows to lower the error probability toward the AWGN limit.

In this chapter, we analyse the performance of several bit-interleaving strategies applied to NB-LDPC codes over the Rayleigh fading channel. We prove that this technique provides important gains in terms of frame error rate and error detection. In particular, we focus on the ultra-sparse non-binary LDPC codes with constant symbol-node connection $d_s = 2$. These regular $(2, d_c)$-NB-LDPC codes demonstrate the best waterfall performance, thanks to their large girths and improved decoding thresholds. But these codes have codewords with low Hamming weights that cause convergence to erroneous codewords at a high SNR regime and, consequently, lead to high error floors. We demonstrate that the interleaving helps to avoid these drawbacks.

Also, this work demonstrates the importance of the way the bit-interleaving is designed, and proposes a design of an optimized bit-interleaver inspired from the Progressive Edge-Growth (PEG) algorithm. This optimization algorithm depends on the topological structure of a given LDPC code and can also be applied to any degree distribution and code realization. Simulations show excellent results of the proposed interleaver compared to the random interleaver as well as to the system without interleaver.

The rest of the chapter is organized as follows. After motivating this work in Section 6.1, we give some background and notation in Section 6.2. Then, we introduce the interleaving technique in Section 6.3. The proposed PEG interleaver is described in Section 6.4. In Section 6.5 simulation results and comparisons are shown. Finally, the conclusions are drawn in Section 6.6.

6.1 Motivations

Channel model. The assumption behind the AWGN channel is that the transmission is in a free space propagation. Often this is not the case and, for example in indoor wireless system or in mobile communication in urban environment, the transmission may be affected by other phenomena like the ground reflection or the presence of all around objects. Thus the transmitted signals undergo to a *multipath* propagation that causes fluctuations of the amplitude, phase and arrival angle of the received signals. When there is not a dominant light-of-sight, this process can be modeled by the Rayleigh model [51]. As illustrated in the Section 2.2.1 (page 17), this model assumes that the channel response envelope varies according to the Rayleigh distribution (Eq. (2.6)).

In this work we consider NB-LDPC codes modulated by a complex QAM constellation and then transmitted over the Rayleigh fading channel. We also assume that the non-binary alphabet of the codes matches the complex constellation used at the transmitter (that is, a q-QAM modulation is used for NB-LDPC codes defined over $\mathbb{F}_q$).

Interleaved systems. The interleaving technique is critical for many channels that are not memoryless and in which the errors might occur in bursts. Besides, for some families of error-correcting codes, like Turbo codes, an *inner* interleaver is an integral component and its proper design is crucial for good performance [55, 75]

Moreover, a classical way to fight against the fading effects is to introduce diversity by the mean of a bit-interleaver at the transmitter side [45]. Nevertheless, the use of the interleaver for binary LDPC codes appears instead useless, since the bipartite graph already behaves as an inner interleaver. Therefore, the idea of interleaving LDPC codes is something new, because there is no interest of using an interleaver in the binary case. But with the introduction of non-binary alphabets, the use of a *binary interleaver* turns out to be useful when NB-LDPC are transmitted over the Rayleigh fast fading channel, in which different codeword symbols are affected by different fading coefficients.

Interleaving benefits. The interleaver operates between the encoder and the symbol mapper (Figure 6.1), and it aims at improving error rates in situations involving a fading channel. Without an interleaver the bits of each symbol-node are affected by a unique fading coefficient then they are more sensible to deep fading. The presence of an interleaver allows to spread the deep fading and then helps to mitigate this effect within the sent codeword.

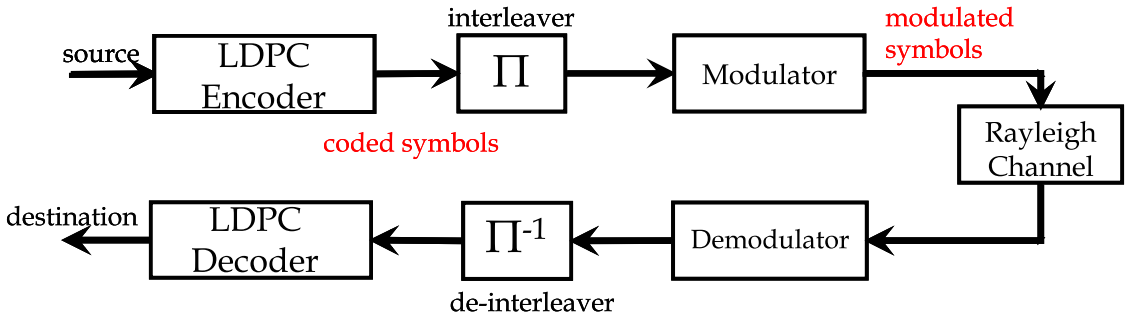

Figure 6.1: Transmission Chain

In Section 6.3 the interleaver is seen as a superimposed graph on a pre-designed Tanner graph, in which the outgoing edges from symbol-nodes correspond to the mapping between the bits of the coded NB-LDPC symbols and the bits that compose the transmitted QAM symbols. The QAM symbol will be the *modulation symbol-node*, which will represent the third kind of node of the whole graph. In our work we aim to design an *interleaving pattern* that gives the position of each bit inside the modulated symbols. At the receiver end this pattern is used after the demodulation for de-interleaving, so that each soft-bit takes the original position in the codeword.

We demonstrate that a random interleaver already shows good simulation results with respect to a no-interleaved system. But we will also show that the performance are even better when the bit-interleaver is adapted to the NB-LDPC code structure. Indeed, in Section 6.4 we propose an optimized interleaving algorithm inspired from the Progressive Edge-Growth algorithm. It associates consecutive channel bits to the most *distant* symbol-nodes in order to render the resulting graph with the largest girth, hence the coded symbols result the more uncorrelated possible.

Ultra-sparse LDPC codes. The case of codes for which the underlying bipartite graph is ultra-sparse, in the sense that each symbol-node is connected to exactly $d_s = 2$ linear constraint-nodes, is of particular interest. First, very large girths can be obtained for Tanner graphs with $d_s = 2$, as demonstrated in [32, 72]. It has also been pointed out [13, 63] that when the size of the non-binary alphabet grows, best decoding thresholds are obtained for average density of edges closer and closer to $d_s = 2$. Practically, for NB-LDPC codes defined over Galois fields $\mathbb{F}_q$ with $q \geq 64$, best codes, both asymptotically and at finite lengths, are *ultra-sparse* codes. For more details the reader can also refer to the Section 4.3.3 (page 76).

Poulliat *et al.* in [54] proposed a design method for regular (2,d_c)-LDPC codes over Finite Field $\mathbb{F}_q$ based on the algebraic properties of the binary images of the parity-check matrix H, with good performance in the waterfall and in the error floor regions. Thanks to these designs we can construct optimized regular (2,d_c)-LDPC codes used for the completeness of this work. In fact, in this work we design ultra-sparse LDPC codes defined in high order alphabet $\mathbb{F}_q$, with $q \geq 64$.

Despite those advantages, the *ultra-sparse* LDPC codes in $\mathbb{F}_q$ suffer from a serious drawback, as their minimum distance d_{min} is limited and grows at best as $\mathcal{O}(log(N))$ [54]. The minimum distance d_{min} of a code is an important parameter: a large d_{min} helps to distinguish two different codewords. As a consequence the decoder can avoid to converge toward erroneous words at high SNR. Conversely, a code with limited d_{min} induces the decoder to choose bad codewords causing poor performance in the error floor region. This limitation is not critical when the desired error rate is above 10^{-5}, which is the case of the wireless transmissions that we target in this thesis. However the bit-interleaver helps the decoder to converge toward the correct codewords that is one of the reasons for which we observe a code performance improving. This will be presented in simulation result section (Section 6.5), when we will talk about the percentages of detected errors of the analysed system.

Marginalization effect. Another aspect on which we will focus is that the introduction of the bit-interleaver yields indirectly to a degradation of the "quality of the demapping". We recall that the decoder input is a probability vector associated to each coded symbol of being one of the elements of the Galois field. The reader can refer to the Section 3.2.4

(page 44) where we have computed the decoder input. Here we make a quality study of the two cases.

1) **Symbol-demapping.** In case of uninterleaved systems (or more in general when there is a one-to-one correspondence between the coded and modulated symbols) these vectors are derived directly from the received complex symbols. This operation is referred, with a little abuse of language, to as *symbol-demapping*.

2) **Bit-demapping.** On the contrary, when we use an interleaver, the computation of the decoder input is more complicated because one needs first to reconstruct the symbol-nodes by the de-interleaver. This means that we need to recompose the symbol-node probabilities bit-by-bit: we have to compute first the probabilities of each bit of being "0" or "1", and then, with these probabilities, we can compute the probabilities of the symbol-nodes. The probability of each symbol of being one of the elements of the Galois field is obtained by the product of the probabilities of each component bit of being "0" or "1". This operation is referred, with a little abuse of language, to as *bit-demapping*.

   We note that the operation of passing from the QAM symbol-level to the bit-level and then to the $\mathbb{F}_q$-symbol level, called commonly *marginalization*, induces a demapping loss. On the other hand, the interleaving system shows very good performance that outperforms the uninterlevead systems and demonstrates that the introduced coding gain is greater than the marginalization loss.

6.2 Background and Notation

6.2.1 Modulation and Channel Model

Channel model. We assume a Rayleigh channel model, typical of a mobile/multipath environment. This model has been presented in Section 2.2.1 (page 17). The received symbol y_k is:

$$y_k = g_k x_k + z_k$$

where x_k stands for the k^{th} transmitted modulated symbol scaled by i.i.d. Rayleigh coefficients g_k, and z_k is the Additive White Gaussian Noise with variance σ^2. Moreover, we assume perfect Channel State Information at the receiver.

Modulation. We denoted by $\mathbb{X}$ the set of the symbols in the $\mathcal{M}$-QAM constellation. Each modulated symbol $x_k \in \mathbb{X}$ has $\mathtt{m}$ bits. We also denote by N the number of modulation symbols, and by $\mathtt{n}_b$ the number of coded bits.

6.2.2 Initialization of the Belief Propagation Decoding

We decode the LDPC codes with the Belief Propagation (BP) decoder [76] in which the messages that circulate on the graph are multidimensional vectors (Section 3.2.1, page 38).

The initialization messages are Likelihood probability weights, dependent on the channel statistics. For NB-LDPC codes, the decoder input consists of N Likelihood vectors $(P(s_n = a))_{a \in \mathcal{A}}$, where s_n denotes the n^{th} transmitted symbol, $n \in \{1, \ldots, N\}$.

In the following we discuss about the computation of the input Likelihood messages, with or without an interleaver. This paragraph follows from the Section 3.2.4, but now we

focus on viewpoint of the demapping quality. Let Y be the vector of received symbols and Γ be the Likelihood messages at the input of symbol-nodes.

Symbol-demapping. In the case of $\mathtt{m} = p$ and no bit-interleaver, there exists a one-to-one correspondence between the modulated and coded symbols, then the Likelihood vectors are directly derived from the received complex symbols (Eq. (3.20)):

$$\begin{array}{c} Y = [\, y_1, \dots y_k, \dots y_{\mathsf{N}} \,] \\ \downarrow \qquad \downarrow \qquad \downarrow \\ \Gamma = [\, \gamma_{s_1}, \ \dots \gamma_{s_n}, \dots \gamma_{s_N} \,] \end{array}$$

Bit-demapping. The other case corresponds when we use a bit-interleaver (or, more in general, when the size of the constellation does not match the size of the coded symbols). In such a case, an intermediate operation, denoted to as *marginalization*, is used to transform the constellation symbol likelihoods into bit-wise likelihoods. In the following framebox we represent the case of the interleaver. First, the marginalization operation transforms each received complex symbol into a vector of $\mathtt{m}$ soft bits, we denoted by Y_{bin} the vector that contains the $\mathtt{n}_b$ soft bits corresponding to the complex symbols (Eq. (3.21) and Eq. (3.22)). Afterwards, the de-interleaving operation rearranges Y_{bin} in order to obtain Γ_{bin}. Finally, the decoder input Γ is obtained according to the Eq. (3.23).

$$\begin{array}{c} Y = [y_1, \dots y_{\mathsf{N}}] \\ \Downarrow \text{ marginalization} \\ Y_{\text{bin}} = [(y_{1,1}, \dots y_{1,\mathtt{m}}), \dots, (y_{\mathsf{N},1}, \dots y_{\mathsf{N},\mathtt{m}})] \\ \Downarrow \text{ deinterleaving} \\ \Gamma_{\text{bin}} = [(\gamma_{s_{1,1}}, \dots, \gamma_{s_{1,1}}), \dots, (\gamma_{s_{N,1}}, \dots, \gamma_{s_{N,p}})] \\ \Downarrow \text{ Likelihood computing} \\ \Gamma = [\gamma_{s_1}, \ \dots \gamma_{s_N}] \end{array}$$

Concerning the non-interleaved system with $\mathtt{m} \neq p$, the only change is that the de-interleaver merely matches the sequence of bits, such that $y_{1,1}$ corresponds to $\gamma_{s_{1,1}}$, $y_{1,2}$ corresponds to $\gamma_{s_{1,2}}$, and so forth.

Hence, the bit-demapper induces a performance loss because of the marginalization effect. Nevertheless, in presence of an interleaver, the loss of information due to marginalization is counterbalanced by the gain that the bit-interleaver brings in the case of fading channels. We show in Section 6.5 that the diversity gain surpasses greatly the loss due to marginalization both in the waterfall and in the error floor regions. In the rest of the analysis, we consider only the case where $\mathtt{m} = p$, so that the marginalization transformation is used only when a bit-interleaver is employed.

6.2.3 Typology of errors

Consider $S \in \mathcal{C}$ a transmitted codeword of an LDPC code $\mathcal{C}$. At each decoding iteration, the decoder determines an estimation $\hat{S}$ of S according to the a posteriori symbol probability relative to each symbol-node.

Usually, the decoder stops in two situations:

i) at some iteration, the syndrome is verified $H\hat{S}^T = 0$, hence a codeword $\hat{S}$ is identified;

ii) the maximum number of iterations is reached, then $\hat{S}$ is computed from the messages at the last iteration (but $\hat{S}$ is not a codeword)

A *detected* error happens if $\hat{S}$ does not belong to the codeword set (that is in situation *ii*). If $\hat{S}$ belongs to the codeword set (situation *i*) but it is not the transmitted codeword ($\hat{S} \neq S$), the decoder makes an *undetected* error.

Undetected errors are due to codewords with low Hamming weight, which is one the weaknesses of the considered (2,d_c)-NB-LDPC codes. We will make in Section 6.5 a detailed study of the percentages of detected and un-detected errors in each situation.

6.3 Interleaving Algorithm

The effect of a bit-interleaver is to spread the coded bits in different modulation symbols, such that the bits composing the same coded symbol are affected by different fading factors. The advantage of using an interleaver is that the deep fading effects are mitigated because the soft bits come from the same QAM-symbol are distributed within more coded symbols after the de-interleaver. Put differently, the decoder input corresponding to each coded symbol is composed by the components of more QAM-symbol: this allows to render the exchanged messages more reliable and to easily recover the codewords.

The bit-interleaver can be seen as the construction of a superimposed regular $(\mathtt{m}, p)$-bipartite graph (from now on called simply *interleaving graph*) on the Tanner Graph. It connects the *modulation-nodes* x_k to the symbol-nodes s_n of a pre-designed Tanner Graph. The modulation-nodes are another type of nodes representing the modulated symbols in the interleaving graph.

In the interleaving graph [1], edges connect N modulation symbol-nodes x_k to N coded symbol-nodes s_n. Actually the degree of a modulation-node must be equal to the constellation order (number of bits per modulation-symbol) and the degree of a symbol-node must be equal to the order of the each modulation symbol (number of bits per symbol-node).

As said, since the number of coded and transmitted bits can be computed either as the number of bits within the N coded-symbols s_n, or as the number of bits within the N modulated-symbols x_k, we have that $\mathtt{n}_b = N\ p = \mathsf{N}\ \mathtt{m}$.

Figure 6.2 shows an example of a superimposed interleaving graph on the Tanner graph of Figure 3.1. The interleaver is represented as a block denoted as $\mathbf{\Pi}$; each modulation-node has $\mathtt{m} = 3$ outgoing edges and each symbol-node has $p = 3$ outgoing edges, so that $\mathtt{m} = p$. In order to distinguish the modulation-node they are drawn in blue.

Therefore, the aim of the a bit-interleaving design is to look for an *interleaving pattern* that contains the scrambled bit positions in the interleaving graph.

Though random interleaving shows already good performance as demonstrated in the Section 6.5, we have devised an algorithm to optimize the interleaver design, which even more improves the performance, especially in the error-floor region. The bit-interleaver optimization algorithm is presented in the next section.

1. Even if we assume $\mathtt{m} = p$ (and then $\mathsf{N} = N$), we will give a generalized version of the interleaving algorithm in order to apply it also in case that $\mathtt{m} \neq p$.

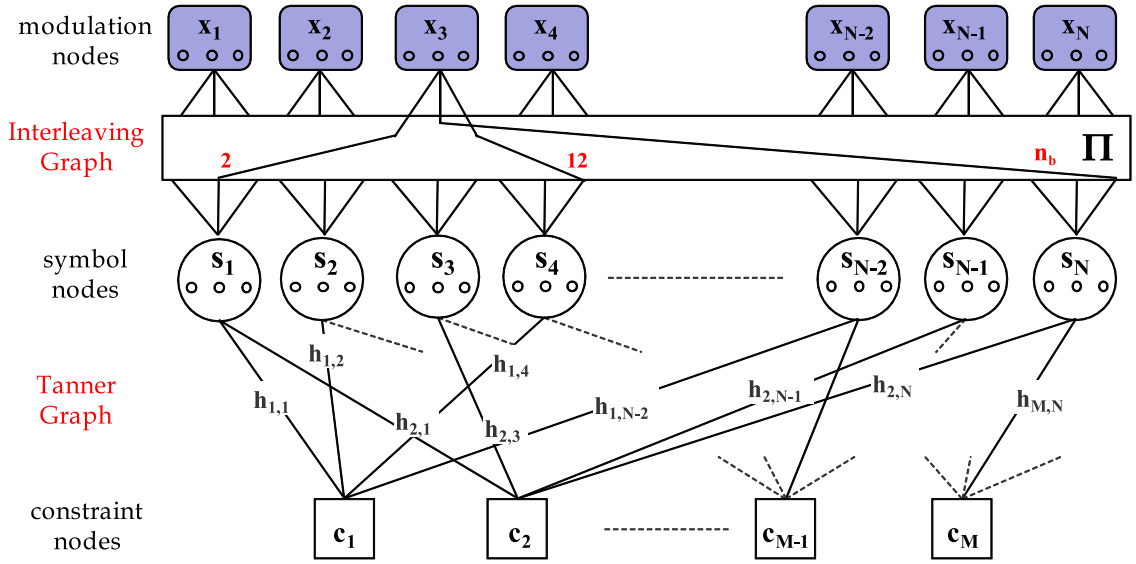

Figure 6.2: Global graph

6.4 PEG Optimization Interleaving

Our bit-interleaver design is inspired from the Progressive Edge-Growth (PEG) method [32] used for constructing Tanner graphs with large girths that progressively connects symbols and constraint-nodes. For this reason, from now on, our optimization algorithm will be referred to as PEG interleaving algorithm.

We have already discussed about the advantages of the PEG algorithm in Section 3.4.1 (page 52) for constructing large girth LDPC graph. The proposed PEG interleaving algorithm is similar to the PEG construction. The PEG interleaving algorithm maps in an efficient way the modulation-nodes to the symbol-nodes. Good connections are meant to give the largest possible girth to both the LDPC Tanner graph and the interleaving graph. Starting with the knowledge of $\mathtt{m}$, p and of the LDPC Tanner graph, the algorithm connects each modulation-node to $\mathtt{m}$ symbol-nodes. The rationale behind the optimization algorithm is to find, for each modulation-node, the $\mathtt{m}$ most distant coded symbol-nodes (from a topological distance point of view), and therefore to build connections in the interleaving graph in order to obtain the largest girth. It should also be noted that the bit-interleaver design is code-dependent. As a matter of fact, the girth computation during the interleaver design takes into account the topology of the already designed LDPC graph. It results in particular that the bit-interleaver built with our algorithm is actually *matched* to a particular NB-LDPC code, which explains the further performance gains that we observe.

PEG interleaving algorithm. We now explain the principles of the PEG interleaving algorithm. Let d_n denotes the coded symbol-node degree and d_k denotes the modulation-node degree. Before the algorithm starts, all the degrees are set to 0. During the algorithm execution, the current node degrees represent the number of established connections for the nodes. Thus, for a coded symbol-node s_n, $n \in \{1, \ldots, N\}$, the degree range is $0 \leq d_n \leq p$; for a modulation-node x_k, $k \in \{1, \ldots, \mathtt{N}\}$, the degree range is $0 \leq d_k \leq \mathtt{m}$.

Let x_k be a modulation-node to be connected. The PEG interleaving algorithm chooses

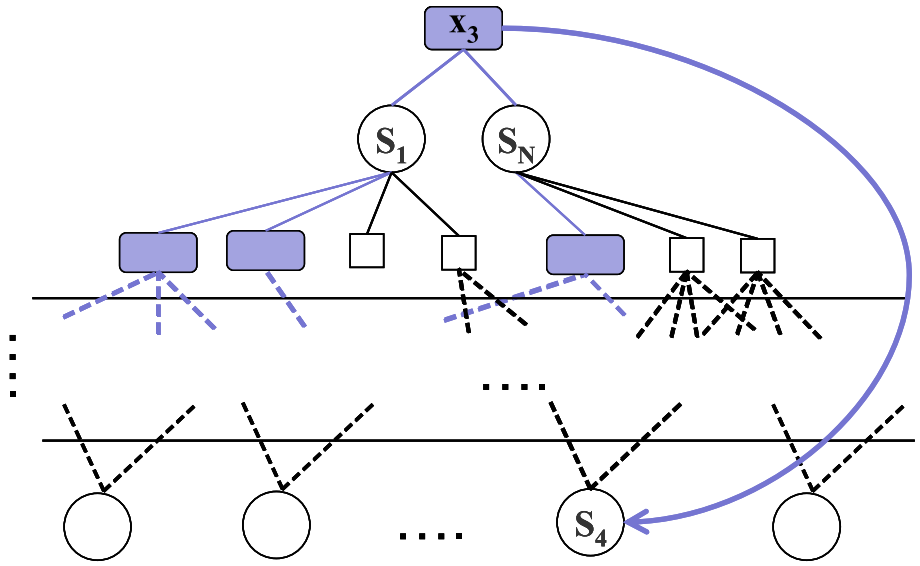

Figure 6.3: Expansion of the Tanner and interleaving graphs

the first connection between x_k and s_w. s_w is a randomly chosen symbol-node among the available ones with the lowest current degree. Then, for the second edge selection, we expand both the Tanner and the interleaving graphs through the three types of nodes taking into account the new connection. Once the graph expansion is complete, the bottom of this graph is the set of symbol-nodes $\{s_z\}$ that are the most distant symbol-nodes from s_w. Hence, the algorithm can connect the modulation-node x_k to one of the coded symbol-node $s_z \in \{s_z\}$ with the lowest degree. A new connection is thereby chosen, and the algorithm goes to the next edge selection, by performing the same steps - graph expansion and node selection. The procedure is iterated until x_k reaches the correct degree $d_k = \mathtt{m}$. The algorithm stops when all the modulation-nodes are connected.

Note that with this PEG interleaving algorithm, only the coded symbol index is important for the global girth of the graph, and not the location of the bit *inside* the coded symbol binary map. Actually, an extra local scrambler could be added — at the coded symbol level — without impacting the girth of the global graph. In our simulations, however, this extra local scrambler did not impact significantly on the error rate performance.

In Figure 6.3 we have illustrated the proposed PEG interleaving algorithm. We assume that the first connection between the modulated-node x_3 and the symbol-node s_1 is randomly chosen. We also admit a partial connections of modulated and coded symbols. Then, for the last connections of x_3, the Tanner graph and the interleaving graph have been developed all together till the expansion is complete. After that, in the bottom of this expansion we are sure to find the most distant symbols from x_3 and, for instance, we proceed with the connection between x_3 and s_4. Finally, a random selection assign to x_3 the coded bit positions $(2, n_b, 12)$, cf. Figure 6.2.

Although the above described construction could be adapted to any NB-LDPC code, including codes with irregular node distributions and $\mathtt{m} \neq p$, we restricted our study to regular (2,d_c)-LDPC codes with $\mathtt{m} = p$. For these codes, the error rate simulation results and the study of detected vs. undetected errors are conducted in the next section.

6.5 Simulation Results

In this section, we present the simulation results for the interleaved and un-interleaved NB-LDPC codes. Motivated by the asymptotically good thresholds that grows with the order field, we have focused on LDPC code alphabets $\mathbb{F}_{64}$ and $\mathbb{F}_{256}$. It has been shown that, for those high orders, the ultra-sparse NB-LDPC codes with node degree $d_s = 2$ is the best choice. We have therefore simulated short-length LDPC codes (each code design presents only ∼100 coded symbols) for those two different alphabets and two different coding rates:

1) Regular (2,6)-NB-LDPC codes corresponding to a rate $R = 2/3$
2) Regular (2,12)-NB-LDPC codes corresponding to a rate $R = 5/6$

These NB-LDPC code Tanner graphs are designed with PEG method [32] (girth $g_{\min} = 6$) and decoded by using a Belief-propagation decoder [16]. As said in the previous section, the size of the alphabet is the same as the size of modulation constellation: codes defined over $\mathbb{F}_{64}$ are transmitted using the 64-QAM modulation ($\mathtt{m} = p = 6$), whereas codes defined over $\mathbb{F}_{256}$ are transmitted using the 256-QAM modulation ($\mathtt{m} = p = 8$).

Each LDPC code is simulated on the Rayleigh fading channel with three possible transmission systems: first, a non-interleaved system with direct mapping from the coded to the modulation symbols, then with a random bit-interleaver and finally with our PEG-optimized interleaver.

Marginalization loss. In Figure 6.4 and 6.5 we have drawn two black curves both ones with empty circles. Both curves correspond to the no-interleaved system, but in order to evaluate the marginalization loss we have demapped the same codes with a symbol demapper and a bit demapper. The solid line represents the performance of the system with a symbol demapper, whereas the dashed line represents the performance of the system with a bit demapper. In these cases, the bit demapper degrades the performance with a marginalization loss included between 0.7 and 0.8dB .

Frame Error Rates. The curves in Figure 6.4 represent the Frame Error Rates (FER) of the considered $(2,d_c)$-LDPC codes defined over $\mathbb{F}_{64}$ for small codewords ($\mathtt{n}_b = 612$ bits, $N = 102$ coded symbols) modulated with a 64-QAM. A significant performance gain can be observed both in the waterfall and in the error floor regions in the presence of a bit-interleaver. This shows that the diversity gain brought by the bit-interleaving surpasses greatly the information loss due to symbol-to-bit marginalization.

As expected, the PEG interleaved system (represented by □) achieves roughly the same gain as random interleaving (represented by $*$) in the waterfall region, but shows also an improved error-floor. This was our goal of building bit-interleavers that optimize the girth of the global graph.

In Figure 6.5 we show the performance of the codes defined over $\mathbb{F}_{256}$ with codelength $\mathtt{n}_b = 816$ ($N = 102$ coded symbols) and modulated with a 256-QAM. As can be seen, when the size of alphabet grows, bit-interleavers still gain in the error floor region, but the gap between the two kinds of interleavers vanishes. Generally, the increase of the size of the alphabet makes the gap between the PEG and a random interleavers reduced. In this figure we can also observe the gap between the interleaved and the un-interleaved curves in the waterfall region: this is due to the different demapping and that the marginalization loss is initially greater than the diversity gain.

Percentage of error detection. Now, let us discuss the effect of bit-interleaving and our optimized construction for the detection of frame errors. We have drawn in Figure 6.6 and 6.7 the percentage of detected frame errors with respect to the FER, for the (2,6)-LDPC codes defined in $\mathbb{F}_{64}$ and in $\mathbb{F}_{256}$, respectively.

As it was expected, this study confirms that the better performance in the error floor region results also from a better detection of frame errors. More precisely, bit-interleaving on a Rayleigh channel helps the decoder to avoid convergence to low-weight codewords, therefore improving at the same time the performance and the probability of error detection. This last feature is very interesting since in most wireless mobile transmissions, a link adaptation strategy implementing retransmission of detected wrong frames is generally used (ARQ or Hybrid-ARQ).

Note that even in the case of codes in $\mathbb{F}_{256}$, our bit-interleaving optimization shows an interesting gain in detected frame errors (100% for all FER simulated) compared to the random-interleavers, although the error rates were the same (see Figure 6.5 and Figure 6.8).

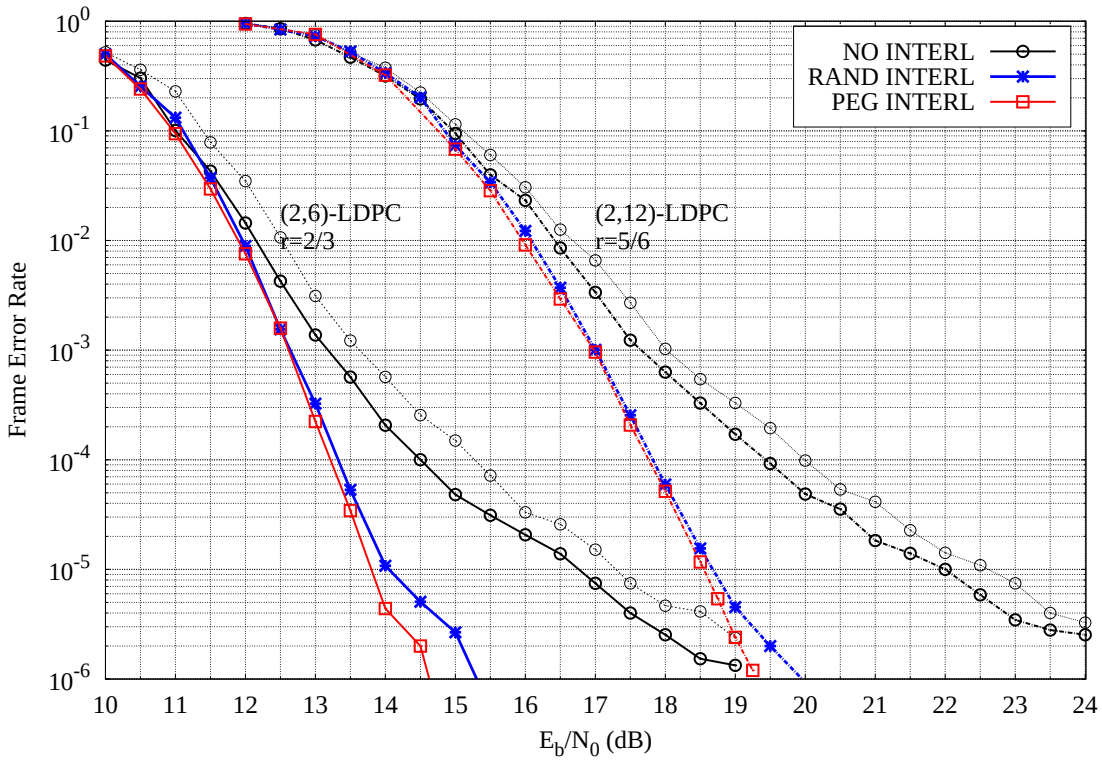

Figure 6.4: Frame Error Rates for (2,6) and (2,12) NB-LDPC codes, $n = 612$, $\mathbb{F}_{64}$, 64-QAM

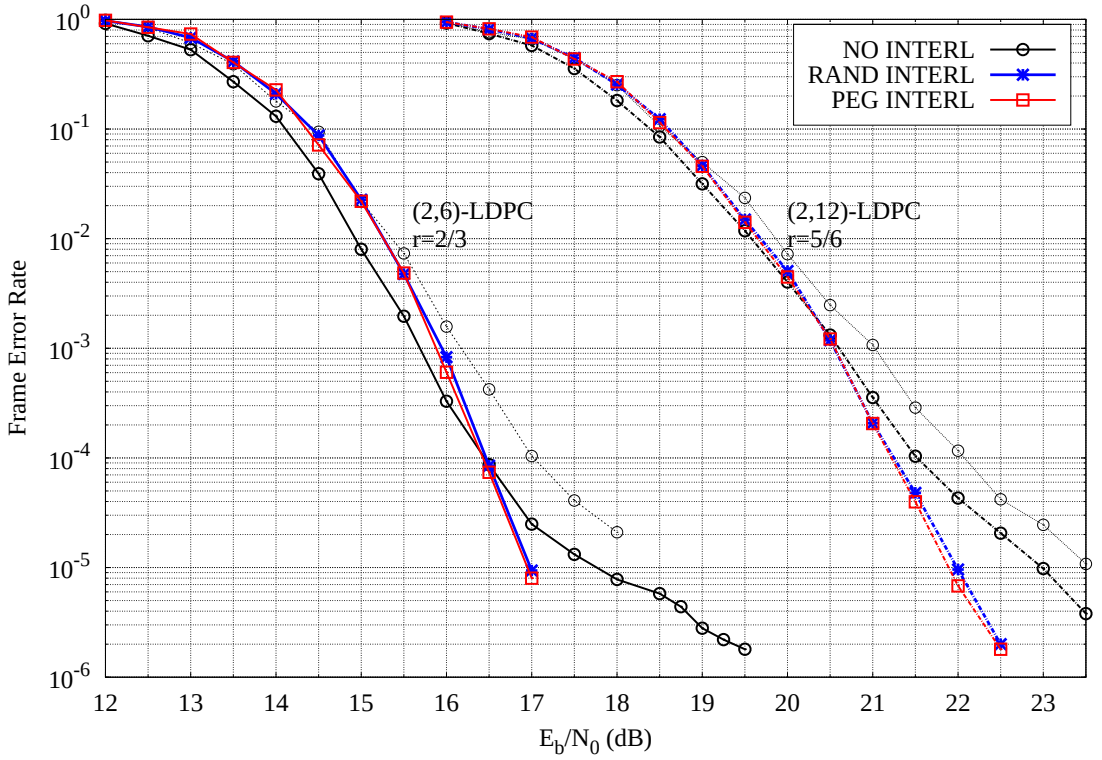

Figure 6.5: Frame Error Rates for (2,6) and (2,12) NB-LDPC codes, $n = 816$, $\mathbb{F}_{256}$, 256-QAM

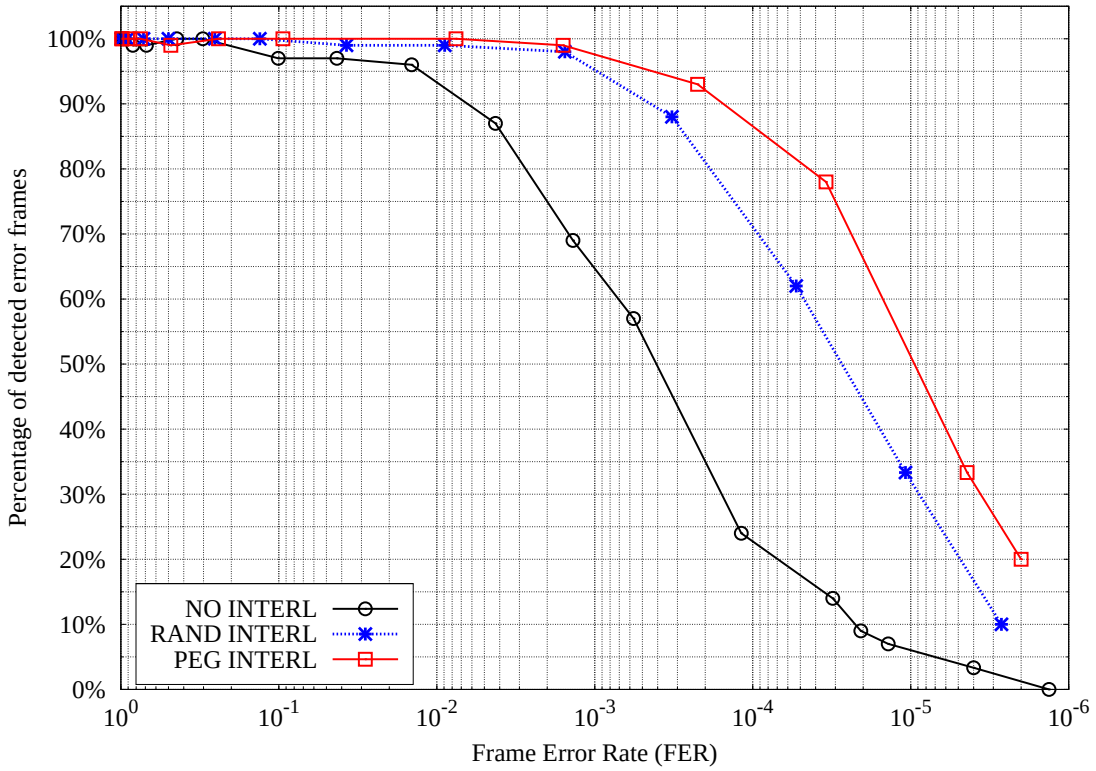

Figure 6.6: Percentage of detected frame errors for a regular (2,6)-NB-LDPC code, $n = 612$, $\mathbb{F}_{64}$, 64-QAM

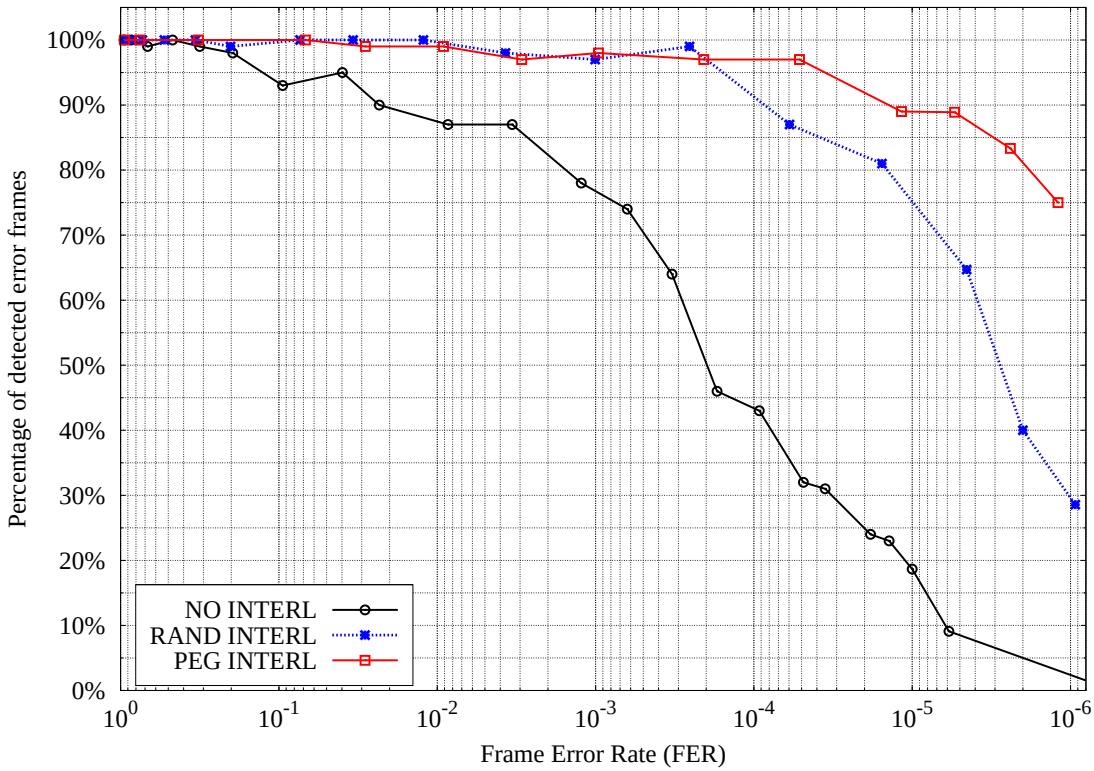

Figure 6.7: Percentage of detected error frames for a regular (2,12)-NB-LDPC code, $n = 612$, $\mathbb{F}_{64}$, 64-QAM

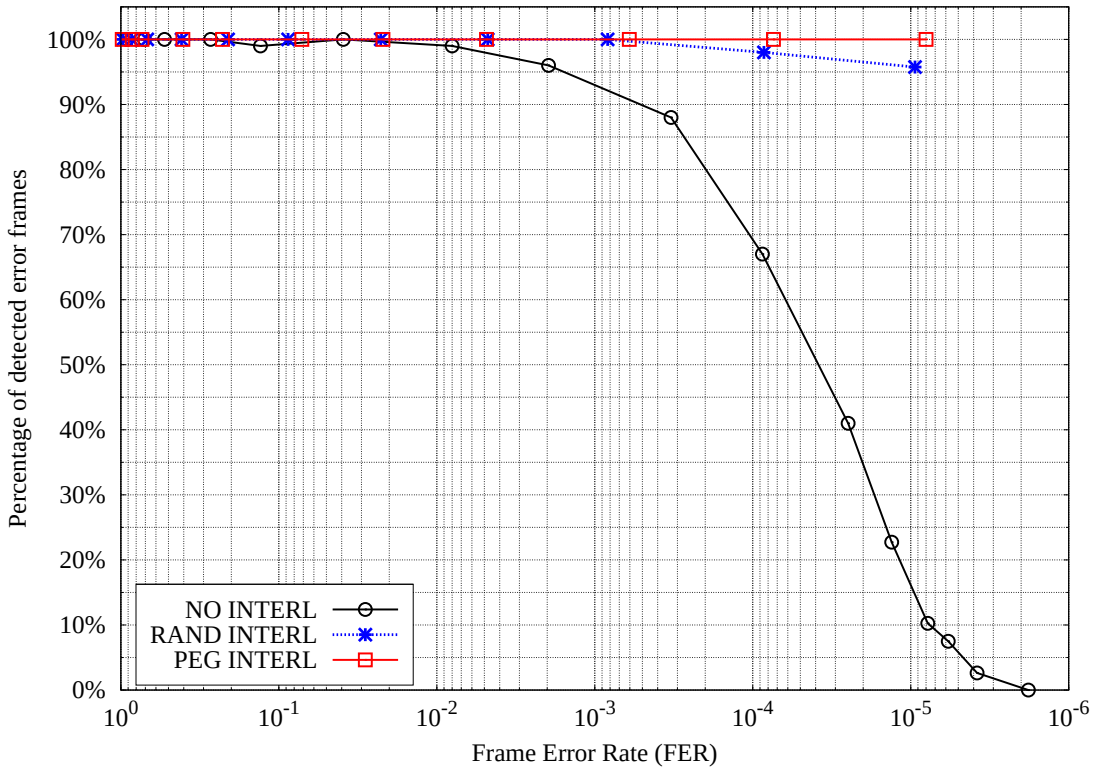

Figure 6.8: Percentage of detected error frames, regular (2,6)-NB-LDPC code, $n = 816$, $\mathbb{F}_{256}$ 256-QAM

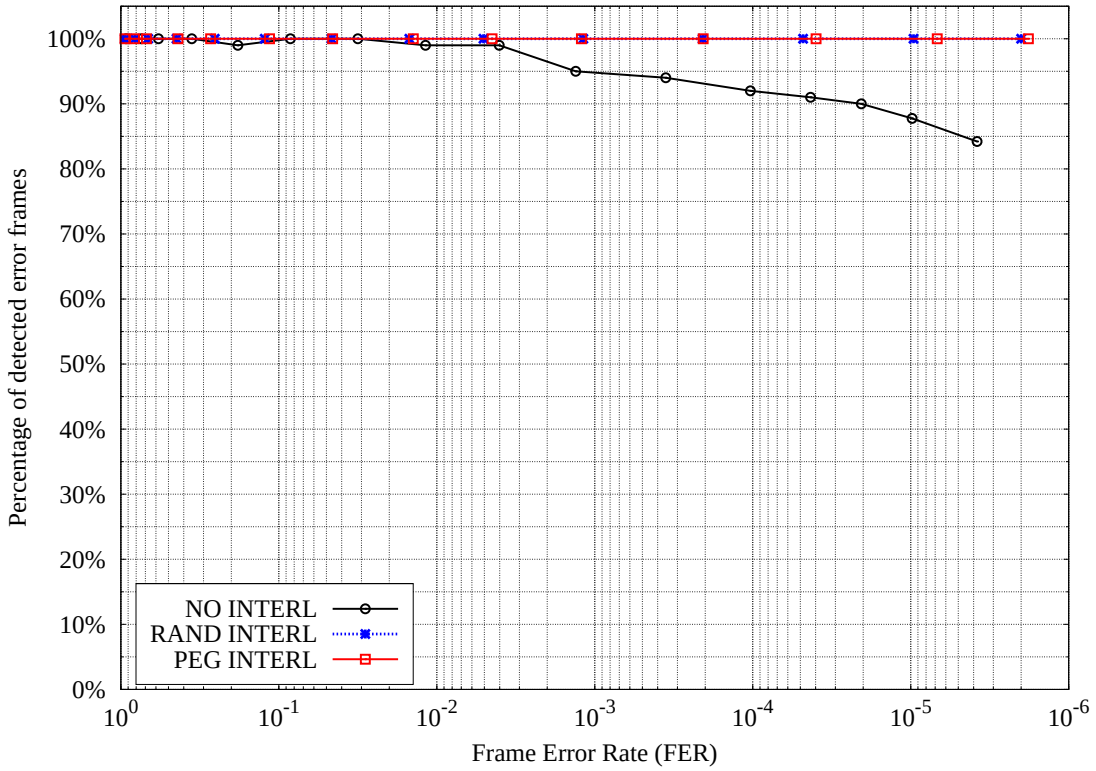

Figure 6.9: Percentage of detected error frames, regular (2,12)-NB-LDPC code, $n = 816$, $\mathbb{F}_{256}$ 256-QAM

6.6 Conclusions

When we transmit in a multipath environment, as in case of an indoor wireless system or a mobile radio, the channel is usually modeled as a Rayleigh model. Coded symbols transmitted over a Rayleigh channel are affected by different fading factors; hence especially for codes with small minimum distance, specific noise realizations could make the decoder converge to erroneous codewords.

Moreover, deep fading could make some codeword symbols totally unrecoverable in case of one-to-one correspondence between modulated and coded symbols, leading to a poor system performance.

In order to avoid this phenomenon, binary diversity can be exploited by using a bit-interleaver module placed between the encoder and the modulator. The bit-interleaver introduction mitigates the fading effects and reduces drastically the error convergence toward close codewords from the transmitted ones.

A random interleaver and an optimized interleaver have been analysed by running simulations over short size regular $(2, d_c)$-NB-LDPC codes. The optimized interleaving algorithm is inspired from the Progressive Edge-Growth (PEG) method and it ensures maximum girth of the global graph.

Although the bit-demapping required by the interleaved system leads to a marginalization loss, it has been demonstrated that the use of an interleaver ensures improved frame-error probabilities compared to a system without it.

Additionally, in all considered cases, the optimized interleaver showed an even better gain with respect to the random interleaver, as far as performance and error detection rates are concerned.

In this chapter we have adapted the interleaving to a given LDPC graph. We have demonstrated that a good mapping between the modulated and the coded symbols yields to improve the performance. As a perspective of a future work, we think that even better performance can be obtained if the PEG construction of the code's bipartite graph is merged with the proposed interleaving algorithm. Thus, in order to have larger girth, the design of the LDPC codes may be developed on the whole graph that includes both the modulation and the symbol-nodes.

Chapter 7

Full-Diversity NB-LDPC Codes over Block-Fading Channels

The block-fading channel model is widely used in the design and analysis of wireless communication systems. In the block-fading (BF) channel model, during the transmission of a codeword the channel is constant within blocks and changes independently from one block to another. This model fits many delay sensitive systems employing frequency hopping (FH), Orthogonal Frequency Division Multiplexing (OFDM), time diversity techniques in time selective channels etc.

BF channels are non-ergodic and their Shannon capacity is zero. A meaningful information-theoretic tool is the *outage probability*, which provides the best slope for the frame error rate (FER) curve of any coded modulation scheme, in the asymptotic limit of the SNR. The slope of the outage probability curve depends on the coding rate. It becomes increasingly steeper as the coding rate decreases from 1 to $1/n_f$, where n_f is the number of fading blocks, and remains unchanged when the coding rate further decreases below this value. Codes with coding rate $r = 1/n_f$, whose FER curve has (asymptotically) the same slope as the outage probability curve are called *full diversity* codes.

In this chapter, we introduce a *flexible* coding scheme that achieves full-diversity over the block-fading channel. The particularity of our approach is to rely on non-binary LDPC codes coupled with multiplicative non-binary codes, so that to easily adapt the coding rate to the number of fading blocks. A simple combining strategy is used at the receiver end before the iterative decoding, so that decoding can be performed on the same mother code. As a consequence, the decoding complexity is the same irrespective of the number of fading blocks. Moreover, we propose an optimization of the non-binary multiplicative coefficients according to the Euclidian distance between the constellation points. Simulation results demonstrate that the proposed technique yields a close-to-outage performance, with significant gains with respect to the binary LDPC codes from the literature.

The rest of the chapter is organized as follows. We motivate this work in Section 7.1. Some backgrounds about the channel model and the coding diversity are discussed in Section 7.2. In Section 7.3 we show the limit of interleaving for this channel model. We discuss several coding issues for the BF channel in Section 7.4. The proposed coding scheme for the block-fading channel is presented in Section 7.5. Performance analysis and simulation results are shown in Section 7.6. Finally, Section 7.7 concludes this chapter.

7.1 Motivations

The block-fading channel [5, 51] was introduced in order to model channels involving slow time-frequency hopping (used in wireless cellular networks), or multicarrier modulation using orthogonal frequency division multiplexing. More generally, this simplified model comes out to be very useful in code designs for slow-varying fading environments.

Coding on block-fading channel. The main characteristic that distinguishes the block-fading model from other channel models is that the former is non ergodic. Consequently, in case that an error-correcting code is used to communicate over the channel, channel realizations corresponding to different codewords have different statistical properties, even if the codewords are allowed to be arbitrarily long. It turns out that there is a non-zero probability, referred to as *outage probability*, that the instantaneous channel capacity (corresponding to the channel realization during the transmission of one single codeword) is smaller than the current coding rate. This also implies that the Shannon capacity of the channel equals 0.

According to the above discussion, the slope of the outage probability curve depends on the current coding rate. It becomes increasingly steeper as the coding rate decreases from 1 to $1/n_f$, where n_f is the number of fading blocks, and remains unchanged when the coding rate further decreases below this value. We also note that in the asymptotic limit of the SNR, the slope of the frame error rate (FER) curve of any coded modulation scheme is lower bounded by the slope of the outage probability curve (with same coding rate). Put differently, the FEC curve decreases, at best, as quickly as the outage probability curve. Moreover, if the code is long enough, the outage probability provides actually a lower bound of the FER performance [1]. As a consequence, the performance of a code over the block-fading channel is usually evaluated in terms of the slope of the FER curve and of the SNR gap to the outage probability. More details will be given in Section 7.2.1.

Binary diversity. Following the study conducted in the previous chapter, concerning binary interleavers for NB-LDPC codes over fast fading channels, we will first study the design of a binary interleaver for NB-LDPC codes transmitted over the BF channel. In order to analyze if the interleaving can bring diversity gain also in case of BF channel we implement an interleaving adapted to this channel model. In this case the interleaving pattern is created so that the coded bits are scattered homogeneously within the fading blocks. However, we will see that the interleaving technique does not give the expected code diversity, especially for a small number of fading blocks.

Hence, regrettably, the binary interleaving technique does not provide good results, as in the case of fast-varying fading channel, where this technique allows to efficiently mitigate the fading effects. The poor performance is ascribed to the fact that the interleaver cannot mitigate the deep fading blocks. In Section 7.3 we will give more details about the implemented interleaver used in the BF transmissions and we show by simulation results that this method is not suitable for such a channel model.

Full-diversity codes. Consider a code $\mathcal{C}$ that is used to communicate over a BF channel with n_f fading blocks. We assume that ML decoding is performed at the receiver, and denote by $P_e(\mathcal{C}, \rho)$ the (word) error probability for a given SNR value ρ.

1. However, for very short codes, the FER curve might be below the outage probability curve!

1) The SNR exponent of the error probability is defined by:

$$d^* = \lim_{\rho \to +\infty} \frac{-\log P_e(\mathcal{C}, \rho)}{\log \rho}$$

2) Under the ML decoding (as assumed above!), $d^* = d_H \in [0, n_f]$, where d_H be the minimum blockwise Hamming distance. Consequently d^* is also referred to as *block diversity* of the code $\mathcal{C}$.

3) A code is said to be *full-diversity* if $d^* = n_f$.

For full-diversity codes $r \leq 1/n_f$ (this follows from the Singleton bound [33, 38, 48]) and the gap between the Frame Error Rate (FER) and the outage probability curves is maintained constant for arbitrarily large SNR values.

The block-erasure channel (BlEC) can be seen as a particular (and limiting) case of the block-fading channel and corresponds to the case in which a block is completely erased with probability p_e or correctly received with probability $1 - p_e$, independently from a block to another. Considering this last model, a full-diversity code has the particularity that any codeword can be entirely recovered if at least one of the n_f blocks is correctly received. By intuition, a full-diversity design for the BlEC will also performs well over the BF channel, where codewords go through deep and weak block fades. Note also that the above definition depends on the considered decoder. But codes that fulfill the full-diversity condition (3) under the ML decoder, might not be full-diversity for other decoders (*e.g.* belief-propagation decoder). More details will be given in Section 7.2.3.

Full-diversity coding schemes. Several coding schemes for the block-fading and the block-erasure channels have been already proposed in the literature, starting from the early 90's with the analysis of convolutional codes with random and periodic interleavers over the block-erasure channel [40]. More recently, interleaving techniques for parallel turbo codes over the block-fading channel have been analysed in [7], and a general construction of turbo-like codes, obtained by concatenating linear binary encoders through interleavers, has been proposed in [33].

Low-Density Parity-Check codes with belief-propagation decoding have attracted an increased interest over the last decade, due to the fact that they can be optimized to perform very close to the channel capacity, for a broad class of ergodic channel models [60]. In case of block-fading channels, a class of full-diversity binary LDPC codes, referred to as *root-LDPC* codes, has been proposed in [8]. A more general definition of root LDPC is introduced in Section 7.4; following this definition we propose a general version of root LDPC codes by defining triangular parity-check matrices for a limited number of fade blocks.

NB repetition symbols. Recently, in [36] Kasai *et al.* have identified an advantage of NB-LDPC codes, consisting on the design of flexible coding transmission in a very simple, though efficient way. The proposed approach is to concatenate non-binary multiplicative codes to a mother NB-LDPC, which leads to extra redundancy built from "non-binary repetition" symbols[2]. When the multiplicative coefficients of the NB-repetition symbols are properly designed, it results that the coding gain is greatly increased compared to binary repetition coding, especially when $q \geq 64$.

2. If s is a non-binary symbol of a transmitted codeword, a *non-binary repetition* symbol is a symbol $s' = hs$, where h is a non-zero element of the Galois field on which the code is defined. If $h = 1$, then $s' = s$, which corresponds to the classical repetition coding, referred to as the *binary repetition* coding.

In our work we investigate the use of non-binary LDPC codes over block-fading channels. We make use of the concatenation of NB-LDPC codes and non-binary repetition codes to design our full-diversity coding scheme, as presented in Section 7.5. Our approach is based on NB-LDPC codes and the recently introduced technique of multiplicative non-binary coding [35, 36]. Using a root NB-LDPC code with rate 1/2 as a mother code and non-binary repetition symbols, we design codes with rate $1/n_f$ that achieve full-diversity over channels with $n_f \geq 2$ fading blocks. The receiver collects the received codeword of the root NB-LDPC code and the non-binary repetition symbols, and combines them before the iterative decoding. Hence, the iterative decoding complexity is the same irrespective of the fading blocks number n_f, while combining the codeword and the additional non-binary repetition symbols brings an effective coding gain.

Moreover, the optimization of the edge labels of the non-binary codes is investigated in order to improve the coding gain. We proceed with two optimization algorithms, the first one for the non-zero entries of the mother parity-check matrix, that are optimized according to [17] (see also the algorithm at page 56), and the second optimization algorithm concerning the multiplicative coefficients of the NB-repetition symbols. In *loc. cit.* it has been also analysed an optimization of the NB-repetition codes similar to the parity-check matrix optimization. In this work we propose a different optimization technique, based on the Euclidian distance between the corresponding symbols of the complex constellation. This optimization yields an effective gain (∼0.5dB) with respect to the random coefficients.

In Section 7.6 we show simulation results and we demonstrate that our method outperforms root LDPC codes, while preserving the same decoder in all the transmissions.

7.2 Background and Notation

7.2.1 Block-Fading Channel Model

We consider the block-fading channel model with n_f fading blocks, each one of length L complex symbols. The fading is flat, constant on each block and i.i.d. on different blocks. The baseband equivalent channel model is given by:

$$y_n = \sqrt{\rho} g_j x_n + z_n, \tag{7.1}$$

where $n \in \{1, \ldots, n_f L\}$, $j = \lceil \frac{n}{L} \rceil \in \{1, \ldots, n_f\}$, x_n and y_n are the n^{th} input and output symbols, g_j is the fading coefficient on block j, and $z_n \sim \mathcal{N}_{\mathbb{C}}(0, 1)$ is the i.i.d. circular complex Gaussian noise with unitary variance. We remind that the transmitted symbols x_n belong to a complex signal set $\mathbb{X}$ (e.g. quadrature amplitude modulation), with cardinality $|\mathbb{X}| = 2^{\mathtt{m}}$ and unit energy, i.e. $2^{-\mathtt{m}} \sum_{x \in \mathbb{X}} |x|^2 = 1$.

In order to have a one-to-one correspondence between coded and modulation symbols, in our work we assume that the constellation order is equal to the Galois field order, hence $\mathtt{m} = p$; in this case the fading coefficients affect directly the corresponding symbol-nodes [3].

Rayleigh fading corresponds to the case $g_j \sim \mathcal{N}_{\mathbb{C}}(0, 1)$, or equivalently, the envelop

$$|g_j| = \sqrt{\Re(g_j)^2 + \Im(g_j)^2}$$

3. In the previous chapters, we used the index k for denoting the k^{th} complex symbol x_k. Since we assume that $\mathtt{m} = p$, we can directly use the index n, so that we will denote the modulated symbol by x_n. This is also valid for the received symbol y_n and the noise component z_n. For more detail, see the Section 6.2.2 (page 104).

is Rayleigh distributed and the real and imaginary parts are Gaussian random variable with variance $\sigma^2 = 1/2$. Hence, the fading coefficients are normalized, that is, $\mathbb{E}[|g_j|^2] = 1$. It follows that the parameter ρ from Eq. (7.1) is equal to the average received SNR, while $\rho|g_j|^2$ represents the instantaneous SNR on block j.

We shall further assume that the average received SNR ρ and the fading coefficients $\mathbf{g} = (g_1, \ldots, g_{n_f})$ are perfectly known at the receiver, but not known at the transmitter. For a fixed ρ value, $I_\rho(\mathbf{g})$ denotes the instantaneous mutual information between channel input and output, assuming that the channel input is uniformly distributed over the complex signal set $\mathbb{X}$. Hence, $I_\rho(\mathbf{g})$ is a random variable taking values in $[0, \mathtt{m}]$ and its cumulative distribution function is referred to as the *information outage probability*. Precisely:

$$P_{\text{out}}(\rho, R) = \Pr\left(I_\rho(\mathbf{g}) < R\right) \tag{7.2}$$

is the probability of the instantaneous mutual information $I_\rho(\mathbf{g})$ being less than the information rate value $R \in]0, \mathtt{m}]$ bits per channel use. Therefore, in the asymptotic limit of L, $P_{\text{out}}(\rho, R)$ gives the optimal word error rate (WER) [4] value, as a function of ρ, for any code with alphabet $\mathbb{X}$ and information rate R.

7.2.2 Block-Erasure Channel Model

We consider a block-erasure channel (BlEC) with the same complex input alphabet $\mathbb{X}$. As for the block-fading channel, the transmitted data is split into n_f blocks of L complex symbols, but any block is either perfectly received (without noise) or erased. This can be conveniently expressed by the following formula:

$$y_n = \sqrt{\rho} g_j x_n, \tag{7.3}$$

where $n \in \{1, \ldots, n_f L\}$, $j = \lceil \frac{n}{L} \rceil \in \{1, \ldots, n_f\}$, x_n and y_n are the n^{th} input and output symbols, respectively, and $g_j \in \{0, 1\}$ is the channel coefficient on block j. The channel "erasure" state corresponds to $g_j = 0$ (we assume perfect channel state information at the receiver) and the block-erasure probability is defined by $p_e = \Pr(g_j = 0)$ and it is constant for all blocks $j \in \{1, \ldots, n_f\}$. We also define the *average received SNR* by $\rho = \dfrac{1}{p_e}$

7.2.3 Coding Diversity

Let $\mathcal{C}$ be a block code, with alphabet $\mathbb{X}$ used to communicate over the block-fading channel (Eq.(7.1)). Since any codeword spans exactly n_f fading blocks and that $\mathtt{m} = p$, it follows that the codelength verifies $N = n_f L$ (complex symbols) [5]. The *information rate* of $\mathcal{C}$, in bits per complex symbol, is denoted by $R \in]0, \mathtt{m}]$, whereas the *coding rate* is defined as $r = R/\mathtt{m} \in]0, 1]$. The *block diversity* of $\mathcal{C}$ is defined as the SNR exponent of the error probability $P_e(\mathcal{C}, \rho)$ of $\mathcal{C}$ [33]:

$$d^* = \lim_{\rho \to +\infty} \frac{-\log P_e(\mathcal{C}, \rho)}{\log \rho} \tag{7.4}$$

The blockwise Hamming weight of a codeword S is the number of blocks j on which S is non zero (that is, it has at least one non-zero entry). The minimum blockwise Hamming

4. Or the Frame Error Rate (FER).

5. In the previous chapters we have denoted by N the number of complex symbols. In this chapter the number of the complex symbols is equal to the number of the codelength because $\mathtt{m} = p$. Thus, we do not distinguish between N and N.

distance $d_H \in [0, n_f]$ of a code $\mathcal{C}$ is the minimum number of blocks on which two given codewords differ. Equivalently d_H is equal to the minimum blockwise Hamming weight of a non-zero codeword.

Under the optimal Maximum-Likelihood (ML) decoding, and assuming Rayleigh fading and a sufficiently large L value, the SNR exponent d^* is equal to the minimum blockwise Hamming distance d_H [47]. Consequently $d^* \in [0, n_f]$ and we say that $\mathcal{C}$ achieves *full-diversity* if $d^* = n_f$. By the Singleton bound [38] we have

$$d^* \leq 1 + \lfloor n_f(1-r) \rfloor,$$

yielding a compromise between achievable block diversity and coding rate. In particular, the coding rate of a full-diversity code must verify $r \leq \frac{1}{n_f}$.

7.2.4 BP Decoder Initialization

In this section we present the computation of the probability distributions used for the initialization of the BP decoder presented in Section 3.2.1 (page 38). In this work we focus on the case when the code field $\mathbb{F}_q$ and the complex signal set $\mathbb{X}$ have the same cardinality (hence, $\mathtt{m} = p$), this allows us to use a symbol-demapping (except when a bit-interleaving is used between encoder and modulator; we shall refer to Section 6.2.2 for more details about the initialization of the BP decoder in this case).

Let μ be a bijective function mapping the code field into the complex signal set:

$$\mu : \mathbb{F}_q \to \mathbb{X} \tag{7.5}$$

A codeword $S = (s_1, \ldots, s_N)$ is mapped into a vector of complex symbols $(x_1, \ldots, x_N)$, where $x_n = \mu(s_n)$, which is transmitted over the block-fading channel (7.1).

According to the discussion in Section 3.2.4 (page 44) about the decoder initialization, we compute the probability $P_n(a) = \Pr(s_n = a \mid y_n)$ of the transmitted symbol s_n being equal to a conditioned on the received complex symbol y_n. Since we assumed perfect CSI at the receiver this probability is:

$$P_n(a) = \gamma \exp(-|y_n - \sqrt{\rho} g_j \mu(a)|^2), \tag{7.6}$$

where j is the block index corresponding to codeword index n, and γ is a normalization constant such that $\sum_a P_n(a) = 1$.
These probability distributions are used for the initialization of the BP decoder and are iteratively updated according to the sum-product rules.

7.3 Interleaving in BF Channel

In the previous chapter we have applied the binary interleaving technique in case that a NB-LDPC code is used to communicate over a fast fading channel. In this section we investigate a binary interleaving that spreads the coded bits over the different fading blocks of a BF channel. We will show however that this technique does not efficiently mitigate the fading effects in this case.

Binary-Interleaving for BF channels. The proposed interleaving splits up each symbol among the different fading blocks. Thus, at the receiver end, after the de-interleaving, each

symbol receives contributions from the different fading blocks of the channel. An example of the interleaving method is shown in Figure 7.1, where we have represented the coded symbols by circles and the complex modulated (QAM) symbols by rectangles. The number of fading blocks n_f is equal to 3 and they are represented by #1,#2 and #3. The code is defined over $\mathbb{F}_8$ and each QAM symbol has $\mathtt{m} = 3$ bits. There is a rationale behind this choice: we fixed $p = n_f = 3$ because the decoder input for each symbol-node is computed from exactly 3 bits, each one modulated by a complex (QAM) symbol in a different fading block. By sake of simplicity, in the following we shall assume that $p = n_f$ or that p is a multiple of n_f.

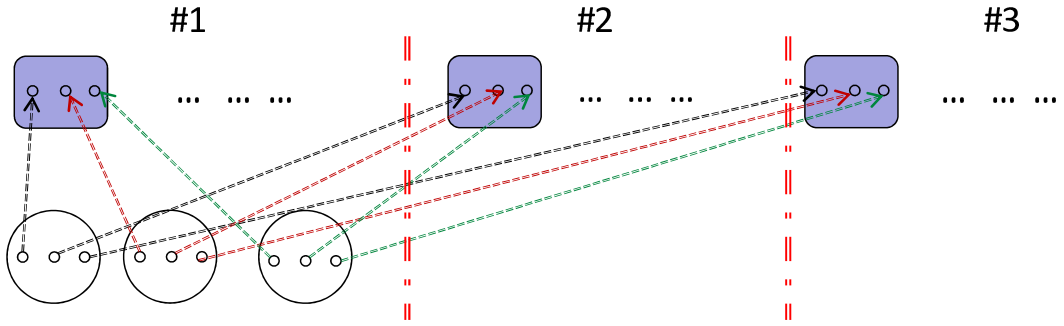

Figure 7.1: Interleaving design for the BF channel. The proposed NB-LDPC code is defined over $\mathbb{F}_8$ and transmitted over a BF channel with 3 fading blocks

For the example illustrated in Figure 7.1, the corresponding bit-interleaving pattern is shown below:

$$\|(1\ 4\ 7)\ (10\ 13\ 16)\ \ldots\ \|\ (2\ 5\ 8)\ (11\ 14\ 17)\ \ldots\ \|\ (3\ 6\ 9)\ (12\ 15\ 18)\ \ldots\ \|$$

where the number correspond to the bit positions within the (binary image of the) NB-LDPC codeword. Positions between parentheses correspond to the same complex (QAM) symbol, and fading blocks are separated by $\|$.

Code design. We have designed *ultra-sparse* $(2,d_{\mathrm{c,avg}})$ codes, meaning that all symbol-nodes are of degree $d_v = 2$. Codes are constructed by the PEG algorithm with a girth $g_{\min} = 10$ and defined over $\mathbb{F}_{16}$. These codes are not full-diversity, since we want to observe if the binary-interleaving can improve the diversity of the system.

For coding rates $r = 1/2$ and $r = 1/3$, we consider regular (2,4) and regular (2,3) LDPC codes, respectively, whereas for the rate $r = 1/4$ the code is semi-regular with a node-perspective distribution polynomials $L(x) = x^2$ and $R(x) = \frac{1}{3}x^2 + \frac{2}{3}x^3$ ($d_{\mathrm{c,avg}} = 2.5$).

Simulation results. We present some comparisons between the interleaved and uninterleaved systems. We have simulated different numbers of fading blocks: we report only the significant simulations in order to make a comparison. In this section we discuss two cases, corresponding to $n_f = 2$ and $n_f = 4$. In both cases, codes are defined over $\mathbb{F}_{16}$. Then, for $n_f = 2$ each coded symbol receives 2 contributions from the first fading block and other 2 contributions from the second fading block; for $n_f = 4$ the coded symbols receive the contributions of all the fading blocks.

Let us discuss the first case ($n_f = 2$). We suppose a study case in which the codes are transmitted over the BF in which only one block undergoes a deep fading, then two cases may happen:

1) If we use an interleaver, half of each symbol may be not recoverable. The question is: could the BP decoder recover all the symbol-nodes starting from only half information?
2) If we do not use the interleaver, a whole block of coded symbols may be lost. In this case the question is: could the BP decoder recover the lost block from the other one?

A similar discussion is valid for $n_f = 4$. We give some answers through simulations. In Figure 7.2 and 7.3 we have drawn the performance results for $n_f = 2$ and 4, respectively; in both cases the same codes are simulated. The used modulation is the 16-QAM. The curves marked by asterisks correspond to the outage probability curves for $r = 1/2$, $r = 1/3$, and $r = 1/4$.

We can observe that in general the interleaved codes (curves with filled markers) do not perform better than the same codes without interleaver (curves with empty markers). This means that it is better to have concentrated deep fading blocks than to spread them in order to mitigate the fading. The interleaver has some advantages as the number of blocks increases (e.g. for $n_f = 4$). Indeed, when the channel approaches the fast-fading model, the interleaver performs better and better.

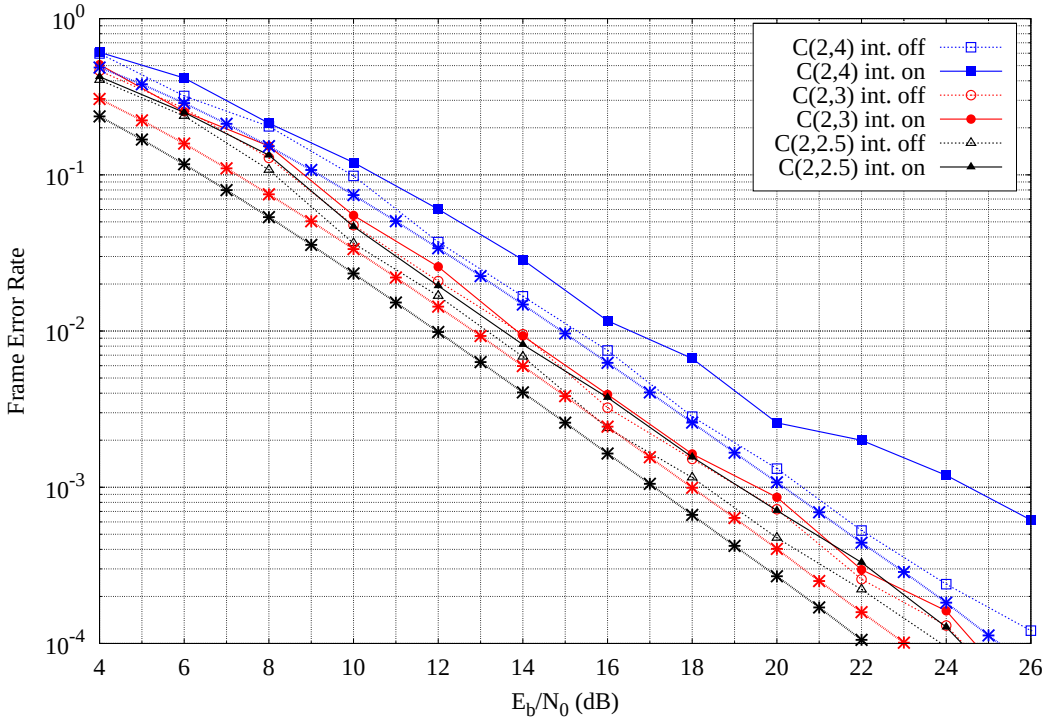

Figure 7.2: Performance comparisons between de-interleaved and interleaved systems for some codes with different rates and defined over $\mathbb{F}_{16}$ and transmitted over a BF channel with $n_f = 2$

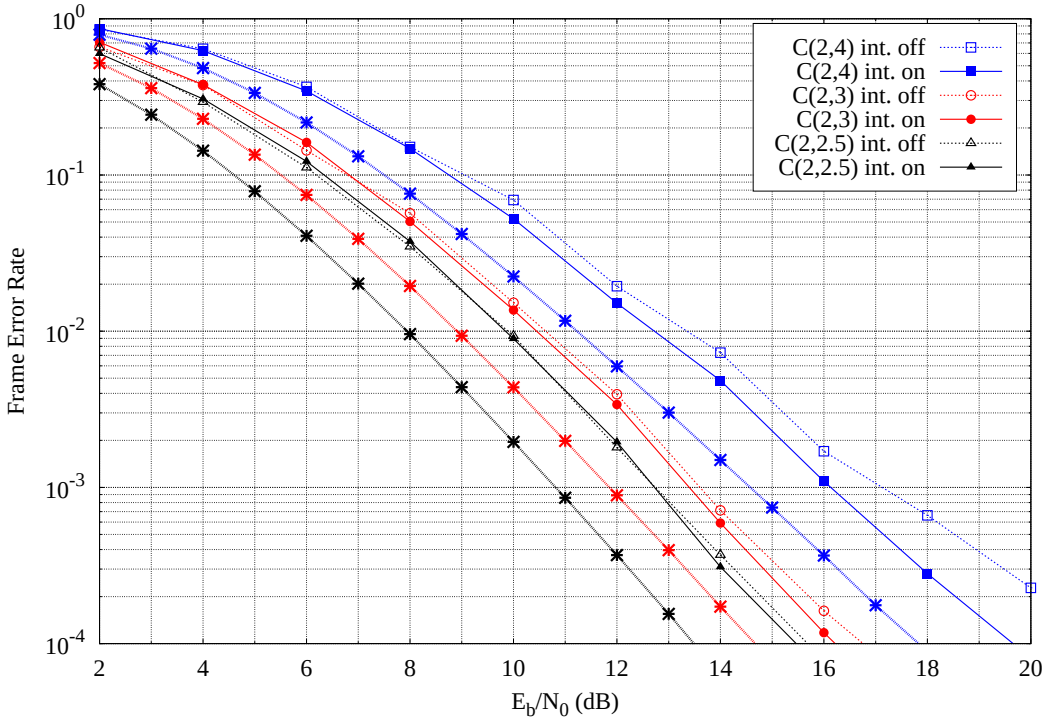

Figure 7.3: Performance comparisons between de-interleaved and interleaved systems for some codes with different rates and defined over $\mathbb{F}_{16}$ and transmitted over a BF channel with $n_f = 4$

7.4 Coding in BF Channel

7.4.1 Coding Diversity under BP Decoding

First, we note that codes achieving full-diversity under ML decoding need not achieve full-diversity under BP decoding, since the corresponding Frame Error Rate (FER) curves need not have, asymptotically, the same slope. An approach for designing codes that achieve full-diversity under BP decoding over block-fading channels has been proposed in [8], where the code design is inferred from a limiting case, namely the block-erasure channel.

We focus on systematic codes, meaning that the source symbols are embedded in the encoded codeword. Over the block-erasure channel, a systematic code of rate $r = 1/n_f$ achieves full-diversity if and only if the source symbols can be recovered (by iterative erasure decoding) from any block of symbols among the n_f blocks determined by the channel.

7.4.2 Root LDPC Codes

A special family of full-diversity LDPC codes, referred to as *Root-LDPC codes*, has been proposed in [8]. We give below an equivalent definition of Root-LDPC codes, which better suits our exposition. In the sequel, we will denote by B_j the set of source symbols that are transmitted in channel block j.

Definition 7.1. *An LDPC code is called a Root-LDPC code if for any source symbol s and any block index j such that $s \notin B_j$, there exists a constraint-node connected only to s and coded symbols transmitted in block j. Such a constraint-node is referred to as a root constraint-node.*

It follows that Root-LDPC codes are a particular case of full-diversity codes, for which the source-symbols can be recovered after one single iteration. Indeed, if only one among the n_f blocks is received, say block j, then any source symbol $s \notin B_j$ can be recovered at the first iteration of the erasure decoding, by using a root constraint-node connected only to s and symbols received in the block j.

A root-parity-check matrix for a n_f-BF channel of rate $r = 1/n_f$ will be referred to as H^{root}_{1/n_f}. An example of a parity-check matrix $H^{\text{root}}_{1/2}$, according to [8], is depicted in Figure 7.4. For hatched matrices, the parameters given in parentheses are the row and column weights (number of non-zero entries); if for instance $d = 2$ we obtain a Root-(2,4) LDPC code. Note that for NB-LDPC codes the matrix entries are taken from the finite field $\mathbb{F}_q$. S_j and P_j denote the source and parity symbols that are transmitted in fading block $j = 1, 2$ (hence $B_j = S_j \cup P_j$). The diagonal corresponds to an identity submatrix.

However, "good" codes with lower coding rates need sparser parity-check matrices, hence this kind of design is not suitable for a larger number of blocks. This is even more true for non-binary codes, since when the size of the non-binary alphabet grows, best codes are obtained for average density of edges closer and closer to $d_v = 2$ [13].

For a number of fading blocks $n_f > 2$, or equivalently rates $r = 1/n_f < 1/2$, the design proposed in [8] is more complicated and the number of entries per column is at least $n_f - 1$. The general structure of a root LDPC code as defined in [8] is formed by squared submatrix of dimension N/n_f^2. Figure 7.5 presents an example of parity-check matrix for $n_f = 3$. In particular the 6×9 squared submatrices of $H^{\text{root}}_{1/3}$ have size $N/3^2$. The green

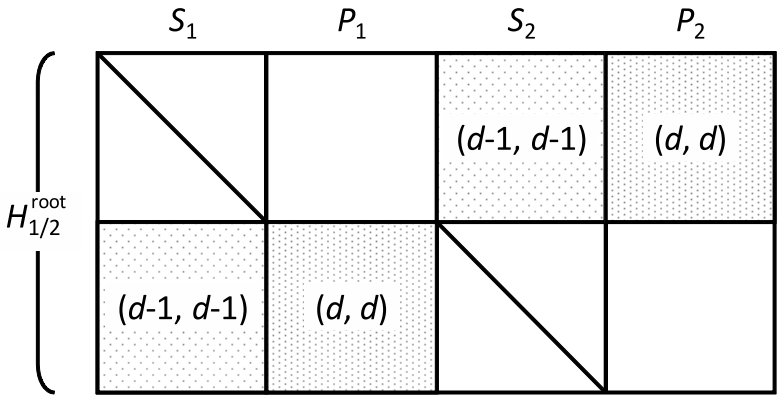

Figure 7.4: Parity check matrix of rate 1/2, regular $(d, 2d)$-Root-LDPC codes

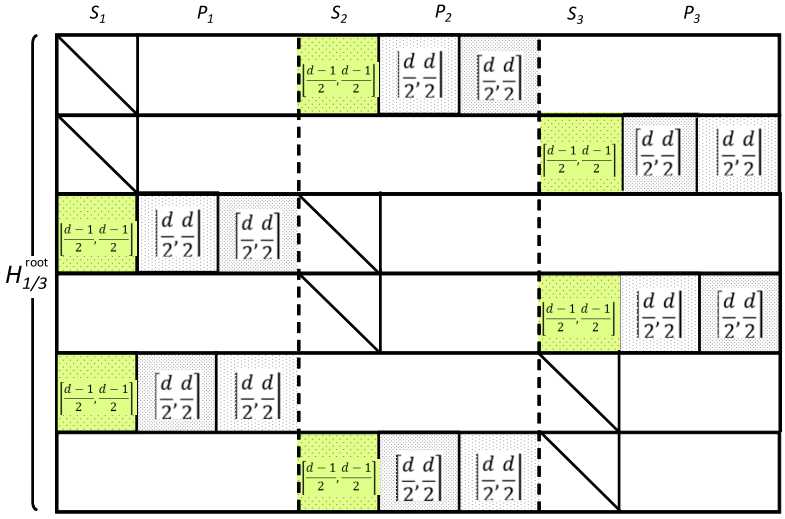

Figure 7.5: Parity check matrix of rate 1/3, regular $(d, 3/2d)$-Root-LDPC codes

parts are used in case of binary-Root-LDPC while they are zero matrices in case of higher Galois field order, when we have to consider ultra-sparse matrices ($d_s = 2$).

In this work we consider LDPC codes defined over $\mathbb{F}_{64}$ for which the best codes are ultra-sparse codes. Thus, we have tried to minimize the number of entries of the parity-check matrices and we have succeeded for the $H_{1/2}^{\text{root}}$ and $H_{1/3}^{\text{root}}$. One can define in a similar way a parity-check matrix $H_{1/4}^{\text{root}}$ that is full-diversity for a 4-BF channel but the corresponding code is not ultra-sparse since $d_s \geq 3$.

7.4.3 Triangular-LDPC Codes

In this section we propose an alternative design of full-diversity codes that shows interesting aspects. They follow from a generalization of the Root-LDPC codes given in the previous section. One of the main motivations for which we have proposed this design is to avoid that the symbol-node degree increases with the number of fading blocks n_f, which is a critical requirement for codes defined over higher order alphabets.

For any block index $j = 1, \ldots, n_f$, we denote by S_j (resp. $S_{\neq j}$) the set of source

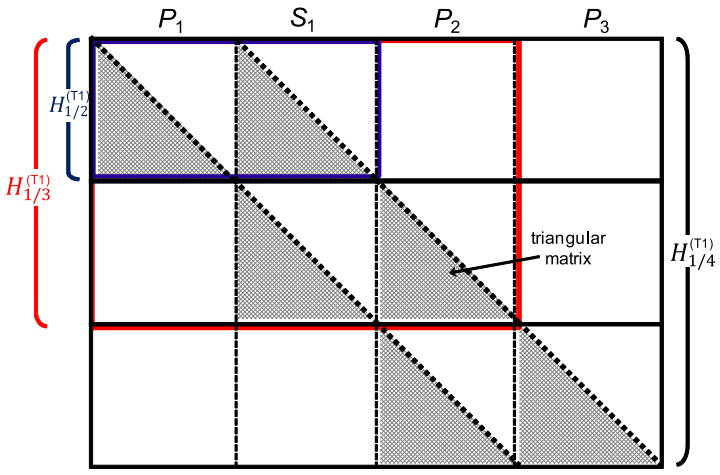

Figure 7.6: T1-Design – Full-diversity parity-check matrices $H_{1/n_f}^{(T1)}$, with $n_f = 2, 3$ and 4

symbols that are transmitted on block j (resp. on blocks $j' \neq j$). We also denote by $H_{S_{\neq j}}$ the matrix determined by the columns of H corresponding to source symbols in $S_{\neq j}$.

In case of the BlEC channel model, the following Lemma follows easily from the discussion in Section 7.4.1.

Lemma 7.1. *A systematic code of coding rate $r = 1/n_f$ achieves full-diversity under iterative erasure decoding iff there is no stopping set contained in any of $S_{\neq j}$, for $j = 1, \ldots, n_f$. Equivalently, this means that any matrix $H_{S_{\neq j}}$ can be* triangularized *(by a row/column permutation) with only non-zero entries in the main diagonal (a triangular matrix is such that all the entries above*[6] *the main diagonal are zero, but it needs not be a square matrix!).*

The idea behind this lemma is that we can design triangular-like parity-check matrices that avoids the creation of stopping-sets so that the decoder can gradually determines the source symbols.

T1-Design. The full-diversity construction illustrated in in Figure 7.6 will be referred to as T1-design and the parity-check matrix will be denoted by $H_{1/n_f}^{(T1)}$. Let us discuss this design. $H_{1/2}^{(T1)}$ (drawn in blue) is a a bi-triangular matrix that respects the Lemma 7.1. Then, $H_{1/3}^{(T1)}$ (drawn in red) is formed by the superposition of two bi-triangular matrices, which are shifted of one position with respect to the above one. Hence, the parity-check matrix $H_{1/n_f}^{(T1)}$ is composed by the superposition of $n_f - 1$ bi-triangular matrices, each one shifted of one position with respect to the adjacent ones, in order to form a cascade of bi-triangular matrices. In this case each triangular matrix is a $[N' \times N']$ square matrix, where $N' = N/n_f$. These codes will be referred to as T1-LDPC codes.

The T1-design has some advantages with respect to the Root design. First of all, it results simpler with respect to Root LDPC code design that becomes complicated especially for an increasing number of blocks. The only request is to have triangular submatrices that demonstrate more flexibility when we use the PEG construction. Another advantage lies on the fact that the entire codeword (source and parity parts) can be recovered from only one single received block (in the Root design, the parity symbol protection is abandoned). In

6. Or below.

particular, at the transmitted side, encoding can be performed on the parity-check matrix, by performing an erasure decoding submitted only with the values of the source symbols. Note also that source symbols could be placed on any of the n_f blocks. Finally, we also note that the T1-Design does not fulfill the conditions of Definition 7.1. For instance, considering the matrix $H_{1/4}^{(T1)}$ from Figure 7.6, if the P_3 block is received, then the source block S_1 is recovered after 2 iterations, instead of one single iteration as for Root-LDPC codes.

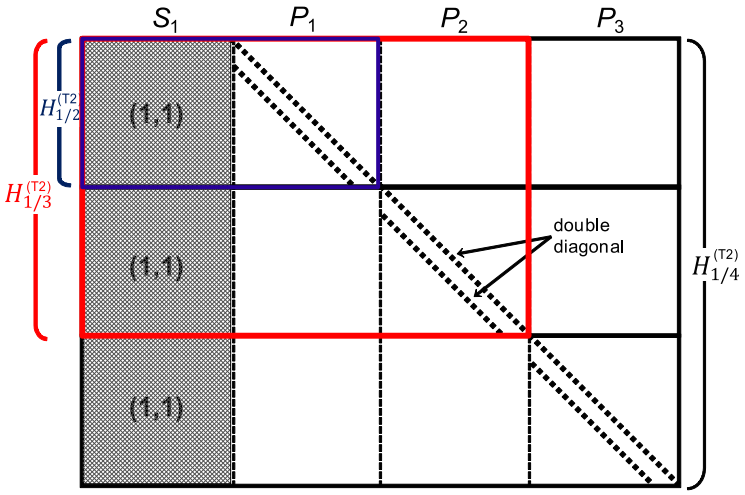

Figure 7.7: T2-Design – Full-diversity Root-parity-check matrix $H_{1/4}^{(T2)}$

T2-Design. Another proposed parity-check matrix of a full-diversity code is presented in Figure 7.7. $H_{1/4}^{(T2)}$ is the parity-check matrix of a code of rate $r = 1/4$: this designed code results to be full-diversity, according to Lemma7.1. The matrix is composed by 4 parts, on the first part we have put the source symbols S_1, whereas the other three parts are the parity parts. Also this design results be simpler with respect to the Root-LDPC codes, because the parity-check matrix is formed by $n_f - 1$ overlapped 1×1 square matrix and $n_f - 1$ triangular matrices, each one shifted with respect to the other ones. This design will be simply referred to as T2-Design, and the corresponding codes will be referred to as T2-LDPC codes. Note that these codes fulfill the conditions of Definition 7.1, hence they are actually a particular case of Root-LDPC codes. As said, when the Galois field order increases the best codes are the ultra-sparse code and this design does not fulfill this criterion, even if we have limited the increasing symbol-node degree only to source symbol nodes. Finally, we note that, as for T1-LDPC codes, T2-LDPC codes can also be encoded by using the parity-check matrix.

7.5 Non-Binary Repetition Symbols

For $n_f > 2$, our approach to design full-diversity non-binary LDPC codes of rate $r = 1/n_f$ is based on the recently introduced technique of multiplicative non-binary coding [36]. In our settings, a *mother* Root-NB-LDPC code, of rate 1/2, is extended with extra *non-binary repetition symbols*, so that to reach the target rate value $r = 1/n_f$. Put differently, the mother code is extended with multiplicative versions of (some or all) coded-symbols.

Precisely, a non-binary repetition symbol $s' = \lambda s$ is given by the multiplication between a codeword symbol s and a non-zero coefficient $\lambda \in \mathbb{F}_q$, which will be referred to as the *multiplicative coefficient*. The case $\lambda = 1$ corresponds to the usual repetition coding. We also note that for $q = 2$ (binary codes), we necessarily have $\lambda = 1$ and therefore $s' = s$.

Now, let $\mathcal{C}_{1/2}$ be a Root-NB-LDPC code of rate 1/2, and $(S_1, P_1; S_2, P_2)$ be a codeword of $\mathcal{C}_{1/2}$, as depicted in Figure 7.4. The semicolon in the above notation is used to separate the fading blocks (in this case $n_f = 2$). For $n_f > 2$, the code $\mathcal{C}_{1/n_f}$ of rate $1/n_f$, referred to as the *Repeated-Root-NB-LDPC code*, is defined by extending the mother code $\mathcal{C}_{1/2}$ with $\left\lceil \frac{n_f-2}{2} \right\rceil$ multiplicative repeated versions of the source symbols (S_1, S_2) and $\left\lfloor \frac{n_f-2}{2} \right\rfloor$ multiplicative repeated versions of the parity symbols (P_1, P_2). This is detailed in Table 7.1 for $2 \leq n_f \leq 6$, where prime and double-prime notation is used to denote multiplicative repeated versions of source and parity symbols. It can be easily seen that a Repeated-Root-NB-LDPC code of rate $r = 1/n_f$ verifies the Definition 7.1, therefore it achieves full-diversity under iterative erasure decoding over the block-erasure channel with n_f erasure blocks.

An example of the transmission with the proposed coding scheme is represented in Figure 7.8. The non-binary repetition module makes copies of the K source symbols and K parity symbols according to the Table 7.1. Since the code is systematic, the encoding needs to compute only the parity symbols [7]. The Parallel to Serial (P/S) module sends these symbols to the modulator according the order indicated in the Table 7.1.

Table 7.1: Repeated-Root-NB-LDPC codes $\mathcal{C}_{1/n_f}$, for $2 \leq n_f \leq 6$

n_f	$\mathcal{C}_{1/n_f}$ – codewords
2	$(S_1, P_1; S_2, P_2)$
3	$(S_1, P_1; S_2, P_2; S_1', S_2')$
4	$(S_1, P_1; S_2, P_2; S_1', S_2'; P_1', P_2')$
5	$(S_1, P_1; S_2, P_2; S_1', S_2'; P_1', P_2'; S_1'', S_2'')$
6	$(S_1, P_1; S_2, P_2; S_1', S_2'; P_1', P_2'; S_1'', S_2''; P_1'', P_2'')$

The decoding algorithm is explained in Section 3.2.1; the only change is the decoder input

7. The parity part is the output of the Root-encoder with rate $r = 1/2$; for this reason $M = K$.

and this is described in the following.

7.5.1 Joint Decoding Strategy

We discuss now the joint decoding strategy. Assume that a codeword of a Repeated-Root-NB-LDPC code $\mathcal{C}_{1/n_f}$ is sent over the channel. Let s_n be a coded symbol of the mother Root-NB-LDPC, and $s_{n'} = \lambda s_n$ be a multiplicative version of s_n, within transmitted codeword of $\mathcal{C}_{1/n_f}$. At the receiver side, the soft demapper computes the probability vectors $\dot{\mathbf{P}}_n = \left[\dot{P}_n(a)\right]_{a\in\mathbb{F}_q}$ and $\mathbf{P}'_n = [P'_n(a)]_{a\in\mathbb{F}_q}$ according to the Table 7.1 and Eq. (7.6), where $\dot{P}_n(a) = \Pr(s_n = a \mid y_n)$ and $P'_n(a) = \Pr(s_{n'} = a \mid y_{n'})$. Since $s_{n'} = \lambda s_n$, the probability $P_n(a) = \Pr(s_n = a \mid y_n, y_{n'})$, of the transmitted symbol s_n being equal to a conditioned on both y_n and $y_{n'}$, can be computed by:

$$\begin{aligned} P_n(a) &= \gamma \dot{P}_n(a) P'_n(\lambda a) \\ &= \gamma \exp(-\rho |g_j|^2 \cdot |y_n - \mu(a)|^2) \exp(-\rho |g_{j'}|^2 \cdot |y_{n'} - \mu(\lambda a)|^2) \end{aligned} \tag{7.7}$$

where γ is a normalization constant such that $\sum_{a\in\mathbb{F}_q} P_n(a) = 1$. The above computation can be generalized, in a straightforward manner, to the case when several multiplicative versions of s_n belong to the transmitted codeword of $\mathcal{C}_{1/n_f}$. The *joint-probability vectors* $\mathbf{P}_n = [P_n(a)]_{a\in\mathbb{F}_q}$, $n = 1, \ldots, N$, are then used for the initialization of the BP decoder of the mother Root-NB-LDPC code $\mathcal{C}_{1/2}$, which is used to decode the source symbols (S_1, S_2). Hence, the same BP decoder (of the mother Root-NB-LDPC code) is used, regardless of the Repeated-Root-NB-LDPC code that is used to communicate over the channel.

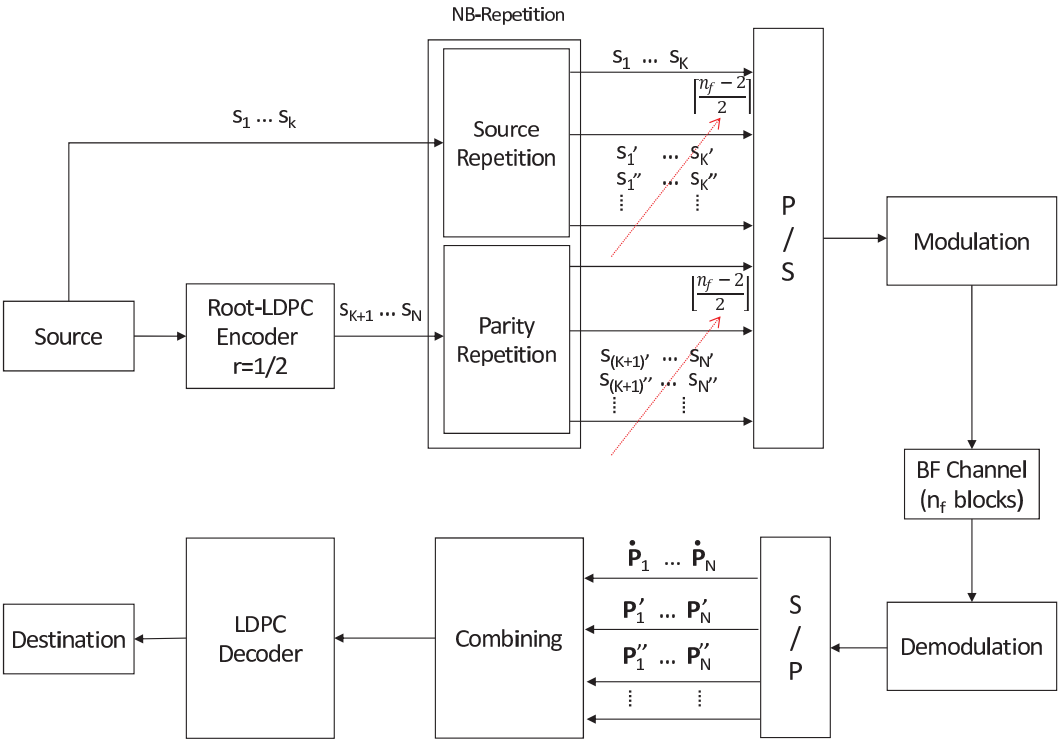

Figure 7.8: Transmission chain scheme – Joint Decoding

7.5.2 Optimization of the Non-Binary Multiplicative Coefficients

In order to optimize the non-binary multiplicative coefficients, we investigate the impact of the non-binary repetition symbols on the joint-probability computation at the receiver. We note that the our approach is different from the one proposed in [17], in that our optimization is done with respect to the Euclidian distance between the points of the complex signal set $\mathbb{X}$, while in *loc. cit.* the optimization is performed with respect to the Hamming distance of the binary image of the component codes.

Let $\mu : \mathbb{F}_q \to \mathbb{X}$ be the mapping function from the code field $\mathbb{F}_q$ to the complex signal set $\mathbb{X}$ (7.5). For any $s \in \mathbb{F}_q$ and any $\delta \geq 0$, let:

$$\mathcal{D}(s, \delta) = \{a \in \mathbb{F}_q \mid |\mu(s) - \mu(a)| \leq \delta\},$$

be the set of non-binary symbols $a \in \mathbb{F}_q$ such that the distance between the complex signal points corresponding to a and s is less than or equal to δ.

Now, let s_n be a codeword symbol of the mother Root-NB-LDPC code $\mathcal{C}_{1/2}$. If δ is large enough, symbols $a \notin \mathcal{D}(s_n, \delta)$ have, in average, small $\dot{P}_n(a)$ values. This follows from Eq. (7.6), by averaging over all possible noise realizations. Put differently, symbols $a \in \mathcal{D}(s_n, \delta)$ are, after the soft demapping, the most likely values for symbol-node n.

Similarly, if $s_{n'} = \lambda s_n$ is a multiplicative version of s_n, the most likely values for symbol-node n' are symbols $a' \in \mathcal{D}(s_{n'}, \delta) = \mathcal{D}(\lambda s_n, \delta)$. Therefore, as determined by the joint-probability vector $P_n(a)$, the most likely values for symbol-node n are symbols a such that $a \in \mathcal{D}(s_n, \delta)$ and $\lambda a \in \mathcal{D}(\lambda s_n, \delta)$. Now, symbols $a \neq s_n$ satisfying the above condition provide misleading information to the BP decoder. In the ideal situation, we should have $\mathcal{D}(s_n, \delta) \cap \lambda^{-1}\mathcal{D}(\lambda s_n, \delta) = \{s_n\}$, with δ as large as possible. Therefore, we define:

$$\delta(\lambda) = \sup\left\{\delta \mid \mathcal{D}(s, \delta) \cap \lambda^{-1}\mathcal{D}(\lambda s, \delta) = \{s\},\ \forall s \in \mathbb{F}_q\right\},$$

and choose $\lambda_1 \in \mathbb{F}_q$, such that $\delta(\lambda_1) = \max_\lambda \delta(\lambda)$. Hence, if s_n has a unique multiplicative version $s_{n'}$ within the transmitted codeword of $\mathcal{C}_{1/n_f}$, we always define $s_{n'} = \lambda_1 s_n$.

In case that there are several multiplicative versions of s_n within the transmitted codeword of $\mathcal{C}_{1/n_f}$, the non-binary multiplicative coefficients are optimized in a similar manner. For instance, in case of two multiplicative versions, we define $\delta(\lambda', \lambda'')$ as the supremum value of δ such that any two among the three sets $\mathcal{D}(s, \delta)$, $\lambda'^{-1}\mathcal{D}(\lambda' s, \delta)$, and $\lambda^{-1}\mathcal{D}(\lambda'' s, \delta)$ have only the symbol s in common, for any $s \in \mathbb{F}_q$. Then we choose (λ_1, λ_2), such that $\delta(\lambda_1, \lambda_2) = \max_{\lambda', \lambda''} \delta(\lambda', \lambda'')$, and define the multiplicative versions of s_n by $s_{n'} = \lambda_1 s_n$ and $s_{n''} = \lambda_2 s_n$.

In our design, we consider Repeated-Root-NB-LDPC over $\mathbb{F}_{64}$, where $\mathbb{F}_{64}$ is defined as the quotient field of $\mathbb{F}_2[\alpha]$ by the primitive polynomial $\alpha^6 + \alpha + 1$. The complex signal set $\mathbb{X}$ is the 64-QAM complex constellation, and for any $s \in \mathbb{F}_{64}$, $\mu(s)$ is defined as the Gray mapping of the *binary image* of s (vector of the coefficients of s expressed as a binary polynomial of degree less that 6). Within these settings, we obtained $\lambda_1 = \alpha^{21}$ in case of one single multiplicative symbol and $(\lambda_1, \lambda_2) = (\alpha^{21}, \alpha^{42})$ in case of two multiplicative symbols. We note the the value of λ_1 is the same in both cases. In Figure 7.9 we represented the Tanner graph of the Repeated-Root-NB-LDPC code $\mathcal{C}_{1/5}$. Source symbols of the mother Root-NB-LDPC code have two multiplicative repeated versions, while parity symbols have only one. The fact that $s_{n'} = \lambda s_n$ is expressed by adding a new check-node connected to both s_n and $s_{n'}$, with coefficient λ on the edge connecting it to s_n. For the first NB-repeated symbols the multiplicative coefficients are equal to λ_1, while for the second ones the multiplicative coefficients are equal to λ_2.

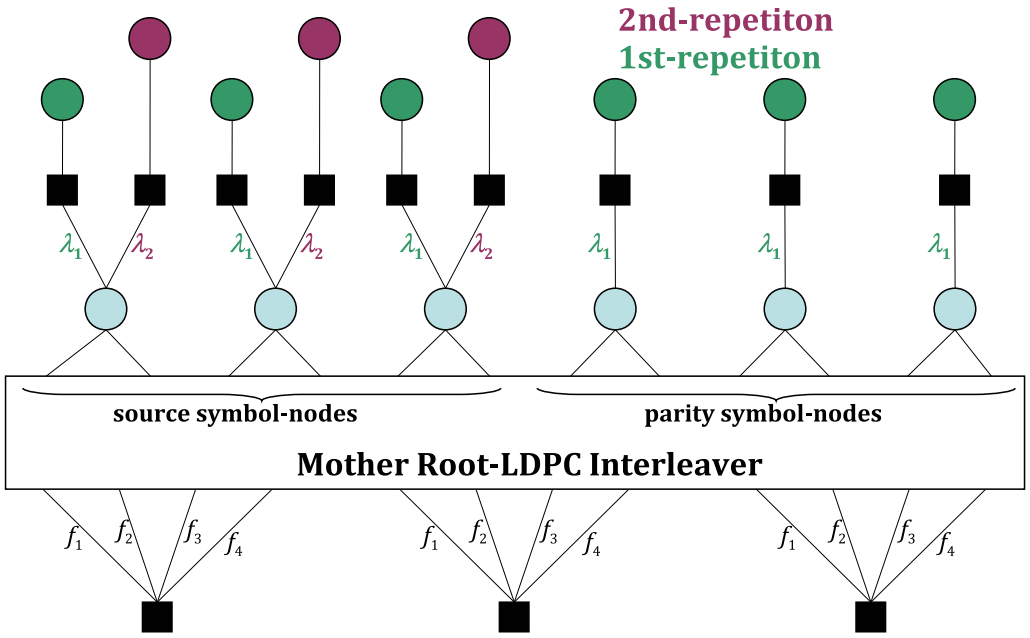

Figure 7.9: Tanner graph of Repeated-Root-NB-LDPC codes

The non-binary coefficients of the Root-NB-LDPC parity-check matrix also appear on the edges connecting the Root-NB-LDPC check-nodes (bottom) to the Root-NB-LDPC interleaver. We note that these coefficients have been optimized according to the method introduced in [17]. In particular, all the check-node share the same set of 4 coefficients, possibly permuted, given by $(\alpha^0, \alpha^9, \alpha^{26}, \alpha^{46})$ (see also Section 3.4.2).

7.6 Simulation Results

7.6.1 Performance of the Repeated-Root-NB-LDPC Codes

In this section we present the performance of the proposed Repeated-Root-NB-LDPC codes over the block-fading channel. We compare the performance of codes over the Galois field $\mathbb{F}_{64}$ and of binary codes. In each case, we constructed a mother regular $(d_v, d_c = 2d_v)$-Root-LDPC code with rate 1/2, as depicted in Figure 7.4. Codes have been constructed by using the PEG algorithm, with $d_v = 2$ for codes over $\mathbb{F}_{64}$, and $d_v = 3$ for the binary case. Furthermore, we extended the mother codes by using – non-binary, for $\mathbb{F}_{64}$ – repetition symbols, so that to obtain coding rates 1/3, 1/4, and 1/6, as explained in Section 7.5. All the codes $\mathcal{C}_{1/n_f}$ have the same binary dimension, equal to 300 bits. Figure 7.10 shows the Frame Error Rates (FER) obtained by Monte-Carlo simulation over the Rayleigh block-fading channel. The number of fading blocks n_f is equal to $2, 3, 4, 6$ for coding rates $1/2, 1/3, 1/4$, and $1/6$ respectively. For each of the above coding rates, we plotted the corresponding outage probability (asterisk markers) and the FER for the non-binary (full markers) and binary (empty markers) codes. As expected, all codes achieve full-diversity. However, in contrast with the binary repetition, the use of non-binary repetition symbols, with optimized multiplicative coefficients, results in an impressive coding gain. As it can be seen, the gap to the outage probability remains constant (approx. 1 dB) for all coding rates, from 1/2 to 1/6.

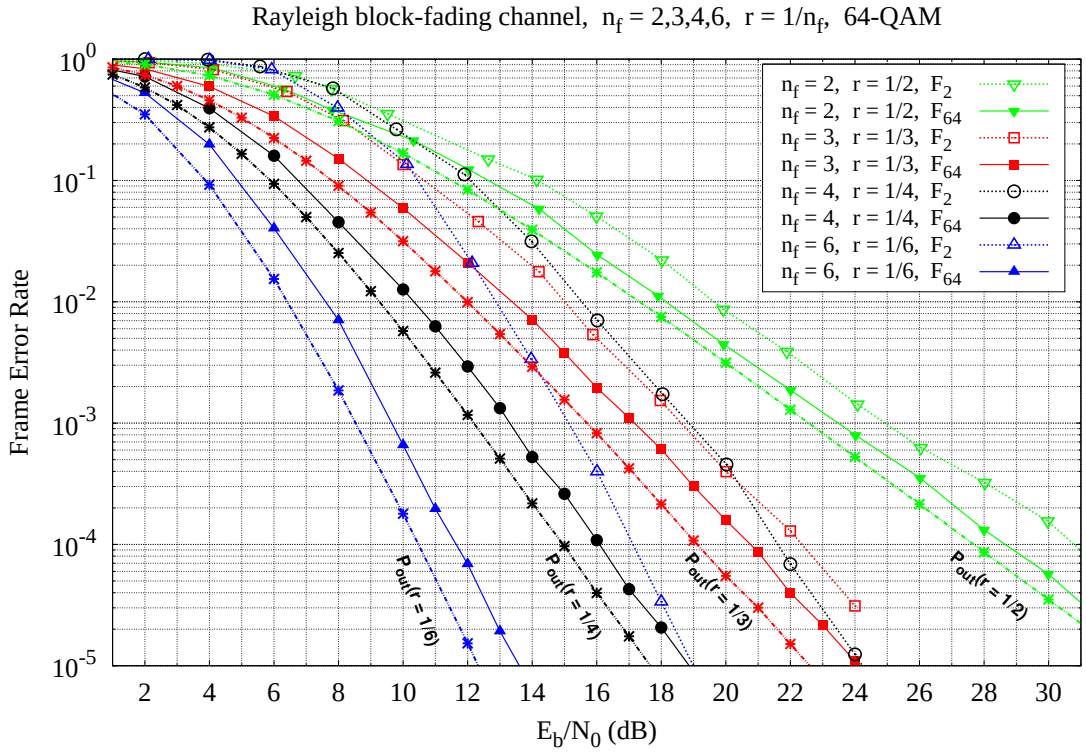

Figure 7.10: FER performance comparison of Repeated-Root-NB-LDPC codes

Moreover, let us analyze the impact of the multiplicative coefficient optimization according to the Euclidian distance of the modulated complex symbols. In Figure 7.11 we show the performance comparison of two optimized Repeated-Root-NB-LDPC codes for $n_f = 3$, or equivalently $r = 1/3$, and $n_f = 4$, or equivalently $r = 1/4$ (with the $\bullet$ markers) [8], and the same codes with randomly chosen repetition coefficients (with the $\circ$ markers). The mother Root-LDPC code is the same for all the repeated-Root-NB-LDPC codes and it has optimized coefficients $(\alpha^0, \alpha^9, \alpha^{26}, \alpha^{46})$ as discussed in the previous section. For the code with rate $r = 1/3$ we have repeated only the source symbols (hence half of the codelength), whereas for the code with rate $r = 1/4$ we have repeated all the coded symbols. It can be observed that the proposed optimization of the multiplicative coefficients leads to a gain of about 0.5 dB in all the cases.

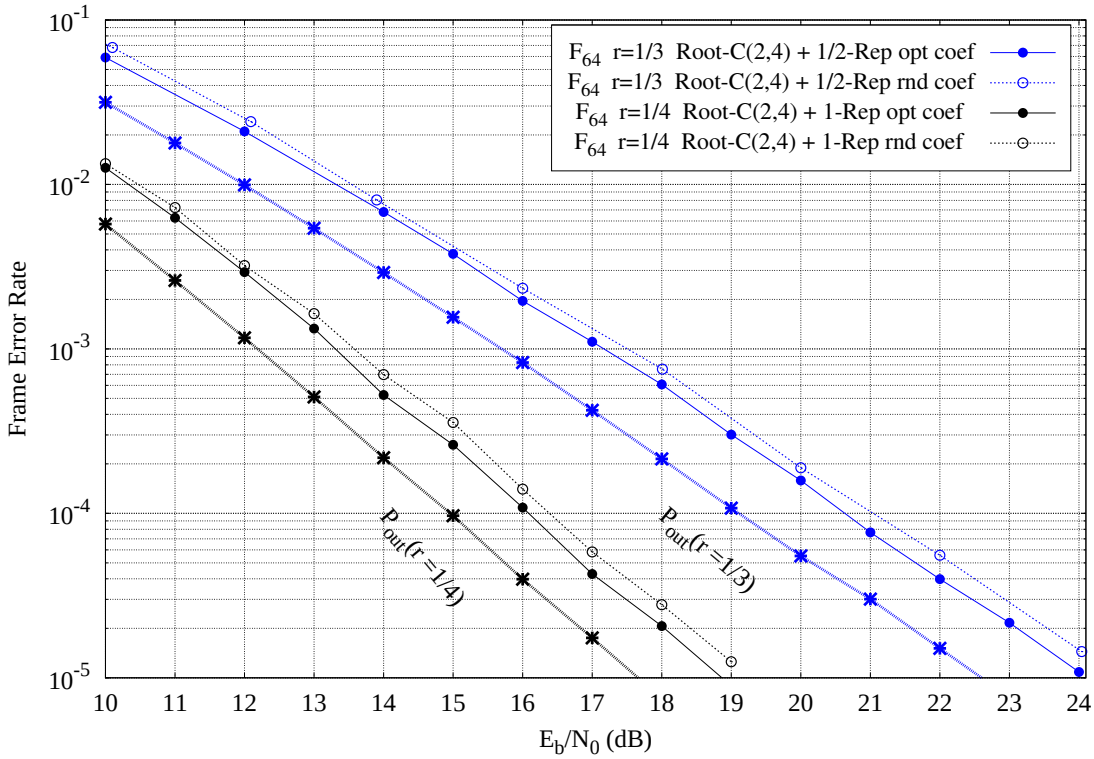

Figure 7.11: FER performance comparison of NB-LDPC codes with optimized and random multiplicative coefficients for $n_f = 3$ ($r = 1/3$) and $n_f = 4$ ($r = 1/4$).

8. These two curves correspond to the curves "$n_f = 3$, $r = 1/3$, $\mathbb{F}_{64}$" and "$n_f = 4$, $r = 1/4$, $\mathbb{F}_{64}$" of Figure 7.10.

7.6.2 Comparison of Simulation Results

In this section we evaluate the performance of the LDPC codes mentioned in Section 7.4.2 and 7.4.3. We present simulation results for the block-fading channel with 3 and 4 fading blocks. All the considered codes $\mathcal{C}_{1/n_f}$ have been constructed by using the PEG algorithm and they have the same binary dimension, equal to 300 bits. Moreover, the codes are modulated with a 64-QAM. All these codes are full-diversity but they show different coding gains that depend on the alphabet size and on the parity-check matrix design.

3 fading blocks. Figure 7.12 presents the FER obtained by Monte-Carlo simulation of codes transmitted over the Rayleigh block-fading channel with $n_f = 3$. All the considered codes have the same coding rate $r = 1/3$.

First of all we focus on the binary case (corresponding to the black curves). Three root-designs have been realized according to the parity-check matrix $H_{1/3}^{\text{root}}$ (see Figure 7.5) with $d_v = 2, 3$ and 4; the corresponding Root-LDPC codes are regular for $d_v = 2$ and $d_v = 4$ with $d_c = 3$ and $d_c = 6$, respectively. For $d_v = 3$ the code results semi-regular with an average check-node degree $d_{\text{c,avg}} = 4.5$ [9]. We have plotted the outage probability P_{out} (the orange curve) and the FER for these codes (empty markers): we can observe that the Root-C(3,4.5) has the best performance for $\mathbb{F}_2$.

Furthermore, we have extended the Root-(3,6) LDPC code, which is the best code for rate $r = 1/2$, with one binary repetition of the source symbols. The FER for this design (black filled circles) demonstrates that the Repeated-Root-C(3,6) LDPC code has worse performance that the Root-C(3,4.5) and Root-C(2,3) codes with rate 1/3.

Finally, we analyse the LDPC codes defined over $\mathbb{F}_{64}$ (corresponding to the blue curves). We constructed a Root-(2,3)-NB-LDPC code, as depicted in Figure 7.5 (the green parts have been omitted) with random coefficients in $\mathbb{F}_{64}$. We have compared this design with the optimized Repeated-Root-NB-LDPC with rate $r = 1/3$. It can be observed that the Root-NB-LDPC code (with the ∇ markers) outperforms the Repeated-Root-NB-LDPC (with the $\bullet$ markers) by $\sim$0.5 dB .

Also, we have constructed the Triangular-LDPC codes according to the parity-check matrices $H_{1/3}^{(T1)}$ of Figure 7.6 and $H_{1/3}^{(T2)}$ of Figure 7.7 with random coefficients from the alphabet $\mathbb{F}_{64}$. The former has an average symbol degree $d_{\text{s,avg}} = 2.1$ while the latter is regular with $d_s = 2$. The red circles correspond to the performance of the T1-Design, whereas the red asterisks correspond to performance of the T2-Design. These codes have very simple designs, but only the latter shows very good performance with respect to the Root-LDPC and the Repeated-Root LDPC codes. The poor performance of T1-design are probably due to the triangular construction that does not protect equally all the symbols. Also, the T1-design does not fully exploit the potential of the PEG algorithm. We have observed that the T2-design, allows constructing graphs with higher girths than the T1-design.

9. Consider Figure 7.5. For Root-(2,3)-LDPC, the green parts have been removed; for Root-(3,4.5)-LDPC only one green matrix is kept for each column of the source part, finally, for Root-(4,6)-LDPC all the green parts are kept.

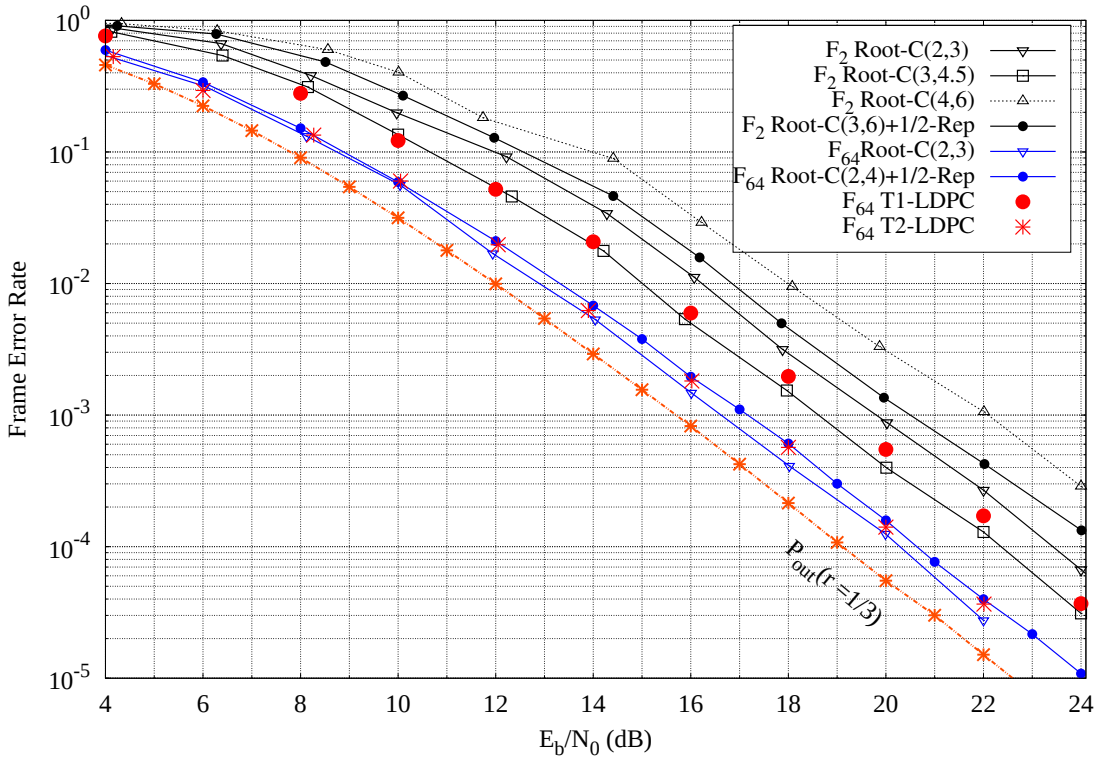

Figure 7.12: FER performance comparison for full-diversity codes with $r = 1/3$ transmitted over BF channel with $n_f = 3$

4 fading blocks. Figure 7.13 presents the FER obtained by Monte-Carlo simulation of codes transmitted over the Rayleigh block-fading channel with $n_f = 4$. All the considered codes have the same coding rate $r = 1/4$. The orange curve corresponds to the outage probability.

We discuss first the binary case (corresponding to black curves). The parity-check matrix $H_{1/4}^{\text{root}}$ is more complicate and it is constructed following the concepts of Section 7.4.2 (page 124). The binary repeated-Root code is constructed from a Root-(3,6)-LDPC code, for which all the coded symbols have been repeated once: this code performs $\sim$3 dB above the Root-(3,4)-LDPC code, which is the best simulated code for $\mathbb{F}_2$.

Concerning NB-LDPC, we have simulated codes defined over $\mathbb{F}_{64}$ (corresponding to blue curves). We constructed a (3,4) Root-LDPC code with rate $r = 1/4$ with random coefficients in $\mathbb{F}_{64}$. The corresponding FER curve is distinguished by ∇ markers. We have also constructed two Repeated-Root-NB-LDPC codes, with rate $r = 1/4$, as follows:

1) The first one uses the Root-LDPC code with rate $r = 1/2$ as mother code, with one non-binary repetition of all the coded symbols. This code has also been evaluated in Section 7.6.1, and the corresponding FER curve is distinguished by $\bullet$ markers.
2) The second one uses the Root-LDPC code with rate $r = 1/3$ as mother code, with one non-binary repetition of the source symbols (repetition of 1/3 of the codelength). The corresponding FER curve is distinguished by $\blacktriangle$ markers.

It can be observed that the rate-1/4 Root-LDPC code (∇ markers) performs worse than the binary code, whereas the two Repeated-Root designs ($\blacktriangle$ and $\bullet$ markers) have similar

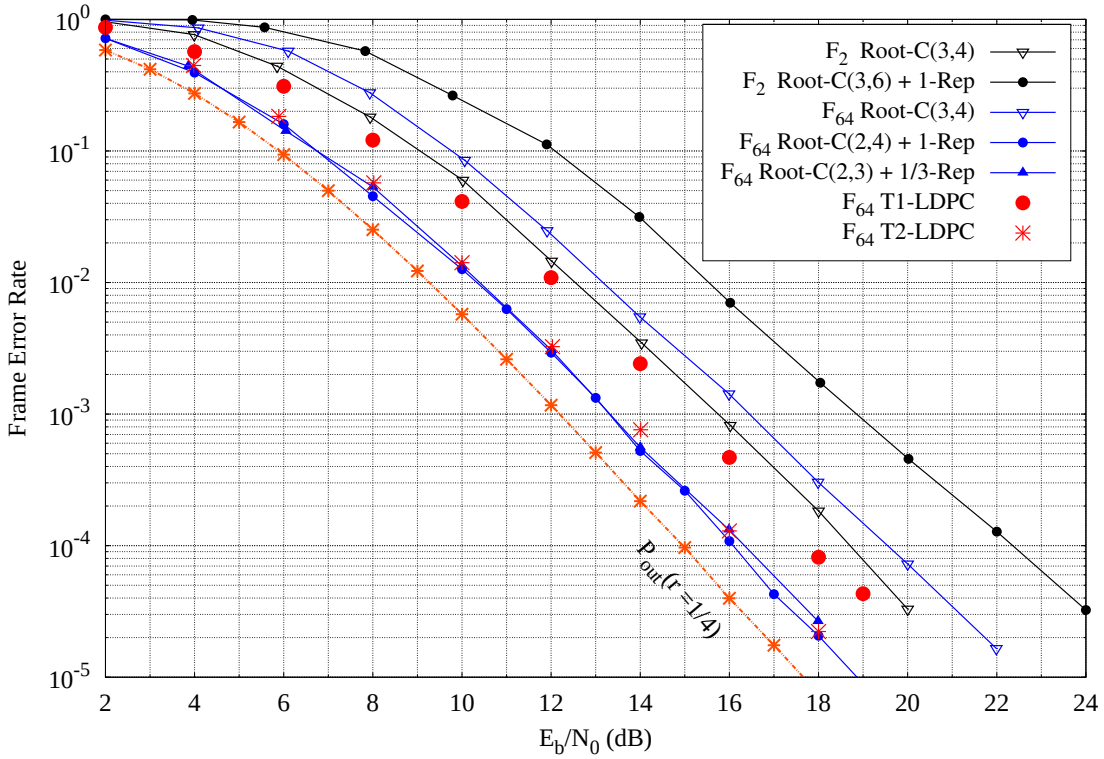

Figure 7.13: FER performance comparison for full diversity codes with $r = 1/4$ transmitted over BF channel with $n_f = 4$

performance and both curves are close to the outage probability. Hence, besides the flexibility advantage, the proposed Repeated-Root design also presents the best performance for small coding rates.

Finally, we have also constructed the Triangular-LDPC codes according to the parity-check matrices $H_{1/4}^{(T1)}$ of Figure 7.6 and $H_{1/4}^{(T2)}$ of Figure 7.7, with random coefficients from the alphabet $\mathbb{F}_{64}$. The former has an average symbol degree $d_{\text{s,avg}} = 2.3$ while the latter has $d_{\text{s,avg}} = 2.25$. Red filled circles correspond to the T1-Design, whereas red asterisks correspond to the T2-Design. Again, only the T2-LDPC code demonstrates very good performance, similar to the Repeated-Root-LDPC design. Furthermore, if there does not exist any flexibility constraints, this code design can be successfully used for high order alphabets and coding rates $r \leq 1/4$.

7.7 Conclusion

This chapter was concerned with the study of NB-LDPC codes for the block-fading channel. First of all we have investigated the use of the binary-interleaving technique and we showed that this technique does not provide enough diversity and is not suitable for slow-varying fading channel.

Then we investigated the design of full-diversity NB-LDPC codes, with a particular attention to codes defined over high order alphabets. For these codes, it is known that the best decoding performance is obtained for an average density of edges very close to $d_s = 2$. Consequently, the classical Root-design used for binary codes is not appropriate in the non-binary case. We therefore proposed two alternative full-diversity designs. One of these new designs results more simple and robust with respect to the Root-design, especially for a number of blocks greater than 4.

Another research direction was dedicated to constructing a coding scheme with close-to-outage performance and, at the same time, can be easily adapted to the number of fading blocks. Then a flexible coding-scheme coding has been investigated, based on non-binary Root-LDPC codes and non-binary repetition coding. The proposed coding-scheme allows designing full-diversity codes over channels with n_f fading blocks, for a wide range of n_f values, while the proposed joint-decoding allows using the same decoder irrespective of the n_f value. Moreover, we have also proposed an optimization procedure for the multiplicative coefficients used for non-binary repetition coding. The performance of the proposed Repeated-Root-NB-LDPC codes over the Rayleigh block-fading channel has been evaluated by Monte-Carlo simulation, and we showed that they perform close to the outage probability, even for very small coding rates.

Conclusions and Perspectives

In this thesis we investigated the analysis and the design of non-binary LDPC codes transmitted over slow and fast fading channels.

The first contribution was the development of the Density Evolution approximation via Monte-Carlo simulations. We showed that the Monte-Carlo technique provides quickly accurate and precise estimates of non-binary ensemble thresholds, and makes possible the optimization of non-binary codes for a wide range of applications and channel models.

The analysis and design of optimized puncturing distributions for NB-LDPC codes is the second contribution of the thesis. First, we showed that the non-binary LDPC codes are more robust to puncturing than their binary counterparts, thanks to the fact that non-binary symbol-nodes can be only partially punctured. Secondly, we analyzed puncturing distributions for several ensembles of regular and semi-regular codes. For irregular codes with higher punctured rate an optimization procedure is needed. For these codes, the optimized puncturing distributions exhibit a gap to capacity between 0.2 and 0.5 dB , for punctured rates varying from 0.5 to 0.9.

With the third contribution we investigated the non-binary LDPC codes transmitted over a Rayleigh (fast) fading channel, in which different modulated symbols are affected by different fading factors. A bit interleaver, operating between the encoder and the symbol mapper, introduces a binary diversity that mitigates these fading effects. Moreover, an optimized interleaving algorithm, inspired from the Progressive Edge-Growth (PEG) method, has been proposed. This algorithm ensures maximum girth of the global graph that extends the bipartite graph of the code with a new ensemble of nodes representing the modulated symbols. The optimized interleaver shows a gain with respect to the random interleaver, as far as performance and error detection rates are concerned.

Finally, the last contribution consists of a flexible coding scheme that achieves full-diversity over the block-fading channel. The proposed LDPC codes, based on Root non-binary LDPC codes coupled with multiplicative non-binary symbols, are close to the outage limit for various coding rates and number of fading blocks. Moreover, at the transmitter end this scheme demonstrated to easily adapt the coding rate to the number of fading blocks, while a simple combining strategy is used at the receiver end before the iterative decoding. As a consequence, the decoding complexity is the same, irrespective of the number of fading blocks and coding rate, while the proposed technique brings an effective coding gain.

General Conclusion. The main goal of this thesis was to demonstrate that besides the gain in the decoding performance, the use of non-binary LDPC codes can bring additional benefits in terms of "flexibility" and "diversity", which may offset the extra cost in decoding complexity.

The flexibility issue has been addressed through the use of puncturing and non-binary repetition (multiplicative) coding. These techniques are complementary: while puncturing is used to increase the coding rate of a mother code, the non-binary repetition coding is used to decrease this rate. At the receiver end, both techniques allows using the same decoder irrespective of the current coding rate. We investigated these techniques in different situations, and proposed appropriate optimization solutions, allowing one to fully exploit the their potential.

The diversity issue has been investigated through the use of the binary-interleaving technique. We showed that this technique is well suited for fast fading channels and proposed optimized constructions of binary interleavers for NB-LDPC codes, which allow improving the code performance on the error-floor region.

Perspectives. While the classical puncturing technique can be applied to both binary and non-binary LDPC codes in order to increase the coding rate of a mother code, the latter family of codes can also be advantageously coupled with non-binary repetition (multiplicative) coding so that to design codes with lower coding rates. Although both techniques have been investigated, in different situations, in this thesis, we still lack a systematic and unified treatment of them. Hence, a first perspective of this thesis would consist in the optimization of flexible coding schemes able to accommodate *almost* any coding rate in $[0, 1]$ (*e.g.* any coding rate in $[0.1, 0.9]$). Such a flexibility would be certainly of great interest for many communication systems, and could offset the extra cost in the decoding complexity of non-binary codes, especially for codes defined over $\mathbb{F}_q$, with *e.g.* $q \leq 16$. This is because the same decoder (of the mother code) can be used for any coding rate, which is in contrast with most of the current standards, which requires the implementation of a different decoder for each of the specified coding rates.

Considering the diversity brought by binary-interleavers for NB-LDPC codes over fast fading channel, a second perspective of this thesis would consist in the joint design of the non-binary bipartite graph and of the binary interleaver. The binary interleaver can be seen as a superimposed graph on the code's Tanner graph, in which nodes representing coded-symbol are connected to nodes representing the complex modulated symbols. Hence, the joint design would consist in an optimized (*e.g.* PEG-like) construction of a graph containing constraint-nodes, coded-symbol-nodes, and modulated-symbol-nodes. While connections between constraint- and coded-symbol- nodes define the NB-LDPC code, the connections between coded-symbol- and modulated-symbol- nodes define the binary-interleaving. We believe that this joint optimization should further improve the performance of NB-LDPC codes over fast fading chanels, especially in the error floor region.

Bibliography

[1] URL `www.http://ipgdemos.epfl.ch/ldpcopt/`.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *Information Theory, IEEE Transactions on*, 20(2):284–287, 1974.

[3] A. Bennatan and D. Burshtein. Design and analysis of nonbinary ldpc codes for arbitrary discrete-memoryless channels. *IEEE Transactions on Information Theory*, 52(2):549–583, 2006.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1064–1070, 1993.

[5] E. Biglieri, J. Proakis, and S. Shamai. Fading channels: Information-theoretic and Communications aspects. *IEEE Trans. on Inf. Theory*, 44(6):2619–2692, 1998.

[6] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[7] J.J. Boutros, E.C. Strinati, and A.G. i Fabregas. Turbo code design for block fading channels. In *Proc. 42nd Annual Allerton Conference on Communication, Control and Computing*, 2004. Allerton, IL.

[8] J.J. Boutros, A. Guillén i Fabregás, E. Biglieri, and G. Zémor. Design and analysis of Low-Density Parity-Check codes for block-fading channels. In *IEEE Inf. Theory and App. Workshop*, pages 54–62, 2007.

[9] J. Campello, D.S. Modha, and S. Rajagopalan. Designing ldpc codes using bit-filling. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 55–59, 2001.

[10] W. Chen, C. Poulliat, D. Declercq, L. Conde-Canencia, A. Al-Ghouwayel, and E. Boutillon. Non-binary ldpc codes defined over the general linear group: finite length design and practical implementation issues. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–5, 2009.

[11] S.Y. Chung, G.D. Forney Jr, T.J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *Communications Letters, IEEE*, 5(2):58–60, 2001.

[12] S.Y. Chung, T.J. Richardson, and R.L. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation. *Information Theory, IEEE Transactions on*, 47(2):657–670, 2001.

[13] M. C. Davey and D. J. C. MacKay. Low density parity check codes over GF(q). *IEEE Commun. Lett.*, 2:165–167, 1998.

[14] M. C. Davey and D. J.C. MacKay. Monte Carlo simulations of infinite low density parity check codes over GF(q). In *International Workshop on Optimal Codes and Related Topics (OC98)*, 1998.

[15] M.C. Davey. Error-correction using low-density parity-check codes. *Univ. of Cambridge PhD dissertation*, 1999.

[16] D. Declercq and M. Fossorier. Decoding algorithms for non-binary LDPC codes over GF(q). *IEEE Trans. Commun.*, 55(4):633–643, 2007.

[17] D. Declercq, V. Savin, and S. Pfletschinger. Multi-relay cooperative NB-LDPC coding with non-binary repetition codes. In *International Conference on Networks (ICN)*, March 2012. St-Gilles, Reunion-Island.

[18] C. Di, D. Proietti, IE Telatar, TJ Richardson, and RL Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Transactions on Information Theory*, 48(6):1570–1579, 2002.

[19] D. Divsalar, H. Jin, and R.J. McEliece. Coding theorems for "turbo-like" codes. In *Proceedings of the annual Allerton Conference on Communication control and Computing*, volume 36, pages 201–210. University of Illinois, 1998.

[20] G.D. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[21] M.P.C. Fossorier. Quasicyclic low-density parity-check codes from circulant permutation matrices. *Information Theory, IEEE Transactions on*, 50(8):1788–1793, 2004.

[22] R. Gallager. Low density parity check codes. *IEEE Trans. Inform. Theory*, IT-8:21–28, 1962.

[23] R. G. Gallager. *Low Density Parity Check Codes*. M.I.T. Press, 1963.

[24] A. Goupil, M. Colas, G. Gelle, and D. Declercq. Fft-based bp decoding of general LDPC codes over abelian groups. *Communications, IEEE Transactions on*, 55:644–649, 2007. ISSN 0090-6778. URL `http://ieeexplore.ieee.org/iel4/26/4155106/04155120.pdf`.

[25] J. Ha, J. Kim, and S.W. McLaughlin. Rate-compatible puncturing of low-density parity-check codes. *Information Theory, IEEE Transactions on*, 50(11):2824–2836, 2004.

[26] J. Ha, J. Kim, D. Klinc, and S.W. McLaughlin. Rate compatible punctured low-density parity-check codes with short block lengths. *IEEE Trans. on Inform. Theory*, IT-52:728–738, 2006.

[27] R.W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[28] RVL Hartley. Transmission of information1. 1928.

[29] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2(2):147–56, 1959.

[30] X. Hu. *Low-delay low-complexity error-correcting codes on sparse graphs.* PhD thesis, EPFL, Lausanne, Switzerland, 2002.

[31] X. Y. Hu and M. P. C. Fossorier. On the computation of the minimum distance of low density parity check codes. In *IEEE, Inter. Conf. Comm.*, volume 2, pages 767–771, 2004.

[32] X. Y. Hu, E. Eleftheriou, and D. M. Arnold. Regular and irregular progressive edge-growth tanner graphs. *IEEE Trans. Inform. Theory*, 51(1):386–398, 2005.

[33] A. Guillén i Fàbregas and G. Caire. Coded modulation in the block-fading channel: coding theorems and code construction. *IEEE Trans. on Inf. Theory*, 52(1):91–114, 2006.

[34] H. Jin, A. Khandekar, and R. McEliece. Irregular repeat-accumulate codes. In *Proc. 2nd Int. Symp. Turbo codes and related topics*, pages 1–8, 2000.

[35] K. Kasai and K. Sakaniwa. Fountain codes with multiplicatively repeated non-binary ldpc codes. In *Int. Symp. on Turbo Codes and Iterative Information Processing (ISTC)*, pages 374–378, 2010.

[36] K. Kasai, D. Declercq, C. Poulliat, and K. Sakaniwa. Multiplicatively repeated non-binary LDPC codes. *IEEE Trans. on Inf. Theory*, 57(10):6788–6795, 2011.

[37] D. Klinc, J. Ha, and S.W McLaughlin. On rate-adaptability of non-binary LDPC codes. In *5th International Symposium on Turbo codes and related topics*, 2008.

[38] R. Knopp and P.A. Humblet. On coding for block fading channels. *IEEE Trans. on Inf. Theory*, 46(1):189–205, 2000.

[39] Y. Kou, S. Lin, and M. P. C. Fossorier. Low density parity check codes based on finite geometries: a rediscovery and new results. *IEEE, Trans. Inform. Theory*, 47(7): 2711–2735, 2001.

[40] A. Lapidoth. The performance of convolutional codes on the block erasure channel using various finite interleaving techniques. *IEEE Trans. on Inf. Theory*, 40(5):1459–1473, 1994.

[41] F. Lehman. *Les systèmes de décodage itératif et leurs applications aux modems filaires et non-filaires.* PhD thesis, INPG, 2002. Grenoble.

[42] F. Lehmann and G.M. Maggio. Analysis of the iterative decoding of ldpc and product codes using the gaussian approximation. *Information Theory, IEEE Transactions on*, 49(11):2993–3000, 2003.

[43] G. Li, I. J. Fair, and W. A. Krzymie. Density evolution for nonbinary LDPC codes under gaussian approximation. *IEEE Trans. Inform. Theory*, 55(3), March 2009.

[44] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Improved low-density parity-check codes using irregular graphs. *Information Theory, IEEE Transactions on*, 47(2):585–598, 2001.

[45] E. Lutz. Code and interleaver design for data transmission over fading channels. In *Proceedings Global Telecommunications Conference*, pages 12–4, 1984.

[46] D. J. C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inform. Theory*, 45(2):399–431, 1999.

[47] E. Malkamaki and H. Leib. Coded diversity on block-fading channels. *IEEE Trans. on Inf. Theory*, 45(2):771–781, 1999.

[48] E. Malkamaki and H. Leib. Evaluating the performance of convolutional codes over block fading channels. *IEEE Trans. on Inf. Theory*, 45(5):1643–1646, 1999.

[49] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang. Stopping sets and the girth of tanner graphs. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, page 2, 2002.

[50] A. Orlitsky, K. Viswanathan, and J. Zhang. Stopping set distribution of ldpc code ensembles. *Information Theory, IEEE Transactions on*, 51(3):929–953, 2005.

[51] L.H. Ozarow, S. Shamai, and A.D. Wyner. Information theoretic considerations for cellular mobile radio. *Vehicular Technology, IEEE Transactions on*, 43(2):359–378, 1994.

[52] S. Pfletschinger and D. Declercq. Non-binary coding for vector channels. In *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 26–30, 2011.

[53] L. Pham Sy. *Codes correcteurs d'erreurs au niveau applicatif pour les télécommunications par satellite*. PhD thesis, ENSEA, Cergy-Pontoise, 2012.

[54] C. Poulliat, M. Fossorier, and D. Declercq. Design of regular $(2, d_c)$-ldpc codes over gf(q) using their binary images. *IEEE Trans. on Communications*, 56(10):1626–1635, 2008.

[55] J. Ramsey. Realization of optimum interleavers. *Information Theory, IEEE Transactions on*, 16(3):338–345, 1970.

[56] V. Rathi and R. Urbanke. Density Evolution, Thresholds and the Stability Condition for Non-binary LDPC Codes. *IEE Proceedings Communications*, 152(6), 2005.

[57] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[58] T. Richardson. Error floors of LDPC codes. In *Proc. Annual Allerton Conference on Coomunication Control and Computing*, volume 41, pages 1426–1435, 2003.

[59] T. J. Richardson and R. L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inform. Theory*, 47(2):599–618, 2001.

[60] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity approaching irregular low density parity check codes. *IEEE Trans. Inform. Theory*, 47 (2):619–637, 2001.

[61] T.J. Richardson and R.L. Urbanke. Efficient encoding of low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2):638–656, 2001.

[62] T.J. Richardson and R.L. Urbanke. *Modern coding theory*. Cambridge Univ Pr, 2008.

[63] L. Sassatelli and D. Declercq. Non-binary hybrid ldpc codes: Structure, decoding and optimization. In *Information Theory Workshop, 2006. ITW'06 Chengdu. IEEE*, pages 71–75, 2006.

[64] V. Savin. Non binary LDPC codes over the binary erasure channel: density evolution analysis. In *IEEE Int. Symp. Applied Sciences on Biomedical and Communication Technologies (ISABEL)*, 2008.

[65] V. Savin. Min-Max decoding for non binary LDPC codes. In *IEEE Int. Symp. on Information Theory*, July 2008.

[66] C.E. Shannon. A mathematical theory of communications, i and ii. *Bell Syst. Tech. J*, 27:379–423, 1948.

[67] H. Song and JR Cruz. Reduced-complexity decoding of q-ary ldpc codes for magnetic recording. *Magnetics, IEEE Transactions on*, 39(2):1081–1087, 2003.

[68] D. Sridhara and T.E. Fuja. Low density parity check codes over groups and rings. In *IEEE Information Theory Workshop. Proceedings of the 2002*, pages 163–166, 2002.

[69] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[70] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[71] R. M. Tanner. Minimum distance bounds by graph analysis. *IEEE Transactions on Information Theory*, 47(2):808–821, 2001.

[72] A. Venkiah, D. Declercq, and C. Poulliat. Design of cages with a randomized progressive edge-growth algorithm. *Communications Letters, IEEE*, 12(4):301–303, 2008.

[73] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

[74] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard. Low-complexity, low-memory ems algorithm for non-binary LDPC codes. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 671–676, 2007. URL `http://ieeexplore.ieee.org/iel4/4288670/4288671/04288786.pdf`.

[75] B. Vucetic and J. Yuan. *Turbo codes: principles and applications*. Springer, 2000.

[76] N. Wiberg. *Codes and decoding on general graphs*. PhD thesis, Likoping University, 1996. Sweden.

[77] H. Wymeersch, H. Steendam, and M. Moeneclaey. Log-domain decoding of ldpc codes over gf (q). In *Communications, 2004 IEEE International Conference on*, volume 2, pages 772–776, 2004.