Analysis and Design of Modern Coding Schemes for Unequal Error Protection and Joint Source-Channel Coding

by Humberto Vasconcelos Beltrão Neto

A thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in Electrical Engineering

Dissertation Committee:
Prof. Dr.-Ing. Werner Henkel, Jacobs University Bremen
Prof. Dr. Valdemar Cardoso da Rocha Jr., Universidade Federal de Pernambuco
Prof. Dr. Jon Wallace, Jacobs University Bremen
Prof. Dr. Gerhard Kramer, Technische Universität München
Prof. Dr. Ricardo Campello de Souza, Universidade Federal de Pernambuco

---

Date of Defense: July 16, 2012
School of Engineering and Science - Jacobs University Bremen

To Lúcia, Miriam, and Antônio

Acknowledgments

This work is the result of my research in the School of Engineering and Science at Jacobs University Bremen developed in cooperation with the Department of Electronics and Systems at the Federal University of Pernambuco.

First of all, I would like to thank my both advisors Prof. Dr.-Ing. Werner Henkel and Prof. Dr. Valdemar da Rocha Jr. for their supervision and friendship. I am very grateful not only for their orientation, but also for the confidence and academic liberty they gave me. Furthermore, I want to thank Prof. Dr. Jon Wallace, Prof. Dr. Ricardo Campello, and Prof. Dr. Gerhard Kramer for accepting to join the dissertation committee.

I also would like to thank my colleagues Neele von Deetzen, Khaled Hassan, Abdul Wakeel, Fangning Hu, Jalal Etesami, Alexandra Filip, Khodr Saaifan, Oana Graur, and Apirath Limmanee for their company, friendship, interesting discussions, and support.

I do not have enough words to express my gratitude to Patricia Wand and Moritz Lehmann. They were more than my flatmates and German teachers, they were my family. I also have to thank my old and great friend Leandro Rocha, who even not being in Bremen, was a constant and important presence in my life. My stay in Germany would not have been half as enjoyable without them, and I can surely say they are friends for life.

This section would not be complete if I did not mention the people to whom I dedicate this thesis: my mom Lúcia, my dad Antônio, and my grandma Miriam. Even being thousands of miles away, they were a constant presence and source of inspiration. I would not have gone so far without them, who sometimes gave up their dreams so that I could make mine come true. Thank you for everything!

Finally, I would like to thank the National Counsel of Technological and Scientific Development (CNPq), the German Academic Exchange Service (DAAD), and the German Research Foundation (DFG) for the financial support.

I hereby declare that I have written this Ph.D. thesis independently, unless where clearly stated otherwise. I have used only the sources, the data, and the support that I have clearly mentioned. This Ph.D. thesis has not been submitted for conferral of degree elsewhere.

Humberto Vasconcelos Beltrão Neto
Bremen, July 2012

Abstract

In this thesis, we investigate systems based on error-correcting codes for unequal-error-protecting and joint source-channel coding applications. Unequal error protection (UEP) is a desirable characteristic for communication systems where source-coded data with different importance levels is being transmitted, and it is wasteful or even infeasible to provide uniform protection for the whole data stream. In such systems, we can divide the coded stream into classes with different protection requirements. Among the possible ways to achieve UEP, we focus on solutions based on error-correcting codes.

Regarding UEP solutions by means of coding, we first introduce an analysis of a hybrid concatenation of convolutional codes, which typically arises in the context of turbo coding schemes with unequal-error-protecting properties. We show that the analysis of such a system can be reduced to the study of serial concatenated codes, which simplifies the design of such hybrid schemes.

Additionally, we also investigate the application of graph-based codes for systems with UEP requirements. First, we perform a multi-edge-type analysis of unequal-error-protecting low-density parity-check (LDPC) codes. By means of such an analysis, we derive an optimization algorithm, which aims at optimizing the connection profile between the protection classes within a codeword of a given unequal-error-protecting LDPC code. This optimization allows not only to control the differences in the performances of the protection classes by means of a single parameter, but also to design codes with a non-vanishing UEP capability when a moderate to large number of decoding iterations is applied.

As a third contribution to UEP schemes, we introduce a multi-edge-type analysis of unequal-error-protecting Luby transform (UEP LT) codes. We derive the density evolution equations for UEP LT codes, analyze two existing techniques for constructing UEP LT codes, and propose a third scheme, which we named flexible UEP LT approach. We show by means of simulations that our proposed codes have better performances than the existing schemes for high overheads and have advantages for applications where a precoding of data prior to the channel encoding is needed.

In the last part of the thesis, we investigate joint source-channel coding schemes where low-density parity-check codes are applied for both source and channel encoding. The investigation is motivated by the fact that it is widely observed that for communication

systems transmitting in the non-asymptotic regime with limited delay constraints, a separated design of the source and channel encoders is not optimum, and gains in complexity and fidelity may be obtained by a joint design strategy. Furthermore, regardless of the fact that the field of data compression has reached a state of maturity, there are state-of-the-art applications which do not apply data compression, thus failing to take advantage from the source redundancy in the decoding.

Within this framework, we propose an LDPC-based joint source-channel coding scheme and by means of the multi-edge analysis previously developed, we propose an optimization algorithm for such systems. Based on syndrome source encoding, we propose a novel system where the amount of information about the source bits available at the decoder is increased by improving the connection profile between the factor graphs of the source and channel codes that compose the joint system.

Lastly, by means of simulations, we show that the proposed system achieves a significant reduction of the error floor caused by the encoding of messages that correspond to uncorrectable error patterns of the LDPC code used as source encoder in comparison to existent LDPC-based joint source-channel coding systems.

Contents

Chapter 1

Introduction

Digital communication systems are so ingrained in our every-day life that it becomes increasingly difficult to imagine a world without it. They are everywhere from mobile telephones to deep-space communication and have been developing at breathtaking pace since 1948 with the publication of Shannon's landmark paper "A mathematical theory of communication" [1]. At a time when it was believed that increasing the rate of information transmission over a channel would increase the probability of error, Shannon proved in his channel coding theorem that this is not true. Communication with a vanishing error probability is indeed possible as long as the transmission rate is kept below the channel capacity. The way to achieve it: coding.

Since the development of the first non-trivial error-correcting codes by Golay [2] and Hamming [3], a lot of work has been done on the development of efficient coding and decoding methods for error control of transmissions over noisy channels. Nevertheless, it was not until the 1990's that practical capacity achieving coding schemes were developed with the advent of turbo codes by Berrou, Glavieux, and Thitimajshima [4] and the rediscovery of Gallager's low-density parity-check codes [5]. Those two schemes share in common the fact that their most used decoding algorithms are based on iterative techniques and, together with Luby transform codes [6], are central to this thesis where we investigate their application for unequal-error-protecting and joint source-channel coding systems.

Unequal error protection is a desirable characteristic for communication systems where source bits with different sensitivities to errors are being transmitted, and it is wasteful or even infeasible to provide uniform protection for the whole data stream. There are mainly three strategies to achieve unequal error protection on transmission systems: bit loading, multilevel coded modulation, and channel coding [7]. In the present work, we focus on the latter, more specifically on low-density parity-check and Luby transform codes. Additionally, we present some results applicable to the design of concatenated coding schemes used within an unequal error protection framework.

Last but not least, we study the problem of joint source-channel coding. The main idea when dealing with the joint source-channel coding/decoding problem is to take advantage of the residual redundancy arising from an incomplete data compression in order to improve the error rate performance of the communication system. This possibility was already mentioned by Shannon in [1] and quoted by Hagenauer in [8]: "However, any redundancy in the source will usually help if it is utilized at the receiving point. In particular, if the source already has redundancy and no attempt is made to eliminate it in matching to the channel, this redundancy will help combat noise." The approach we chose for joint source-channel coding in this thesis is based on low-density parity-check codes and syndrome source encoding.

The outline of this thesis is as follows: First, in Chapter 2, we introduce the communication system and transmission model we are going to assume. Furthermore, we present some basic concepts on coding that are essential for a full understanding of the subsequent chapters. Chapter 3 describes the relation between the two different kinds of extrinsic information transfer charts that arise in the analysis of a hybrid concatenated turbo coding scheme used to achieve unequal error protection capabilities. From this analysis, it is shown that both kinds of charts can be used to analyze the iterative decoding procedure of such hybrid concatenated codes. Finally, it is shown that the analysis of the hybrid turbo codes can be reduced to the study of the component serial concatenated codes.

Chapter 4 deals with low-density parity-check codes with unequal-error-protecting capabilities. It is known that irregular low-density parity-check codes are particularly well-suited for transmission schemes that require unequal error protection of the transmitted data due to the different connection degrees of its variable nodes. However, this capability is strongly dependent on the connection profile among the protection classes defined within a codeword. We derive a multi-edge-type analysis of low-density parity-check codes to optimize such connection profiles according to the performance requirements of each protection class. This allows the construction of low-density parity-check codes where the difference between the performance of the protection classes can be adjusted and with an unequal error protection capability that does not vanish as the number of decoding iterations grows.

In Chapter 5, a multi-edge framework for unequal-error-protecting Luby transform codes similar to the one presented in Chapter 4 is derived. Under the framework introduced, two existing techniques for the design of unequal-error-protecting Luby transform codes can be evaluated and explained in a unified way. Furthermore, we propose a third design methodology for the construction of unequal-error-protecting Luby transform codes which compares favorably to the design techniques already present in the literature.

The multi-edge framework applied in chapters 4 and 5 is then used in Chapter 6 to the joint source-channel coding problem. The approach followed in this chapter relies on a graphical model where the structure of the source and the channel codes are jointly exploited. More specifically, we are concerned with the optimization of joint systems that perform linear encoding of the source output and channel input by means of low-density parity-check codes. We present a novel system where the amount of information about the source bits available at the decoder is increased by improving the connection profile between the factor graphs of the source and channel codes that compose the joint system and propose the application of low-density parity-check codes to the syndrome-based source encoding. Furthermore, we propose an optimization strategy for the component codes based on a multi-edge-type joint optimization.

Chapter 7 summarizes the results of this thesis and considers possible future work.

Chapter 2

Basic Concepts

The present chapter describes the system model considered in the development of this thesis. Furthermore, it introduces concepts, notation, and techniques which will be necessary for the understanding of the forthcoming chapters.

2.1 Communication system

The digital communication system model considered throughout this thesis was first established by Shannon in [1]. Figure 2.1 shows the components of such a model at both the transmitter and receiver sides including the transmission channel.

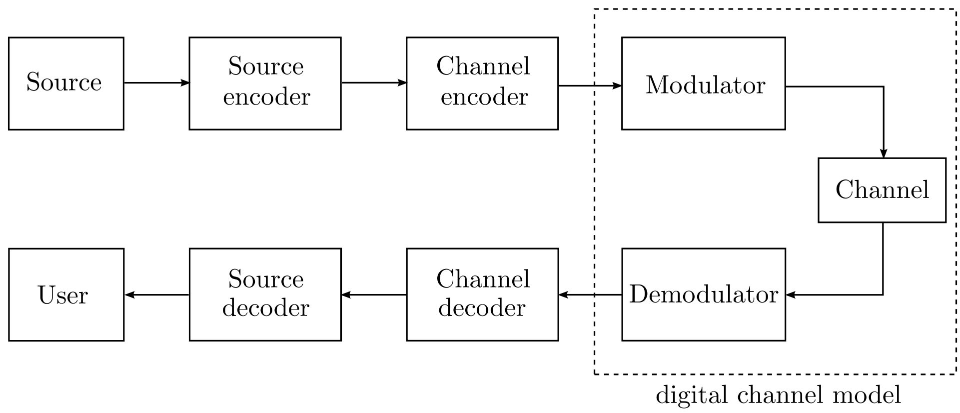

Figure 2.1: Basic digital communication system block diagram.

Throughout this work, we assume that the source is digital with output consisting of a stream of symbols over GF(2). However, systems with analog sources can be easily included into this framework assuming that the source output has been digitized before its delivery to the source encoder.

As a first step prior to transmission, the information received from the source is *compressed* by the source encoder, i.e., the source encoder turns its representation into one with fewer symbols. The compression consists in reducing the redundancy present in the source output to a minimum in order to transmit only the information essential for the reconstruction of the original source output at the receiver. The compressed information is then delivered to the channel encoder which adds redundancy to the received symbol sequence in order to protect it against the effects of distortion, interference, and noise present in the communication channel. The channel encoded sequence is then modulated and transmitted.

The role of the modulator is to turn the output of the channel encoder into a form suitable for transmission. For wireless transmissions for example, the size of the transmitting antenna is proportional to the wavelength of the signal to be transmitted. This means that in order to use antennas of reasonable size, the original bit stream should be represented by a high-frequency signal. We will consider that the modulator, channel, and demodulator form a single block which we will refer to as the digital channel as indicated in Fig. 2.1.

After its arrival at the receiver, the transmitted information is first demodulated and then forwarded to the channel decoder. The channel decoder makes use of the redundancy introduced at the transmitter side to correct possible errors introduced by the transmission channel. After that, the information is finally decompressed by the source decoder and delivered to the user.

This simplified model of a digital communication system is sufficient to describe the work presented in this thesis. Except when specifically stated, we assume that the input to the channel encoders are sequences of binary digits perfectly compressed by the source, i.e., with no leftover redundancy. This is equivalent to assuming that the occurrences of both binary symbols are i.i.d. and have the same probability. Later, we deal with the problem of how to combine source and channel encoders and decoders in order to take advantage of any redundancy resulting from an imperfect compression.

2.2 Transmission model

The transmission of information between the transmitter and receiver takes place over a communication channel. Broadly speaking, a channel is a physical medium through which the information is transmitted or stored. For our purposes, we will adopt an information theoretic approach and follow the channel definition of [9], i.e., we define a channel as a system consisting of an input alphabet $\mathcal{X}$, an output alphabet $\mathcal{Y}$, and a probability transition matrix $p(y|x)$ that expresses the probability of observing the output symbol y given that the symbol x was transmitted, i.e., a matrix of conditional probabilities of y given x.

Among the myriad of channel models present in the literature, we are mainly interested in three models: the binary symmetric channel (BSC), the binary erasure channel (BEC), and the binary input additive white-Gaussian-noise (BI-AWGN) channel . The formal definition of these three models is given as follows

Definition 1 (Binary symmetric channel) *A binary symmetric channel (BSC) is a channel with input alphabet* $\mathcal{X} = \{0, 1\}$*, output alphabet* $\mathcal{Y} = \{0, 1\}$*, and the following set of conditional probabilities*

$$p(y = 0|x = 0) = p(y = 1|x = 1) = 1 - p$$
$$p(y = 1|x = 0) = p(y = 0|x = 1) = p\ .$$

A graphical representation of the BSC can be seen in Fig. 2.2.

The BSC channel is maybe the simplest channel model, but still it captures most of the features of the general transmission problem. The next definition regards another simple but important model: the binary erasure channel (BEC). Introduced by Elias in [10], this model is particularly well-suited to modeling channels where the transmission is done by means of *packets* that are either received correctly or completely lost. Since this kind of transmission is ubiquitous in the Internet, the BEC, which was previously regarded as a toy example, became a real-world channel model.

Definition 2 (Binary erasure channel) *A binary erasure channel (BEC) is a channel with input alphabet* $\mathcal{X} = \{0, 1\}$*, output alphabet* $\mathcal{Y} = \{0, 1, ?\}$*, where* ? *indicates an* *erasure, and the following set of conditional probabilities*

$$p(y=0|x=0) = p(y=1|x=1) = 1-\epsilon$$
$$p(y=1|x=0) = p(y=0|x=1) = 0$$
$$p(y=?|x=0) = p(y=?|x=1) = \epsilon\ .$$

A graphical representation of the BEC can be seen in Fig. 2.3.

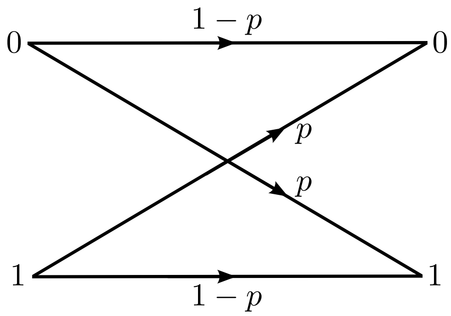

Figure 2.2: Binary symmetric channel.

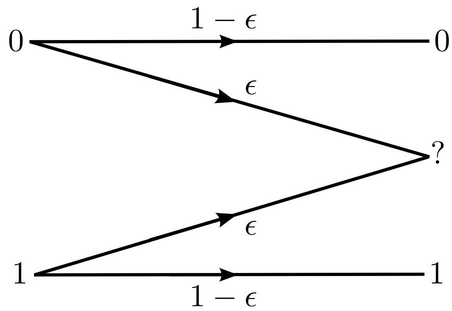

Figure 2.3: Binary erasure channel.

The last channel model we introduce is the binary input additive white-Gaussian-noise channel (BI-AWGNC). We define the BI-AWGNC as follows

Definition 3 (Binary input additive white Gaussian-noise channel) *A binary input additive white Gaussian-noise channel is a channel with input alphabet* $\mathcal{X} = \{-1,+1\}$ *and output alphabet* $\mathcal{Y} = \mathbb{R}$*, with the following set of conditional probabilities*

$$p(y|x) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-(y-x)^2/(2\sigma_n^2)\right]\ , \tag{2.1}$$

where σ_n^2 *is the variance of a zero-mean Gaussian noise sample* n *that the channel adds to the transmitted value* x*, so that* $y = x + n$*. The graphical model of the BI-AWGN channel can be seen in Fig. 2.4.*

Note that for the transmission over the BI-AWGN channel, we consider that each of the binary digits emitted by the channel encoder $c \in \{0,1\}$ is mapped to channel inputs $x \in \{-1,+1\}$ prior to the transmission following the rule $x = (-1)^c$, so that $x = +1$ when $c = 0$.

An important figure of merit of communication channels is their capacity. The capacity of a channel is defined as the maximum amount of information that can be transmitted

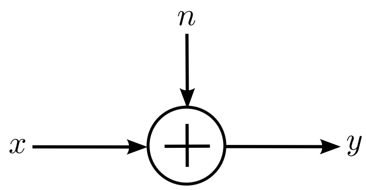

Figure 2.4: Binary input additive white Gaussian-noise channel.

per channel use. In order to mathematically define the channel capacity, we need to introduce two basic information theoretic definitions: entropy and mutual information.

Definition 4 (Entropy) *The entropy (or uncertainty) of a random variable X with probability mass function $p(x)$ is defined as*

$$H(X) = -\sum_{x} p(x) \log_2 p(x). \tag{2.2}$$

Furthermore, the conditional entropy between two random variables (X, Y) is defined as

$$H(X|Y) = -\sum_{x,y} p(x, y) \log_2 p(x|y). \tag{2.3}$$

The entropy may be interpreted as the amount of information we receive when observing the outcome of a random variable X, i.e., the uncertainty we have about the outcome of X. Given the concept of entropy, we can present a central definition in information theory.

Definition 5 (Mutual information) *Let X and Y be two random variables. The mutual information $I(X;Y)$ between X and Y is defined as*

$$I(X;Y) = H(X) - H(X|Y) . \tag{2.4}$$

The mutual information is simply the reduction of the uncertainty about the outcome of X that we get from knowing the outcome of Y. This posed, the capacity of a channel with input X and output Y is defined as

$$C = \max_{p(x)} I(X;Y) . \tag{2.5}$$

The channel capacity has a central role in information theory due to the fact that

Shannon demonstrated in its noisy-channel coding theorem [1] that communication with infinite reliability is possible as long as the transmission rate is kept below the capacity of the communication channel. A more detailed approach to channel capacity, including its computation for a series of important channel models, can be found in [9].

2.3 Channel coding

Channel coding is an essential feature of modern communication and storage systems. In a world where data needs to be transmitted at ever increasing speeds, it becomes essential to find coding schemes capable of providing reliable communication with the highest possible transmission rate. The noisy-channel coding theorem states that reliable communication at rates up to the channel capacity is possible, but its proof is unfortunately not constructive.

The search of practical coding schemes able to approach capacity has been subject of a lot of research, and until the 1990's it was thought that capacity achieving codes were impractical. With the invention of turbo codes [4] and the rediscovery of low-density parity-check codes [5], it was demonstrated that codes that operate very close to capacity are indeed practical.

In this section, we lay down some principles of channel coding, which will be necessary to understand the underlying principles of these capacity achieving codes.

2.3.1 Linear block codes

In this work, we assume that the information emitted by the source is a sequence of k binary symbols $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$. A block code is a bijective mapping that maps each length-k message block into a length-n codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$. If the linear combination of any pair of codewords from a block code is also a codeword, the code is said to be a linear block code. We can define linear block codes using vector space theory as follows [11]

Definition 6 (Linear block code) *A block code of length n and 2^k codewords is called a linear $C(n, k)$ code if and only if its 2^k codewords form a k-dimensional subspace of the vector space of all the n-tuples over the field $GF(2)$.*

Let V_n denote the vector space of all the n-tuples over the field $GF(2)$ and $\mathbf{G}$ be a $(k \times n)$ matrix whose rows form a basis of a k-dimensional subspace of V_n. It is not difficult to see that the k rows of $\mathbf{G}$ span the linear code $C(n,k)$. The matrix $\mathbf{G}$ is called the generator matrix of the code $C(n,k)$. The rate of a code is defined as

Definition 7 (Code rate) *The rate of a binary block code is defined as follows*

$$R = \frac{k}{n} \ . \tag{2.6}$$

Note that every codeword is a linear combination of the rows of the generator matrix, i.e., for a message vector $\mathbf{u}$ the corresponding codeword $\mathbf{c}$ is given by

$$\mathbf{c} = \mathbf{u} \cdot \mathbf{G} \ , \tag{2.7}$$

where the "$\cdot$" denotes the inner product over $GF(2)$. Another important matrix used in the decoding of linear block codes is the $(n-k) \times n$ parity-check matrix $\mathbf{H}$. The parity-check matrix is defined as the matrix that generates the null-space of the code $C(n,k)$, i.e., for every codeword $\mathbf{c}$ in $C(n,k)$ the following equality holds

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0} \ . \tag{2.8}$$

Suppose that we transmitted the codeword $\mathbf{c}$ and received the vector $\mathbf{r} = \mathbf{c} + \mathbf{e}$, where $\mathbf{e}$ is called the error vector. According to Eq. 2.8, we have

$$\mathbf{r} \cdot \mathbf{H}^T = (\mathbf{c} + \mathbf{e}) \cdot \mathbf{H}^T = \underbrace{\mathbf{c} \cdot \mathbf{H}^T}_{=0} + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \ , \tag{2.9}$$

that is, we can detect an error in the transmission by computing the inner product between the received vector and the transpose of the parity-check matrix. The above definitions are sufficient for the purposes of this thesis. A more elaborated description of linear block codes can be found in [11].

2.3.2 Convolutional codes

Convolutional codes were proposed by Elias in [12] as an alternative to block codes. In contrast to block codes, the n output symbols of a convolutional encoder at a certain time do not only depend on the current k, but also on the past M input symbols,

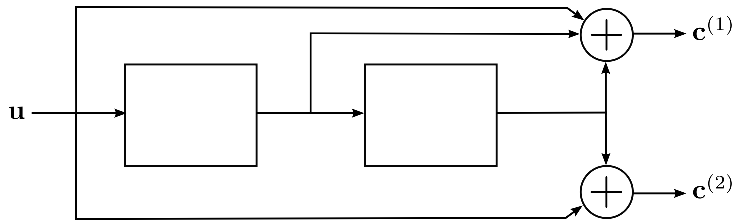

Figure 2.5: Four-state, rate 1/2, convolutional code encoder.

where M is referred to as the memory of the code. Our goal in this section is to introduce a simple description of convolutional codes, their encoding, and mathematical representation. A more detailed presentation can be found in [11, 13].

The most common way of introducing convolutional codes is through the block diagram representation of their encoder. Figure 2.5 depicts a binary convolutional encoder. The boxes represent the memory elements, and the state of a convolutional encoder is defined to be the contents of its binary memory elements. Note that for the encoder of Fig. 2.5, we have $n = 2$ outputs for each $k = 1$ input, so it is a rate-1/2, four-state convolutional encoder. Unlike block codes, the input and output of convolutional codes are (infinite) sequences.

A linear convolutional code may be represented using *generator polynomials*. In general, there are $k \times n$ generator polynomials, which are degree M polynomials $\mathbf{g}_i^{(j)}(D)$ whose coefficients are the response at output j to an impulse applied at input i. For example, the encoder of Fig. 2.5 has impulse responses $\mathbf{g}^{(1)} = [1\ 1\ 1]$ and $\mathbf{g}^{(2)} = [1\ 0\ 1]$, where we omit the index i, since there is only one input. Thus, its generator polynomials are $\mathbf{g}^{(1)} = 1 + D + D^2$ and $\mathbf{g}^{(2)} = 1 + D^2$, where D is equivalent to the discrete time delay operator z^{-1}.

A compact way to represent the encoding of convolutional codes is through the following matrix expression

$$\mathbf{C}(D) = \mathbf{U}(D)\mathbf{G}(D) \ , \tag{2.10}$$

where $\mathbf{U}(D) = [\mathbf{u}^{(1)}(D), \mathbf{u}^{(2)}(D), \cdots, \mathbf{u}^{(k)}(D)]$ is the k-tuple of input sequences, $\mathbf{C}(D) = [\mathbf{c}^{(1)}(D), \mathbf{c}^{(2)}(D), \cdots, \mathbf{c}^{(n-1)}(D)]$ is the n-tuple of output sequences, and $\mathbf{G}(D)$ is the $k \times n$ matrix with $\mathbf{g}_i^{(j)}(D)$ as the elements at line i and column j. The matrix $\mathbf{G}(D)$ is called the code's generator matrix. For the code of Fig. 2.5 we have $\mathbf{G}(D) =$

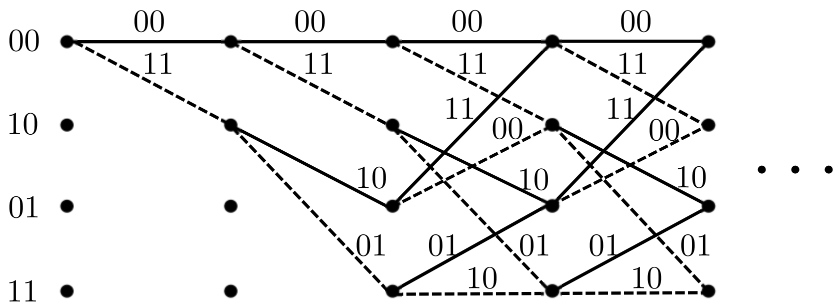

Figure 2.6: Trellis of the rate-1/2, NSC, convolutional code of Fig. 2.5.

$[1 + D + D^2 \quad 1 + D^2]$, thus, Eq. 2.10 can be written as

$$[\mathbf{c}^{(1)}(D) \quad \mathbf{c}^{(2)}(D)] = \mathbf{u} \cdot [1 + D + D^2 \quad 1 + D^2] \, . \tag{2.11}$$

Convolutional encoders are mostly represented as non-recursive non-systematic convolutional (N$\bar{\text{S}}$C) or as recursive systematic convolutional (RSC) encoders. We say that an encoder is recursive if it presents a feedback in its realization and, as a consequence, has a generator matrix $\mathbf{G}(D)$ with at least one rational function among its entries. Conversely, a non-recursive encoder does not have any feedback on its realization, and thus, its $\mathbf{G}(D)$ matrix does not have any rational function among its entries. The encoder of Fig. 2.5, for example, is a non-recursive non-systematic encoder.

As finite-state machines, convolutional encoders have a trellis representation where each encoded sequence is represented by a path on the trellis. Figure 2.6 shows the trellis corresponding to the convolutional encoder of Fig. 2.5. Convolutional codes have several trellis-based decoding algorithms, e.g., list decoding [13], Viterbi algorithm [14], and the BCJR algorithm [15]. A detailed description of the decoding of convolutional codes is out of the scope of the thesis, and we will simply refer to the given literature.

2.4 Low-density parity-check codes

Low-density parity-check (LDPC) codes are linear block codes whose parity-check matrix is sparse. Due to its central role in the development of this thesis, we proceed to a more thorough exposure of the theory involving this class of block codes. LDPC codes can be conveniently represented by bipartite graphs[1] where a set of nodes, the variable (or symbol) nodes, represent the code bits and the other set, the check (or constraint) nodes, represent the parity-check equations which define the code.

The number of edges connected to a node is called the degree of the node. A graph is said to be (d_v, d_c)-regular if all variable nodes have the same degree d_v and all check nodes have the same degree d_c. Figure 2.7 depicts the regular factor graph of a block code with the following parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} . \tag{2.12}$$

Throughout this work, we will focus on irregular LDPC codes, since they are known to approach the capacity more closely than regular LDPC codes. An ensemble of irregular

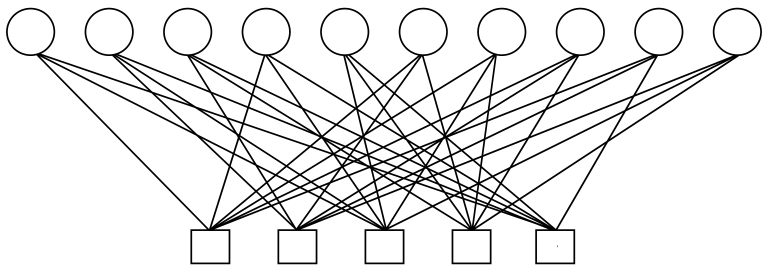

Figure 2.7: Factor graph of a (3,6) regular block code of size $n = 10$.

LDPC codes is specified by a codeword size and two degree distributions. Let n be the codeword size of an LDPC code and Λ_i be the number of variable nodes of degree i, so that $\sum_i \Lambda_i = n$. Similarly, let P_i be the number of check nodes of degree i, so that $\sum_i P_i = n(1-R)$, where R is the design rate.[2] Following a polynomial notation, we have

$$\Lambda(x) = \sum_{i=1} \Lambda_i x^i \ , \qquad P(x) = \sum_{i=1} P_i x^i \ ,$$

i.e., $\Lambda(x)$ and $P(x)$ are integer coefficient polynomials which represent the number

[1]The graph representation of linear block codes is also known as Tanner graphs or factor graphs. We will use those terms interchangeably through this work.

[2]The design rate is the rate of the code assuming that all constraints are linearly independent.

of nodes of a specific degree. Such polynomials are known as variable and check node degree distribution from a *node perspective*, respectively. It is also possible, for convenience, to make use of normalized degree distributions

$$\tilde{\lambda}(x) = \frac{\Lambda(x)}{\Lambda(1)}\ , \qquad \tilde{\rho}(x) = \frac{P(x)}{P(1)}\ .$$

For the asymptotic analysis, it is more convenient to utilize the degree distributions from an *edge perspective* defined by

$$\lambda(x) = \sum_i \lambda_i x^{i-1} = \frac{\Lambda'(x)}{\Lambda'(1)} = \frac{\tilde{\lambda}'(x)}{\tilde{\lambda}'(1)}\ , \quad \rho(x) = \sum_i \rho_i x^{i-1} = \frac{P'(x)}{P'(1)} = \frac{\tilde{\rho}'(x)}{\tilde{\rho}'(1)}\ .$$

Note that, λ_i (ρ_i) is the fraction of edges connected to variable (check) nodes of degree i, i.e., λ_i (ρ_i) is the probability that a randomly and uniformly chosen edge is connected to a variable (check) node of degree i. The design rate of an irregular LDPC code is given by [16]

$$R = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}\ .$$

2.4.1 Iterative decoding

The algorithm employed for decoding LDPC codes throughout the thesis is the belief propagation algorithm (BP), which is a soft-input soft-output bitwise iterative decoding algorithm. The operation of the BP algorithm consists in determining the *a posteriori* probability of each message symbol based on the received signal, code constraints, and channel conditions. The reliability of each symbol at the end of each iteration is then used as an input to the next iteration. The reliability measure used here is the log-likelihood ratio (LLR).

Before presenting the BP algorithm, we need to introduce some notation. Let $\mathcal{N}(c)$ denote the neighborhood of a check node c.[3] Similarly, let $\mathcal{M}(v)$ denote the neighborhood of a variable node v. The set $\mathcal{N}(c)$ with the node v excluded is indicated by $\mathcal{N}(c)\backslash v$. Let $q_{v\to c}$ be the messages sent from a symbol node v to the check node c. Finally, let $r_{c\to v}$ be the message sent from the check node c to the variable node v. Having introduced

[3] The neighborhood of a node is composed of all its adjacent nodes. Two node are said to be adjacent if they are connected through an edge.

the notation, we can now describe the four parts of the BP algorithm following the concepts introduced in [17] and the presentation of [18]: initialization, check nodes update, variable nodes update, and termination as follows.

Algorithm 1 Belief propagation

1. **Initialization-** Let x_v denote the represented value of a symbol node v and y_v be the channel observation regarding x_v. At the initialization step, each variable node computes an initial LLR $L(y_v|x_v) = \ln(p(y_v|x_v = 0)/p(y_v|x_v = 1))$. Then, every variable node sends to their neighbors the message

$$L(q_{v\to c}) = L(y_v|x_v) \ .$$

2. **Check nodes update-** The cth check node receives the messages $L(q_{v\to c})$, where $v \in \mathcal{N}(c)$, and updates the messages $L(r_{c\to v})$ according to

$$L(r_{c\to v}) = 2 \cdot \tanh^{-1}\left(\prod_{v'\in\mathcal{N}(c)\setminus v} \tanh\left(\frac{L(q_{v'\to c})}{2}\right)\right) \ .$$

3. **Variable nodes update-** The vth variable node receives $L(r_{c\to v})$, where $c \in \mathcal{M}(v)$, and updates $L(q_{v\to c})$ according to

$$L(q_{v\to c}) = L(y_v|x_v) + \sum_{c'\in\mathcal{M}(v)\setminus c} L(r_{c'\to v}) \ .$$

4. **Termination-** The decoder computes the *a posteriori* information regarding the symbol v through the sum of the channel information and all the messages transmitted to v by its neighboring check nodes,

$$A_v = L(y_v|x_v) + \sum_{c\in\mathcal{M}(v)} L(r_{c\to v}) \ .$$

The algorithm stops if a valid codeword is found, i.e., the hard decision $\hat{\mathbf{x}}$ of the vector $A = (A_1, A_2, ..., A_n)$, where

$$\hat{x}_v \triangleq \begin{cases} 0, & \text{if } \ A_v \geq 0 \ ; \\ 1, & \text{otherwise}, \end{cases}$$

fulfills the condition $\hat{\mathbf{x}}\mathbf{H}^{\mathbf{T}} = \mathbf{0}$, or a predetermined maximum number of iterations is reached.

Notice that the BP algorithm is optimal for cycle-free graphs. Since the majority of the known practical codes (and the codes we are dealing with belong to this set) do not have a cycle-free bipartite graph representation, the BP algorithm will be sub-optimal in such cases. A more detailed description of the BP algorithm can be found in [17].

2.4.2 Density evolution

The common way to access the performance of iterative decoders is by means of density evolution. When dealing with iteratively decoded LDPC codes, density evolution aims at tracking the evolution of the error probability of the variable nodes at each iteration, which is a function of the probability density functions of their incoming messages. Since this method turns out to be computationally prohibitive, the probability density functions are typically approximated by a single parameter. The mutual information between the variables associated with the variable nodes and the message received or emitted by them is typically chosen as such a parameter.

The use of mutual information leads to a description of the convergence behavior of a code by means of mutual information transfer functions. These transfer functions, usually referred to as extrinsic mutual information transfer functions, enable a simple convergence analysis of iteratively decoded systems [19] and the design of regular and irregular LDPC codes [20, 21]. In the forthcoming description, we assume infinitely long LDPC codes (asymptotic assumption) and that the messages exchanged within the decoding graph are Gaussian (Gaussian approximation) [21]. The asymptotic assumption allows us to consider that the messages arriving at a node through different edges are independent, since the corresponding graph will be cycle-free.

Under this independence assumption, the variance of the outgoing message of a degree-d_v variable node can be written as $\sigma_v^2 = \sigma_{ch}^2 + (d_v - 1)\sigma_r^2$, where σ_{ch}^2 is the variance of the received channel message, and σ_r^2 is the variance of the messages sent by the neighboring check nodes. Note that for antipodal transmission over the AWGN, the variance of the received channel message is given by $\sigma_{ch}^2 = 4/\sigma_n^2$, where σ_n^2 is the variance of the Gaussian noise. Furthermore, under the Gaussian approximation, the mutual information between the outgoing message of a degree-d_v variable node and its represented value at iteration l is given by

$$I_{v,l} = J\left(\sqrt{\sigma_{ch}^2 + (d_v - 1)[J^{-1}(I_{c,l-1})]^2}\right) , \tag{2.13}$$

where $I_{c,l-1}$ is the mean mutual information between the messages sent from a degree-d_c check node at iteration $l-1$ and the represented value of v. The $J(.)$ function relates variance and mutual information and is defined in [22] as

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{e^{\frac{(\xi-\sigma^2/2)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} \cdot \log_2[1 + e^{-\xi}]d\xi \ . \tag{2.14}$$

The function $J(\sigma)$ cannot be expressed in closed form, but it is monotonically increasing and thus invertible. According to [23], Eq. (2.14) and its inverse can be closely approximated by

$$J(\sigma) \approx (1 - 2^{-H_1\sigma^{2H_2}})^{H_3} \ , \tag{2.15}$$

$$J^{-1}(I) \approx (-\frac{1}{H_1}\log_2(1 - I^{\frac{1}{H_3}}))^{\frac{1}{2H_2}} \ , \tag{2.16}$$

with $H_1 = 0.3073$, $H_2 = 0.8935$, and $H_3 = 1.1064$.

For the computation of $I_{c,l-1}$, note that a degree-d_c check node and a degree-d_v variable node can be modeled as a length-d_c single parity-check code (SPC) and a length-d_v repetition code (REP), respectively. Thus, we can make use of the duality property for SPC and REP codes derived in [20] and write $I_{c,l-1}$ as

$$I_{c,l-1} = 1 - J\left(\sqrt{(d_c - 1)[J^{-1}(1 - I_{v,l-1})]^2}\right) . \tag{2.17}$$

For irregular LDPC codes, the mutual information between the outgoing messages of variable and check nodes and its represented values can be computed by averaging it over the different degrees. This posed, the mutual information between the messages sent from the check to variable nodes and from variable to check nodes at iteration l and their represented values, computed by means of density evolution using the Gaussian approximation, are given by

$$I_{v,l} = \sum_{i=2}^{d_{v_{max}}} \lambda_i J\left(\sqrt{4/\sigma_n^2 + (i-1)[J^{-1}(I_{c,l-1})]^2}\right) \ , \tag{2.18}$$

$$I_{c,l} = 1 - \sum_{i=2}^{d_{c_{max}}} \rho_i J\left(\sqrt{(i-1)J^{-1}(1 - I_{v,l})^2}\right) \ , \tag{2.19}$$

where $d_{v_{max}}$ and $d_{c_{max}}$ are the maximum degrees of variable and check nodes, respectively. The density evolution for LDPC codes can then be written as a function of the mutual information at the previous iteration, the noise variance, and degree distributions as,

$$I_l = F(\lambda(x), \rho(x), \sigma_n^2, I_{l-1}) \; . \tag{2.20}$$

Using Eq. (2.20), we can predict the decoding behavior and also optimize the degree distributions of an irregular LDPC code. The optimization is performed under the constraint that the mutual information between the variables nodes and their represented values should increase at every decoding iteration until its convergence to unity, i.e.,

$$F(\lambda(x), \rho(x), \sigma_n^2, I_v) \geq I_v, \; \forall \; I_v \in [0, 1) \; . \tag{2.21}$$

2.4.3 Stability condition

In order to guarantee the convergence of the error probability to zero as the number of decoding iterations tends to infinity, a given degree distribution (λ, ρ) has to fulfill the stability condition. This condition was first derived in [16] for general binary-input memoryless output symmetric channels and is a important constraint to be considered in the optimization of the degree distributions of LDPC codes. In the following, we present the stability conditions for the BIAWGN and BSC channels. A formal proof of these conditions can be found in [16, 24].

Theorem 1 (Stability condition) *Assume we are given a degree distribution pair* (λ, ρ)*. The stability condition for the binary-input AWGN channel is given by*

$$\lambda'(0)\rho'(1) < e^{\frac{1}{2\sigma_n^2}} \; ,$$

where σ_n^2 *denotes the variance of the Gaussian noise. For the binary symmetric channel, the stability condition can be written as*

$$\lambda'(0)\rho'(1) < \frac{1}{2\sqrt{p(1-p)}} \; ,$$

where p *is the crossover probability of the BSC channel.*

2.4.4 Multi-edge-type LDPC codes

Multi-edge-type LDPC codes [24,25] are a generalization of irregular and regular LDPC codes. Diverting from standard LDPC ensembles where the graph connectivity is constrained only by the node degrees, in the multi-edge setting, several edge classes can be defined, and every node is characterized by the number of connections to edges of each class. Within this framework, the code ensemble can be specified through two node-perspective multinomials associated to variable and check nodes, which are defined respectively by [24]

$$\nu(\mathbf{r},\mathbf{x}) = \sum \nu_{\mathbf{b},\mathbf{d}} \mathbf{r}^{\mathbf{b}} \mathbf{x}^{\mathbf{d}} \quad \text{and} \quad \mu(\mathbf{x}) = \sum \mu_{\mathbf{d}} \mathbf{x}^{\mathbf{d}}, \tag{2.22}$$

where $\mathbf{b}$, $\mathbf{d}$, $\mathbf{r}$, and $\mathbf{x}$ are vectors which are explained as follows. First, let m_e denote the number of edge types used to represent the graph ensemble and m_r the number of different received distributions. The number m_r represents the fact that the different bits can go through different channels and thus, have different received distributions. Each node in the ensemble graph has associated to it a vector $\mathbf{x} = (x_1, \ldots, x_{m_e})$ that indicates the different types of edges connected to it and a vector $\mathbf{d} = (d_1, \ldots, d_{m_e})$ referred to as *edge degree vector* which denotes the number of connections of a node to edges of type i, where $i \in (1, \ldots, m_e)$.

For the variable nodes, there is additionally the vector $\mathbf{r} = (r_0, \ldots, r_{m_r})$, which represents the different received distributions and the vector $\mathbf{b} = (b_0, \ldots, b_{m_r})$, which indicates the number of connections to the different received distributions (b_0 is used to indicate the puncturing of a variable node). In the sequel, we assume that $\mathbf{b}$ has exactly one entry set to 1 and the rest set to zero. This simply indicates that each variable node has access to only one channel observation at a time. We use $\mathbf{x}^{\mathbf{d}}$ to denote $\prod_{i=1}^{m_e} x_i^{d_i}$ and $\mathbf{r}^{\mathbf{b}}$ to denote $\prod_{i=0}^{m_r} r_i^{b_i}$. Finally, the coefficients $\nu_{\mathbf{b},\mathbf{d}}$ and $\mu_{\mathbf{d}}$ are non-negative reals such that if n is the total number of variable nodes, $\nu_{\mathbf{b},\mathbf{d}} n$ and $\mu_{\mathbf{d}} n$ represent the number of variable nodes of type ($\mathbf{b}$,$\mathbf{d}$) and check nodes of type $\mathbf{d}$, respectively.[4] Furthermore, we have the additional notations defined in [24]

$$\nu_{x_j}(\mathbf{r},\mathbf{x}) = \frac{\partial \nu(\mathbf{r},\mathbf{x})}{\partial x_j} \quad \text{and} \quad \mu_{x_j}(\mathbf{x}) = \frac{\partial \mu(\mathbf{x})}{\partial x_j} \; . \tag{2.23}$$

Note that, in a valid multi-edge ensemble, the number of connections of each edge type

[4] We will frequently refer to nodes with edge degree vector $\mathbf{d}$ as "type $\mathbf{d}$" nodes.

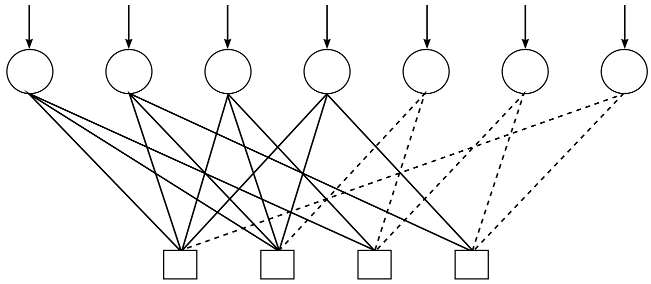

Figure 2.8: Multi-edge graph with two different edge types and one received distribution.

should be the same at both variable and check nodes sides. This gives rise to the *socket count equality* constraint, which can be written as

$$\nu_{x_j}(\mathbf{1},\mathbf{1}) = \mu_{x_j}(\mathbf{1}) \ , \quad j = 1,\ldots,m_e \ , \tag{2.24}$$

where $\mathbf{1}$ denotes a vector with all entries equal to 1, with length being clear from the context.

2.5 Luby transform codes

First introduced by Luby in [6], Luby transform (LT) codes form together with Raptor [26] and Online codes [27] the class of the so-called rateless codes. Rateless codes are particularly suitable for the transmission of data through channels that can be represented by the binary erasure channel with unknown erasure probabilities, a situation where traditional erasure correcting codes turn out to be suboptimal. Rateless codes are also very interesting for multicast transmission, since they eliminate the requirement for retransmission.

2.5.1 LT encoding

The encoding algorithm for LT codes can be described as follows. Suppose we like to encode a message composed of k input symbols. Each output symbol is formed by first determining its degree i according to a probability distribution $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$, where Ω_i denotes the probability of i being chosen. The output symbol is then formed choosing i input symbols uniformly and at random and performing an XOR operation on them. The process is repeated until a sufficient number of output symbols $n = \gamma k$

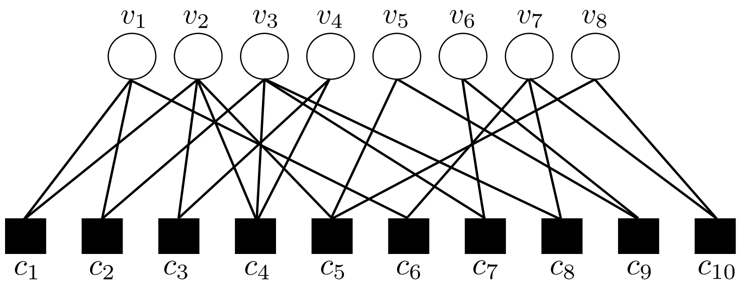

Figure 2.9: Factor graph representing the result of LT encoding for a code with $k = 8$ and $\gamma = 10/8$.

arrives at the receiver. The quantity $\gamma \geq 1$ is called the overhead. We can describe the formation of an output symbol following the LT encoding in a step by step manner as follows:

1. Randomly choose the output symbol degree i from the degree distribution $\Omega(x)$.
2. Choose uniformly and at random i symbols among the original k input symbols.
3. Form the output symbol performing the bitwise exclusive-or of the chosen i symbols.

The encoding procedure can be depicted as a bipartite graph with k variable nodes and n check nodes. Figure 2.9 shows the bipartite graph resulting from the encoding of $k = 8$ input symbols into $n = 10$ output symbols ($\gamma = 10/8$).

2.5.2 Iterative decoder

The decoding algorithm of LT codes can easily be described with the help of the graph induced by the encoding as follows

1. Find an output symbol c_j, for $j = 1, \ldots, n$, that is connected to only one input symbol v_i. In case there is no output symbol fulfilling this condition, the decoding is halted and more output symbols will be required for successful decoding.
   (a) Determine v_i as $v_i = c_j$,
   (b) Add the value of v_i to all its neighboring output symbols,
   (c) Remove v_i together with all edges emanating from it from the graph.

2. Repeat (1) until every v_i, for $i = 1, \ldots, k$, is recovered.

Note that it is supposed here that the decoder knows the degree and the set of neighbors of each output symbol. Strategies to accomplish this can be found in [6]. The description done so far considers LT codes where every input symbol has the same protection requirements (equal-error-protecting LT codes). In Chapter 5, we present modifications to the LT encoding procedure in order to derive LT codes with unequal-error-protecting capability.

2.6 Extrinsic information transfer charts

Introduced by ten Brink in [28], extrinsic information transfer (EXIT) charts are a simple but powerful method to investigate the convergence behavior of iterative decoding. Let I_a denote the average mutual information between the bits represented by the variable nodes and the *a priori* LLR values at the decoder input. In the same way, let I_e denote the mutual information between the bits represented by the variable nodes and the *extrinsic* log-likelihood values at the decoder output.

An EXIT chart is a graphical representation of the transfer functions $I_e = T(I_a)$ of the constituent decoders inside the same plot, i.e., it shows the relation between the *a priori* information at the input and the extrinsic information at the output of both constituent decoders of a iterative system. Drawing both transfer curves into the same plot is only possible due to the fact that the extrinsic information of one constituent decoder becomes the *a priori* information of the other at each decoding iteration.

Herein, we consider systems with two constituent decoders, and consequently, our EXIT charts will be two-dimensional. Nevertheless, for systems with more than two constituent decoders, two-dimensional EXIT charts can be constructed if all the decoders have the same transfer functions, e.g., symmetric multiple concatenated codes [29]. In the following, we construct the EXIT chart of a regular LDPC code to clarify the concepts we just mentioned.

The structure of an LDPC decoder is depicted in Fig. 2.10 [30]. The edge interleaver connects the variable (VND) and check nodes (CND). Throughout the decoding, each component decoder in Fig. 2.10 converts *a priori* log-likelihood ratios (L-values) into *a posteriori* L-values. If we subtract the *a priori* L-values from the resulting *a posteriori*

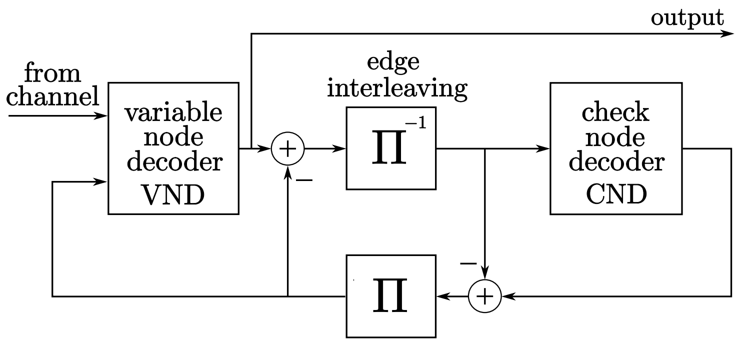

Figure 2.10: Iterative decoder structure of an LDPC code.

L-value, we obtain what is called the *extrinsic* L-value. In the following iteration, the extrinsic L-value sent by a component decoder is used as *a priori* information by the other one.

Assuming a Gaussian approximation for the messages exchanged between variable and check nodes, the transfer function of a degree-d_v variable node is given by Eq. (2.13) substituting $I_{c,l-1}$ by I_a, i.e.,

$$I_{e,VND} = J\left(\sqrt{4/\sigma_n^2 + (d_v - 1)[J^{-1}(I_a)]^2}\right) . \tag{2.25}$$

Similarly, replacing $I_{v,l-1}$ by I_a in Eq. (2.17), we can write the transfer function of a degree-d_c check node as

$$I_{e,CND} = 1 - J\left(\sqrt{(d_c - 1)J^{-1}(1 - I_a)^2}\right) . \tag{2.26}$$

With the transfer functions for both the VND and CND, we can construct the EXIT chart of the system shown in Fig. 2.10 and hence predict the convergence behavior of the code. In our example, we consider a regular LDPC with $d_v = 3$ and $d_c = 6$. The resulting EXIT chart is depicted in Fig. 2.11.

Figure 2.11 shows the curves $I_{e,VND}$ versus $I_{a,VND}$ (solid line) and $I_{a,CND}$ versus $I_{e,CND}$ (dashed line). Note that only $I_{e,VND}$ is a function of the channel condition, since only the variable nodes have access to the channel observation. This means that for every different noise variance σ_n^2, we have a different curve $I_{e,VND}$ versus $I_{a,VND}$ and consequently, a different EXIT chart. The reason to plot the extrinsic transfer function of the check nodes on reversed axis is that it allows us to construct

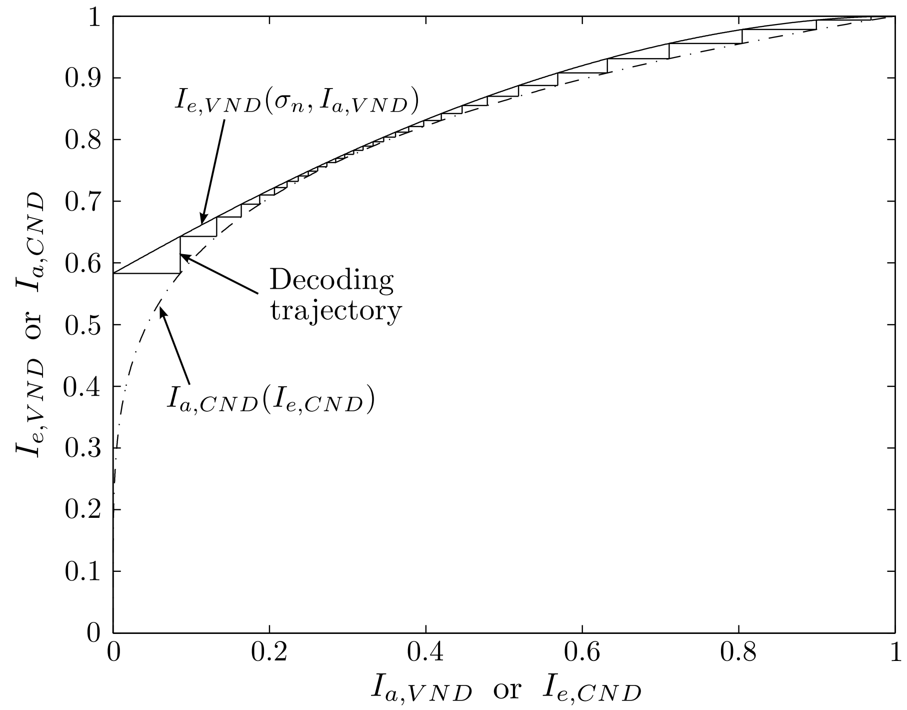

Figure 2.11: EXIT chart for the $(d_v, d_c) = (3, 6)$ regular LDPC code ensemble at $E_b/N_0 =$ 1.25 dB.

the decoding trajectory of the iterative system, since the extrinsic information of one decoder becomes the *a priori* information of the other. The decoding trajectory depicts the amount of information being exchanged between the constituent decoders.

If we increase the signal-to-noise ratio, the upper curve shifts upwards opening the "tunnel" between the two curves and thus speeding up the convergence, since a lower number of iterations will be needed to achieve the point (1,1), which indicates full knowledge of the transmitted bits. Conversely, if the signal-to-noise ratio is lowered, the "tunnel" becomes narrower. If both curves intersect, the decoding trajectory does not go all the way to the point (1,1), what means that the iterative decoder won't converge. As for LDPC codes, it is possible to construct EXIT charts for a vast variety of iterative decoded systems such as serial and parallel concatenated codes. For such systems, we refer the reader to [31] for a very comprehensive and detailed description.

Chapter 3

Asymptotic Analysis of Hybrid Turbo Codes

This chapter describes the relation between the two different kinds of EXIT charts that arise in the analysis of a hybrid concatenated turbo coding scheme used to achieve unequal-error-protecting capabilities. From this analysis, it is shown that both kinds of charts can be used to analyze the iterative decoding procedure of such hybrid concatenated codes. Finally, it is shown that the analysis of the hybrid turbo codes proposed in [32] can be reduced to the study of its component serial concatenated codes.

3.1 Hybrid turbo codes

Turbo codes [4] were originally defined as parallel concatenated codes (PCCs), i.e., a parallel concatenation of two binary convolutional codes with the parallel branches separated by one interleaver of appropriate size, decoded by an iterative decoding algorithm. Later, Benedetto *et al.* [33] introduced a serial concatenation of interleaved codes. Those serially concatenated codes (SCCs) in general exhibit lower error floors than PCCs, but SCCs usually converge further away from channel capacity. A further form of concatenated code, hybrid concatenated codes (HCCs), consists of a combination of parallel and serial concatenation, offering the opportunity to exploit the advantages of parallel and serially concatenated codes. There are several different hybrid concatenated structures proposed in literature, e.g., [34, 35]. Herein, we study the hybrid scheme proposed in [32], which is depicted in Fig. C.1. This kind of concatenation consists of a parallel concatenation of two serially concatenated interleaved codes and arise in the context of turbo coding schemes with unequal-error-protecting properties.

In [36], the authors showed that a pruning procedure can be employed to adapt the rate and distance for different protection levels in UEP turbo codes. Pruning can simply be accomplished by a concatenation of a mother code and a pruning code, which leads to a selection of only some paths in the decoding trellis. In Fig. C.1, the codes G_{11} and G_{21} can be referred to as the pruning codes and G_{12} and G_{22} as the mother codes of such a UEP scheme. As a tool for investigating the iterative decoding behavior of this hybrid concatenation, we make use of ten Brink's EXIT charts [28]. We can however define two different EXIT charts for the studied concatenation. The first one, which we call local EXIT chart, examines the iterative decoding behavior of the serial concatenated codes. The second one, which we call global EXIT chart, deals with the exchange of information between each parallel branch during the decoding procedure. Our objective is to derive the relation between these different charts, showing that the design of hybrid turbo codes can, by means of the local EXIT chart, be reduced to that of serially interleaved concatenated codes.

In the following, all component codes of the hybrid concatenation shown in Fig. C.1 are assumed to be recursive systematic convolutional codes. The interleavers in the upper and lower branch are denoted as Π_1 and Π_2, respectively. Since the output of each parallel branch is systematic, the information bits only have to be transmitted once. The example codes we use in this chapter are given by

$$G_{11} = G_{21} = \begin{pmatrix} 1 & \frac{D^2}{1+D+D^2} \end{pmatrix} \tag{3.1}$$

and

$$G_{12} = G_{22} = \begin{pmatrix} 1 & 0 & \frac{1+D+D^2}{1+D^2} \\ 0 & 1 & \frac{1}{1+D^2} \end{pmatrix} . \tag{3.2}$$

In this example, the outer codes have rates $R_{11} = R_{21} = 1/2$ and the inner codes rates are $R_{12} = R_{22} = 2/3$. The systematic coded bit stream is formed as follows

$$\begin{aligned} \mathbf{c} \;=\; & (c_{1,1}(1)\; c_{1,2}(1)\; c_{1,3}(1)\; c_{2,2}(1)\; c_{2,3}(1) \\ & \; c_{1,1}(2)\; c_{1,2}(2)\; c_{1,3}(2)\; c_{2,2}(2)\; c_{2,3}(2)\; \ldots\;), \end{aligned}$$

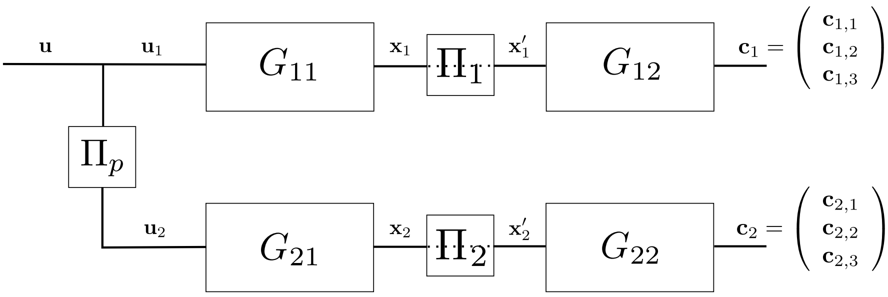

Figure 3.1: Encoder structure of a hybrid turbo code.

where $c_{1,1}(1) = u(1)$, $c_{1,1}(2) = u(2)$ and so on. We assume that only the parity bits of $\mathbf{x}_1$ and $\mathbf{x}_2$ are interleaved in the serial concatenations. This is done in order to keep the overall system systematic. This assumption is represented in Fig. C.1 by the dotted lines dividing the serial interleavers. Note that $c_{2,1}(.)$ is not transmitted, since we do not want to transmit the systematic information twice. Thus, the overall rate of our example code is $R = 1/5$. The decoding of such codes is divided into a local decoding corresponding to each serial branch, and a global decoding where the parallel branches exchange extrinsic information between them. In the following, we explain the local decoding operation and then show how to connect the partial results for each parallel branch to form the global decoding system. As component decoders, we assume *a posteriori* decoders (APP decoders, e.g., BCJR, logMAP) which have two inputs and two outputs in form of log-likelihood ratios (L-values).

3.1.1 Iterative decoding of the parallel concatenated codes

Both decoders of the parallel concatenation receive as first input the channel observation (intrinsic information). As this information can be interpreted as *a priori* information concerning the coded stream, we will call it $L_{\mathrm{a}}(\hat{\mathbf{c}}_j)$ where the indices $j = 1$ and $j = 2$ refer to the upper and lower branch, respectively. The second input represents the *a priori* information concerning the uncoded bit streams denoted by $L_{\mathrm{a}}(\hat{\mathbf{u}}_j)$. The decoder outputs two L-values corresponding to the coded and uncoded bit streams denoted by $L(\hat{\mathbf{c}}_j)$ and $L(\hat{\mathbf{u}}_j)$, respectively. Figure 3.2 shows the corresponding system. For systematic codes, the decoder outputs are composed of the two *a priori* values and some extrinsic information gained by the decoding process. In order to avoid statistical dependencies, the two decoders only exchange the extrinsic L-values corresponding to the uncoded bit stream $L_{\mathrm{e}}(\hat{\mathbf{u}}_j) = L(\hat{\mathbf{u}}_j) - L_{\mathrm{a}}(\hat{\mathbf{u}}_j) - L_{\mathrm{a}}(\hat{\mathbf{c}}_j)$.

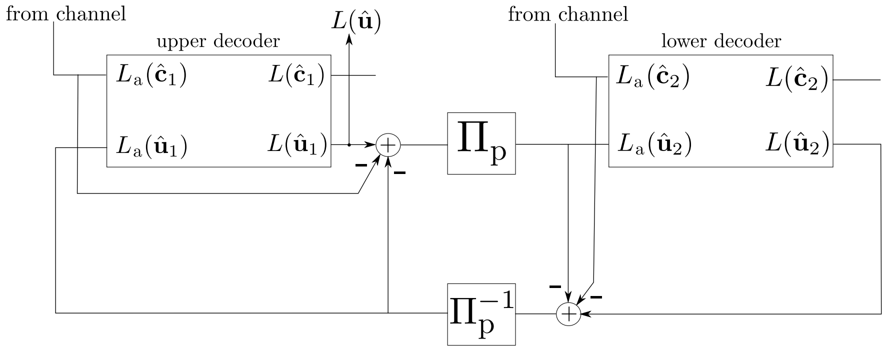

Figure 3.2: Decoder structure of the parallel concatenation present in the hybrid turbo code.

3.1.2 Iterative decoding of the serially concatenated codes

For a serial concatenation with interleaver Π_j, let $\mathbf{u}_j$ and $\mathbf{x}_j$ be the input and output of the outer encoder, and let $\mathbf{x}'_j$ and $\mathbf{c}_j$ be the input and output of the inner encoder, respectively. For each iteration, the inner decoder receives the intrinsic information $L_\mathrm{a}(\hat{\mathbf{c}}_j)$ and the *a priori* knowledge on the inner information bits $L_\mathrm{a}(\hat{\mathbf{x}}'_j)$. Accordingly, the inner decoder outputs two L-values corresponding to the coded and uncoded bit streams denoted by $L(\hat{\mathbf{c}}_j)$ and $L(\hat{\mathbf{x}}'_j)$, respectively. The difference $L(\hat{\mathbf{x}}'_j) - L_\mathrm{a}(\hat{\mathbf{x}}'_j)$, which combines extrinsic and channel information, is then passed through a bit deinterleaver to become the *a priori* input $L_\mathrm{a}(\hat{\mathbf{x}}_j)$ of the outer decoder. The outer decoder feeds back extrinsic information $L_\mathrm{e}(\hat{\mathbf{x}}_j) = L(\hat{\mathbf{x}}_j) - L_\mathrm{a}(\hat{\mathbf{x}}_j)$ which becomes the *a priori* knowledge $L_\mathrm{a}(\hat{\mathbf{x}}'_j)$ for the inner decoder. It is worth noting that the *a priori* information concerning the uncoded input of the outer decoder is zero all the time, since there is no information from this side of the decoder.[1] Furthermore, the outer decoder does not pass information corresponding to the uncoded, but to the coded bits to the inner decoder, since the (interleaved) coded output of the outer encoder corresponds to the uncoded input of the inner encoder. The decoder structure of the upper branch ($j = 1$) is depicted in Fig. 3.3.

[1] This is assumed here because we are dealing solely with the decoding procedure of one serial branch. When dealing with the whole hybrid system, $L_\mathrm{a}(\hat{\mathbf{u}}'_j)$ will vary, since it is the information exchanged between the parallel branches.

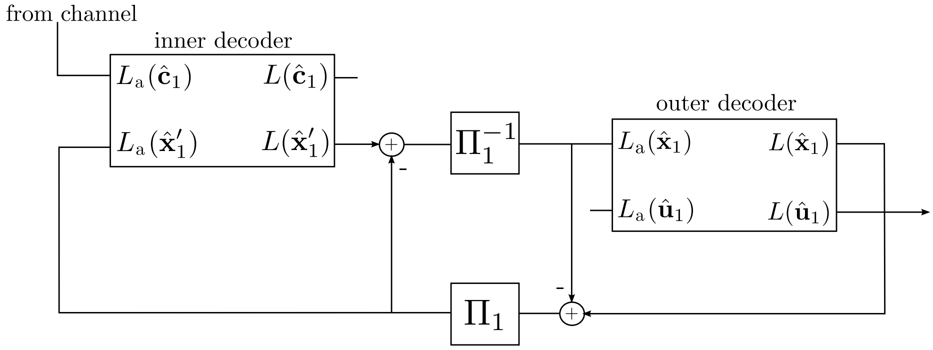

Figure 3.3: Decoder structure of the upper branch for the hybrid turbo code

3.1.3 Hybrid turbo code decoding

Once we know how the serial and parallel decoding is performed, we are able to describe the decoding procedure of the whole system. At first, the channel provides information about the outputs corresponding to the two inner encoders, i.e., $L_a(\hat{\mathbf{c}}_1)$ and $L_a(\hat{\mathbf{c}}_2)$. By now, assume the upper branch to be decoded first. Thus, the upper inner decoder calculates the estimated vector of L-values $L(\hat{\mathbf{x}}'_1)$, subtracts its *a priori* L-values $L_a(\hat{\mathbf{x}}'_1)$, and then passes it to the outer decoder (note that for the first iteration $L_a(\hat{\mathbf{x}}'_1) = 0$). The outer decoder receives $L_a(\hat{\mathbf{x}}_1) = \Pi_1^{-1}(L(\hat{\mathbf{x}}'_1) - L_a(\hat{\mathbf{x}}'_1))$, calculates its estimated L-values $L(\hat{\mathbf{x}}_1)$, and then passes the extrinsic information $L_e(\hat{\mathbf{x}}_1) = L(\hat{\mathbf{x}}_1) - L_a(\hat{\mathbf{x}}_1)$ to the upper inner decoder. This procedure is performed for a certain number of iterations $n_{it,1}$. At the end of these iterations, the upper branch calculates the extrinsic information regarding the information bits $L_e(\hat{\mathbf{u}}_1) = L(\hat{\mathbf{u}}_1) - L_a(\hat{\mathbf{u}}_1) - L_a(\hat{\mathbf{c}}_1)$ and passes it on to the lower branch. Note that $L_a(\hat{\mathbf{u}}_1) = 0$ is zero when this value is calculated for the first time. The decoding of the lower branch starts with the activation of the outer decoder, which receives *a priori* information from the upper branch $L_a(\hat{\mathbf{u}}_2) = \Pi_p(L_e(\hat{\mathbf{u}}_1))$ and from the channel $L_a(\hat{\mathbf{x}}_2)$. Note that, since the inner encoder is systematic, the channel information regarding the output of the outer encoder is the systematic part of the inner encoder output, thus it can be passed on to the outer decoder without activating the inner decoder. The outer decoder computes the values $L(\hat{\mathbf{x}}_2)$ and passes the extrinsic information $L_e(\hat{\mathbf{x}}_2) = L(\hat{\mathbf{x}}_2) - L_a(\hat{\mathbf{x}}_2)$ on to the lower inner decoder. The inner decoder then subtracts its *a priori* values $L_a(\hat{\mathbf{x}}'_2) = \Pi_2(L_e(\hat{\mathbf{x}}_2))$ from its estimated values and forwards it to the outer decoder. The lower branch is decoded with $n_{it,2}$ iterations ending with an activation of the outer decoder, which subtracts the initial channel information $L_a(\hat{\mathbf{c}}_2)$ and the *a priori* values coming from the upper branch $L_a(\hat{\mathbf{u}}_2)$ from

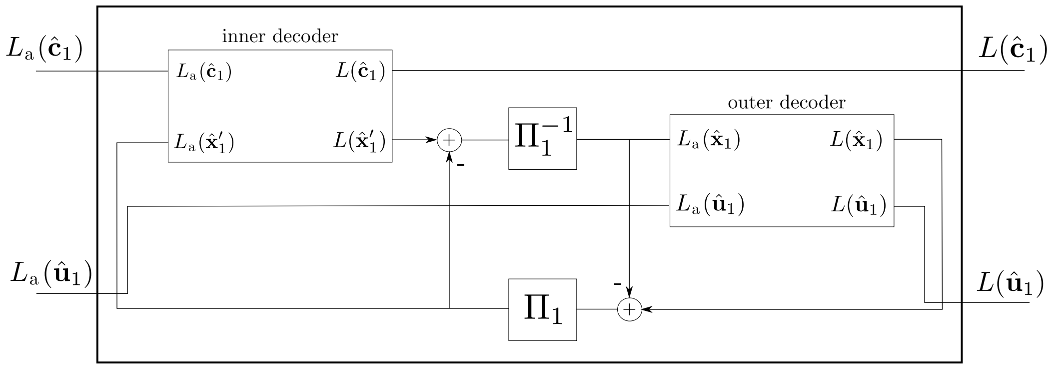

Figure 3.4: Decoder structure of a serial concatenation inside an APP decoder.

its estimation $L(\hat{\mathbf{u}}_2)$. This whole process is executed for $n_{it,g}$ iterations, called global iterations. Each subsequent decoding of a branch is performed as described for the lower branch, since the *a priori* information for each branch will be non-zero, i.e., it is formed by the extrinsic information from the other branch. The iterations within the branches are called local iterations. As we stated before, the hybrid turbo code can be seen as a parallel concatenation of two serially concatenated codes. In that way, the lower and upper decoders of the parallel concatenation can be represented as constituent blocks as shown in Fig. 3.4. Those blocks are then connected as shown in Fig. 3.2.

3.2 Global and local EXIT charts

In order to analyze the iterative decoding procedure, we can construct EXIT charts corresponding to the local as well as to the global iterations. In [32], the authors studied the convergence behavior of hybrid turbo codes by means of global EXIT charts, but the local charts and the relation between the latter and global charts were not explored. In this section, we study the construction of the local charts and review the construction of the global one.

3.2.1 Local EXIT charts

The construction of the local EXIT charts consists in drawing the transfer characteristic of a serial concatenated coding scheme [37]. In the previous section, we mentioned that

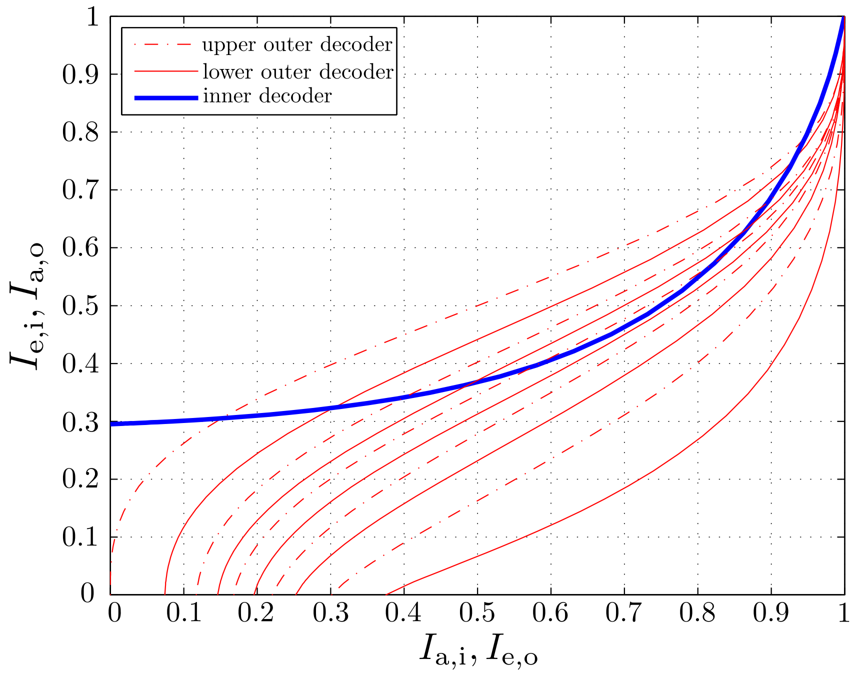

Figure 3.5: Local EXIT chart of the example hybrid turbo code for E_b/N_0= -1.22 dB. For this SNR, it can be noticed from this chart that the system will converge for $n_{it,g} \geq 3$.

for the serially concatenated codes the *a priori* information concerning the uncoded input of the outer decoder is zero during the whole decoding procedure. When we deal with the whole system, this is not valid anymore, since at the end of each local decoding, one parallel branch shall send information concerning the uncoded bits to its adjacent branch. That is, when dealing with the whole hybrid turbo code, we now state that the information concerning the uncoded input of the outer decoder is constant during the local decoding (decoding within each serial branch).

Figure 3.5 illustrates this situation by representing the local EXIT chart of a hybrid turbo code for a total of five global iterations ($n_{it,g} = 5$), where the vertical axis denotes the extrinsic (*a priori*) information of the inner (outer) decoder $I_{\mathrm{e,i}}$ ($I_{\mathrm{a,o}}$) and the horizontal axis denotes the *a priori* (extrinsic) information of the inner (outer) decoder $I_{\mathrm{a,i}}$ ($I_{\mathrm{e,o}}$). The solid and the dashed thin lines represent the transfer characteristic of the upper and lower outer decoder, respectively. The bold line represents the transfer characteristic of the inner decoder which remains unchanged during the decoding procedure. This is due to the fact that the *a priori* information in the beginning of the decoding procedure is solely due to the intrinsic information, which remains constant during the global decoding procedure.

The transfer characteristic of the outer decoder starts at the abscissa zero (first local decoding operation) and is shifted to the right at the beginning of each further local decoding. This is due to the fact that at each local iteration, new information regarding the uncoded bits ($I_{\mathrm{a}}(\hat{\mathbf{u}})$) will be received from the adjacent parallel branch. It is this gain of information that enables us to generate a different transfer characteristic curve for each local decoding operation. From now on, we will refer to this set of information transfer curves of the outer decoder for different $I_{\mathrm{a}}(\hat{\mathbf{u}})$ (together with the transfer curve of the inner decoder) as local EXIT charts.

Each global iteration is represented by a pair of curves for the outer decoders (one dashed line together with one solid line). The convergence of the decoding procedure for the represented SNR can be inferred from Fig. 3.5, since there will be an "open tunnel" between the transfer characteristic of the inner and outer decoders for $n_{it,g} \geq 3$, i.e., the mutual information exchanged between them will converge to one in a limited number of iterations.

3.2.2 Global EXIT charts

For the construction of the global EXIT chart, we consider each serial decoding structure of a branch as one component decoder in a parallel concatenation. The global EXIT chart depicts the mutual information concerning the *a priori* values $L_{\mathrm{a}}(\hat{\mathbf{u}}_j)$ and the extrinsic values $L_{\mathrm{e}}(\hat{\mathbf{u}}_j) = L(\hat{\mathbf{u}}_j) - L_{\mathrm{a}}(\hat{\mathbf{u}}_j) - L_{\mathrm{a}}(\hat{\mathbf{c}}_j)$. The global EXIT chart for $E_b/N_0 = 1$ dB is shown in Fig. 3.6.

Note that due to the different code rates for the global and local systems, the corresponding local EXIT chart is depicted in Fig. 3.5 for $E_b/N_0 = -1.22$ dB, i.e.,

$$\frac{E_b}{N_0}\Big|_{dB,R=1/5} = \frac{E_b}{N_0}\Big|_{dB,R=1/3} + 10\log_{10}\frac{5}{3}\ . \tag{3.3}$$

As expected from the analysis of the corresponding local EXIT chart depicted in Fig. 3.5, the iterative decoding converges.

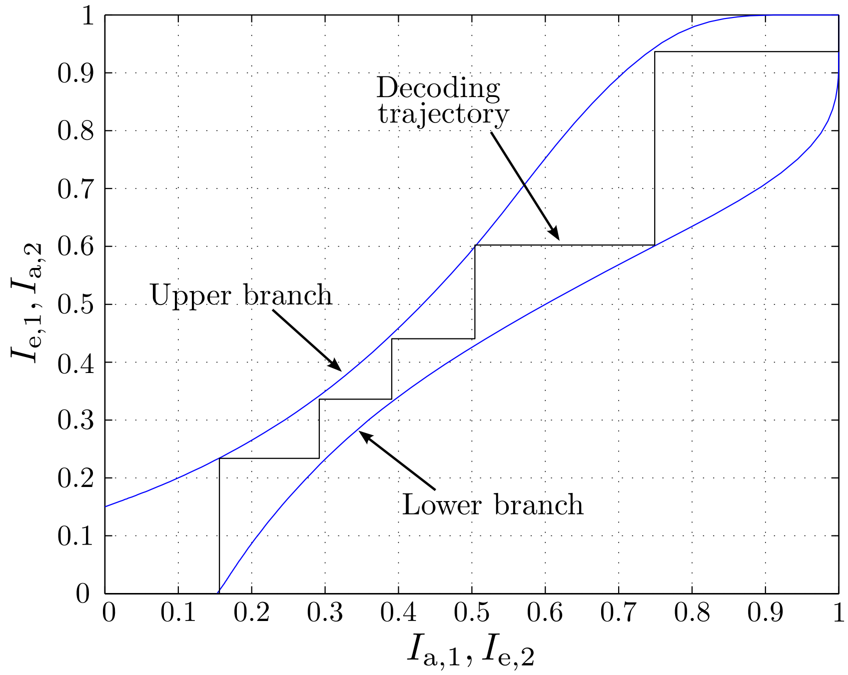

Figure 3.6: Global EXIT chart with decoding trajectory for the example hybrid turbo code with $n_{it,1} = n_{it,2} = 2$ and $E_b/N_0 = 1$ dB (lower decoder activated first).

3.3 Relation between local and global EXIT charts

The analysis of both local and global EXIT charts provides a good insight into the iterative decoding procedure. Since they lead to the same conclusion about the decoder convergence, a mathematical relation between them is to be expected. The derivation of this relation is the subject of the present section.

The local EXIT charts relate the *a priori* and extrinsic information concerning the codewords of the outer decoder, i.e., they show the relation between $I(\mathbf{x}; L_\mathrm{a}(\hat{\mathbf{x}})) = I_{\mathrm{a,o}}(\hat{\mathbf{x}})$ and $I(\mathbf{x}; L_\mathrm{e}(\hat{\mathbf{x}})) = I_{\mathrm{e,o}}(\hat{\mathbf{x}})$. Applying Eq. (2.4) for finitely long sequences, the mutual information between some data sequence $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ and the corresponding L-values $L(\hat{\mathbf{x}})$ can be written as

$$I = I(\mathbf{x}; L(\hat{\mathbf{x}})) = E\{1 - \log_2(1 + e^{-x_v \cdot L(\hat{x}_v)})\} \approx 1 - \frac{1}{n} \sum_{i=1}^{n} \log_2(1 + e^{-x_i \cdot L(\hat{x}_i)}) \,. \quad (3.4)$$

The global EXIT chart, instead, plots the relation between the mutual information of *a priori* and extrinsic values regarding the information bits, i.e., $I(\mathbf{u}; L_\mathrm{a}(\hat{\mathbf{u}})) = I_{\mathrm{a},j}(\hat{\mathbf{u}})$ and $I(\mathbf{u}; L_\mathrm{e}(\hat{\mathbf{u}})) = I_{\mathrm{e},j}(\hat{\mathbf{u}})$ where $j = 1$ (upper branch), 2 (lower branch).

In the local EXIT charts, we should now focus on the points where $I_{\mathrm{a,o}}(\hat{\mathbf{x}}) = 0$, i.e., the points where the *a priori* information regarding the output bits of the outer encoder is zero. In this situation, all the knowledge that the outer decoder has about $\hat{\mathbf{x}}$ comes from the information regarding $\hat{\mathbf{u}}$ provided by the other branch. Since the outer code is systematic, we can write $\mathbf{x} = [\mathbf{u}, \mathbf{p}]$, where $\mathbf{p}$ represent the parity bits resulting from the encoding of $\mathbf{u}$ by the outer encoder. Moreover, the information regarding the parity bits $\mathbf{p}$ is zero in the points where $I_{\mathrm{a,o}}(\hat{\mathbf{x}}) = 0$. Thus, by means of Eq. 3.4, we can write

$$I_{\mathrm{e,o}}(\hat{\mathbf{x}}) = R_\mathrm{o} \cdot I_{\mathrm{e},j}(\hat{\mathbf{u}}) + (1 - R_\mathrm{o}) \cdot \underbrace{I_{\mathrm{e},j}(\hat{\mathbf{p}})}_{=0} = R_\mathrm{o} \cdot I_{\mathrm{e},j}(\hat{\mathbf{u}}) \ , \tag{3.5}$$

where R_o is the rate of the outer code and $j = 1$ or 2 depending whether the upper or lower decoder was activated in the corresponding local decoding, respectively.

Equation (3.5) relates two quantities that are depicted in different EXIT charts, thus it can be used to link both representations. On the one hand, from the local EXIT chart, we will be able to calculate the global decoding trajectory from the points of zero ordinate ($I_{\mathrm{a,o}}(\hat{\mathbf{x}}) = 0$), since at these points, all the knowledge that the outer decoder has about $\hat{\mathbf{x}}$ comes from the information regarding $\hat{\mathbf{u}}$. Then, using Eq. (3.5), we can compute the corresponding $I_{\mathrm{e},j}(\hat{\mathbf{u}})$. On the other hand, by directly evaluating the global EXIT chart where the decoding trajectory and the transfer curve of the active branch meet, one can compute the points of the local EXIT chart where $I_{\mathrm{a,o}}(\hat{\mathbf{x}}) = 0$. This situation is shown in Fig. 3.7 for the local and global EXIT charts depicted in figs. 3.5 and 3.6. Note that since the $R_\mathrm{o} = 0.5$, $I_{\mathrm{e,o}}(\hat{\mathbf{x}}) = 0.5 \cdot I_{\mathrm{e},j}(\hat{\mathbf{u}})$, where $j = 1$ for the upper branch (dashed arrows) and $j = 2$ for the lower one (solid arrows).

Figure 3.8 depicts the local EXIT chart for $E_b/N_0 = -1.77$ dB (or $E_b/N_0 = 0.5$ dB if we refer to the whole system). From this chart, we can observe that the decoding will not converge for this SNR due to the intersection between the transfer curves of the inner and outer decoder. In this example, note that the more local iterations are performed, the closer the transfer curves of the outer decoder lie to each other. This reflects the fact that there is no further gain of information about $\hat{\mathbf{u}}$. This is depicted in the corresponding global EXIT chart of Fig. 3.9 when the transfer curves of each branch intersect. As stated in Eq. (3.5), the point in the local chart to where the transfer curves converge is exactly half of the ordinate of the point where the transfer

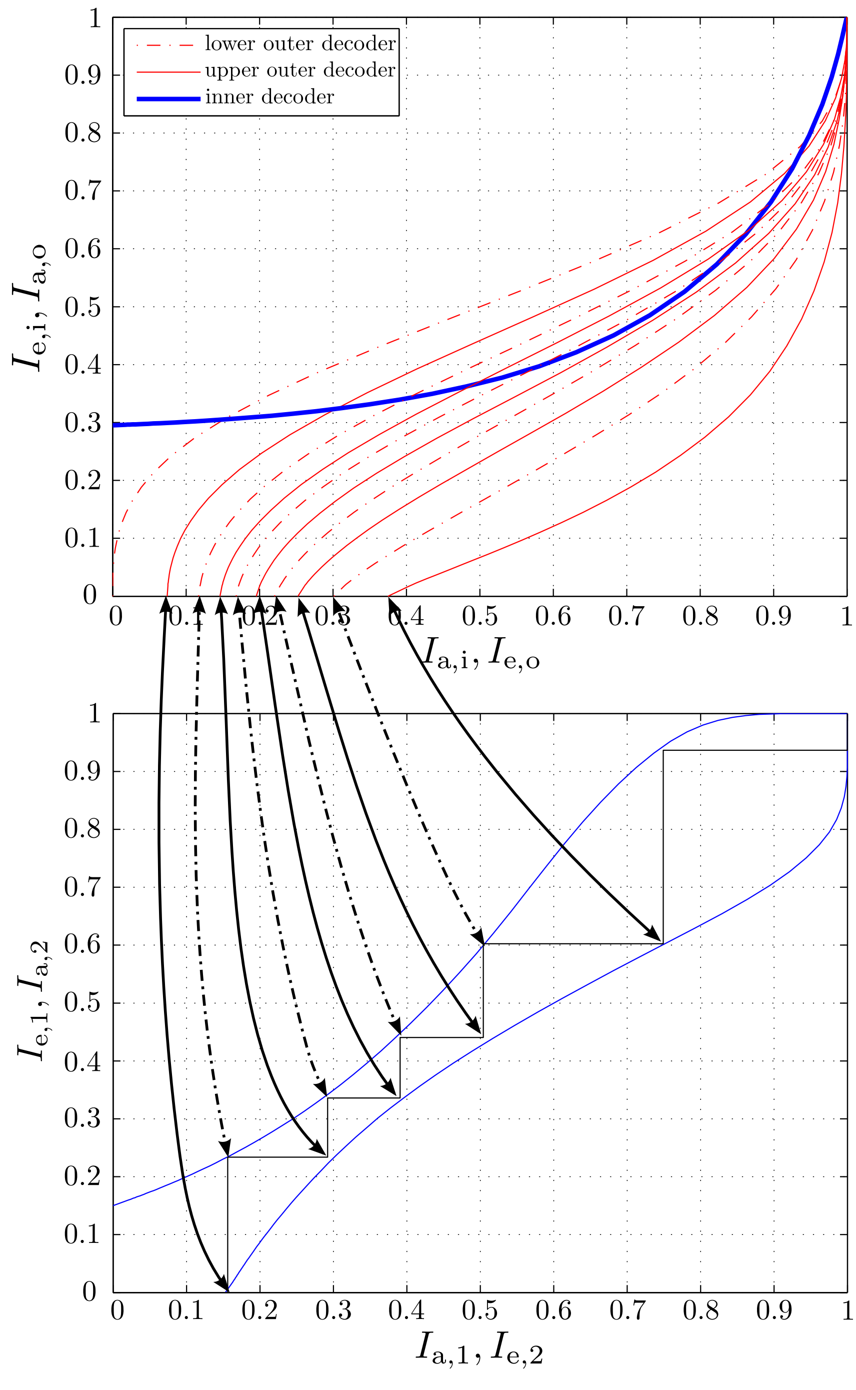

Figure 3.7: Depiction of the relation between the points $I_{a,o}(\hat{\mathbf{x}}) = 0$ in the local EXIT chart and the global decoding trajectory for the example hybrid turbo code. The charts were constructed for $E_b/N_0 = 1$ dB (related to the global system) with the lower decoder being activated first.

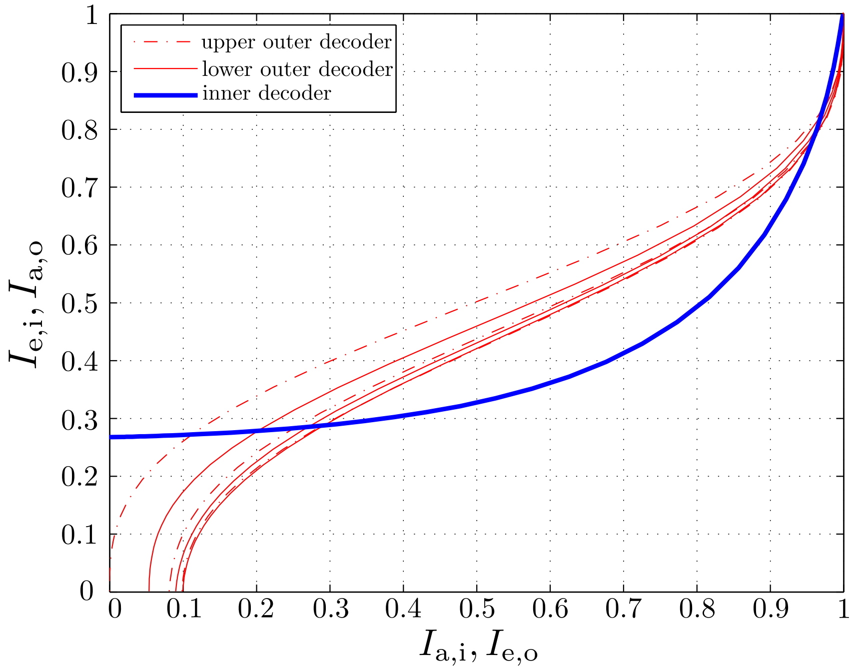

Figure 3.8: Local EXIT chart for the example hybrid turbo code for E_b/N_0=0.5 dB (relating to the whole system). Local decoders do not converge for this SNR.

curves of the global EXIT chart intersect.

3.4 Construction of the local EXIT chart from the transfer characteristic of the inner and outer codes

We still need to show how to construct the local EXIT charts from the transfer characteristic of the inner and outer codes. This reduces the design of good hybrid turbo codes to the well-known design of serially concatenated convolutional codes [37]. In order to construct the whole local EXIT chart, we must be able to compute the *a priori* information regarding the message bits ($I_{a,j}(\hat{\mathbf{u}})$), since for each $I_{a,j}(\hat{\mathbf{u}})$ (that remains constant during each local decoding operation) we will have a different transfer curve for the outer decoder.

The problem can be formulated as follows: given $I_{a,o}(\hat{\mathbf{x}})$, $I_{e,o}(\hat{\mathbf{x}})$, and $I_{a,j}(\hat{\mathbf{u}})$, compute $I_{e,j}(\hat{\mathbf{u}})$ (which will be used as *a priori* information regarding the message bits in the next local decoding). It should be clear that $I_{a,o}(\hat{\mathbf{x}})$ and $I_{e,o}(\hat{\mathbf{x}})$ can be evaluated from

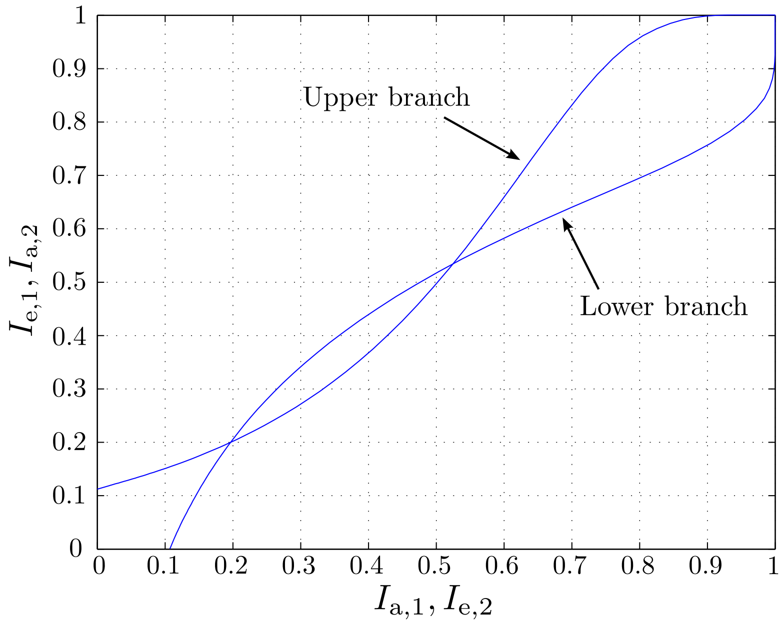

Figure 3.9: Global EXIT chart of a hybrid turbo code for $E_b/N_0 = 0.5$ dB.

the EXIT chart of the serial concatenation for a given $I_{a,j}(\hat{\mathbf{u}})$. Let $I_{a,j}(\hat{\mathbf{u}})^{(it,g)}$ denote the *a priori* mutual information regarding the message bits at the global iteration it, g, where it, g is a non-negative integer. Note that $I_{a,j}(\hat{\mathbf{u}})^{(it,g)}$ is calculated recursively, that is, $I_{a,1}(\hat{\mathbf{u}})^{(it,g)} = I_{e,2}(\hat{\mathbf{u}})^{(it,g)}$ for $it, g \geq 1$, and $I_{a,2}(\hat{\mathbf{u}})^{(it,g)} = I_{e,1}(\hat{\mathbf{u}})^{(it,g-1)}$ for $it, g > 1$, with $I_{a,2}(\hat{\mathbf{u}})^{(1)} = 0$.[2]

The soft output of the outer decoder concerning the information bits can be written as [38]

$$L_e(\hat{\mathbf{u}}) = L(\hat{\mathbf{u}}) - L_a(\hat{\mathbf{u}}) - L_{ch}(\hat{\mathbf{u}}) \ , \tag{3.6}$$

where $L_{ch}(\hat{\mathbf{u}})$ is the intrinsic information concerning the uncoded bits. As indicated by simulations, we assume the involved random variables to have a symmetric Gaussian distribution. Under the Gaussian assumption and assuming independence between the random variables involved (since we are considering independent channel observations), we can write

$$\sigma_e^2 = \sigma_u^2 - \sigma_a^2 - \sigma_{ch}^2 \ , \tag{3.7}$$

[2]Note that we are assuming the lower branch to be decoded first.

where the variances σ_a^2, σ_{ch}^2, and σ_u^2 are calculated inverting the $J(.)$ function, i.e.,

$$\sigma_{\mathrm{a}}^2 \approx J^{-1}(I_{\mathrm{a}}(\hat{\mathbf{u}}))^2, \sigma_{\mathrm{ch}}^2 \approx J^{-1}(I_{\mathrm{ch}}(\hat{\mathbf{u}}))^2, \sigma_u^2 \approx J^{-1}(I(\hat{\mathbf{u}}))^2 \ ,$$

where $I_{\mathrm{ch}}(\hat{\mathbf{u}}) = I(\mathbf{u}; L_{\mathrm{ch}}(\hat{\mathbf{u}}))$ and $I(\hat{\mathbf{u}}) = I(\mathbf{u}; L(\hat{\mathbf{u}}))$. Note that $I_{\mathrm{ch}}(\hat{\mathbf{u}})$ can be inferred from the local EXIT chart from the point where the inner decoder information transfer curve intersects the ordinate axis. $I(\hat{\mathbf{u}})$ can be calculated from the local EXIT chart in the following way.

Since $L(\hat{\mathbf{x}}) = L_{\mathrm{a}}(\hat{\mathbf{x}}) + L_{\mathrm{e}}(\hat{\mathbf{x}})$, and assuming that $L_{\mathrm{a}}(\hat{\mathbf{x}})$ and $L_{\mathrm{e}}(\hat{\mathbf{x}})$ are independent Gaussian distributed variables, we can write

$$\sigma_x^2 = \sigma_{\mathrm{a,o}}^2 + \sigma_{\mathrm{e,o}}^2 \ , \tag{3.8}$$

where $\sigma_{\mathrm{a,o}}^2 \approx J^{-1}(I_{\mathrm{a,o}}(\hat{\mathbf{x}}))^2$ and $\sigma_{\mathrm{e,o}}^2 \approx J^{-1}(I_{\mathrm{e,o}}(\hat{\mathbf{x}}))^2$. Since $I_{\mathrm{a,o}}(\hat{\mathbf{x}})$ and $I_{\mathrm{e,o}}(\hat{\mathbf{x}})$ in the end of each local iteration are known[3], we can compute $I(\mathbf{x}; L(\hat{\mathbf{x}})) = I(\hat{\mathbf{x}}) = J(\sigma_x)$. Finally, note that $I(\hat{\mathbf{x}})$ contains information regarding both parity and information bits. Thus, we can write

$$I(\hat{\mathbf{x}}) = I(\hat{\mathbf{u}}) \cdot R_{\mathrm{o}} + I(\hat{\mathbf{p}}) \cdot (1 - R_{\mathrm{o}}) \ , \tag{3.9}$$

where R_{o} is the rate of the outer code and $I(\hat{\mathbf{p}})$ is the information regarding the parity-check bits. Assuming that the L-values carry approximately the same amount of information for every bit, we can say that $I(\hat{\mathbf{u}}) \approx I(\hat{\mathbf{p}})$ and then

$$I(\hat{\mathbf{u}}) \approx I(\hat{\mathbf{x}}) \ . \tag{3.10}$$

Note that Eq. (3.5) can also be derived from (3.9) by noticing that in the points of the local EXIT chart where $I_{\mathrm{a,o}}(\hat{\mathbf{x}}) = 0$, the information about the parity bits equals zero, i.e., $I(\hat{\mathbf{p}}) = 0$. By means of eqs. (3.7), (2.14), (3.8), and (3.10), we can compute the extrinsic information regarding the message bits $I_{\mathrm{e}}(\hat{\mathbf{u}})$ and then compute the transfer curve of the outer decoder when $I_{\mathrm{a}}(\hat{\mathbf{u}}) \neq 0$ thus deriving the complete local EXIT chart.

With the local EXIT chart and Eq. (3.5), we are able to predict the convergence behavior of the global system without the need of constructing the whole global EXIT chart. That is, the convergence of the system may be predicted locally by analyzing the local EXIT charts. This reduces the analysis of the global system to the study of

[3]We assume that enough iterations are are performed to make the decoding trajectory reach the intersection point between the transfer curves of the inner and the outer decoder.

a serial concatenated code, since the convergence behavior of the global system can be predicted from the local EXIT chart.

Chapter 4

Multi-Edge-Type Unequal-Error-Protecting LDPC Codes

Herein, a multi-edge-type analysis of LDPC codes is described. This analysis leads to the development of an algorithm to optimize the connection profile between the different protection classes defined within a codeword. The developed optimization algorithm allows the construction of unequal-error-protecting low-density parity-check (UEP LDPC) codes where the difference between the error rate performance of the protection classes can be adjusted. Concomitantly, it enables the construction of LDPC codes with UEP capabilities that do not vanish as the number of decoding iterations grows.

4.1 Unequal-error-protecting LDPC codes

When the performance of an LDPC code is considered, it is widely noticed that, at least for a limited number of decoding iterations, the connection degree of a variable node affects the error rate of the symbol it represents, i.e., a higher connection degree lowers the probability of an erroneous decoding of a variable node. This observation led to the investigation of irregular LDPC codes for applications where unequal error protection is desired [39–41], since these codes inherently provide different levels of protection within a codeword due to the different connection degrees of its variable nodes. Other strategies to generate UEP LDPC codes include adapting its check node degree distribution as done in [42], and using an algebraic method based on the Plotkin construction developed in [43]. In the present chapter, we consider only UEP LDPC codes designed by means of the optimization of its variable node degree distribution while the check node degree distribution is fixed.

The UEP LDPC codes considered herein were introduced by Poulliat et al. in [40]. The idea behind the development of these codes is based on dividing a codeword into different protection classes and defining local variable degree distributions, i.e., each protection class is described by a polynomial $\lambda^{(j)}(x) = \sum_{i=2}^{d_{v_{max}}^{(k)}} \lambda_i^{(j)} x^{i-1}$, where $\lambda_i^{(j)}$ represents the fraction of edges connected to degree i variable nodes within the protection class C_j. Based on the observation that the error rate of a given protection class depends on the average connection degree and on the minimum degree of its variable nodes, the authors propose an optimization algorithm where the cost function is the maximization of the average variable node degree subject to a minimum variable node degree $d_{v_{min}}^{(j)}$. A detailed description of the hierarchical optimization algorithm applied in the derivation of the UEP LDPC codes considered in this chapter can be found in [40].

Furthermore, the authors in [40] interpret the unequal-error-protecting properties of an LDPC code as different local convergence speeds, i.e., the most protected class is the one that converges with the smallest number of decoding iterations to its right value. This assumption is made in order to cope with the observation that, even though irregular LDPC codes present UEP capabilities for a low number of message-passing iterations regardless of the construction algorithm used, this capability vanishes for some LDPC constructions as the number of iterations grow. This phenomenon was also observed in [43], where the authors argue that no difference between the performance of the protection classes can be detected after 50 iterations. On the other hand, the results presented in [41] show significant UEP capabilities even after 200 decoding iterations.

This discrepancy was studied in [44], where it is pointed out that the connection degree among the variable nodes belonging to different protection classes is a determining property for the preservation of the UEP capabilities of an LDPC code when the number of message passing iterations for decoding grows. More specifically, the authors show that LDPC codes defined by the same pair of degree distributions present different UEP capabilities for a moderate to large number of decoding iterations when constructed by means of distinct computer-based design algorithms. For example, codes constructed with the random [16] and ACE [45] algorithms preserve the UEP capability independently of the number of decoding iterations, while this same capability vanishes as the number of decoding iterations grows for codes constructed by means of the PEG [46] or the PEG-ACE [47] algorithms.

These observations motivated us to investigate the application of a multi-edge-type analysis for UEP LDPC codes, since it allows to distinguish the messages exchanged among the different protection classes during the iterative decoding. This ability to distinguish messages according to its originating protection class provides us with the means for controlling the connectivity among the different classes. This enables not only the construction of LDPC codes with non-vanishing UEP capabilities for a moderate to large number of iterations, but also to control the difference in the error-rate performance among the protection classes. Before proceeding to the actual multi-edge-type analysis, we introduce some notation that will be useful in the forthcoming description.

4.1.1 System model and notation

The transmission of information over an AWGN channel with noise variance σ_n^2 using BPSK signaling is assumed. The unequal-error-protecting LDPC codes considered herein are binary, systematic, rate $R = k/n$, irregular LDPC codes with variable and check nodes degree distributions defined by $\lambda(x) = \sum_{i=2}^{d_{v_{max}}} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=2}^{d_{c_{max}}} \rho_i x^{i-1}$, where $d_{v_{max}}$ and $d_{c_{max}}$ are the maximum variable and check node degrees of the code, respectively. The bits within a codeword are divided into N_c disjoint protection classes $(C_1, C_2, \ldots, C_{N_c})$ with decreasing levels of protection. Furthermore, we consider that all the $n-k$ redundant bits are associated with the less protected class C_{N_c} and that the vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_{N_c-1}\}$ represents the fraction of information bits associated with the first $N_c - 1$ protection classes.

4.2 Multi-edge-type unequal-error-protecting LDPC codes

Unequal-error-protecting LDPC codes can be included in a multi-edge framework in a straightforward way. This can be done by distinguishing between the edges connected to the different protection classes defined within a codeword. According to this strategy, the edges connected to variable nodes within a protection class are considered to be all of the same type. For example, consider the factor graph of Fig. C.2 where the arrows represent the received channel information and the variable nodes are divided into 2 disjoint protection classes C_1 and C_2 represented by the gray and white variable nodes, respectively. A multi-edge-type description arises by letting the edges connected to the variable node of class C_1 and C_2 be defined as type-1 (depicted by solid lines) and type-2 (depicted by the dashed lines) edges, respectively.

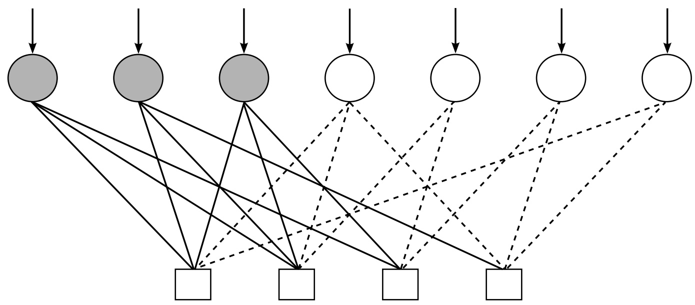

Figure 4.1: Multi-edge-type factor graph of a code with 2 protection classes.

Considering that each variable node has access to only one channel observation and that there are no punctured bits (i.e., $\mathbf{b} = (0, 1)$ for all variable nodes), the variable and check node multinomials for this example are given by

$$\nu(\mathbf{r}, \mathbf{x}) = \frac{3}{7} r_1 x_1^3 + \frac{1}{7} r_1 x_2^3 + \frac{3}{7} r_1 x_2^2, \qquad \mu(\mathbf{x}) = \frac{2}{7} x_1^3 x_2^2 + \frac{1}{7} x_1^2 x_2^2 + \frac{1}{7} x_1 x_2^3 \,.$$

It is worth noting that as opposed to the variable nodes, the check nodes admit connections with edges of different types simultaneously as can be inferred from Fig. C.2. In the following, we will divide the variable nodes into m_e protection classes $(C_1, C_2, \ldots, C_{m_e})$ with decreasing levels of protection, i.e., $m_e = N_c$.

4.2.1 Edge-perspective notation

The connection between the protection classes occurs through the check nodes, since they are the only nodes that can have different types of edges attached to them. Consider irregular LDPC codes with node-perspective variable and check node multi-edge multinomials $\nu(\mathbf{r}, \mathbf{x}) = \sum \nu_{\mathbf{b},\mathbf{d}} \mathbf{r}^{\mathbf{b}} \mathbf{x}^{\mathbf{d}}$ and $\mu(\mathbf{x}) = \sum \mu_{\mathbf{d}} \mathbf{x}^{\mathbf{d}}$, respectively. In the following, we consider unpunctured codes and that the variable nodes have access to only one observation. Furthermore, variable nodes within the protection class C_j are only connected to edges of type j.

In order to optimize the amount of connection between the protection classes, it will be more convenient to work with the edge, instead of the node perspective. We now define the following edge-perspective multi-edge multinomials

$$\lambda^{(j)}(\mathbf{r}, \mathbf{x}) = \frac{\nu_{x_j}(\mathbf{r}, \mathbf{x})}{\nu_{x_j}(\mathbf{1}, \mathbf{1})} = \sum_{d_j} \lambda^{(j)}_{d_j} r_1 x_j^{d_j - 1} = r_1 \sum_{i=1}^{d_{vmax}} \lambda^{(j)}_i x_j^{i-1}, \tag{4.1}$$

$$\rho^{(j)}(\mathbf{x}) = \frac{\mu_{x_j}(\mathbf{r}, \mathbf{x})}{\mu_{x_j}(\mathbf{1}, \mathbf{1})} = \sum_{\mathbf{d}} \rho^{(j)}_{\mathbf{d}} \mathbf{x}^{\mathbf{d}'} x_j^{d_j - 1}, \tag{4.2}$$

where the rightmost term of Eq. (4.1) is obtained letting $d_j = i$. Furthermore, $\lambda^{(j)}_i$ denote the fraction of type j edges connected to variable nodes of degree i, $\rho^{(j)}_{\mathbf{d}}$ denote the fraction of type j edges connected to check nodes with edge degree vector $\mathbf{d}$, $\mathbf{x}^{\mathbf{d}'} = \prod_{i=1}^{m_e} x_i^{d_i}$ with $d_j = 0$, and $\mathbf{1}$ denotes a vector with all entries equal to 1 with length being clear from the context. In the next section, we will use eqs. (4.1) and (4.2) in the derivation of the optimization algorithm for the connection profile among the protection classes of an UEP LDPC code.

4.2.2 Asymptotic analysis

Our main objective is, given a UEP LDPC code with overall variable ($\lambda(x)$) and check node ($\rho(x)$) degree distributions, to optimize the connection profiles between the different protection classes in order to control the amount of protection of each class. A second goal is to be able to construct UEP LDPC codes with non-vanishing UEP capabilities when a moderate to large number of decoding iterations is used. The algorithm we derive here can be applied for any irregular pair of degree distributions. However, in order to reduce the search space of the optimization, we suppose from now on that the LDPC code to be optimized is check-regular, i.e., all the check nodes have the same degree d_c.

Despite having the same degree, each check node may have a different number of edges belonging to each of the m_e classes. To understand this, consider for example a check node with an associated edge degree vector $\mathbf{d} = (d_1, d_2, \ldots, d_{m_e})$, where d_j is the number of connections to the protection class i and $\sum_{j=1}^{m_e} d_j = d_c$. If we then consider a code with $m_e = 3$ protection classes, each check node is connected to d_1 edges of class 1, d_2 edges of class 2, and d_3 edges of class 3, where d_1, d_2, and d_3 are not necessarily equal. This posed, we can compute the evolution of the iterative decoding by means of density evolution. We assume standard belief propagation decoding of LDPC codes where the messages exchanged between the variable and check nodes are independent log-likelihood ratios having a symmetric Gaussian distribution (variance equals twice the mean).

Let $I_{v,l}^{(j)}$ ($I_{c,l}^{(j)}$) denote the mutual information between the messages sent through type-j edges at the output of variable (check) nodes at iteration l and the associated variable node value. We can write $I_{v,l}^{(j)}$ and $I_{c,l}^{(j)}$ as

$$I_{v,l}^{(j)} = \sum_{i=2}^{d_{v_{max}}} \lambda_i^{(j)} J\left(\sqrt{\sigma_{ch}^2 + (i-1)[J^{-1}(I_{c,l-1}^{(j)})]^2}\right) , \tag{4.3}$$

$$I_{c,l}^{(j)} = 1 - \sum_{i=1}^{d_{c_{max}}} \sum_{\mathbf{d}:d_j=i} \rho_{\mathbf{d}}^{(j)} J\left(\sqrt{(d_j-1)[J^{-1}(1-I_{v,l}^{(j)})]^2 + \sum_{s\neq j} d_s[J^{-1}(1-I_{v,l}^{(s)})]^2}\right) . \tag{4.4}$$

Equation (4.3) can be derived from Eq. (2.18) replacing λ_i by $\lambda_i^{(j)}$ and $I_{c,l-1}$ by $I_{c,l-1}^{(j)}$, since we are considering each protection class individually, and variable nodes in class j only receive messages from check nodes through edges of type j. Similarly, Eq. (4.4) can be derived from Eq. (2.19) with the addition of an extra term (rightmost sum) in order to consider the messages arriving from protection classes others than j. Furthermore, the coefficients ρ_i in Eq. (2.19) are replaced by the sum $\sum_{\mathbf{d}:d_j=i} \rho_{\mathbf{d}}^{(j)}$ in order to take into account all possible edge degree vectors $\mathbf{d}$ with $d_j = i$.

As done in Section 2.4.2 for irregular LDPC codes, we can combine eqs. (4.3) and (4.4) summarizing the density evolution as a function of the mutual information at the previous iteration, the mutual information contribution from the other classes, noise variance, and degree distributions, i.e.,

$$I_{c,l}^{(j)} = F(\boldsymbol{\lambda}, \boldsymbol{\rho}_{\mathbf{d}}^{(j)}, \sigma_n^2, I_{c,l-1}^{(j)}, \boldsymbol{I}_{c,l-1}) , \tag{4.5}$$

where the bold symbols represent sequences of values defined as $\boldsymbol{\lambda} = \{\{\lambda_i^{(j)}\}_{i=2}^{d_{v_{max}}}\}_{j=1}^{m_e}$,

$\boldsymbol{\rho}_{\mathbf{d}}^{(j)} = \{\rho_{\mathbf{d}:d_j=i}^{(j)}\}_{i=1}^{d_{c_{max}}}$, and $\boldsymbol{I}_{c,l-1} = \{I_{c,l-1}^{(s)}\}_{s=1}^{m_e}$ with $s \neq j$. By means of Eq. (4.5), we can predict the convergence behavior of the iterative decoding and then optimize the degree distribution $\rho^{(j)}(\mathbf{x})$ under the constraint that the mutual information must increase with the number of iterations, i.e.,

$$F(\boldsymbol{\lambda}, \boldsymbol{\rho}_{\mathbf{d}}^{(j)}, \sigma_n^2, I^{(j)}, \boldsymbol{I}) > I^{(j)} \ . \tag{4.6}$$

4.2.3 Optimization algorithm

In [43] and [44], it was pointed out that the UEP capabilities of a code depend on the amount of connection among the protection classes, i.e., if the most protected class is well connected to a less protected one, the performance of the former will decrease while the performance of the latter will be improved. For example, let us assume a code with 2 protection classes and $d_c = 4$. The possible values for the check nodes' edge degree vectors $\mathbf{d} = (d_1, d_2)$ are (0,4), (1,3), (2,2), (3,1), and (4,0). On the one hand, if a code has a majority of check nodes with $\mathbf{d} = (4, 0)$, the first protection class will be very isolated from the second one, which will lead to an enhanced performance difference between the two classes. On the other hand, if a large amount of the check nodes are of type $\mathbf{d} = (2, 2)$, the protection classes will be very connected, which favors the overall performance but mitigates the UEP capability of a code at a moderate to large number of decoding iterations. This example suggests that to control the UEP capability of an LDPC code and to prevent this characteristic from vanishing as the number of decoding iterations grows, it is necessary to control the distribution of the check nodes' edge degree vectors, i.e., optimize $\rho^{(j)}(\mathbf{x})$.

These observations about the influence of the connection between the protection classes on the UEP characteristics of a code can be further analyzed by means of a detailed computation of the mutual information, which may be performed by considering the edge-based mutual information messages traversing the graph instead of node-based averages. Such an analysis and its results will be presented in Section 4.4. In this section, we introduce an algorithm developed to optimize the connection profile between the various protection classes present in a given UEP LDPC code, i.e., $\rho^{(j)}(\mathbf{x})$. Initially, the algorithm computes the variable node degree distribution of each class $\lambda^{(j)}(\mathbf{r}, \mathbf{x})$ based on the node-perspective overall variable node degree distribution $\tilde{\lambda}(x)$ from the given UEP LDPC code and the number of nodes in each protection class. The way this computation is done will be outlined in the following example.

Example 1 *Consider a length-100, rate-1/2, UEP LDPC code with* $m_e = 3$ *protection classes, variable node degree distribution from a node perspective given by* $\tilde{\lambda}(x) = 0.05x^{10} + 0.2x^7 + 0.2x^5 + 0.55x^2$*, and* $\boldsymbol{\alpha} = (0.2, 0.8)$*. The first protection class will then be formed by the first 10 variable nodes with higher degree, and the second class by the remaining 40 systematic variable nodes. Following this strategy, the first class contains 5 degree-10 and 5 degree-7 variable nodes. The second protection class is formed by 15 degree-7, 20 degree-5, and 5 degree-2 variable nodes. Finally, the third protection class is composed of the redundant bits and has only degree-2 variable nodes. This results in the following multi-edge degree distribution*

$$\nu(\boldsymbol{r}, \boldsymbol{x}) = 0.05 r_1 x_1^{10} + 0.05 r_1 x_1^7 + 0.15 r_1 x_2^7 + 0.2 r_1 x_2^5 + 0.05 r_1 x_2^2 + 0.5 r_1 x_3^2 \ .$$

Applying Eq. (4.1), we obtain

$$\lambda^{(1)}(\boldsymbol{r}, \boldsymbol{x}) = 0.58824 r_1 x_1^9 + 0.41176 r_1 x_1^6 \ ,$$

$$\lambda^{(2)}(\boldsymbol{r}, \boldsymbol{x}) = 0.48837 r_1 x_2^6 + 0.46512 r_1 x_2^4 + 0.04651 r_1 x_2 \ ,$$

$$\lambda^{(3)}(\boldsymbol{r}, \boldsymbol{x}) = r_1 x_3 \ .$$

◇

Once the degree distributions $\lambda^{(j)}(\mathbf{r}, \mathbf{x})$ for $j = 1, \ldots, m_e$ are known, the algorithm proceeds sequentially optimizing the distributions $\rho^{(j)}(\mathbf{x})$ for $j = 1, \ldots, m_e$, proceeding from the least protected class to the most protected one. This scheduling is done to control the amount of messages from the less protected classes that are forwarded to the more protected ones, i.e., the check nodes should have the minimum number of connections to the least protected classes in order to avoid that unreliable messages are forwarded to better protected ones.

Since we are using linear programming with a single objective function, we chose it to be the minimization of the average check node degree within the class being optimized, i.e., it minimizes the average number of edges of such a class connected to the check nodes. This minimization aims at diminishing the amount of unreliable messages (i.e., the ones coming up from the less protected variable nodes) that flows through a check node.

In addition, in order to control the amount of connection among the protection classes, we introduce the set of vectors $\boldsymbol{\delta}^{(j)} = (\delta_1^j, \ldots, \delta_j^j)$ for $j = 2, \ldots, m_e$ defined as follows. Given a check node with edge degree vector $\mathbf{d} = (d_1, \ldots, d_{m_e})$, for $1 \leq i \leq j$ each coordinate δ_i^j defines the maximum allowed d_i when $d_j > 0$. For example, assume an UEP LDPC code with $m_e = 3$ protection classes, $\boldsymbol{\delta}^{(3)} = (2, 3, 3)$, and $\boldsymbol{\delta}^{(2)} = (2, 4)$. This implies that a check node with connections to the protection class C_3 (i.e., $d_3 > 0$) can be connected to a maximum of $\delta_1^3 = 2$ edges of type 1, $\delta_2^3 = 3$ edges of type 2, and $\delta_3^3 = 3$ edges of type 3. In turn, each check node with connections to C_2 but no connections to C_3 can be connected to a maximum of $\delta_1^2 = 2$ edges of type 1 and $\delta_1^2 = 4$ edges of type 2. Roughly speaking, each vector $\boldsymbol{\delta}^{(j)}$ adjusts the degree of connection between C_j and the more protected protection classes $C_{j'}$, $j' < j$ while setting the maximum allowed d_j. We will refer to the vectors $\boldsymbol{\delta}^{(j)}$ as *interclass connection vectors.*[1]

Given σ_n^2 and $\boldsymbol{\delta}^{(j)}$ for $j = 2, \ldots, m_e$, the check node profile optimization algorithm for a given UEP LDPC code with parameters $\tilde{\lambda}(x)$, n, R, $\boldsymbol{\alpha}$, and d_c, can be written as stated in Algorithm 2. The optimization is successful, when a solution $\rho^{(j)}(\mathbf{x})$ is found which converges for the given σ_n^2 and $\boldsymbol{\delta}^{(j)}$. Note that the optimization can be solved by

Algorithm 2 Check node profile optimization algorithm

For $j = m_e$ to 1

1. Compute $\lambda^{(j)}(x)$

2. Minimize the average check node degree $\sum_{s=1}^{d_c} s \cdot \sum_{\mathbf{d}:d_j=s} \rho_{\mathbf{d}}^{(j)}$ under the following constraints,

$$\mathcal{C}_1 : \sum_{s=1}^{d_c} \sum_{\mathbf{d}:d_j=s} \rho_{\mathbf{d}}^{(j)} = 1 \ ,$$

$$\mathcal{C}_2 : d_i \leq \delta_i^j, \ \forall \, i = 1, \ldots, j \ \text{ and } \mathbf{d} : d_j > 0 \ ,$$

$$\mathcal{C}_3 : F(\boldsymbol{\lambda}, \boldsymbol{\rho}_{\mathbf{d}}^{(j)}, \sigma_n^2, I, \boldsymbol{I}) > I \ , \quad \forall \, I \in [0, 1) \ ,$$

$$\mathcal{C}_4 : \forall \, \mathbf{d} : 1 \leq d_{j'} \leq d_c \text{ and } j' > j \ , \quad \rho_{\mathbf{d}}^{(j)} \text{ is fixed} \ .$$

end

[1]Note that in the optimization algorithm proposed herein, it makes no sense to define a vector $\boldsymbol{\delta}^{(1)}$, since the distribution $\rho^{(1)}(\mathbf{x})$ is completely determined by the optimization of the other protection classes.

linear programming since the cost function and the constraints $(\mathcal{C}_1)$ and $(\mathcal{C}_3)$ are linear in the parameters $\boldsymbol{\rho}_{\mathbf{d}}^{(j)}$. The constraints $(\mathcal{C}_2)$ and $(\mathcal{C}_4)$ are the interclass connection and the previous optimization constraints, respectively. Once we have optimized the check-node profile, the code can be realized through the construction of a parity-check matrix following the desired profile.

4.3 Simulation results

In this section, simulation results for multi-edge-type UEP LDPC codes with optimized check node connection profile are presented. We designed UEP LDPC codes of length $n = 4096$, $m_e = 3$ protection classes, rate 1/2, and $d_{v_{max}} = 30$ following the algorithm of [40]. The proportions of the classes are chosen such that C_1 contains 20 % of the information bits and C_2 contains 80 %. The third protection class C_3 contains all parity bits. Therefore, we are mainly interested in the performances of classes C_1 and C_2.

The variable and check node degree distribution for the designed UEP LDPC code are given by $\lambda(x) = 0.2130x + 0.0927x^2 + 0.2511x^3 + 0.2521x^{17} + 0.0965x^{18} + 0.0946x^{29}$ and $\rho(x) = x^8$, respectively. All parity-check matrices were realized using protograph-based constructions [48]. In order to show the role of the interclass connection vector, we optimized the multi-edge check node degree distribution of the above described UEP LDPC code for different interclass connection vectors. This resulted in four codes (referred to as codes I, II, III, and IV) with different sets of $\boldsymbol{\delta}^{(j)}$ according to Table 4.1. The multi-edge distributions from an edge perspective for the optimized codes are shown in Tables 4.2 and 4.3.

4.3.1 Low number of iterations

In this subsection, we describe simulation results for a total of 7 decoding iterations aiming at systems with computational complexity and decoding time constraints (as most of the practical decoding schemes).

Table 4.1: Interclass connection vectors for four optimized multi-edge-type UEP LDPC codes.

	I	II	III	IV
$\boldsymbol{\delta}^{(3)}$	(2,2,7)	(2,2,7)	(2,4,7)	(0,2,7)
$\boldsymbol{\delta}^{(2)}$	(6,5)	(4,5)	(6,5)	(6,5)

Table 4.2: Local variable degree distributions. The coefficients $\lambda_i^{(j)}$ represent the fraction of edges connected to variables nodes of degree i within the class C_j.

C_1	C_2	C_3
$\lambda_4^{(1)} = 0.00197$	$\lambda_3^{(2)} = 0.23982$	$\lambda_2^{(3)} = 0.93901$
$\lambda_{18}^{(1)} = 0.57263$	$\lambda_4^{(2)} = 0.76018$	$\lambda_3^{(3)} = 0.06099$
$\lambda_{19}^{(1)} = 0.21085$		
$\lambda_{30}^{(1)} = 0.21455$		

Figure 4.2 shows the performance of codes I and II. The difference between these codes results from the interclass connection vector of C_2. Since Code II has a lower δ_1^2, the classes C_1 and C_2 are more isolated from each other. This can be concluded from Table 4.3 by the presence of check nodes with edge degree vector $\mathbf{d} = (9, 0, 0)$ in Code II, which indicates check nodes connected only to type-1 edges. Due to its higher isolation, it is expected for Code II that C_1 has a better performance while the error rate of C_2 is worsened as can be observed from Fig. 4.2.

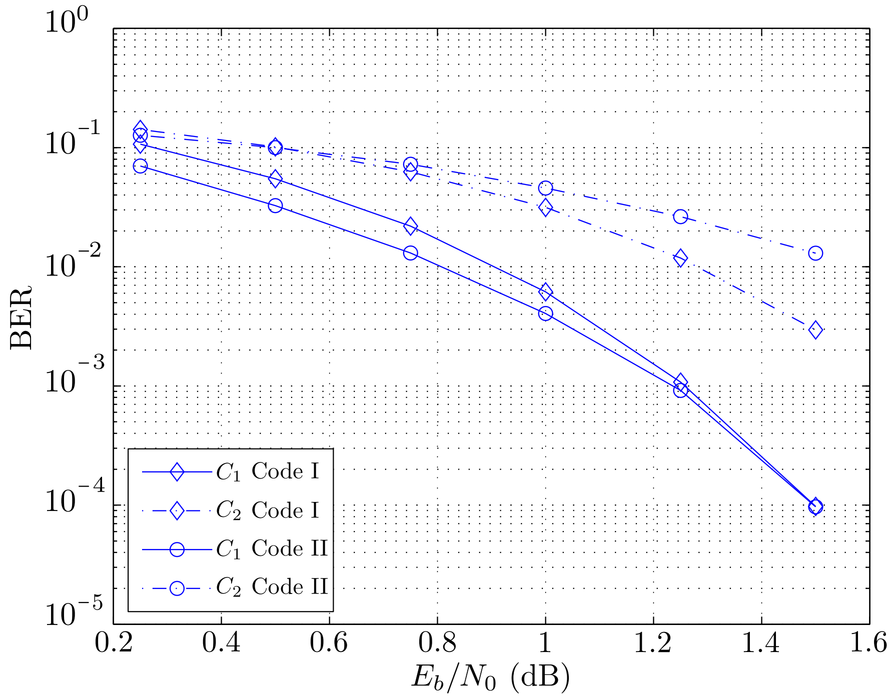

Figure 4.2: Error performance of codes I and II for 7 decoding iterations.

Table 4.3: Multi-edge check node degree distributions from an edge perspective. The coefficients $\rho_{\mathbf{d}}^{(j)}$ represent the fraction of edges of class j connected to check nodes of type $\mathbf{d}$.

	I			II			III			IV		
$\mathbf{d}$	$\rho_{\mathbf{d}}^{(1)}$	$\rho_{\mathbf{d}}^{(2)}$	$\rho_{\mathbf{d}}^{(3)}$	$\rho_{\mathbf{d}}^{(1)}$	$\rho_{\mathbf{d}}^{(2)}$	$\rho_{\mathbf{d}}^{(3)}$	$\rho_{\mathbf{d}}^{(1)}$	$\rho_{\mathbf{d}}^{(2)}$	$\rho_{\mathbf{d}}^{(3)}$	$\rho_{\mathbf{d}}^{(1)}$	$\rho_{\mathbf{d}}^{(2)}$	$\rho_{\mathbf{d}}^{(3)}$
(0,2,7)	0	0	0	0	0	0	0	0	0	0	0.19681	1
(2,2,5)	0.20589	0.27553	1	0.20589	0.27553	1	0	0	0	0	0	0
(2,4,3)	0	0	0	0	0	0	0.34315	0.91844	1	0	0	0
(3,6,0)	0	0	0	0	0	0	0	0	0	0	0	0
(4,5,0)	0.099360	0.16621	0	0.43308	0.72447	0	0	0	0	0.035770	0.059840	0
(5,4,0)	0.23258	0.24900	0	0	0	0	0.12189	0.081560	0	0.24449	0.26175	0
(6,3,0)	0.46217	0.30926	0	0	0	0	0	0	0	0.71974	0.48160	0
(9,0,0)	0	0	0	0.36103	0	0	0.53496	0	0	0	0	0

Note that varying δ_1^2 does not change the degree of connection of C_1 and C_2 to C_3. This can be inferred from Table 4.3 which shows that the fraction of edges connected to check nodes of type $\mathbf{d} = (2, 2, 5)$ remains constant for both codes.

Let us now compare the performances of codes I and III. Since Code III has an interclass connection vector $\boldsymbol{\delta}^{(3)} = (2, 4, 7)$ while Code I has $\boldsymbol{\delta}^{(3)} = (2, 2, 7)$, the variable nodes within protection class C_2 will be more connected to the least protected bits of C_3 in the former code. In fact, from Table 4.3, we see that for Code III about 92 % of the type 2 edges are connected to check nodes of type $\mathbf{d} = (2, 4, 3)$ while only 27 % type-2 edges have connections to check nodes connected to C_3 on Code I. This worsens the performance of C_2 for Code III as depicted in Fig. 4.3.

Note however that regardless of its higher isolation, C_1 in Code III does not have a lower error rate than in Code I for high signal-to-noise ratios. This is explained by the fact that C_1 on Code I profits from its higher connection degree to C_2 while in Code III, C_1 is very isolated from the other protection classes and also does not profit much from the connections to C_2 due to the poor performance of the latter.

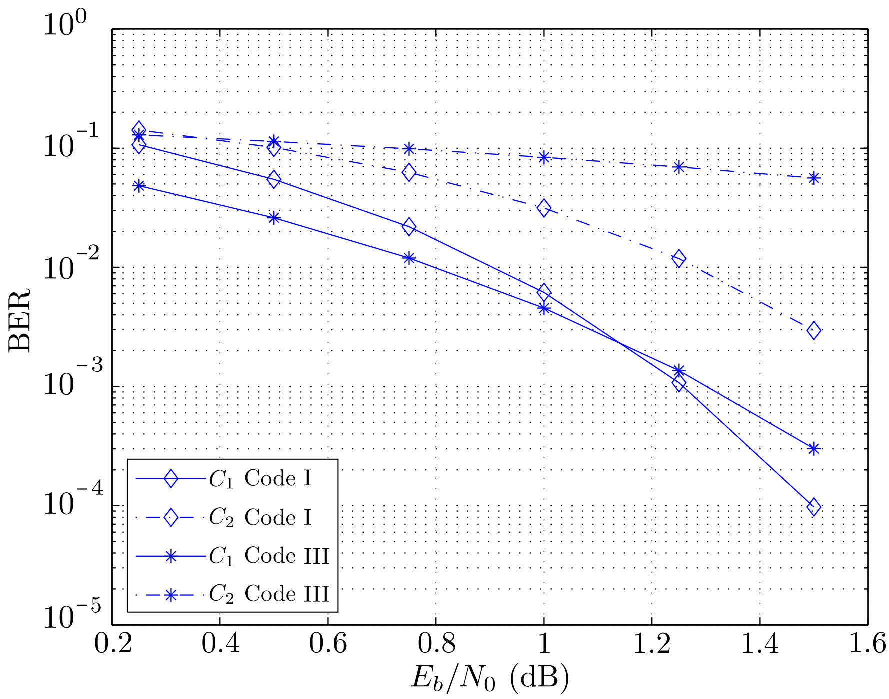

Figure 4.3: Error performance of codes I and III for 7 decoding iterations.

These observations can be inferred from Fig. 4.3. This indicates that there is a limit where the isolation of a protection class starts to be counterproductive to its performance.

For the next simulation, we set $\delta^{(3)} = (0, 2, 7)$ and $\delta^{(2)} = (6, 5)$. This gives rise to Code IV where the protection class C_1 is not connected to any check node that has connections to the least protected class C_3. At the same time, C_1 and C_2 have a high connectivity between themselves. This has two expected effects. First, since C_1 and C_2 are well connected, there is not such a huge difference between their performances as in Code III. Second, as both C_1 and C_2 have the lowest connection degree to C_3 among all designed codes, their performances are expected to be enhanced in comparison to codes I, II, and III. Those observations are confirmed by the simulations depicted in Fig. 4.4. Furthermore, the isolation from C_3 is indicated by the multi-edge check node degree distribution from an edge perspective of Code IV (Table 4.3).

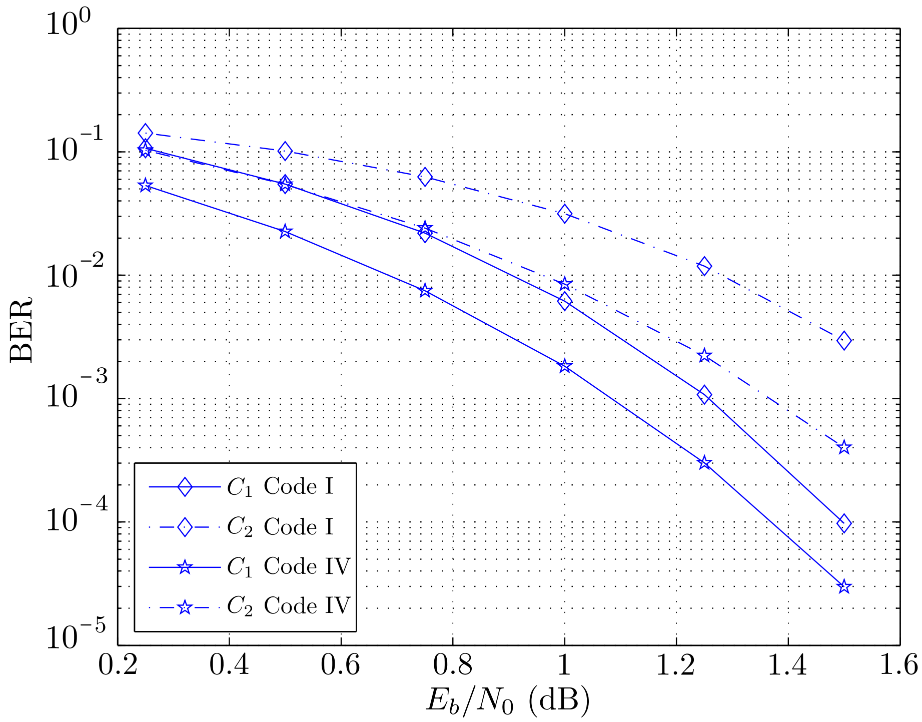

Figure 4.4: Error performance of codes I and IV for 7 decoding iterations.

Finally, Fig. 4.5 compares the performances of Code IV and a code constructed by means of ACE [45] without any optimization of the interclass connection degree. We chose ACE as computer-based construction algorithm due to the fact that it is shown in [44] that it generates LDPC codes with good UEP capabilities. Note that our multi-edge UEP LDPC code shows better performances for both protection classes, the considered signal-to-noise ratio range, and number of decoding iterations. For the most protected class C_1, our scheme has a coding gain of 0.25 dB for a BER $= 3 \cdot 10^{-4}$, while for the protection class C_2, Code IV exhibits a gain of more than 0.75 dB for BER's smaller than $= 2 \cdot 10^{-2}$ when compared with the ACE code.

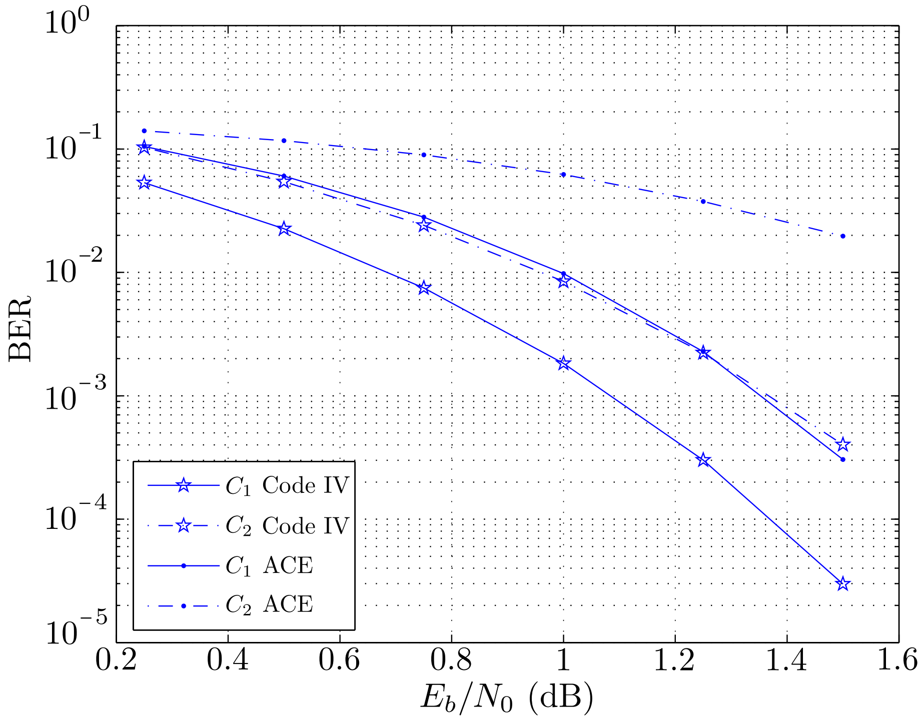

Figure 4.5: Error performance of Code IV and of a code constructed by means of ACE without optimization of the interclass connection degree for a total of 7 decoding iterations.

4.3.2 High number of iterations

Herein, we present the results of simulations performed for a high number of decoding iterations. We show that by means of our developed multi-edge check node degree distribution optimization algorithm, it is possible to construct unequal error protecting LDPC codes with non-vanishing UEP capabilities for a moderate to large number of decoding iterations. All the bit-error curves depicted in this subsection were obtained for a total of 50 decoding iterations.

We divide the codes of Table 4.1 into 2 sets. The set composed by codes II and III has non-vanishing UEP capabilities for a moderate to large number of decoding iterations. In the second set, Code I shows UEP capabilities for high signal-to-noise ratios and code IV does not show any significant difference between the performance of the protection classes for a high number of decoding iterations. Figures 4.6 and 4.7 show the simulation results for both sets for a total of 50 decoding iterations. A close analysis of such results together with the distributions of Table 4.3 leads to the conclusion that the isolation between the protection classes is in fact a key parameter to be observed if a non-vanishing UEP capability is desired for a moderate to large number of decoding iterations.

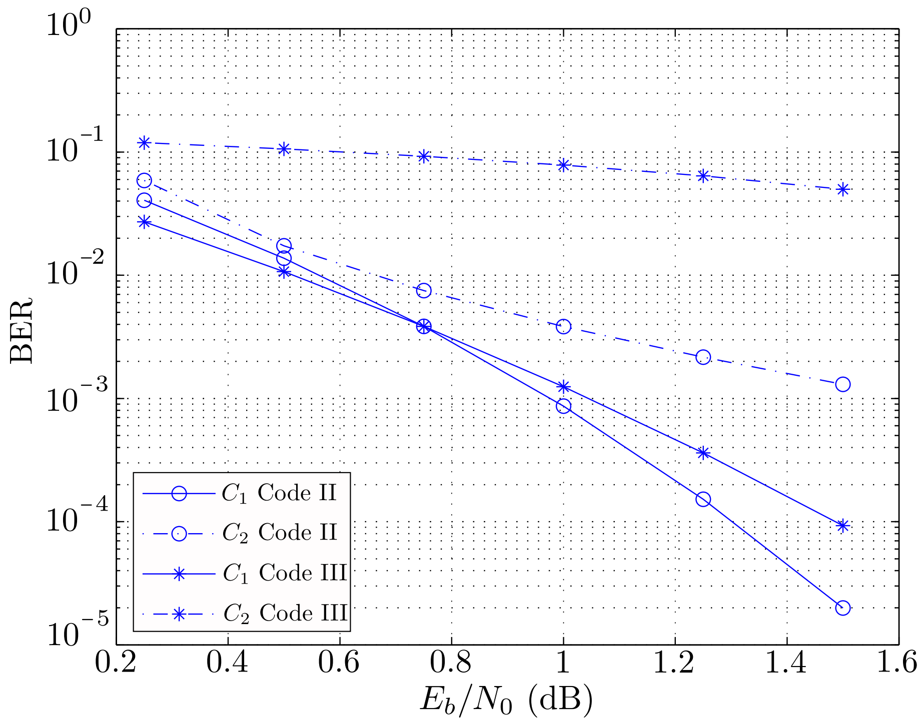

Figure 4.6: Error performance of codes II and III for 50 decoding iterations.

Note that while C_1 and C_2 have a large interconnection degree for codes I and IV (there is no check node with connections solely to one of the protection classes), codes II and III have check nodes only connected to C_1 variable nodes (check nodes with edge degree vector $\mathbf{d}$=(9,0,0)). On the one hand, to isolate the systematic protection classes enhances the difference between their performances. On the other hand, the higher the performance difference, the more penalized the less protected systematic class will be.

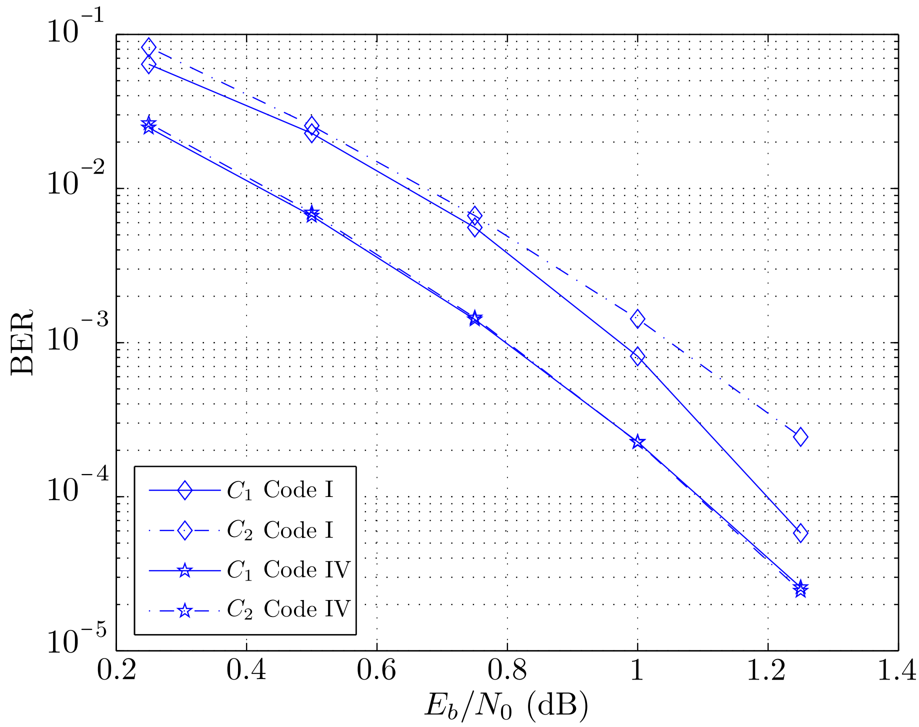

Figure 4.7: Error performance of codes I and IV for 50 decoding iterations.

The simulations of figs. 4.6 and 4.7 indicate that if an UEP LDPC code is desired for applications where a moderate to large number of decoding iterations will be used, a high isolation degree between the systematic protection classes is not a desired feature, since it penalizes too much the performance of both protection classes, i.e., there is a compromise between class isolation and average performance.

In order to show that our optimization can also generate good performance UEP LDPC codes with non-vanishing UEP capabilities for a large number of decoding iterations, we optimized a 3-class code with interclass connection vectors $\boldsymbol{\delta}^{(3)} = (4, 4, 4)$ and $\boldsymbol{\delta}^{(2)} = (4, 5)$ referred to as Code V. For this code, the least protected class is evenly connected to C_1 and C_2. Nevertheless, since Code IV has $\boldsymbol{\delta}^{(2)} = (6, 5)$, and Code V has $\boldsymbol{\delta}^{(2)} = (4, 5)$, the protection classes C_1 and C_2 are more connected in the former code. As a consequence, Code V has UEP capabilities for a high number of decoding iterations while Code IV has not. This can be concluded from figs. 4.7 and 4.8. Figure 4.8 shows the bit-error rate curve of Code V together with the performance of the ACE code already described in the previous section. While presenting a comparable performance for low SNR's, code V has a better performance than the ACE code for $E_b/N_0 > 1$ dB.

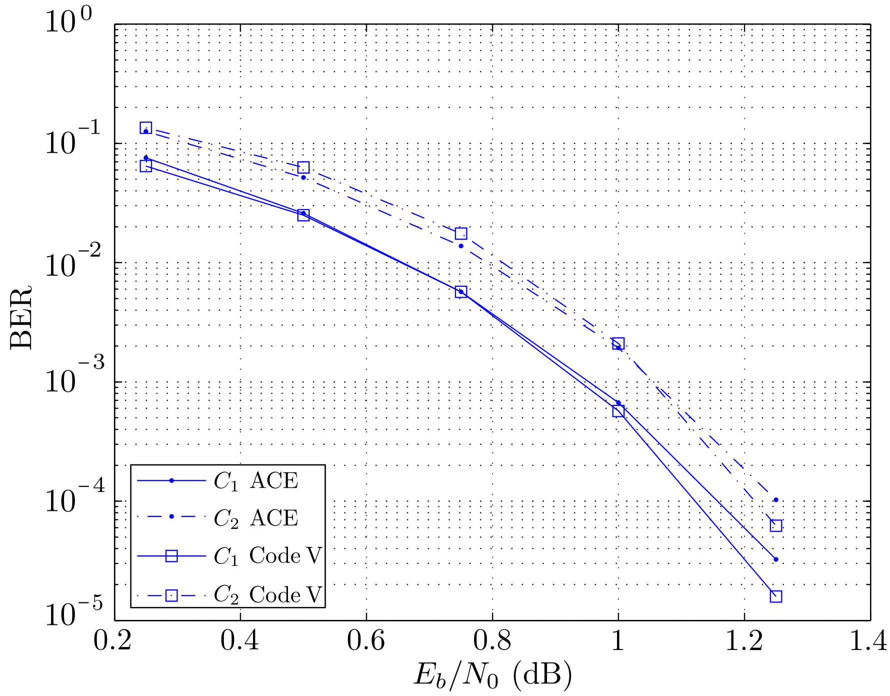

Figure 4.8: Comparison between the error performance of Code V and a code constructed by means of ACE without optimization of the interclass connection degree. Simulation done with a total of 50 decoding iterations.

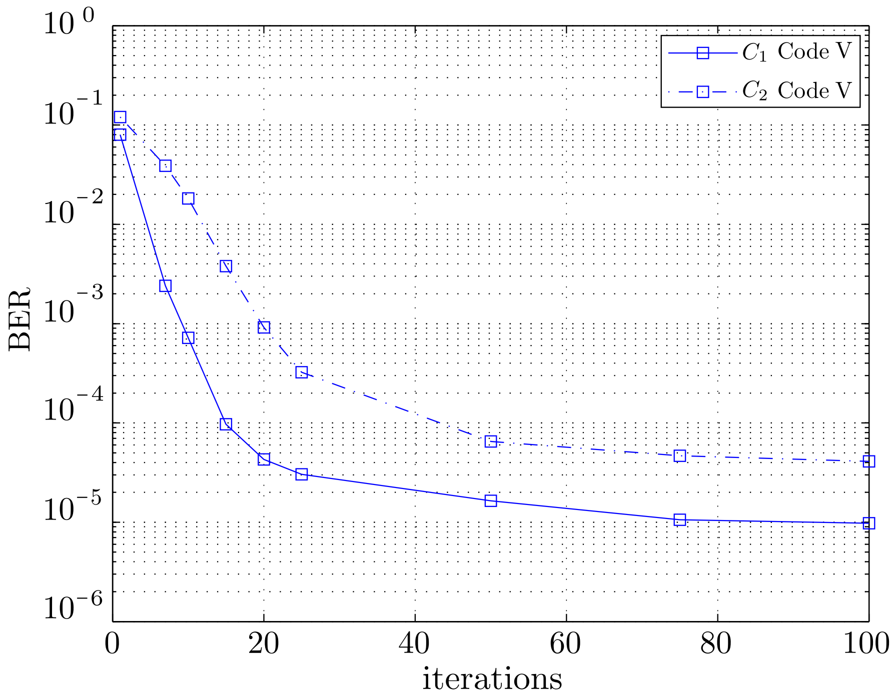

Figure 4.9: BER as a function of the number of decoder iterations for the multi-edge UEP LDPC code at $E_b/N_0 = 1.25$ dB.

Lastly, Fig. 4.9 shows the bit-error rate of Code V as a function of the number of decoding iterations at a signal-to-noise ratio of 1.25 dB. Note the resilience of the UEP capabilities as the number of iterations grows. Similarly to codes I, II, III, and IV, we constructed Code V using a protograph-based construction, since as opposed to PEG and ACE algorithms, it can generate LDPC codes with a very irregular check node degree distribution.

In summary, for applications with a low number of decoding iterations, it is desirable to keep the most protected classes as isolated as possible from the least protected protection class. However, when a high number of decoding iterations is to be applied, there is a significant performance improvement when the protection class composed of the parity bits is evenly connected with the protection classes formed by the systematic bits. Notwithstanding, for both cases, the higher the isolation between the systematic protection classes, the higher will be the difference between their performances.

4.4 Detailed mutual information evolution

The analysis of the unequal-error-protecting capabilities of the optimized codes proposed in this chapter can be done by means of mutual information (MI) evolution. However, as pointed out in [49], the MI analysis as developed in Section 2.1.2 for regular and irregular LDPC codes generally cannot be applied to the study of multi-edge-type LDPC codes. The reasoning behind this fact is twofold. First, eqs. (2.18) and (2.19) consider an average MI computed for the whole ensemble defined by the degree distributions pair $\lambda(x)$ and $\rho(x)$. This means that the different UEP capabilities of the codes studied here could not be detected, since they share the same overall degree distributions. Second, the analysis based solely on $\lambda(x)$ and $\rho(x)$ considers the convergence behavior of the codewords to its right value as a whole, not allowing to investigate the convergence of the protection classes separately. To overcome such limitations, [50] proposes a detailed MI evolution analysis based on the results of [49].

Before describing the detailed mutual information evolution, recall that during the iterative decoding of LDPC codes, variable and check nodes act as serially concatenated decoders exchanging extrinsic information. The extrinsic mutual information between the output of a variable node and its corresponding codeword bit ($I_{e,VND}$) becomes *a priori* information for its neighboring check nodes ($I_{a,CND}$). Analogously, the extrinsic mutual information between the output of a check node and its corresponding codeword bit ($I_{e,CND}$) becomes *a priori* information for its neighboring variable nodes ($I_{a,VND}$). Using Eq. (2.13), we can write the extrinsic mutual information between the sth output message of a degree-d_v variable node and its corresponding codeword bit as [49]

$$I_{e,VND|s} = J\left(\sqrt{\sum_{r=1,r\neq s}^{d_v} [J^{-1}(I_{a,VND|r})]^2 + [J^{-1}(I_{ch})]^2}\right) , \tag{4.7}$$

where $I_{a,VND|r}$ is the *a priori* mutual information of the message received by the variable node through the rth edge and $I_{ch} = J(\sigma_{ch})$. For degree-d_c check nodes, we can use Eq. (2.17) and approximate its extrinsic mutual information by

$$I_{e,CND|s} = 1 - J\left(\sqrt{\sum_{r=1,r\neq s}^{d_c} [J^{-1}(1 - I_{a,CND|r})]^2}\right) , \tag{4.8}$$

where $I_{a,CND|r}$ is the *a priori* mutual information of the message received on its rth edge.

In the following, we present the extrinsic information transfer analysis described in [50] and [44]. The algorithm aims at computing the *a posteriori* MI of each variable node (instead of node-based averages) at the end of each decoder iteration. This allows to evaluate the decoding convergence of each protection class. The algorithm was proposed originally for the analysis of LDPC codes designed with protographs. However, we will apply it substituting the protograph base matrix by the parity-check matrix of our LDPC codes. We will denote each element at location (i, j) of the parity-check matrix by $h_{i,j}$. Furthermore, $I_{e,VND}(i, j)$, $I_{e,CND}(i, j)$ denote the extrinsic mutual information between the message sent by the variable node V_j to the check node C_i, and from C_i to V_j respectively, and the associated codeword bit. The detailed MI is summarized in Algorithm 3. Note that the difference between Algorithm 3 and the standard MI evolution for LDPC codes is that the former omits the averaging through the degree distributions. Thus, it is possible that codes belonging to the same ensemble have a different detailed MI evolution. This is an essential feature when investigating UEP capabilities of a given LDPC code realization.

By tracking the mean *a posteriori* MI (I_{appv}) of each protection class, we can study the UEP capabilities of a given LDPC code, i.e., it can be investigated if distinct protection classes have different error protection capabilities for given channel conditions and number of decoding iterations. For example, in Fig. 4.10, we depict the difference

Algorithm 3 Detailed mutual information evolution

1. **Initialization**
   Compute the channel information $I_{ch} = J(\sigma_{ch})$ with
$$\sigma_{ch}^2 = \frac{4}{\sigma_n^2} \ .$$

2. **Variable-to-check update**
   (a) For $i = 1, \ldots, n-k$ and $j = 1, \ldots, n$, if $h_{i,j} = 1$, calculate
$$I_{e,VND}(i,j) = J\left(\sqrt{\sum_{s\in\mathcal{M}(i), s\neq i} [J^{-1}(I_{a,VND}(s,j))]^2 + [J^{-1}(I_{ch})^2]}\right) \ ,$$
   where $\mathcal{M}(i)$ is the set of check node incident to variable node v_i.
   (b) If $h_{i,j} = 0$, $I_{e,VND}(i,j) = 0$.
   (c) For $i = 1, \ldots, n-k$ and $j = 1, \ldots, n$, set $I_{a,CND}(i,j) = I_{e,VND}(i,j)$.

3. **Check-to-variable update**
   (a) For $i = 1, \ldots, n-k$ and $j = 1, \ldots, n$, if $h_{i,j} = 1$, calculate
$$I_{e,CND}(i,j) = 1 - J\left(\sqrt{\sum_{s\in\mathcal{N}(j), s\neq j} [J^{-1}(1 - I_{a,CND}(i,s))]^2}\right) \ ,$$
   where $\mathcal{N}(j)$ is the set of variable nodes incident to check node c_j.
   (b) If $h_{i,j} = 0$, $I_{e,CND}(i,j) = 0$.
   (c) For $i = 1, \ldots, n-k$ and $j = 1, \ldots, n$, set $I_{a,VND}(i,j) = I_{e,CND}(i,j)$.

4. ***A posteriori* mutual information evaluation**
   For $j = 1, \ldots, n-k$, calculate
$$I_{appv}(j) = J\left(\sqrt{\sum_{s\in\mathcal{M}(i)} [J^{-1}(I_{a,VND}(s,j))]^2 + [J^{-1}(I_{ch})]^2}\right) \ .$$

5. **Repeat steps 1 to 4 until a maximum desired number of iterations is reached**.

between the mean for the variable nodes *a posteriori* MI of a certain protection class and its maximum achievable value for each protection class of codes I and IV as a function of the number of decoding iterations at $E_b/N_0 = 1$ dB. As done in [44] and [50], this difference is depicted, since for values near to 1, small differences in the MI can lead to significant differences in the error rate performance [22].

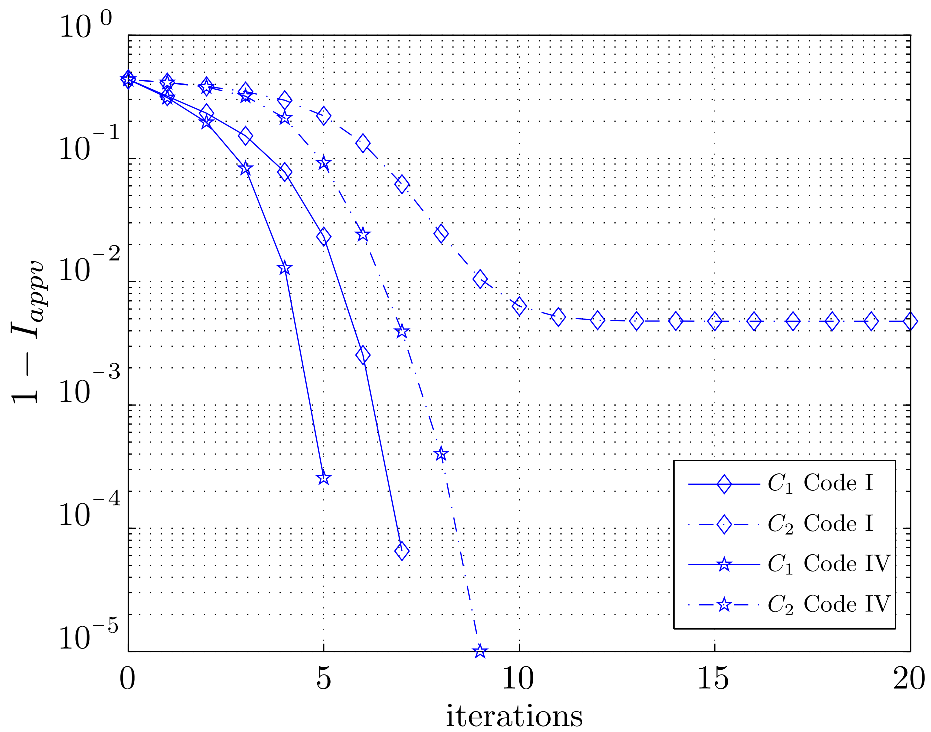

Figure 4.10: Distance of the variable node *a posteriori* MI to the maximum MI as a function of the number of decoder iterations at $E_b/N_0 = 1$ dB.

For the protection class C_1 of codes I and IV, there is no detectable gap to the optimum mutual information $(1 - I_{appv})$ for more than 7 and 5 decoder iterations, respectively. The same occurs to the protection class C_2 of Code IV when more than 9 iterations are considered. This indicates that for Code IV, the BER tends asymptotically to zero for both protection classes after a moderate number of decoding iterations and thus, there will be no UEP at this SNR ($E_b/N_0 = 1$ dB). However, $(1 - I_{appv})$ of the protection class C_2 of code I has a constant non-zero gap to 1 for a number of decoding iterations greater than 10, which indicates that this code has UEP capabilities for a moderate to large number of iterations at this SNR. These conclusions are supported by the simulations depicted in Fig. 4.7. It is worth noting that the results in Fig. 4.10

show the convergence behavior of the protection classes for a specific SNR. The fact that class C_2 of Code I does not converge for $E_b/N_0 = 1$ dB shows that, for this SNR, C_1 and C_2 show different convergence behaviors and thus will have different protection levels, i.e., UEP. It should not be concluded from Fig. 4.10 that Code I has a bad overall performance for a high number of iterations (see Fig. 4.7).

Since the MI analysis assumes cycle-free codes and a Gaussian approximation of the messages exchanged between the variable and check nodes, its results are just approximations. Nevertheless, as shown by our simulations, they provide good predictions regarding the UEP capabilities of a code.

Chapter 5

Multi-Edge-Type Unequal-Error-Protecting LT Codes

In the present chapter, a multi-edge framework for unequal-error-protecting LT codes is derived by distinguishing between the edges connected to each protection class on the factor graph induced by the encoding. As UEP LDPC codes, unequal-error-protecting LT codes are interesting coding schemes for systems where the source bits being transmitted have different sensitivities to errors.

The development of unequal-error-protecting rateless codes was first presented by Rahnavard *et al.* in [51], where the authors propose the partitioning of the block to be transmitted into protection classes with symbols on distinct classes having different probabilities of being chosen during the LT encoding. Another UEP scheme was presented in [52], where the authors achieve UEP properties by means of a windowing technique. Herein, we show that these two schemes can be interpreted as particular cases of multi-edge-type unequal-error-protecting LT codes, which provides us with a common framework for comparison. Furthermore, we propose a third construction algorithm for UEP LT codes which compares favorably to the existing techniques examined herein.

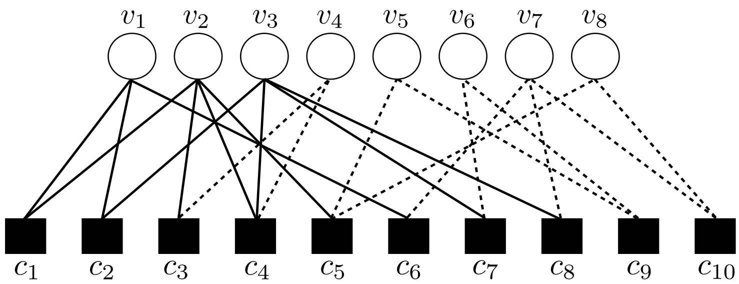

Figure 5.1: Multi-edge graph with two different edge types for an LT code with $k = 8$ and $n = 10$.

5.1 Multi-edge-type unequal-error-protecting LT codes

A multi-edge-framework for unequal-error-protecting LT codes can be developed in a similar way as discussed for LDPC codes. The edges connected to symbols of a protection class in the bipartite graph induced by the encoding are considered to be all of the same type.

For example, in Fig. C.3, we divided the input symbols (variable nodes) into two classes. The first three variable nodes belong to the first class. Consequently, all the edges connected to those symbols are defined as type-1 edges (depicted by solid lines). Additionally, the last five variable nodes form another protection class, whose connected edges are defined as type-2 edges (depicted by dashed lines). Note that as for UEP LDPC codes, only the check nodes (output symbols) admit connections to edges of different types simultaneously. The multi-edge degree distributions for the code depicted in Fig. C.3 are given by

$$\nu(\mathbf{x}) = \frac{1}{8}x_1^3 + \frac{2}{8}x_1^4 + \frac{4}{8}x_2^2 + \frac{1}{8}x_2^3 \; , \tag{5.1}$$

$$\mu(\mathbf{x}) = \frac{2}{8}x_1^2 + \frac{4}{8}x_1 x_2 + \frac{1}{8}x_1^2 x_2 + \frac{1}{8}x_1 x_2^2 + \frac{2}{8}x_2^2 \; . \tag{5.2}$$

Note that for LT codes, the multi-edge-type variable node distribution is not a function of the vector $\mathbf{r}$. This is explained by the fact that the factor graph representation of the encoding does not include any channel observation, i.e., all the entries of the vector $\mathbf{b}$ are equal to 0. In the following, we will divide the variable nodes into m_e protection classes $(C_1, C_2, \ldots, C_{m_e})$ with monotonically decreasing levels of protection.

5.1.1 Node-perspective degree distributions

We now determine the multi-edge-type variable and check node degree distributions $\nu(\mathbf{x})$ and $\mu(\mathbf{x})$ for UEP LT codes. In order to determine $\mu(\mathbf{x}) = \sum_{\mathbf{d}} \mu_{\mathbf{d}} \mathbf{x}^{\mathbf{d}}$, we need to compute the fraction of check nodes of type $\mathbf{d}$, i.e., the coefficients $\mu_{\mathbf{d}}$. Recall that according to the encoding algorithm of LT codes, the probability of an output symbol having degree i is Ω_i (in the UEP context, we call $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$ the overall output symbol degree distribution). Given that the degree of an output symbol corresponds to the number of edges connected to it, i.e., $i = \sum_{j=1}^{m_e} d_j$ where d_j denotes the number of edges of type j connected to a check node, we have

$$
\begin{aligned}
\mu_{\mathbf{d}} &= \Omega_i \cdot \frac{\binom{k_1}{d_1} \cdot \binom{k_2}{d_2} \cdots \binom{k_{m_e}}{d_{m_e}}}{\binom{k}{i}} \\
&= \Omega_i \cdot \frac{\binom{k}{i}^{-1}}{d_1! \cdots d_{m_e}!} \frac{k_1!}{(k_1 - d_1)!} \cdots \frac{k_{m_e}!}{(k_{m_e} - d_{m_e})!} \ ,
\end{aligned}
\tag{5.3}
$$

where k_j is the number of input symbols of class j, and $\mu_{\mathbf{d}}$ is the probability of an output symbol (check node) having edge degree vector $\mathbf{d} = (d_1, \ldots, d_{m_e})$. Equation (5.3) is derived by considering the LT encoding which is a choice without replacement, i.e., a degree-i output symbol has exactly i distinct neighbors.

In order to simplify our description, we consider that the encoding of LT codes is analogous to a choice with replacement. Such an approximation becomes more exact as the number of symbols in each protection class increases, since as k_j grows, it will become more and more unlikely to choose the same input symbol more than once during the encoding of an output symbol. In this case, we have

$$
\mu_{\mathbf{d}} = \Omega_i \cdot \frac{i!}{d_1! d_2! \cdots d_{m_e}!} \omega_1^{d_1} \omega_2^{d_2} \cdots {\omega_{m_e}}^{d_{m_e}} \ , \tag{5.4}
$$

where ω_j is the probability of an input symbol of the class C_j being chosen among the k input symbols and $i = \sum_{j=1}^{m_e} d_j$. The check node degree distribution is then given by

$$
\mu(\mathbf{x}) = \sum_{\mathbf{d}} \Omega_i \cdot \frac{i!}{d_1! d_2! \cdots d_{m_e}!} \omega_1^{d_1} \omega_2^{d_2} \cdots {\omega_{m_e}}^{d_{m_e}} \mathbf{x}^{\mathbf{d}} \ . \tag{5.5}
$$

In order to compute the variable node degree distribution $\nu(\mathbf{x}) = \sum_{\mathbf{d}} \nu_{\mathbf{d}} \mathbf{x}^{\mathbf{d}}$, first recall that according to our previous definitions, the variable nodes belonging to C_j are only connected to type-j edges. This means that for the variable nodes, the edge degree vectors $\mathbf{d}$ have only one non-zero component, e.g., for the two-class case $\mathbf{d} = (d_1, 0)$ or $(0, d_2)$.

Let ν_{d_j} represent $\nu_{\mathbf{d}}$ when d_j is the only non-zero component of $\mathbf{d}$. By simple combinatorial arguments, the probability of a variable node of class C_j having degree d_j is given by

$$\nu_{d_j} = \binom{\beta_j \gamma k}{d_j} p_j^{d_j} (1 - p_j)^{\beta_j \gamma k - d_j} , \tag{5.6}$$

where $\beta_j = \mu_{x_j}(\mathbf{1})$ is the average of type-j edges connected to a check node, $\gamma = n/k$, and $p_j = 1/k_j$ is the probability of an input symbol being chosen among the k_j symbols of C_j. Note that the product $\beta_j \gamma k$ represents the total number of type-j edges present in the multi-edge graph induced by the LT encoding. The variable node degree distribution can then be written as

$$\nu(\mathbf{x}) = \sum_{\mathbf{d}, j} \binom{\beta_j \gamma k}{d_j} p_j^{d_j} (1 - p_j)^{\beta_j \gamma k - d_j} \mathbf{x}^{\mathbf{d}} . \tag{5.7}$$

Since the edge degree vectors $\mathbf{d}$ for the variable nodes have only one non-zero component, Eq. (5.7) reduces to

$$\nu(\mathbf{x}) = \sum_{j=1}^{m_e} \sum_{d_j=1}^{\beta_j \gamma k} \binom{\beta_j \gamma k}{d_j} p_j^{d_j} (1 - p_j)^{\beta_j \gamma k - d_j} x_j^{d_j} . \tag{5.8}$$

Equations (5.5) and (5.8) are quite general and apply for any unequal-error-protecting LT code with m_e protection classes. In order to clarify the concepts in a simple manner, in our example and finite length simulations, we will consider codes with only two protection classes, i.e., codes with $m_e = 2$. In this particular case, eqs. (5.5) and (5.7) can be written as

$$\mu(\mathbf{x}) = \sum_{d_1, d_2} \Omega_{d_1 + d_2} \cdot \binom{d_1 + d_2}{d_1} \omega_1^{d_1} (1 - \omega_1)^{d_2} \cdot x_1^{d_1} x_2^{d_2} , \tag{5.9}$$

$$\nu(\mathbf{x}) = \sum_{j=1}^{2} \sum_{d_j=1}^{\beta_j \gamma k} \binom{\beta_j \gamma k}{d_j} p_j^{d_j} (1 - p_j)^{\beta_j \gamma k - d_j} \cdot x_j^{d_j} . \tag{5.10}$$

5.1.2 Encoding and decoding

The encoding algorithm of multi-edge-type LT codes is similar to the encoding of traditional LT codes described in Section 2.5.1. However, instead of selecting the output symbol degree i according to $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$, we must select an edge degree vector $\mathbf{d}$ according to $\mu(\mathbf{x}) = \sum_{\mathbf{d}} \mu_{\mathbf{d}} \mathbf{x}^{\mathbf{d}}$. After that, an output symbol with edge degree vector $\mathbf{d} = (d_1, \ldots, d_{m_e})$ is formed by selecting uniformly and at random d_j input symbols from C_j for $j = 1, \ldots, m_e$, and performing a bitwise XOR operation between them. In summary, the multi-edge LT encoding of an output symbol can be described in a step by step manner as follows

1. Randomly choose the output symbol edge degree vector $\mathbf{d} = (d_1, \ldots, d_{m_e})$ accordingly to the degree distribution $\mu(\mathbf{x})$.
2. For $j = 1, \ldots, m_e$, choose uniformly and at random d_j symbols among the k_j input symbols of protection class C_j.
3. Form the output symbol performing the exclusive-or of the chosen $i = \sum_{j=1}^{m_e} d_j$ symbols.

The output symbol is then transmitted, and the encoding process is repeated until a sufficient number of symbols is obtained at the receiver or a pre-defined number of γk output symbols is generated. The decoding algorithm for multi-edge-type LT codes is exactly the same as the iterative decoder for LT codes described in Section 2.5.2.

5.2 Construction algorithms for unequal-error-protecting LT codes

Herein, we proceed to a multi-edge-type analysis of the unequal-error-protecting LT codes presented in [51] and [52] and propose a novel construction strategy for such a class of codes.

5.2.1 Weighted approach

The first strategy (to which we will refer as the weighted approach) for the construction of UEP LT codes was introduced in [51]. In that work, the authors proposed the partitioning of the k variable nodes into m_e sets (protection classes) of sizes $\alpha_1 k, \alpha_2 k, \ldots, \alpha_{m_e} k$ such that $\sum_{j=1}^{m_e} \alpha_j = 1$. Let q_j be the probability of an edge being connected to a particular variable node within the set j. By introducing a bias on the probabilities[1] q_j, some sets of symbols become more likely to be selected during the encoding procedure, which makes the symbols that compose that set more protected, i.e., the biasing gives rise to an unequal-error-protecting capability.

Consider for example the two-class case. For this case, the authors in [51] divide the input symbols into two sets: *more important bits* (MIB) and *less important bits* (LIB) composed by αk and $(1-\alpha)k$ symbols, respectively. Furthermore, they set $q_1 = \frac{k_M}{k}$ and $q_2 = \frac{k_L}{k}$ for some $0 < k_L < 1$ and $k_M = \frac{1-(1-\alpha)k_L}{\alpha}$. The difference between the performances of the MIB and LIB can then be controlled by varying k_M. Note that $k_M = 1$ corresponds to the equal-error-protecting LT codes (also referred to as non-UEP LT codes).

For the two-class case, the encoding algorithm is defined as follows. First, define the output symbol degree i according to a degree distribution $\Omega(x)$, and define $d_1 = \min([\alpha i k_M], \alpha k)$ and $d_2 = i - d_1$. Second, choose d_1 and d_2 symbols among the MIB and the LIB, respectively. Finally, the output symbol is generated performing a bitwise XOR operation over the $i = d_1 + d_2$ selected input symbols. One drawback of this algorithm is that the extension for the $m_e > 2$ case is not trivial. We solve this problem including the weighted scheme in the multi-edge framework and applying the encoding algorithm for multi-edge-type UEP LT codes described in Section 5.1.2.

The multi-edge framework derivation is straightforward once we notice that for the weighted scheme, the probability of an input symbol of set j being chosen among the k input symbols (ω_j) can be written as: $\omega_j = q_j k_j$. Replacing these values into Eq. (5.4), we obtain the coefficients of the multi-edge check node degree distribution of the weighted scheme. Since the selection of the input symbols within a protection class during encoding remains uniform and random, the variable node degree distribution is obtained by setting $p_j = \frac{1}{\alpha_j k}$ in Eq. (5.7). In summary, the multi-edge-type degree distributions for the weighted approach are given by

$$\mu(\mathbf{x}) = \sum_{\mathbf{d}} \Omega_i \cdot \frac{i!}{d_1! \cdots d_{m_e}!} (q_1 \alpha_1 k)^{d_1} \cdots (q_{m_e} \alpha_{m_e} k)^{d_{m_e}} \mathbf{x}^{\mathbf{d}} \,, \tag{5.11}$$

[1] For equal-error-protecting LT codes, $q_1 = \ldots = q_{m_e} = \frac{1}{k}$.

$$\nu(\mathbf{x}) = \sum_{j=1}^{m_e} \sum_{d_j=1}^{\beta_j \gamma k} \left[\binom{\beta_j \gamma k}{d_j} \left(\frac{1}{\alpha_j k} \right)^{d_j} \left(1 - \frac{1}{\alpha_j k} \right)^{\beta_j \gamma k - d_j} \right] x_j^{d_j} . \tag{5.12}$$

5.2.2 Windowed approach

The second UEP LT code construction strategy investigated in the present chapter (from now on referred to as the windowed approach) was introduced in [52]. Similar to [51], the windowed approach partitions the input symbols into protection classes composed by $k_1, k_2, \ldots, k_{m_e}$ symbols such that $k_1 + k_2 + \cdots + k_{m_e} = k$. As for the weighted approach, a decreasing protection level for the classes is assumed, i.e., the ith class is more important than the jth class if $i < j$. Furthermore, another partitioning of the input symbols (which the authors call windows) is considered. The ith window is defined as the set of the first $w_j = \sum_{i=1}^{j} k_i$ input symbols and consequently, the most important symbols form the first window while the whole block comprises the final m_eth window.

In the windowed scheme, each output symbol is encoded first selecting a window j following a probability distribution $\Gamma(x) = \sum_{j=1}^{m_e} \Gamma_j x^j$, where Γ_j is the probability that the jth window is chosen. Once the jth window is defined, each output symbol is formed according to the regular LT encoding algorithm considering only the symbols inside the selected window and following a degree distribution $\Omega^{(j)}(x) = \sum_{i=1}^{w_j} \Omega_i^{(j)} x^i$.

The derivation of the multi-edge-type check node degree distribution for the windowed approach is not as direct as for the weighted. Nevertheless, it can be shown by means of simple combinatorics that the coefficients $\mu_{\mathbf{d}}$ for the windowed approach are given by

$$\mu_{\mathbf{d}} = \frac{i!}{d_1! d_2! \ldots d_{m_e}!} \left\{ \sum_{j=2}^{m_e} \Omega_i^{(j)} \Gamma_j \prod_{r=1}^{j} \alpha_r^{d_r} \cdot [1 - \operatorname{sgn}(\sum_{s>j} d_s)] \right\}$$
$$+ \Omega_i^{(1)} \Gamma_1 [1 - \operatorname{sgn}(\sum_{s>1} d_s)] , \tag{5.13}$$

where $i = \sum_{j=1}^{m_e} d_j$ and $\alpha_j = k_j / k$. The reasoning behind Eq. (5.13) can be made clearer if we consider the $m_e = 3$ case. In this case, consider the probability $\mu_{\mathbf{d}}$ of selecting an edge degree vector $\mathbf{d} = (d_1, d_2, d_3)$. If the window w_1 is selected (which happens with probability Γ_1), we can write

$$\mu_{\mathbf{d}} = \begin{cases} \Omega^{(1)}_{d_1+d_2+d_3}\Gamma_1, & \text{if } \ d_2 = d_3 = 0 \ , \\ 0, & \text{if } \ d_2 > 0 \ \text{ or } \ d_3 > 0 \ . \end{cases} \tag{5.14}$$

Note that Eq. (5.14) can be written in a compact form as

$$\mu_{\mathbf{d}} = \Omega^{(1)}_{d_1+d_2+d_3}\Gamma_1[1 - \operatorname{sgn}(d_2 + d_3)] \ . \tag{5.15}$$

Suppose now that the window w_2 is selected. In this case, we can write

$$\mu_{\mathbf{d}} = \begin{cases} \frac{(d_1+d_2+d_3)!}{d_1!d_2!d_3!}\left(\Omega^{(2)}_{d_1+d_2+d_3}\Gamma_2\alpha_1^{d_1}\alpha_2^{d_2}\right), & \text{if } \ d_3 = 0 \ , \\ 0, & \text{if } \ d_3 > 0 \ . \end{cases} \tag{5.16}$$

Note that the equation for case $d_3 = 0$ can be derived multiplying Eq. (5.4) by Γ_2 and substituting ω_j by α_j for $j = 1, 2$. In compact notation, we can write Eq. (5.16) as

$$\mu_{\mathbf{d}} = \frac{(d_1 + d_2 + d_3)!}{d_1!d_2!d_3!}\left(\Omega^{(2)}_{d_1+d_2+d_3}\Gamma_2\alpha_1^{d_1}\alpha_2^{d_2}[1 - \operatorname{sgn}(d_3)]\right) \ , \tag{5.17}$$

where the leftmost term indicates the number of permutations of $(d_1+d_2+d_3)$ elements with the repetition of d_1, d_2, and d_3 elements, $\Omega^{(2)}_{d_1+d_2+d_3}$ is the probability of choosing a degree $(d_1 + d_2 + d_3)$ according to the distribution $\Omega^{(2)}(x)$, Γ_2 is the probability of the window w_2 being selected, and $\alpha_j = k_j/k$.

Lastly, suppose now that the window w_3 is selected. Once again, we can use Eq. (5.4) and write

$$\mu_{\mathbf{d}} = \frac{(d_1 + d_2 + d_3)!}{d_1!d_2!d_3!}\left(\Omega^{(3)}_{d_1+d_2+d_3}\Gamma_3\alpha_1^{d_1}\alpha_2^{d_2}\alpha_3^{d_3}\right) \ . \tag{5.18}$$

In summary, adding eqs. (5.15), (5.17), and (5.18) we have

$$\mu_{\mathbf{d}} = \frac{(d_1 + d_2 + d_3)!}{d_1!d_2!d_3!}\left\{\Omega^{(2)}_{d_1+d_2+d_3}\Gamma_2\alpha_1^{d_1}\alpha_2^{d_2} \cdot [1 - \operatorname{sgn}(d_3)] + \Omega^{(3)}_{d_1+d_2+d_3}\Gamma_3\alpha_1^{d_1}\alpha_2^{d_2}\alpha_3^{d_3}\right\} + \Omega^{(1)}_{d_1+d_2+d_3}\Gamma_1[1 - \operatorname{sgn}(d_2 + d_3)] \ , \tag{5.19}$$

which can be written as

$$\mu_{\mathbf{d}} = \frac{i!}{d_1!d_2!d_3!}\left\{\sum_{j=2}^{3}\Omega_i^{(j)}\Gamma_j\prod_{r=1}^{j}\alpha_r^{d_r}\cdot[1-\mathrm{sgn}(\sum_{s>j}d_s)]\right\} + \Omega_i^{(1)}\Gamma_1[1-\mathrm{sgn}(\sum_{s>1}d_s)]\ , \tag{5.20}$$

where $i = \sum_{j=1}^{3} d_j$. If we apply the same reasoning to the windowed approach with m_e protection classes, we obtain Eq. (5.13). The variable node degree distribution can be obtained as for the weighted case, i.e., setting $p_j = \frac{1}{\alpha_j k}$ in Eq. (5.7). In summary, the multi-edge degree distributions for the windowed approach are given by

$$\begin{aligned}\mu(\mathbf{x}) = \sum_{\mathbf{d}}\frac{i!}{d_1!d_2!\dots d_{m_e}!}&\left\{\sum_{j=2}^{m_e}\Omega_i^{(j)}\Gamma_j\prod_{r=1}^{j}\alpha_r^{d_r}\cdot[1-\mathrm{sgn}(\sum_{s>j}d_s)]\right\}\\ &+\Omega_i^{(1)}\Gamma_1[1-\mathrm{sgn}(\sum_{s>1}d_s)]\mathbf{x}^{\mathbf{d}}\ ,\end{aligned} \tag{5.21}$$

$$\nu(\mathbf{x}) = \sum_{j=1}^{m_e}\sum_{d_j=1}^{\beta_j\gamma k}\left[\binom{\beta_j\gamma k}{d_j}\left(\frac{1}{\alpha_j k}\right)^{d_j}\left(1-\frac{1}{\alpha_j k}\right)^{\beta_j\gamma k - d_j}\right]x_j^{d_j}\ . \tag{5.22}$$

5.2.3 Flexible UEP LT codes

We showed in the previous subsections that both the weighted and the windowed schemes rely on the modification of the probability of occurrence of an output symbol of type $\mathbf{d}$, i.e., they modify the coefficients $\mu_{\mathbf{d}}$ of a non-UEP LT code (Eq. (5.4)) in order to favor the selection of some class of input symbols during encoding. In the following, we propose a scheme that works by biasing the coefficients $\mu_{\mathbf{d}}$ in order to increase the average number of edges of the most protected classes. In fact, we transfer edges from one less important class to another more important one.

In order to understand the idea, consider an LT code with two protection classes. Its check node degree distribution can be written as

$$\begin{aligned}\mu(\mathbf{x}) =&\mu_{(1,0)}x_1 + \mu_{(0,1)}x_2 + \mu_{(2,0)}x_1^2 + \mu_{(1,1)}x_1x_2 +\\ &\mu_{(0,2)}x_2^2 + \mu_{(3,0)}x_1^3 + \cdots + \mu_{(0,i_{max})}x_2^{i_{max}},\end{aligned} \tag{5.23}$$

where $\mu_{(d_1,d_2)} = \mu_{\mathbf{d}}$ for $\mathbf{d} = (d_1, d_2)$ and $i_{max} = \max(i|\Omega_i > 0)$. In order to keep the original overall output symbol degree distribution $\Omega(x)$, the coefficients $\mu_{\mathbf{d}}$ must satisfy the following condition

$$\sum_{\mathbf{d}} \mu_{\mathbf{d}} = \Omega_i, \text{ for all } \mathbf{d} : \sum_{j=1}^{m_e} d_j = i \,. \tag{5.24}$$

In the two-class case for example, $\mu_{(1,0)} + \mu_{(0,1)} = \Omega_1$, $\mu_{(2,0)} + \mu_{(1,1)} + \mu_{(0,2)} = \Omega_2$, and so on. We like to keep the overall degree distribution constant in order to keep the overall performance of the rateless scheme unchanged.

The idea of our proposed scheme is to increase the probability of selection of the most important input symbols by increasing the occurrence probability of output symbols which are more connected to input symbols of the most sensitive class. For example, in the two-class case we increase the values of the coefficients $\mu_{(d_1,d_2)}$ with $d_1 > d_2$ while observing the condition given in Eq. (5.24). More generally, in order to favor the selection of the most important input symbols during encoding, we increase the values of the coefficients $\mu_{(d_1,d_2,\ldots,d_{m_e})}$ with $d_j > d_{j+1}$ (since it is assumed that the class C_j is more important than C_{j+1}) for $j = 1, \ldots, m_e - 1$ while observing condition (5.24).

Our proposed strategy to favor the selection of the most important symbols during the LT encoding is described as follows. Consider an output symbol with edge degree vector $\mathbf{d} = (d_1, \ldots, d_{j-1}, d_j, \ldots, d_{m_e})$ where $0 < d_{j-1} < d_j$. The fraction of output symbols of this kind is given by $\mu_{\mathbf{d}=(d_1,\ldots,d_{j-1},d_j,\ldots,d_{m_e})}$. Let $a = d_{j-1}$ and $b = d_j$. In order to favor the selection of symbols in the most protected class $j-1$, we can convert a fraction Δ_j of the output symbols with $\mathbf{d} = (d_1, \ldots, a, b, \ldots, d_{m_e})$ where $0 < a < b$ into symbols with $\mathbf{d} = (d_1, \ldots, b, a, \ldots, d_{m_e})$. Following this strategy, it is not difficult to see that since $a < b$, the selection during LT encoding of input symbols of class $j-1$ will become more likely, while the selection of the symbols on class j will become less probable. According to this scheme, we can define an LT code with the check node degree coefficients $\mu_{\mathbf{d}}^{UEP}$ as follows.

Let $\boldsymbol{\Delta} = (\Delta_2, \ldots, \Delta_{m_e})$ be a vector whose components Δ_j denote the fraction of the check node degree coefficients $\mu_{\mathbf{d}}$ with $d_j > d_{j-1}$ to be reduced in order to favor the selection of bits of class $j-1$ during the LT encoding. Given $\boldsymbol{\Delta}$ and an LT code with overall degree distribution $\Omega(x)$, an UEP LT code with m_e protection classes can be generated according to Algorithm 4.

Algorithm 4 Flexible UEP LT construction algorithm

1. Compute $\mu(\mathbf{x})$ according to Eq. (5.5)
2. for $j = m_e$ to 2

        for all $\mathbf{d}$ with $\mu_{\mathbf{d}} > 0$
            if $0 < d_{j-1} < d_j$
                $a \longleftarrow d_{j-1}$
                $b \longleftarrow d_j$
                $\mu^{UEP}_{(d_1,\ldots,a,b,\ldots,d_{m_e})} = \mu_{(d_1,\ldots,a,b,\ldots,d_{m_e})} - \Delta_j \cdot \mu_{(d_1,\ldots,a,b,\ldots,d_{m_e})}$
                $\mu^{UEP}_{(d_1,\ldots,b,a,\ldots,d_{m_e})} = \mu_{(d_1,\ldots,b,a,\ldots,d_{m_e})} + \Delta_j \cdot \mu_{(d_1,\ldots,a,b,\ldots,d_{m_e})}$
            else
                $\mu^{UEP}_{\mathbf{d}} = \mu_{\mathbf{d}}$
            end
        end
        $\mu_{\mathbf{d}} = \mu^{UEP}_{\mathbf{d}}$
    end

The following example should clarify the flexible UEP LT construction algorithm.

Example 2 *Consider an LT code with overall degree distribution* $\Omega(x) = 0.15x + 0.55x^2 + 0.30x^3$ *and consider* $\boldsymbol{\Delta} = (0.3)$*. We intend to construct a two-class unequal-error-protecting LT code where 10 % of the input symbols belong to the most protected class, i.e.,* $\alpha_1 = 0.1$*. The coefficients* $\mu_{\boldsymbol{d}}$ *for the non-UEP case can be computed by means of Eq. (5.9) with* $\omega_1 = \alpha_1$ *and* $\omega_2 = 1-\alpha_1$*. In order to generate a UEP LT code, we compute the coefficients* $\mu^{UEP}_{\boldsymbol{d}}$ *of the UEP LT multi-edge-type check node degree distribution according to Algorithm 4. In the present example, for* $\boldsymbol{d} = (d_1, d_2) = (1, 2)$ *and* $\Delta_2 = 0.3$ *we have*

$$\mu^{UEP}_{(2,1)} = \mu_{(2,1)} + 0.3\mu_{(1,2)} \ , \tag{5.25}$$

$$\mu^{UEP}_{(1,2)} = \mu_{(1,2)} - 0.3\mu_{(1,2)} \ . \tag{5.26}$$

For every other $\boldsymbol{d}$*,* $\mu^{UEP}_{(d_1,d_2)} = \mu_{(d_1,d_2)}$*. The multi-edge check node degree distributions of the original and UEP LT codes are depicted in Table 5.1.*

◇

The flexible UEP LT approach has advantages of both the weighted and the windowed approach. First, the difference between the performance of the different protection

Table 5.1: Flexible UEP LT construction example.

$\mathbf{d}$	(0,1)	(1,0)	(0,2)	(2,0)	(0,3)	(3,0)	(2,1)	(1,2)	(1,1)
$\mu_{\mathbf{d}}$	0.135	0.015	0.4455	0.0055	0.2187	0.0003	0.0081	0.0729	0.099
$\mu_{\mathbf{d}}^{UEP}$	0.135	0.015	0.4455	0.0055	0.2187	0.0003	0.02997	0.05103	0.099

classes can be controlled through the vector $\mathbf{\Delta}$ by adjusting its components, i.e., the higher Δ_j, the greater the difference between the performance of classes j and $j-1$. Second, its encoding procedure is easily generalized for the case $m_e > 2$, a characteristic that the weighted approach does not possess.

Additionally, our scheme is more suitable than the windowed one for applications where a precoding is needed, e.g., Raptor codes [26]. For these codes, the input symbols are encoded by a traditional erasure correcting code prior to the LT encoding. This guarantees a successful recovery of all input symbols while allowing a linear time encoding and decoding. In case we desire to use Raptor codes for UEP applications using our flexible UEP construction, we can do it using only one precoding for the whole data. In contrast, the windowed approach has to precode all defined protection classes separately [52]. Another advantage of our proposed construction is that using only one precoding avoids finite-length effects that may arise from separately encoding protection classes with a low number of symbols.

In the following section, we develop an asymptotic analysis of multi-edge-type UEP LT codes and show how the three approaches presented herein behave when the number of input symbols tends to infinity ($k \to \infty$). Moreover, we use the such analysis to show the role of the parameter $\mathbf{\Delta}$ on the performance of the different protection classes of an UEP LT code constructed using our proposed algorithm.

5.3 Asymptotic analysis of multi-edge-type UEP LT codes

The asymptotic analysis of multi-edge LT codes can be done by means of density evolution. However, in a multi-edge-type analysis, the computation of one density for each edge type is required. With this in mind, we point out that the iterative decoding of LT codes is analogous to the belief-propagation decoding of symbols transmitted through an erasure channel where all the variable nodes are erased and the check node values are given by the output symbols they represent. This analogy allows us to use the results derived for the BEC to compute the error probability of an LT code.

Theorem 2 (LT decoding failure probability) *The erasure probability $y_l^{(j)}$ of an input symbol of class j of a multi-edge LT code with node-perspective degree distribution pair $(\nu(\boldsymbol{x}), \mu(\boldsymbol{x}))$ at iteration $l \geq 0$ is given by*

$$y_l^{(j)} = \sum_{d_j} \nu_{d_j} \cdot (1 - \rho^{(j)}(1 - y_{l-1}^{(1)}, \dots, 1 - y_{l-1}^{(m_e)}))^{d_j} \ , \tag{5.27}$$

where $\forall j$, $y_{-1}^{(j)} = 1$, $\rho^{(j)}(\boldsymbol{x}) = \frac{\mu_{x_j}(\boldsymbol{x})}{\mu_{x_j}(\boldsymbol{1})} = \sum_{\boldsymbol{d}} \rho_{\boldsymbol{d}}^{(j)} \boldsymbol{x}^{\boldsymbol{d}}$, $\rho_{\boldsymbol{d}}^{(j)}$ denotes the fraction of type j edges connected to check nodes of type $\mathbf{d}$, and ν_{d_j} is given by Eq. (5.6).

Proof: Let G denote the bipartite graph induced by an LT encoding. Consider a subgraph G_l^j of G formed by a variable node v^j, chosen uniformly and at random among the ones of class j and all its neighbors within distance $2l$. Asymptotically, the subgraph G_l^j is a tree [53] of depth $2l$ that represents the dependency between the value assumed by v^j and the messages sent by the other variable nodes after l message passing decoding iterations. Let the variable node v^j be the root (depth 0) of G_l^j and assume that $y_l^{(j)}$ is its erasure probability. Consider now that the nodes at depth 2 in G_l^j are the roots of independent G_{l-1}^s trees, for $s \in \{1, \dots, m_e\}$. Consider the variable-to-check messages in the lth iteration. By assumption, each such message is an erasure with probability $y_{l-1}^{(j)}$ and all messages are independent. Recall that in a multi-edge framework, each check node of type $\mathbf{d} = (d_1, d_2, ..., d_{m_e})$ is connected to d_1 edges of type 1, d_2 edges of type 2, and so on. Since we are considering a BP decoding, by definition, a check-to-variable message emitted by a check node of degree i along a particular edge is an erasure iff any of the $i-1$ incoming messages is an erasure. Thus, the erasure probability of an outgoing message of a check node at depth l with edge degree vector $\mathbf{d}$ sent through an edge of type j is equal to $1 - (1 - y_{l-1}^{(1)})^{d_1} \cdots (1 - y_{l-1}^{(j)})^{d_j - 1} \cdots (1 - y_{l-1}^{(m_e)})^{d_{m_e}}$. Since the outgoing edge has probability $\rho_{\mathbf{d}}^{(j)}$ to be connected to a check node of type $\mathbf{d}$, it follows that the expected erasure probability of a check-to-variable message in the lth iteration is equal to $1 - \rho^{(j)}(1 - y_{l-1}^{(1)}, \dots, 1 - y_{l-1}^{(m_e)})$ where $\rho^{(j)}(\mathbf{x}) = \sum \rho_{\mathbf{d}}^{(j)} \mathbf{x}^{\mathbf{d}}$. Now consider the erasure probability of the root node at iteration l. By definition, a variable node will be considered erased if all its incoming messages are erasures. Since a variable node of class j has only connections to edges of type j, the probability of a variable node of degree d_j being erased is $(1 - \rho^{(j)}(1 - y_{l-1}^{(1)}, \dots, 1 - y_{l-1}^{(m_e)}))^{d_j}$. Averaging over all the possible variable node degrees d_j we conclude that the erasure probability of an input symbol of class j at iteration l is given by $\sum_{d_j} \nu_{d_j} \cdot (1 - \rho^{(j)}(1 - y_{l-1}^{(1)}, \dots, 1 - y_{l-1}^{(m_e)}))^{d_j}$ as claimed.

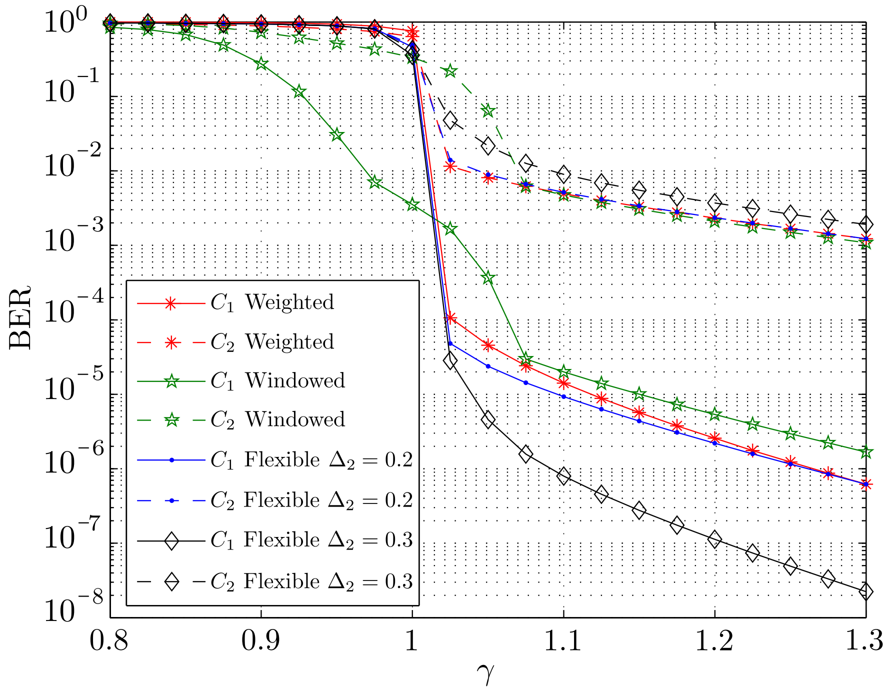

Figure 5.2: Asymptotic performance of the weighted, windowed, and the proposed flexible UEP LT construction strategies.

Equation (5.27) together with the degree distributions $(\nu(\mathbf{x}), \mu(\mathbf{x}))$ allows us to compute the asymptotic $(k \to \infty)$ performance of a multi-edge UEP LT code with overall output symbol degree distribution $\Omega(x)$. Herein, we consider multi-edge UEP LT codes with the overall output symbol degree distribution proposed in [26]

$$\begin{aligned}\Omega(x) = \; & 0.007969x + 0.493570x^2 + 0.166220x^3 + 0.0726464x^4 + 0.082558x^5 \\ & + 0.056058x^8 + 0.037229x^9 + 0.055590x^{19} + 0.025023x^{64} + 0.003135x^{66}. \quad (5.28)\end{aligned}$$

Figure 5.2 shows the asymptotic performance of the three different UEP LT codes construction strategies presented in this paper. The parameters for the weighted $(k_m = 2.077)$ and the windowed $(\Gamma_1 = 0.084)$ approaches are optimized for an overhead $\gamma = n/k = 1.05$ according to [52]. The flexible UEP LT asymptotic analysis was carried out for $\Delta_2 = 0.2$ and $\Delta_2 = 0.3$. Note that as we increase the value of Δ_2 the difference between the performance of both protection classes increases. Furthermore, the MIB in our proposed algorithm with $\Delta_2 = 0.3$ have a better performance than the weighted and windowed approach for $\gamma > 1.025$. Finally, we present in Fig. 5.3 the asymptotic

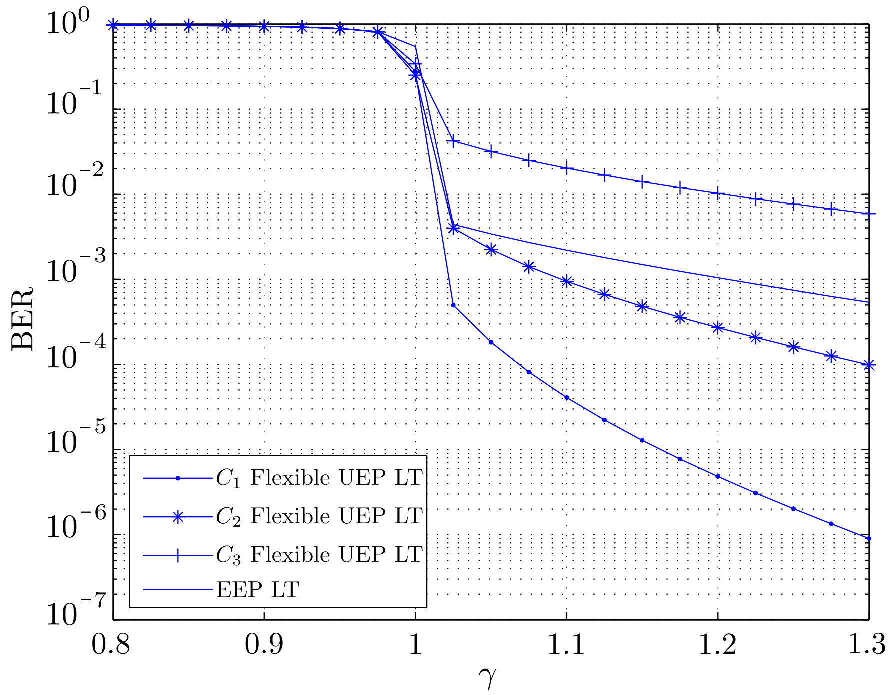

Figure 5.3: Asymptotic performance of the flexible UEP LT construction for 3 protection classes.

analysis results for an UEP LT code designed according to our proposed algorithm with $\alpha = (0.1, 0.3, 0.6)$ and $\mathbf{\Delta} = (0.4, 0.8)$.

5.4 Simulation results

In this section, we present the finite-length simulation results for the weighted, windowed, and our proposed scheme for generating UEP LT codes. Figure 5.4 shows the bit-error rates after performing LT decoding. The parameters for the weighted, windowed, and the flexible UEP LT approaches are the same as with the asymptotic analysis depicted in Fig. 5.2, i.e., for the weighted approach we set $k_m = 2.077$, for the windowed $\Gamma_1 = 0.084$ with both windows using the same degree distribution $\Omega(x)$ (i.e., $\Omega^{(1)}(x) = \Omega^{(2)}(x) = \Omega(x)$), and for the flexible UEP LT we set $\Delta_2 = 0.3$. We assume the transmission of $k = 5000$ input symbols divided into two protection classes. The first protection class is composed of 10 % of the input symbols ($k_1 = 0.1k$), and the second protection class contains the other $k_2 = k - k_1$ input symbols.

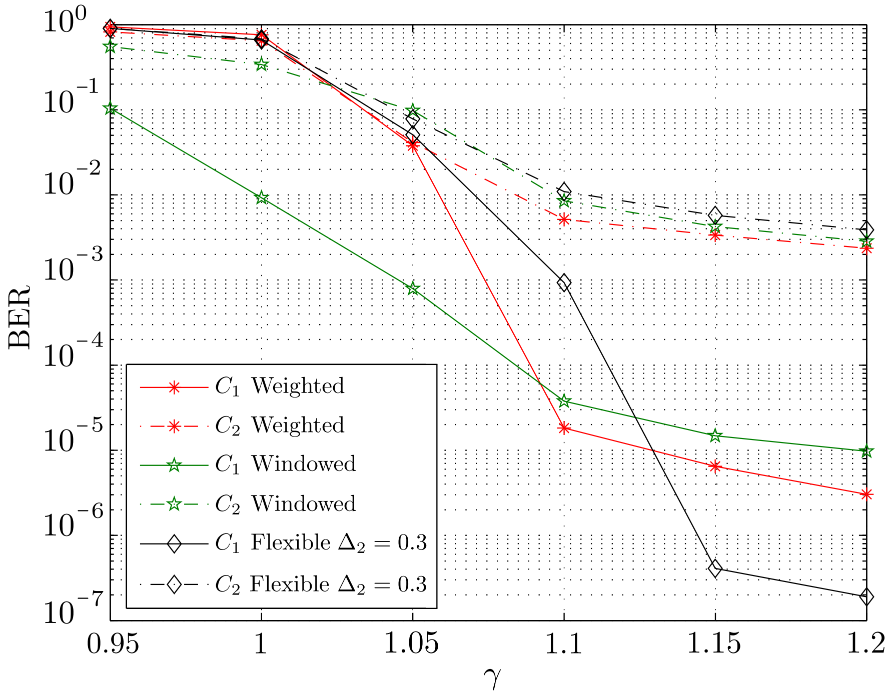

Figure 5.4: Simulation results of the weighted, windowed, and flexible schemes for $k = 5000$.

Note that for the finite-length simulation, the flexible UEP LT strategy reaches a lower BER for the most protected class than the other approaches for high overhead values. As predicted by the asymptotic analysis, the windowed approach has a better performance for low-overheads. This is due to the precoding effect of the windowed scheme, e.g., in the two-class case the windowed scheme is equivalent to first generating $\Gamma_1 \gamma k$ symbols of class one (a precoding) and then proceeding to the regular LT-encoding. The results indicate that the flexible approach is preferred for applications where a lower BER is required, while the windowed can be used for unequal recovery time (URT) applications more efficiently [52].

Nevertheless, for applications where a precoding is needed, our scheme is more suitable than the windowed, since it only uses one precoding for the complete data block while the windowed approach has to precode all defined protection classes separately [52]. Furthermore, a single precoding avoids finite-length effects that may arise from separately precoding protection classes with a low number of bits. Additionally, flexible UEP LT codes can easily be generalized for applications with more than two protection classes, a characteristic which is not supported by the weighted approach.

Chapter 6

LDPC-based Joint Source-Channel Coding

It is widely observed that for communication systems transmitting in the non-asymptotic regime with limited delay constraints, a separated design of source and channel codes is not optimum, and gains in complexity and fidelity may be obtained by a joint design strategy. The approach for joint source-channel coding pursued in this chapter relies on a graphical model where the structures of the source and the channel codes are jointly exploited. More specifically, we are concerned with the optimization of joint systems that perform a linear encoding of the source output and channel input by means of low-density parity-check codes.

Herein, we present a novel LDPC-based joint source-channel coding system where the amount of information about the source bits available at the decoder is increased by improving the connection profile between the factor graphs of the source and channel codes that compound the joint system. Furthermore, we propose an optimization strategy for the component codes based on a multi-edge-type joint optimization.

6.1 Joint source-channel coding

The “separation principle” between source and channel coding is one of the milestones in the development of Information Theory. A consequence of the direct source-channel coding theorem laid by Shannon in his 1948 paper [1], this principle states that there is no loss in asymptotic performance when source and channel coding are performed separately. It is though widely observed that for communication systems transmitting in the non-asymptotic regime with limited delay constraints, the separation principle may not be applicable and gains in complexity and fidelity may be obtained by a joint design strategy [54].

The main idea when dealing with the joint source-channel (JSC) coding problem is to take advantage of the residual redundancy arising from an incomplete data compression in order to improve the error rate performance of the communication system. This possibility was already mentioned by Shannon in [1] and quoted by Hagenauer in [8]: "However, any redundancy in the source will usually help if it is utilized at the receiving point. In particular, if the source already has redundancy and no attempt is made to eliminate it in matching to the channel, this redundancy will help combat noise."

One of the possible approaches to JSC coding, and the one we will pursue in this chapter, relies on a graphical model where the structure of the source and the channel codes are jointly exploited. We are particularly interested in systems that perform linear encoding of sources by means of error-correcting codes. The strategy of such schemes is to treat the source output $\mathbf{u}$ as an error pattern and perform compression calculating the syndrome generated by $\mathbf{u}$, i.e., the source encoder calculates $\mathbf{s} = \mathbf{u}\mathbf{H}^T$, where $\mathbf{H}$ is the parity-check matrix of the linear error-correcting code being considered and the syndrome $\mathbf{s}$ is the compressed sequence.

Compression schemes based on syndrome encoding for binary memoryless sources were developed in the context of variable-to-fixed length algorithms in [55] and [56]. Afterwards, Ancheta [57] developed a fixed-to-fixed linear source code based on syndrome formation. Due to the limitations of the practical error-correcting codes known at that time, this line of research was left aside by the advent of Lempel-Ziv coding [58, 59]. Regardless of the fact that the field of data compression has reached a state of maturity, there are state-of-the-art applications that do not apply data compression, thus failing to take advantage of the source redundancy in the decoding. This is mainly due to a lack of resilience of data compressors to transmission errors and to the fact that such state-of-the-art compression algorithms just have an efficient performance with packet sizes much longer than the ones typically specified in modern wireless standards (e.g., Universal Mobile Telecommunications System) [60].

Such shortcomings together with the availability of linear codes capable of operating near the Shannon limit, most notably turbo and low-density parity-check codes, have been motivating the search for new data compression algorithms to compete with the state-of-the-art methods. The compression of binary memoryless sources using turbo codes was first addressed in [61], where the authors proposed the use of punctured turbo codes to perform near-lossless compression and JSC. Thereafter, Hagenauer et al. [62] introduced a lossless compression algorithm using the concept of *decremental* redundancy, which was then extended in [63] to include the transmission of the compressed data through a noisy channel.

An alternative approach to the source compression by means of error correcting codes is the syndrome-source compression using low-density parity-check codes together with belief propagation decoding presented in [60], which was further extended in [64] to cope with a noisy channel. In contrast to general linear codes, an LDPC code has a sparse parity-check matrix and can thus be used as a linear compressor with linear complexity in the blocklength. In addition, syndrome source-coding schemes can be naturally extended to joint source-channel encoding and decoding configurations.

One of the schemes proposed in [64] for JSC was a serial concatenation of two LDPC codes, where the outer code works as a syndrome-source compressor and the inner code as the channel code. The codeword resulting from such a concatenation is then jointly decoded using the source statistics by means of the belief propagation algorithm applied to the joint source-channel factor graph. Despite its introduction in [64], it was in [65] that this scheme was first studied for a JSC application (Caire et al. did not explore it in [64], rather focusing on the LOTUS codes introduced therein). Simulation results in [66] showed the presence of error floors in the error-rate curves, which are a consequence of the fact that some output sequences emitted by the source form error patterns that cannot be corrected by the LDPC code used as source compressor. These problems can be mitigated either by reducing the source compression rate or increasing the codeword size, but such solutions also come with some drawbacks.

First of all, increasing the size of the codeword would undermine one of the advantages of the JSC scheme, namely the possibility of a better performance in a non-asymptotic scenario. Second, reducing the compression rate is clearly also not desirable, since it pushes the system performance away from capacity. Another possible solution would be the use of the closed-loop iterative doping (CLID) algorithm in conjunction with a library of LDPC codes for source encoding [67], a solution that comes naturally at the expense of an increase of the encoding complexity. Considering the above mentioned problems and known solution options, we can now state the main goal of this chapter: the construction of an LDPC-based joint source-channel coding scheme that overcomes the complexity/performance problems of the existing JSC schemes based on syndrome-source encoding.

6.2 LDPC-based joint source-channel system

In [64], the authors proposed two configurations for a joint source-channel encoding system using LDPC codes for both source compression and channel coding. The first proposed structure was based on a serial concatenation of two LDPC codes where the outer and the inner codes perform syndrome-source compression and channel coding, respectively. The second structure was based on a single systematic LDPC code, where the source output composed the systematic part of the codeword and was punctured prior to transmission so that only the nonsystematic part was sent through the communication channel. We will concentrate our work on the concatenated structure.

In the concatenated approach, a codeword $\mathbf{c}$ was defined by

$$\mathbf{c} = \mathbf{s} \cdot \mathbf{G}_{cc} = \mathbf{u} \cdot \mathbf{H}_{sc}^{T} \cdot \mathbf{G}_{cc} \ ,$$

where $\mathbf{G}_{cc}$ is the $l \times m$ LDPC generator matrix of the channel code, $\mathbf{H}_{sc}$ is the $l \times n$ parity-check matrix of the LDPC code applied for source coding, $\mathbf{s}$ is the $1 \times l$ source compressed sequence, and $\mathbf{u}$ is the $1 \times n$ source output. The factor graph defined by such an encoding scheme is depicted in Fig. 6.1. The variable and the check nodes of the

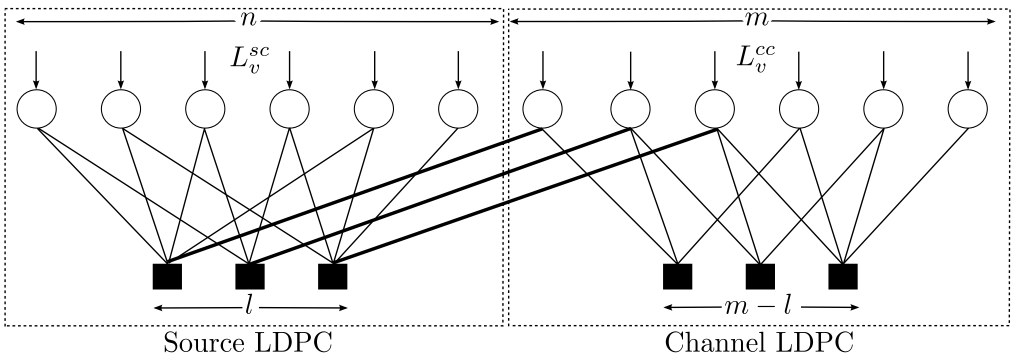

Figure 6.1: Joint source-channel factor graph.

source LDPC (left) represent the source output and the compressed source sequence, respectively. Since we will consider only binary sources, the variable nodes represent binary symbols. In this system, each check node of the source LDPC is connected to a single variable node of the channel code (right) forming the systematic part of the channel codeword (the connections are represented by bold edges). Since only m variable nodes are transmitted, the overall rate is n/m. Furthermore, L_v^{sc} and L_v^{cc} denote the log-likelihood ratios representing the intrinsic information received by the source ($v = 1, \ldots, n$) and channel ($v = n+1, \ldots, n+m$) variable nodes, respectively.

Considering a two-state Markovian source and performing standard belief propagation on the graph of Fig. 6.1, the simulation results in [66] showed the presence of error floors in the error-rate curves, which are nothing but a consequence of the fact that some output sequences emitted by the source form error patterns that cannot be corrected by the LDPC code used as source compressor. The proposed solution to cope with this residual error was either reducing the compression rate, or increasing the source output block length. As we already pointed out, these solutions are not very attractive, since the reduction of the compression rate pushes the system performance away from its asymptotic capacity, and a large block length undermines the application of the proposed JSC system for cases where state-of-the-art compression algorithms turn out to be ineffective, i.e., systems with source data divided in small block lengths.

Our idea to cope with the problem and thus generate a JSC system with competitive performance, even for small source block lengths, while keeping the advantage of the simplified syndrome-source compression is to improve the amount of information about the source bits available at the decoding by inserting new edges connecting the check nodes of the channel code to the variable nodes of the source code. We depict this idea in Fig. C.4, where the inserted edges are represented by dashed lines. The reasoning

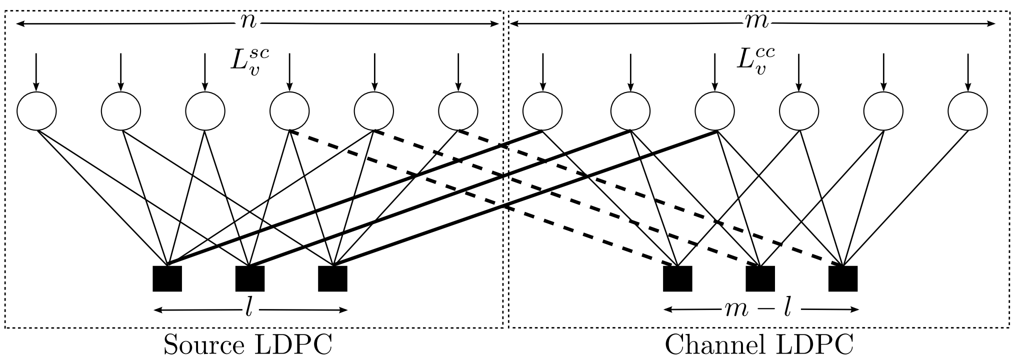

Figure 6.2: Joint source-channel factor graph with inserted edges.

of this strategy is that such an edge insertion will provide an extra amount of extrinsic information to the variable nodes of the source LDPC which will significantly lower the error floor due to uncorrectable source output patterns. We will limit our investigation to memoryless binary sources, but our system can easily be extended to sources with memory if we consider the use of the Burrows-Wheeler transform [68] prior to the syndrome-source compression as done in [67] for the case of pure data compression.

6.2.1 Encoder

To understand our proposed serial encoding strategy, consider the representation of the factor graph depicted in Fig. C.4 by an $m \times (n+m)$ matrix $\mathbf{H}$. According to this notation, we have the following matrix representation for the JSC system of Fig. C.4

$$\mathbf{H} = \left[\begin{array}{c|c|c} \mathbf{H}_{sc} & \mathbf{I} & \mathbf{0} \\ \hline \mathbf{L} & \multicolumn{2}{c}{\mathbf{H}_{cc}} \end{array}\right],$$

where $\mathbf{H}_{sc}$ is the $l \times n$ source encoder parity-check matrix, $\mathbf{H}_{cc}$ is the $(m-l) \times m$ parity-check matrix of the channel code, $\mathbf{I}$ is an $l \times l$ identity matrix, and $\mathbf{L}$ is a $(m-l) \times n$ matrix, to which we will refer as *linking* matrix. Note that for the system depicted in Fig. 6.1, $\mathbf{L} = \mathbf{0}$. The linking matrix $\mathbf{L}$ represents the connections among the check nodes of the channel code to the variable nodes of the source code.

The encoding scheme of our proposed system diverts slightly from the serial approach of [65]. The difference lies in the fact that the word to be encoded by the channel code is formed by the concatenation of the source output $\mathbf{u}$ and its syndrome $\mathbf{s}$ computed by the source code, i.e., a codeword $\mathbf{c}$ is defined by

$$\mathbf{c} = [\mathbf{u}, \mathbf{s}] \cdot \mathbf{G}_L = [\mathbf{u}, \mathbf{u} \cdot \mathbf{H}_{sc}^T] \cdot \mathbf{G}_L \ , \tag{6.1}$$

where $\mathbf{H}_{sc}$ is the $l \times n$ parity-check matrix of the LDPC code applied for source coding, $\mathbf{s}$ is the $1 \times l$ source compressed sequence, $\mathbf{u}$ is the $1 \times n$ source output, and $\mathbf{G}_L$ is a $(n+l) \times (n+m)$ matrix such that the row space of $\mathbf{G}_L$ is the null space of $[\mathbf{L}, \mathbf{H}_{cc}]$, i.e., $\mathbf{G}_L$ is the generator matrix of a linear systematic code whose parity-check matrix is given by the horizontal concatenation of the matrices $\mathbf{L}$ and $\mathbf{H}_{cc}$. In the following proposition, we show that every codeword of the code spanned by $\mathbf{G}_L$ is a codeword of the code spanned by the null space of $\mathbf{H}$, i.e.,

Proposition 1 *Let* $\mathbf{H} = \left[\ [\mathbf{H}_{sc}, \mathbf{I}, \mathbf{0}]^T, [\mathbf{L}, \mathbf{H}_{cc}]^T\ \right]^T$ *denote the matrix whose factor graph representation corresponds to the system depicted in Fig. C.4,* $\mathbf{H}_L = [\mathbf{L}, \mathbf{H}_{cc}]$*, and* $[\mathbf{u}, \mathbf{s}]$ *be the concatenation of the source output* $\mathbf{u}$ *and its syndrome-compressed sequence* $\mathbf{s}$*. A codeword* $\mathbf{c}$ *formed by the encoding of the vector* $[\mathbf{u}, \mathbf{s}]$ *by the linear code spanned by the null space of the matrix* $\mathbf{H}_L$ *is also a codeword of the linear code spanned by the null space of* $\mathbf{H}$*.*

Proof: Let $\mathbf{G}_L$ denote the systematic generator matrix of the null space of the matrix $\mathbf{H}_L$. Since the code spanned by the rows of $\mathbf{G}_L$ is systematic, its codewords can be written as $\mathbf{c} = [\mathbf{u}, \mathbf{s}, \mathbf{p}]$, where $\mathbf{u} = [u_0, u_1, \ldots, u_{n-1}]$ represents the source output, $\mathbf{s} = [s_0, s_1, \ldots, s_{l-1}]$ denotes the syndrome compressed sequence, and $\mathbf{p} = [p_0, p_1, \ldots, p_{m-l-1}]$ is a vector whose elements are the parity bits generated by the inner product between $[\mathbf{u}, \mathbf{s}]$ and $\mathbf{G}_L$. For every codeword $\mathbf{c}$, the following equation holds

$$\mathbf{c} \cdot \mathbf{H}_L^T = \mathbf{c} \cdot [\mathbf{L}, \mathbf{H}_{cc}]^T = \mathbf{0}\ . \tag{6.2}$$

Recall now that according to our compression rule, and since our operations are defined over GF(2), we can write

$$\begin{aligned} [u_0, u_1, \ldots, u_{n-1}] \cdot \mathbf{H}_{sc}^T &= [s_0, s_1, \ldots, s_{l-1}] \\ [u_0, u_1, \ldots, u_{n-1}] \cdot \mathbf{H}_{sc}^T + [s_0, s_1, \ldots, s_{l-1}] \cdot \mathbf{I} &= \mathbf{0}\ , \end{aligned} \tag{6.3}$$

where $\mathbf{I}$ is an $l \times l$ identity matrix, and $\mathbf{0}$ is a vector whose elements are all equal to zero. Note that Eq. (6.3) can be written as

$$[u_0, u_1, \ldots, u_{n-1}, s_0, s_1, \ldots, s_{l-1}] \cdot [\mathbf{H}_{sc}, \mathbf{I}]^T = \mathbf{0}\ . \tag{6.4}$$

Consider now the $l \times (n+m)$ matrix $[\mathbf{H}_{sc}, \mathbf{I}, \mathbf{0}]$. According to Eq. (6.4), for every vector $\mathbf{p} = [p_0, p_1, \ldots, p_{m-l-1}]$, we can write

$$[u_0, u_1, \ldots, u_{n-1}, s_0, s_1, \ldots, s_{l-1}, p_0, p_1, \ldots, p_{m-l-1}] \cdot [\mathbf{H}_{sc}, \mathbf{I}, \mathbf{0}]^T = \mathbf{0}\ ,$$

i.e.,

$$\mathbf{c} \cdot [\mathbf{H}_{sc}, \mathbf{I}, \mathbf{0}]^T = \mathbf{0}\ . \tag{6.5}$$

Finally, consider the inner product

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{c} \cdot \left[\ [\mathbf{H}_{sc}, \mathbf{I}, \mathbf{0}]^T, [\mathbf{L}, \mathbf{H}_{cc}]^T \ \right] = \left[\ \mathbf{c} \cdot [\mathbf{H}_{sc}, \mathbf{I}, \mathbf{0}]^T, \mathbf{c} \cdot [\mathbf{L}, \mathbf{H}_{cc}]^T \ \right] \ . \tag{6.6}$$

Substituting eqs. (6.2) and (6.5) into Eq. (6.6), we have

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0} \ ,$$

i.e., a codeword $\mathbf{c}$ of the code spanned by the null space of $\mathbf{H}_L$ is also a codeword of the code spanned by the null space of $\mathbf{H}$.

□

In our proposed system, the overall rate will be kept constant when compared to the system of Fig. 6.1, since the first n bits of $\mathbf{c}$ will be punctured prior to transmission. The encoding algorithm of our proposed joint source-channel system can be summarized as follows:

1. Given a source output vector $\mathbf{u}$, compute $\mathbf{s} = \mathbf{u} \cdot \mathbf{H}_{sc}^T$.
2. Compute $\mathbf{v} = [\mathbf{u}, \mathbf{s}]$, i.e., the horizontal concatenation of vectors $\mathbf{u}$ and $\mathbf{s}$.
3. Generate the codeword $\mathbf{c} = \mathbf{v} \cdot \mathbf{G}_L$.
4. Transmit $\mathbf{c}$ after puncturing its first n bits.

Steps 1 and 3 are the source and channel encoding steps, respectively. Since $\mathbf{H}_{sc}$ is sparse, the source encoding has a complexity that is linear with respect to the block length. Furthermore, applying the technique presented in [69] for encoding LDPC codes by means of their parity-check matrix, the complexity of the channel encoding can be made approximately linear.

6.2.2 Decoder

The decoding of the LDPC-based joint source-channel system is done by means of the belief propagation algorithm applied to the factor graph of Fig. C.4, whose structure is known to both, the encoder and the decoder. We assume that the decoder knows the statistics of the source. For example, for memoryless Bernoulli sources, the decoder knows the success probability, i.e., the decoder knows the probability p_v of a source symbol assuming the value 1.

Herein, we assume that the source is a memoryless Bernoulli source with success probability p_v, and that the transmission takes place through a binary input AWGN channel. Within this framework, we can write $L_v^{sc} = \log\left(\frac{1-p_v}{p_v}\right)$ and $L_v^{cc} = \frac{2y_v}{\sigma_n^2}$ (where y_v is the received BPSK modulated codeword transmitted through an AWGN with noise variance σ_n^2).

6.3 Multi-edge notation for joint source-channel factor graphs

The factor graph representing the joint source-channel system is composed of two separated LDPC factor graphs that exchange information. In order to combine the evolution of the iterative decoding of both source and channel codes in a single input-output function, we derive in the sequel a multi-edge representation of the joint source-channel factor graph.

In a multi-edge framework for joint source-channel factor graphs, we define four edge types within the corresponding graph, i.e., $m_e = 4$. The first edge type is composed by the edges connected solely to nodes of the source LDPC code. Similarly, the second edge type is composed of the edges connected only to nodes belonging to the channel LDPC code. The third type is formed by the edges that connect the check nodes of the source code to the variable nodes of the channel code. Finally, the edges that connect the variable nodes of the source LDPC codes to the check node of the channel LDPC factor graph compose the fourth edge type.

Note that now we also have two different received distributions corresponding to the source statistics and channel information, respectively. Figure 6.3 depicts the four edge types and received distributions. The solid and dashed lines depict type-1 and type-2 edges, respectively. The type-3 and type-4 edges are depicted by the dash-dotted and dotted lines, respectively. Additionally, the received distributions of the source and channel variable nodes are depicted by solid and dashed arrows, respectively. Since the variable nodes have access to two different observations, the vector $\mathbf{r} = (r_0, r_1, r_2)$ has three components, i.e., $m_r = 2$. The first component (r_0) represents a bit with no available intrinsic information, second component (r_1) corresponds to the observation

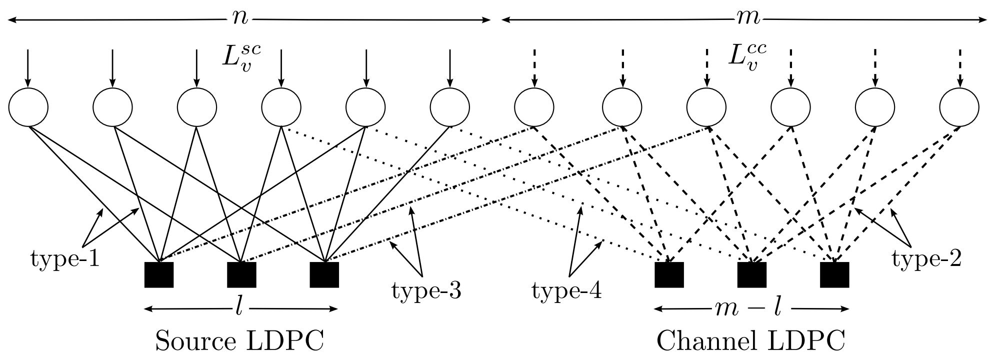

Figure 6.3: Multi-edge joint source-channel factor graph.

accessible to the n source LDPC variable nodes, and the third component (r_2) denotes the channel observations, which are available only to the m channel LDPC variable nodes. Furthermore, since each variable node has access to either the source statistics or the channel observation, we can write $\mathbf{b} = (0, 1, 0)$ for the source and $\mathbf{b} = (0, 0, 1)$ for the channel variable nodes, respectively.

As an example, consider the graph of Fig. 6.3 ($n = 6, m = 6, l = 3$). In this case, the multi-edge degree distributions can be written as

$$\nu(\mathbf{r}, \mathbf{x}) = \frac{3}{12} r_1 x_1^2 + \frac{2}{12} r_1 x_1^2 x_4 + \frac{1}{12} r_1 x_1 x_4 + \frac{1}{12} r_2 x_2 x_3 + \frac{2}{12} r_2 x_2^2 x_3 + \frac{3}{12} r_2 x_2^2 ,$$

$$\mu(\mathbf{x}) = \frac{2}{12} x_1^4 x_3 + \frac{1}{12} x_1^3 x_3 + \frac{1}{12} x_2^3 x_4 + \frac{2}{12} x_2^4 x_4 .$$

6.4 Asymptotic analysis

In this section, we derive the multi-edge-type mutual information evolution equations for LDPC-based joint source-channel coding systems. As previously done, we will use the edge-perspective degree distributions $\lambda^{(j)}(\mathbf{r}, \mathbf{x})$ and $\rho^{(j)}(\mathbf{x})$ to describe the evolution of the mutual information between the messages sent through type-j edges and the associated variable node values. Recall that,

$$\lambda^{(j)}(\mathbf{r}, \mathbf{x}) = \frac{\nu_{x_j}(\mathbf{r}, \mathbf{x})}{\nu_{x_j}(\mathbf{1}, \mathbf{1})} , \tag{6.7}$$

$$\rho^{(j)}(\mathbf{x}) = \frac{\mu_{x_j}(\mathbf{x})}{\mu_{x_j}(\mathbf{1})} , \tag{6.8}$$

where $\nu_{x_j}(\mathbf{r},\mathbf{x})$ and $\mu_{x_j}(\mathbf{x})$ are the derivatives of $\nu(\mathbf{r},\mathbf{x})$ and $\mu(\mathbf{x})$ with respect to x_j, respectively.

Before proceeding to the asymptotic analysis, it is worth mentioning an important result present in [57]. In this work, the author associates to any binary source an *additive channel* in which the source output forms the error pattern. Furthermore, he shows that the average fraction of source digits erroneously reconstructed for syndrome-source-coding of a binary source coincides with the bit error probability when the corresponding syndrome decoder is used with the given linear code on the additive channel associated with the source.

For a memoryless Bernoulli source[1] with a probability of emitting a one equal to p_v, the associated additive channel is a BSC with crossover probability p_v. This means that we can model the received distributions of the source code variable nodes as the distribution of the output of a BSC with crossover probability p_v.

Let $I_{v,l}^{(j)}$ ($I_{c,l}^{(j)}$) denote the mutual information between the messages sent through type-j edges at the output of variable (check) nodes at iteration l and the associated variable node value. Due to the fact that the source and channel variable nodes have channel observations with different distributions, we will describe the mutual information equations for source and channel LDPC multi-edge variable nodes, separately.

Following the notation of [65] we can write for the source code variable nodes, i.e., for $j \in \{1,4\}$

$$I_{v,l}^{(j)} = \sum_{\mathbf{d}} \lambda_{\mathbf{d}}^{(j)} J_{BSC}\left((d_j - 1)[J^{-1}(I_{c,l-1}^{(j)})]^2 + \sum_{s\neq j} d_s [J^{-1}(I_{c,l-1}^{(s)})]^2, p_v \right) , \tag{6.9}$$

where $\lambda_{\mathbf{d}}^{(j)}$ is the probability of a type-j edge being connected to a variable node with edge degree vector $\mathbf{d}$. The function J_{BSC} is given by [65]

$$J_{BSC}(\sigma^2, p_v) = (1-p_v) I(x_v; \mathcal{L}^{(1-p_v)}) + p_v I(x_v; \mathcal{L}^{(p_v)}) , \tag{6.10}$$

where x_v denotes the corresponding bitnode variable, $\mathcal{L}^{(1-p_v)} \sim \mathcal{N}(\frac{\sigma^2}{2} + L_v^{sc}, \sigma^2)$, and $\mathcal{L}^{(p_v)} \sim \mathcal{N}(\frac{\sigma^2}{2} - L_v^{sc}, \sigma^2)$.

The derivation of the function $J_{BSC}(\cdot)$ can be easily understood if we recall that under

[1] We assume a memoryless Bernoulli source with output alphabet $\mathcal{X} = \{0,1\}$.

the Gaussian approximation and for infinitely long codes, the variance of the outgoing message of a degree-d_v variable node can be written as $\sigma_v^2 = \sigma_{ch}^2 + (d_v - 1)\sigma_r^2$, where σ_{ch}^2 is the variance of the received channel message, and σ_r^2 is the variance of the messages sent by the neighboring check nodes. Furthermore, the messages sent from the neighboring check nodes are considered to be independent and Gaussian. Within this assumption, the sum of the $d_v - 1$ messages sent from the check nodes is approximated by a Gaussian with mean $\sigma^2/2$ and variance σ^2.

Consequently, since the equivalent channel for the source code variable nodes is a BSC with crossover probability p_v, the distribution of the messages $q_{v\to c}$ sent from the source code variable nodes will be a mixture of two Gaussian distributions, i.e.,

$$q_{v\to c} \sim (1 - p_v)\mathcal{N}(\frac{\sigma^2}{2} + L_v^{sc}, \sigma^2) + p_v\mathcal{N}(\frac{\sigma^2}{2} - L_v^{sc}, \sigma^2) \ ,$$

and the mutual information $I(x_v; q_{v\to c})$ can be written as in Eq. (6.10), which does not have a closed form, but can be numerically computed recalling that

$$I(x_v; \mathcal{L}) = 1 - E[\log_2(1 + e^{-\mathcal{L}})] \ , \tag{6.11}$$

and that for a Gaussian random variable $x \sim \mathcal{N}(\frac{\sigma^2}{2} + a, \sigma^2)$

$$E[\log_2(1 + e^x)] = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \log_2(1 + e^y) e^{\frac{-(y - \frac{\sigma^2}{2} - a)^2}{2\sigma^2}} dy \ . \tag{6.12}$$

For the channel code, the multi-edge mutual information evolution equations are derived in the same way as done for multi-edge UEP LDPC codes in Chapter 4. This leads to the following mutual information equation for the channel code variable nodes, i.e., for $j \in \{2, 3\}$ we can write

$$I_{v,l}^{(j)} = \sum_{\mathbf{d}} \lambda_{\mathbf{d}}^{(j)} J\left(\sqrt{4/\sigma_n^2 + (d_j - 1)[J^{-1}(I_{c,l-1}^{(j)})]^2 + \sum_{s\neq j} d_s [J^{-1}(I_{c,l-1}^{(s)})]^2}\right) \ . \tag{6.13}$$

Finally, the mutual information between the messages sent by a check node through a type-j edge and its associated variable value for both source and channel LDPC codes (i.e., for all j) can be written as

$$I_{c,l}^{(j)} = 1 - \sum_{i=1}^{d_{cmax}^{(j)}} \sum_{\mathbf{d}:d_j=i} \rho_{\mathbf{d}}^{(j)} J\left(\sqrt{(d_j-1)[J^{-1}(1-I_{v,l}^{(j)})]^2 + \sum_{s\neq j} d_s[J^{-1}(1-I_{v,l}^{(s)})]^2}\right), \tag{6.14}$$

where $\rho_{\mathbf{d}}^{(j)}$ is the probability of a type-j edge being connected to a check node with edge degree vector $\mathbf{d}$, and $d_{cmax}^{(j)}$ is the maximum number of type-j edges connected to a check node.

In order to limit the search space of the forthcoming optimization algorithm, we consider in the following derivations that both source and channel LDPC codes are check-regular. Furthermore, the check nodes of source and channel LDPC codes are considered to have edge degree vectors $\mathbf{d} = (d_{c_1}, 0, 1, 0)$ and $\mathbf{d} = (0, d_{c_2}, 0, 1)$, respectively. As a consequence, the multi-edge check node degree distributions of the source and channel LDPC codes are given by $\rho^{(1)}(\mathbf{x}) = x_1^{d_{c_1}-1}$ and $\rho^{(2)}(\mathbf{x}) = x_2^{d_{c_2}-1}$, respectively.

In our multi-edge representation for the channel LDPC factor graph, the variable nodes only have connections to type-2 and type-3 edges, i.e., all channel code variable nodes have an edge degree vector $\mathbf{d} = (0, d_2, d_3, 0)$ where $d_2 \in \{2, \ldots, d_{vmax}^{(2)}\}$, and $d_3 \in \{0, 1\}$. Thus, for the channel LDPC code, we can summarize the set of mutual information evolution equations as follows:

- variable nodes messages update:

$$I_{v,l}^{(2)}(\mathbf{d}) = J\left(\sqrt{4/\sigma_n^2 + (d_2-1)[J^{-1}(I_{c,l-1}^{(2)}(\mathbf{d}))]^2 + d_3[J^{-1}(I_{c,l-1}^{(3)}(\mathbf{d}))]^2}\right) \tag{6.15}$$

$$I_{v,l}^{(2)} = \sum_{\mathbf{d}} \lambda_{\mathbf{d}}^{(2)} J\left(\sqrt{4/\sigma_n^2 + (d_2-1)[J^{-1}(I_{c,l-1}^{(2)}(\mathbf{d}))]^2 + d_3[J^{-1}(I_{c,l-1}^{(3)}(\mathbf{d}))]^2}\right) \tag{6.16}$$

- check nodes messages update:

$$I_{c,l}^{(2)}(\mathbf{d}) = 1 - J\left(\sqrt{(d_{c_2}-1)[J^{-1}(1-I_{v,l}^{(2)})]^2 + J^{-1}(1-I_{v,l}^{(4)}(\mathbf{d}))^2}\right) \tag{6.17}$$

- channel to source decoder messages update:

$$I_{v,l}^{(3)}(\mathbf{d}) = d_3 \cdot J\left(\sqrt{4/\sigma_n^2 + d_2[J^{-1}(I_{c,l-1}^{(2)}(\mathbf{d}))]^2}\right) \tag{6.18}$$

$$I_{c,l}^{(4)} = 1 - J\left(\sqrt{d_{c_2}[J^{-1}(1 - I_{v,l}^{(2)})]^2}\right) \tag{6.19}$$

- source decoder messages update:

$$I_{v,l}^{(4)}(\mathbf{d}) = T_1(I_{c,l-1}^{(4)}, I_{v,l-1}^{(3)}(\mathbf{d})) \tag{6.20}$$

$$I_{c,l}^{(3)}(\mathbf{d}) = T_2(I_{c,l}^{(4)}, I_{v,l}^{(3)}(\mathbf{d})) \tag{6.21}$$

where $T_1(\cdot)$ and $T_2(\cdot)$ are the transfer functions of the source decoder. Recall that we are considering here that the source decoder is fixed. Given the source code degree distributions $\lambda^{(1)}(\mathbf{r}, \mathbf{x})$ and $\rho^{(1)}(\mathbf{x})$, those functions can be explicitly computed by means of eqs. (6.9) and (6.14) for every edge degree vector $\mathbf{d}$.[2] It is worth noting that in the computation of $T_1(\cdot)$ and $T_2(\cdot)$, the rightmost sum in Eq. (6.14) will be zero if $I_{v,l}^{(3)}(\mathbf{d}) = 0$, since the corresponding check node is not receiving any information through type-3 edges in this case.

Combining eqs. (6.16) - (6.21), we can summarize the mutual information evolution for the channel code as

$$I_{v,l}^{(2)} = F_2(\underline{\lambda}, \underline{d_c}, I_{v,l-1}^{(2)}, p_v, \sigma_n) , \tag{6.22}$$

where $\underline{d_c} = [d_{c_1}, d_{c_2}]$, and $\underline{\lambda} = [\underline{\lambda}^{(1)}, \underline{\lambda}^{(2)}]$ with $\underline{\lambda}^{(j)}$ denoting the sequence of coefficients $\lambda_{\mathbf{d}}^{(j)}$ for all $\mathbf{d}$ and $j \in \{1, 2\}$. The initial conditions are $I_{v,0}^{(3)}(\mathbf{d}) = I_{c,0}^{(2)}(\mathbf{d}) = I_{c,0}^{(3)}(\mathbf{d}) = 0, \forall\, \mathbf{d}$, and $I_{c,0}^{(4)} = 0$.

For the source code factor graph, the variable nodes only have connections to type-1 and type-4 edges, i.e., all source code variable nodes have an edge degree vector $\mathbf{d} = (d_1, 0, 0, d_4)$ where $d_1 \in \{2, \ldots, d_{v_{max}}^{(1)}\}$, and $d_4 \in \{0, 1\}$. Similar to the channel code factor graph, we can summarize the set of mutual information evolution equations as follows:

[2]For the computation of $T_1(\cdot)$, note that by means of eqs. (2.23) and (6.7) we can write $\lambda_{\mathbf{d}}^{(4)}(\mathbf{r}, \mathbf{x}) = \left[\frac{\int \lambda_{\mathbf{d}}^{(1)}(\mathbf{r}, \mathbf{x})}{\int_0^1 \lambda_{\mathbf{d}}^{(1)}(\mathbf{r}, \mathbf{x})}\right]'_{x_4}$, where f'_x denotes the partial derivative of f with respect to x.

- variable nodes messages update:

$$I_{v,l}^{(1)}(\mathbf{d}) = J_{BSC}\left((d_1-1)[J^{-1}(I_{c,l-1}^{(1)}(\mathbf{d}))]^2 + d_4[J^{-1}(I_{c,l-1}^{(4)}(\mathbf{d}))]^2, p_v\right) \quad (6.23)$$

$$I_{v,l}^{(1)} = \sum_{\mathbf{d}} \lambda_{\mathbf{d}}^{(1)} J_{BSC}\left((d_1-1)[J^{-1}(I_{c,l-1}^{(1)}(\mathbf{d}))]^2 + d_4[J^{-1}(I_{c,l-1}^{(4)}(\mathbf{d}))]^2, p_v\right) \quad (6.24)$$

- check nodes messages update:

$$I_{c,l}^{(1)}(\mathbf{d}) = 1 - J\left(\sqrt{(d_{c_1}-1)[J^{-1}(1-I_{v,l}^{(1)})]^2 + [J^{-1}(1-I_{v,l}^{(3)}(\mathbf{d}))]^2}\right) \quad (6.25)$$

- source to channel decoder messages update:

$$I_{v,l}^{(4)}(\mathbf{d}) = d_4 \cdot J_{BSC}\left(d_1[J^{-1}(I_{c,l-1}^{(1)}(\mathbf{d}))]^2, p_v\right) \quad (6.26)$$

$$I_{c,l}^{(3)} = 1 - J\left(\sqrt{d_{c_1}[J^{-1}(1-I_{v,l}^{(1)})]^2}\right) \quad (6.27)$$

- channel decoder messages update:

$$I_{v,l}^{(3)}(\mathbf{d}) = T_4(I_{c,l-1}^{(3)}, I_{v,l-1}^{(4)}(\mathbf{d})) \quad (6.28)$$

$$I_{c,l}^{(4)}(\mathbf{d}) = T_5(I_{c,l}^{(3)}, I_{v,l}^{(4)}(\mathbf{d})) \quad (6.29)$$

where $T_4(\cdot)$ and $T_5(\cdot)$ are the transfer functions of the channel decoder, which is considered to be fixed. Given the channel code degree distribution $\lambda^{(2)}(\mathbf{r},\mathbf{x})$ and $\rho^{(2)}(\mathbf{x})$, those functions can be explicitly computed by means of eqs. (6.13) and (6.14) for every edge degree vector $\mathbf{d}$.[3] Similarly to the channel code, in the computation of $T_4(\cdot)$ and $T_5(\cdot)$, the rightmost sum in Eq. (6.14) will be zero if $I_{v,l}^{(4)}(\mathbf{d}) = 0$, since the corresponding check node is not receiving any information through type-4 edges in this case.

Combining eqs. (6.24) - (6.29) we can summarize the mutual information evolution for the source code as

$$I_{v,l}^{(1)} = F_1(\underline{\lambda}, \underline{d_c}, I_{v,l-1}^{(1)}, p_v, \sigma_n) \,, \quad (6.30)$$

where $\underline{d_c} = [d_{c_1}, d_{c_2}]$, and $\underline{\lambda} = [\underline{\lambda}^{(1)}, \underline{\lambda}^{(2)}]$ with $\underline{\lambda}^{(j)}$ denoting the sequence of coefficients

[3]For the computation of $T_4(\cdot)$, note that by means of eqs. (2.23) and (6.7) we can write $\lambda_{\mathbf{d}}^{(3)}(\mathbf{r},\mathbf{x}) = \left[\frac{\int \lambda_{\mathbf{d}}^{(2)}(\mathbf{r},\mathbf{x})}{\int_0^1 \lambda_{\mathbf{d}}^{(2)}(\mathbf{r},\mathbf{x})}\right]'_{x_3}$, where f'_x denotes the partial derivative of f with respect to x.

$\lambda_{\mathbf{d}}^{(j)}$ for all $\mathbf{d}$ and $j \in \{1, 2\}$. The initial conditions are $I_{v,0}^{(4)}(\mathbf{d}) = I_{c,0}^{(4)}(\mathbf{d}) = I_{c,0}^{(1)}(\mathbf{d}) = 0, \forall\, \mathbf{d}$ and $I_{c,0}^{(3)} = 0$.

By means of eqs. (6.22) and (6.30), we can predict the convergence behavior of the iterative decoding for both channel and source codes and then optimize the multi-edge edge-perspective variable node degree distributions $\lambda^{(1)}(\mathbf{r}, \mathbf{x})$ and $\lambda^{(2)}(\mathbf{r}, \mathbf{x})$ under the constraint that the mutual information for both codes must be increasing as the number of iterations grows.

6.5 Optimization

Having derived the mutual information evolution equations, we are now able to present an optimization algorithm derived to maximize the overall rate of the proposed JSC code. Optimization strategies for LDPC-based JSC schemes present in the literature either consider full knowledge of the channel code [65] or of the source decoder [70] (where the authors also showed that an LDPC code optimized for the AWGN is not necessarily optimum for the JSC problem).

In this section, we introduce an optimization algorithm that takes into account the extra connections between the factor graphs of the source and channel codes we proposed previously. By means of a multi-edge-type framework, the algorithm presented herein extends the optimization technique for LDPC-based JSC coding schemes presented in the literature for the case where the source code variable nodes (and not only the check nodes) are connected to the channel LDPC code factor graph.

In our proposed algorithm, we first compute the rate optimal channel LDPC code assuming that the transmission is carried over an AWGN channel with noise variance σ_n^2. This is a standard irregular LDPC optimization [24] and since we are not considering any connection to the source code in this first step, it can be done by means of eqs. (6.13) and (6.14) with $\mathbf{d} = (0, d_2, 0, 0)$ and $d_2 \in \{2, \ldots, d_{v_{max}}^{(2)}\}$, where $d_{v_{max}}^{(j)}$ denotes the maximum number of type-j edges connected to a variable node. The optimized degree distribution obtained at this step will be denoted as $\lambda_0^{(2)}(\mathbf{r}, \mathbf{x})$.

After having optimized the channel code variable nodes degree distribution, we assign the variable nodes of higher degree to the message bits. This is done in order to better protect the compressed message transmitted through the channel, since the more connected a variable node is, the better its error rate performance. This can be done as follows,

1. Given $\lambda_0^{(2)}(\mathbf{r},\mathbf{x})$, compute the node-perspective multi-edge degree distribution $\nu_0(\mathbf{r},\mathbf{x}) = \frac{\int \lambda_0^{(2)}(\mathbf{r},\mathbf{x})\mathrm{d}x_2}{\int_0^1 \lambda_0^{(2)}(\mathbf{r},\mathbf{x})\mathrm{d}x_2}$.

2. Assign a fraction R_{cc} of nodes (the ones with higher degree) to the systematic part of the codeword, where R_{cc} is the rate of the channel code. This is done by turning a variable node with edge degree vector $\mathbf{d} = (0, d_2, 0, 0)$ into a variable node with edge degree vector $\mathbf{d} = (0, d_2, 1, 0)$. This gives rise to a modified node-perspective degree distribution $\nu(\mathbf{r},\mathbf{x})$, where a fraction of R_{cc} nodes have one connection to type-3 edges.

3. Given $\nu(\mathbf{r},\mathbf{x})$, compute the new edge-perspective multi-edge variable node degree distribution $\lambda^{(2)}(\mathbf{r},\mathbf{x}) = \frac{\nu_{x_2}(\mathbf{r},\mathbf{x})}{\nu_{x_2}(\mathbf{1},\mathbf{1})}$.

Once we have optimized the channel code, we optimize (maximizing its rate) the source LDPC code considering its connections to the channel LDPC code graph.

Let $\underline{d_{v_{max}}} = [d_{v_{max}}^{(1)}, d_{v_{max}}^{(2)}, d_{v_{max}}^{(3)}, d_{v_{max}}^{(4)}]$ be a vector whose components $d_{v_{max}}^{(j)}$ represent the maximum number of connections of a single variable node to type-j edges. Also, recall that the components of the vector $\underline{d_c} = [d_{c_1}, d_{c_2}]$ define the number of connections of the source code check nodes to type-1 edges (d_{c_1}) and the number of connections of the channel code check nodes to type-2 edges (d_{c_2}). Additionally, $\underline{\lambda^{(j)}}$ denote the sequence of the coefficients of $\lambda^{(j)}(\mathbf{r},\mathbf{x})$. Given $\underline{d_{v_{max}}}$, $\underline{d_c}$, p_v, and σ_n, the optimization problem can be written as shown in Algorithm 5.

Since we are considering the convergence only through edges of type-1, the stability condition $\mathcal{C}_3$ remains the same as for non-multi-edge LDPC codes ensembles with codewords transmitted over a BSC with transition probability p_v. Furthermore, the rate constraint $\mathcal{C}_4$ must be considered due to the fact that the number of type-4 edges connected to the source code variable nodes must be equal to the number of channel code check nodes.

For given $\underline{\lambda^{(2)}}$, $\underline{d_c}$, p_v, and σ_n, the constraints $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$ are linear in the parameter $\underline{\lambda^{(1)}}$. This means that the optimization of both source and channel codes can be solved by linear programming. For a given channel condition, every different set of vectors $\underline{d_{v_{max}}}$, $\underline{d_c}$ will give rise to systems with a different overall rate. In practice, we fix the vector $\underline{d_{v_{max}}}$ and vary d_{c_1} and d_{c_2} in order to obtain the joint system with

Algorithm 5 Joint source-channel code optimization

1. Optimize the rate of the channel LDPC code without considering the connections to the factor graph of the source LDPC code. Save the obtained the degree distribution $\lambda_0^{(2)}(\mathbf{r},\mathbf{x})$.
2. Compute $\lambda^{(2)}(\mathbf{r},\mathbf{x})$ by assigning as systematic bits a fraction R_{cc} of the variable nodes with higher degrees of the optimized channel LDPC code.
3. Considering $\underline{\lambda} = [\underline{\lambda^{(1)}}, \underline{\lambda^{(2)}}]$, maximize $\sum_{s=2}^{d_{v_{max}}^{(1)}} \sum_{\mathbf{d}:d_1=s} \lambda_{\mathbf{d}}^{(1)}/s$ under the following constraints,

   $\mathcal{C}_1$ (proportion constraint): $\sum_d \lambda_{\mathbf{d}}^{(1)} = 1$,

   $\mathcal{C}_2$ (convergence constraint): $F_1(\underline{\lambda}, \underline{d_c}, I, p_v, \sigma_n) > I$, $\forall\, I \in [0,1)$,

   $\mathcal{C}_3$ (stability constraint): $\sum_{\mathbf{d}:d_1=2} \lambda_{\mathbf{d}}^{(1)} < \frac{1}{2\sqrt{p_v(1-p_v)}} \cdot \frac{1}{(d_{c_1}-1)}$,

   $\mathcal{C}_4$ (rate constraint): $\sum_{\mathbf{d}:d_4>0} \frac{\lambda_{\mathbf{d}}^{(1)}}{d_1} = 1/(d_{c_1} d_{c_2} \sum_{\mathbf{d}:d_3=1} \frac{\lambda_{\mathbf{d}}^{(2)}}{d_2})$.

maximum overall rate for a binary symmetric source with transition probability p_v and a channel noise variance σ_n^2.

6.6 Simulation results

In this section, we present simulation results obtained with an LDPC-based JSC coding system constructed according to the degree distributions optimized by the algorithm previously proposed. We optimized a system with the following parameters: $p_v = 0.03$, $\sigma_n^2 = 0.95$, $\underline{d_{v_{max}}} = [30, 30, 1, 1]$, and $\underline{d_c} = [22, 6]$. The compression rate obtained for the source code was $R_{sc} = 0.2361$, and the transmission rate obtained for the channel LDPC code was $R_{cc} = 0.4805$. This gives an overall coding rate $R_{over} \simeq 2.03$. Note that the value of p_v was chosen in order to allow a comparison with the results presented in [65] and [66]. The resulting multi-edge distributions are given in Tables 6.1 and 6.2.

Table 6.1: Optimized multi-edge variable node degree distribution for type-1 edges.

$\mathbf{d}$	(2,0,0,0)	(2,0,0,1)	(3,0,0,0)	(9,0,0,0)	(10,0,0,0)	(30,0,0,0)
$\lambda_{\mathbf{d}}^{(1)}$	0.034955	0.098275	0.22059	0.20734	0.22014	0.21870

Table 6.2: Optimized multi-edge variable node degree distribution for type-2 edges.

$\mathbf{d}$	(0,2,0,0)	(0,2,1,0)	(0,3,1,0)	(0,7,1,0)	(0,8,1,0)
$\lambda_{\mathbf{d}}^{(2)}$	0.33334	0.005203	0.31028	0.23786	0.11332

In order to show the merits of the proposed optimization, we compare its performance with two LDPC-based JSC systems with the same overall rate $R_{over} = 2.03$ to which we will refer as systems I and II. Those two systems have only the connections between the check nodes of the source LDPC code and the systematic variable nodes of the channel LDPC code as depicted in Fig. 6.1. For System I, the source and channel LDPC codes were optimized separately for the BSC and the AWGN, respectively. System II consists of a source code jointly optimized with a fixed channel code previously optimized for the AWGN channel as done in [66]. All the performance curves were obtained considering BPSK modulated signal transmitted over an AWGN channel and a total of 50 decoding iterations.

The simulation results for the three systems with a source message of length $n = 3200$ are depicted in Fig. 6.4. The results for System I confirm that codes individually optimized do not have a good performance for the JSC system. System II shows some improvement of the bit error rate by means of the joint optimization of the source and channel LDPC codes, but still shows an error floor for high SNR's. As discussed before, this error floor is a consequence of the compression of source messages that form error patterns not correctable by the source LDPC code. Figure 6.4 shows that by means of our proposed system (depicted as JSC opt), we managed to significantly lower this error floor.

As a second set of simulations, we compare the results of our previously optimized code (whose degree distributions are shown in Tables 6.1 and 6.2) for $n = 3200$ and $n = 6400$. Furthermore, we design a code with $\underline{d_{v_{max}}} = [30, 30, 1, 1]$, $\underline{d_c} = [10, 6]$, $p_v = 0.03$, and $\sigma_n^2 = 0.95$ to which we will refer as System III. The simulation results for such systems are shown in Fig. 6.5, where it can be recognized that the error floor caused by uncorrectable error patterns can be further lowered by increasing the codeword size or lowering the compression rate. The simulation for System III considered a block size of $n = 3200$. Tables 6.3 and 6.4 show the resulting degree distributions for System III, which has an overall rate $R_{over} \simeq 1.76$.

As previously pointed out, lowering the compression rate pushes the overall rate further away from capacity, which can be expressed as $C/H(S)$, where C denotes the capacity

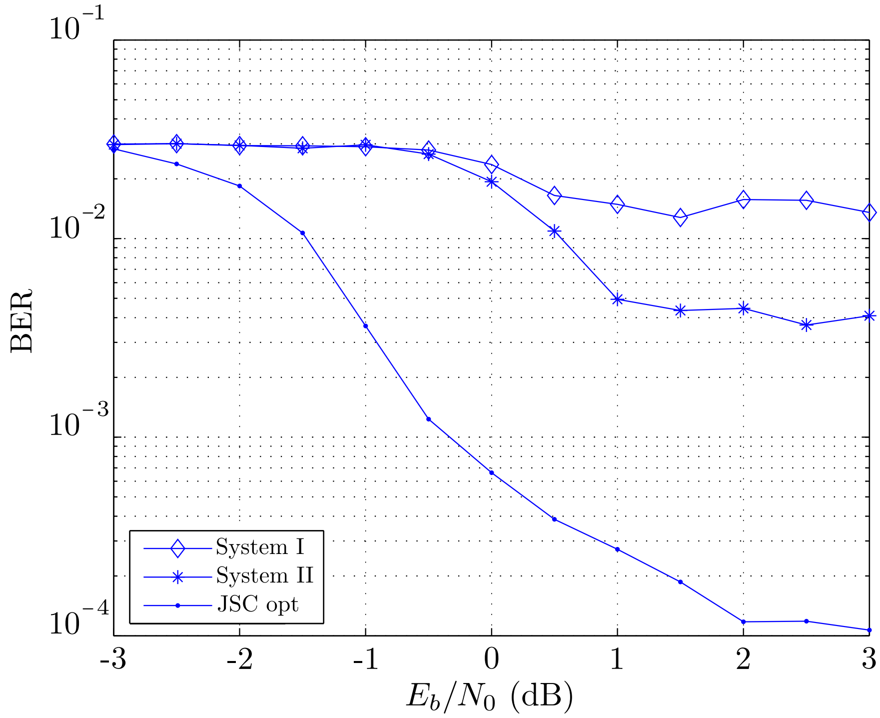

Figure 6.4: Performance of joint source-channel coded systems for $n = 3200$ and $R_{over} = 2.03$.

Table 6.3: Optimized multi-edge variable node degree distribution for type-1 edges.

$\mathbf{d}$	(2,0,0,0)	(2,0,0,1)	(3,0,0,0)	(5,0,0,0)
$\lambda_{\mathbf{d}}^{(1)}$	0.076899	0.21621	0.58665	0.12024

Table 6.4: Optimized multi-edge variable node degree distribution for type-2 edges.

$\mathbf{d}$	(0,2,0,0)	(0,2,1,0)	(0,3,1,0)	(0,7,1,0)	(0,8,1,0)
$\lambda_{\mathbf{d}}^{(2)}$	0.33334	0.005203	0.31028	0.23786	0.11332

of the transmission channel and $H(S)$ denotes the entropy of the source. For the source and channel parameters we used in our optimization, i.e., $p_v = 0.03$ and $\sigma_n^2 = 0.95$, the asymptotically optimal Shannon limit is $C/H(S) \simeq 2.58$ source symbols per channel use.

A possible strategy to enhance the performance of our proposed system is to place infinite reliability on some of the variable nodes (shortening) [71, 72]. The use of this technique should be a matter of further research in order to approach the JSC system capacity more closely without having to increase n or decrease the overall transmission rate. Nevertheless, our results already show a considerable enhancement of the bit-error rate performance when compared with existing LDPC-based JSC systems.

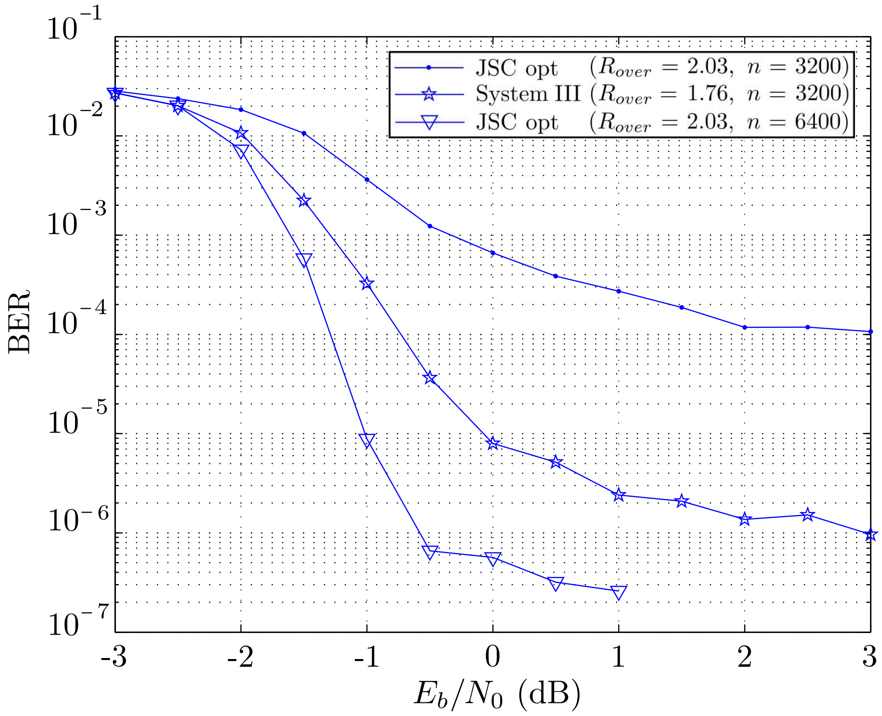

Figure 6.5: Performance curves of joint source-channel coded systems with different overall rates and input block sizes.

Chapter 7

Concluding Remarks

We investigated the asymptotic analysis and design of modern coding schemes for systems with unequal-error-protection requirements and joint source-channel coding applications. Regarding unequal error protection, we started with an asymptotic analysis of hybrid turbo codes and then studied low-density parity-check and LT codes proposing optimization algorithms to enhance the unequal-error-protecting capabilities of both schemes by means of a multi-edge framework. Lastly, an LDPC-based joint source-channel coding system was studied. In the following, we summarize the contributions of the thesis and point out some possible future research topics.

As a first contribution, we derived the construction of local EXIT charts for a hybrid concatenation of convolutional codes, which we named hybrid turbo codes and showed the relation between local and global EXIT charts. Furthermore, we pointed out that by means of this derived relation between local and global convergence, the analysis of the global system can be reduced to the study of a serial concatenated code, since the convergence behavior of the global system can be predicted from the local (serial) EXIT chart.

Afterwards, we performed a multi-edge-type analysis of unequal-error-protecting LDPC codes. By means of such an analysis, we derived an optimization algorithm that aims at optimizing the connection profile between the protection classes defined within a codeword of a given UEP-LDPC code. This optimization allowed us not only to control the differences in the performances of the protection classes by means of a single parameter, the interclass connection vector. It also allowed us to design codes with a non-vanishing UEP capability when a moderate to large number of decoding iterations is applied. Finally, the optimization algorithm introduced herein has the ability to generate UEP-LDPC codes with superior performance for applications where a low or high number of decoding iterations is needed.

Further research in this area might be the investigation of unequal-error-protecting LDPC codes with more than three protection classes defined within a codeword. Also, we restricted our optimizations to check regular LDPC codes. It would be interesting to consider irregular check nodes to verify if extra gains in the error rate performance are achievable applying our proposed optimization.

As a last investigation subject on unequal-error-protecting schemes, we introduced a multi-edge type analysis of unequal-error-protecting LT codes. Furthermore, we derived the density evolution equations for UEP LT codes, analyzed two of the existing techniques for generating UEP LT codes, and proposed a third scheme called flexible UEP LT approach. Finally, we showed by means of simulation that our proposed codes perform better than existing schemes for high overheads and have advantages for applications where precoding is needed, e.g., Raptor codes, since it only uses one precoding for the whole data block avoiding finite-length effects that can arise from separately encoding protection classes with a low number of bits.

A possible extension of the work done on unequal-error-protecting LT codes would be to modify the optimization algorithm in order to optimize the codes according to the protection requirements of each individual protection class, i.e., the bit error rate required for each class would be a parameter of the optimization.

Lastly, we proposed an LDPC-based joint source-channel coding scheme and by means of the multi-edge analysis previously developed for LDPC codes, proposed an optimization algorithm for such systems. Based on a syndrome source-encoding idea, we presented a novel system where the amount of information about the source bits available at the decoder is increased by improving the connection profile between the factor graphs of the source and channel codes that compose the joint system. The presented simulation results show a significant reduction of the error floor caused by the encoding of messages that correspond to uncorrectable error patterns of the LDPC code used as source encoder in comparison to existent LDPC-based joint source-channel coding systems.

This topic offers interesting further research possibilities. One possibility is the construction of an unequal-error-protecting JSC system applying UEP LDPC codes as syndrome-based source compressors. Another possible research topic is the improvement of the performance of our proposed JSC system by lowering the probability of an uncorrectable source pattern using shortening, which means placing infinite reliability on some source LDPC variable nodes. Those infinite reliability nodes are to be punctured prior to the transmission in order to keep the compression rate unchanged, and they have their positions known by both encoder and decoder.

Finally, an iterative optimization of the component codes of the herein proposed JSC system can be developed considering at the initial iteration that one of the component LDPC codes is fixed (we can for example take an LDPC code optimized for the AWGN as channel code), and then optimize the other following the standard approach of computing the extrinsic information transfer chart. In the next iteration, the code previously optimized is fixed and the optimization of the other component code is carried out.

Appendix A

List of Mathematical Symbols

α_j	fraction of input symbols within protection class j
$\mathbf{b} = (b_0, b_1, \ldots, b_{m_r})$	received degree vector
β_j	average number of type-j edges connected to a variable node
$\mathbf{c}$	channel encoder output vector
$\mathbf{C}(D)$	convolutional encode output sequence
C_j	protection class j
D	discrete delay operator
d_v	regular LDPC variable node degree
$d_{v_{max}}$	maximum variable node degree
d_c	regular LDPC check node degree
$d_{c_{max}}$	maximum check node degree
$\mathbf{d} = (d_1, d_2, \ldots, d_{m_e})$	edge degree vector
$\boldsymbol{\delta}^{(j)} = (\delta_1^j, \delta_2^j, \ldots, \delta_j^j)$	interclass connection vector of protection class j
$\mathbf{e}$	error vector
ϵ	erasure probability of a binary erasure channel
$\mathbf{G}$	generator matrix
$\mathbf{G}(D)$	convolutional code generator matrix
γ	LT code overhead
$\Gamma(x)$	window selection degree distribution
Γ_i	probability of window i being selected
$\mathbf{H}$	parity-check matrix
k	number of code bits at the encoder input
$\Lambda(x)$	node perspective variable node degree distribution
Λ_i	number of variable nodes with degree i
$\tilde{\lambda}(x)$	normalized node perspective variable node degree distribution

$\lambda(x)$	normalized edge perspective variable node degree distribution
λ_i	fraction of edges connected to degree-i variable nodes
$\lambda^{(j)}(\mathbf{r},\mathbf{x})$	edge perspective multi-edge variable node degree distribution of type-j edges
$\lambda_{\mathbf{d}}^{(j)}$	fraction of type-j edges connected to type-$\mathbf{d}$ variable nodes
M	memory of a convolutional code
m_e	number of edge types
m_r	number of received channel distributions
$\mu(\mathbf{x})$	multi-edge node perspective check node degree distribution
$\mu_{\mathbf{d}}$	fraction of check nodes of type $\mathbf{d}$
$\mathcal{M}(v)$	neighborhood of a variable node v
$\mathcal{N}(c)$	neighborhood of a check node c
N_c	number of protection classes
n	number of code bits at the encoder output
$n_{it,g}$	number of global decoding iterations
$n_{it,j}$	number of decoding iterations for branch j
$\nu(\mathbf{r},\mathbf{x})$	node perspective multi-edge variable node degree distribution
$\nu_{\mathbf{r},\mathbf{d}}$	fraction of variable nodes of type $(\mathbf{b},\mathbf{d})$
$P(x)$	node perspective check node degree distribution
P_i	number of check nodes with degree i
p	transition probability of a binary symmetric channel
p_j	selection probability of a class-j input symbol among all input class-j input symbols
$q_{v\to c}$	message sent from variable node v to check node c
R	linear block code rate
R_o	rate of outer code
R_{sc}	source code compression rate
R_{cc}	channel code transmission rate
R_{over}	overall coding rate
$r_{c\to v}$	message sent from check node c to variable node v
$\mathbf{r}$	received vector
$\tilde{\rho}(x)$	normalized node perspective check node degree distribution
$\rho(x)$	normalized edge perspective check node degree distribution
ρ_i	fraction of edges connected to degree-i check nodes
$\rho^{(j)}(\mathbf{d})$	edge perspective multi-edge check node degree distribution of type-j edges
$\rho_{\mathbf{d}}^{(j)}$	fraction of type-j edges connected to type-$\mathbf{d}$ check nodes
σ_{ch}^2	variance of channel messages
σ_n^2	noise variance
σ_r^2	variance of messages sent from a check node

σ_v^2	variance of messages sent from a variable node
$\mathbf{U}(D)$	convolutional encoder input sequence
$\mathbf{u}$	source output and channel encoder input vector
V_n	n-dimensional vector space
w_i	window i
$\Omega(x)$	LT code output symbol degree distribution
Ω_i	fraction of output symbols with degree i
$\Omega^{(j)}(x)$	LT code output symbol degree distribution for window j
ω_j	selection probability of a class-j input symbol among all input symbols
$\mathcal{X}$	channel input alphabet
x	channel input symbol
x_v	represented value of a variable node v
$\mathcal{Y}$	channel output alphabet
y	channel output symbol
y_v	channel observation for the value of a variable node v

Appendix B

List of Acronyms

ACE	approximate cycle extrinsic message degree
APP	a posteriori
AWGN	additive white Gaussian noise
BCJR	Bahl-Cocke-Jelinek-Raviv
BEC	binary erasure channel
BER	bit-error rate
BI-AWGN	binary-input additive white Gaussian noise
BP	belief propagation
BPSK	binary phase shift keying
BSC	binary symmetric channel
CLID	closed-loop iterative decoder
CND	check node decoder
DE	density evolution
EEP	equal error protection
EXIT	extrinsic information transfer
GF	Galois field
HCC	hybrid concatenated code
JSC	joint source-channel
LDPC	low-density parity-check
LIB	less important bits
LLR	log-likelihood ratio
LT	Luby transform
MI	mutual information
MIB	most important bits
N$\bar{\text{S}}$C	non-recursive non-systematic convolutional

PCC	parallel concatenated code
PEG	progressive edge-growth
RSC	recursive systematic convolutional
SCC	serial concatenated code
SNR	signal-to-noise ratio
UEP	unequal error protection
URT	unequal recovery time
VND	variable node decoder

Appendix C

Extended Abstract in Portuguese

Resumo

Na presente tese, investigamos sistemas baseados em códigos corretores de erros para as seguintes aplicações: proteção desigual contra erros e codificação conjunta fonte-canal. Proteção desigual contra erros (UEP - *unequal error protection*) é uma característica desejável para sistemas de comunicação nos quais dados com diferentes níveis de importância são transmitidos, e é um desperdício de recursos computacionais, ou mesmo inviável, assegurar uma proteção uniforme para todo o conjunto de dados. Em tais sistemas, pode-se dividir os dados a serem transmitidos em classes com requisitos de proteção diferentes. Entre os métodos possíveis para se obter UEP, vamos nos concentrar em soluções baseadas em códigos corretores de erros.

No concernente a soluções baseadas em códigos corretores de erros para a obtenção de uma proteção desigual dos dados, apresentamos primeiramente uma análise de uma concatenação híbrida de códigos convolucionais, que tipicamente surge no contexto de sistemas de codificação turbo com propriedades UEP. Nós mostramos que a análise de tal sistema pode ser reduzida ao estudo de uma concatenação serial de códigos convolucionais, o que simplifica significativamente o projeto de tais sistemas híbridos.

Além disso, investigamos também a aplicação de códigos baseados em grafos para sistemas com requisitos UEP. Primeiramente, realizamos uma análise tipo multirramos de códigos baseados em matrizes de verificação de paridade esparsas (LDPC - *low-density parity-check*) para sistemas com proteção desigual. Por meio dessa análise, derivamos um algoritmo de otimização que visa otimizar as conexões entre as diferentes classes

de proteção definidas dentro de uma palavra-código de um dado código LDPC com proteção desigual. Esta otimização não apenas permite controlar as diferenças entre os desempenhos das diferentes classes de proteção por meio de um único parâmetro, mas também possibilita a construção de códigos com uma capacidade UEP que não desaparece quando um número moderado a grande de iterações é aplicado na decodificação iterativa feita por meio do algoritmo soma-produto.

Como terceira contribuição relacionada a sistemas UEP, introduzimos uma análise multirramos de códigos *Luby transform* para aplicações com proteção desigual. Nós obtemos as equações de evolução da probabilidade de erro na decodificação de códigos LT com proteção desigual, analisamos duas técnicas existentes para a construção dos mesmos, e propomos um terceiro método de construção, ao qual chamamos de "construção flexível". Através de simulações, mostramos que para regimes de transmissão com altos *overheads*, a construção flexível proposta apresenta desempenhos melhores do que os esquemas existentes e possui vantagens para aplicações onde uma pré-codificação de dados anterior à codificação de canal é necessária.

Na última parte da tese, investigamos esquemas de codificação conjunta fonte-canal nos quais códigos LDPC são aplicados para a codificação de fonte e de canal. Tal investigação é motivada pelo fato de que é amplamente observado que, para sistemas de comunicação operando em regime não-assintótico, um projeto separado dos codificadores de fonte e de canal não é ótimo, e ganhos em termos de complexidade e de fidelidade podem ser obtidos por uma estratégia de construção conjunta. Além disso, independentemente do fato da área de compressão de dados ter atingido um estado de maturidade, existem aplicações estado-da-arte que não aplicam compressão de dados, deixando de tirar proveito da redundância da fonte ao realizar a decodificação.

Dentro deste cenário, propomos um esquema de codificação conjunta fonte-canal baseado em códigos LDPC e, por meio da análise multirramos previamente desenvolvida, propomos um algoritmo de otimização para esses sistemas. Tendo por base a codificação de fonte por meio de formação de síndromes, propomos um novo sistema onde a quantidade de informação disponível no decodificador acerca dos símbolos emitidos pela fonte é aumentada por meio de uma maior conexão entre o grafos-fator correspondentes aos códigos de fonte e de canal que compõem o sistema conjunto.

Por fim, através de simulações, mostramos que o sistema proposto, comparado com sistemas de codificação conjunta fonte-canal baseados em códigos LDPC existentes, apresenta uma redução significativa do *error-floor* ocasionado pela codificação de mensagens correspondentes a padrões de erro incorrigíveis pelo código LDPC de fonte.

C.1 Introdução

Sistemas de comunicação digital encontram-se tão arraigados em nosso dia a dia que se torna cada vez mais difícil imaginar um mundo sem os mesmos. Eles estão por toda parte, de telefones celulares a comunicações espaciais, e têm se desenvolvido em ritmo acelerado desde 1948 com a publicação do trabalho seminal de Shannon "*A mathematical theory of communication*" [1]. Em uma época em que se acreditava que um aumento da taxa de transmissão de informação sobre um canal aumentaria obrigatoriamente a probabilidade de ocorrência de erros na transmissão, Shannon provou em seu teorema da codificação de canal que isso não é verdade. É de fato possível estabelecer uma comunicação com probabilidade de erro tendendo a zero, contanto que a taxa de transmissão seja mantida abaixo da capacidade do canal utilizado. O caminho para isso: codificação.

Desde o desenvolvimento dos primeiros códigos corretores de erros não triviais por Golay [2] e Hamming [3], um grande número de trabalhos tem sido feito acerca do desenvolvimento de métodos de codificação e de decodificação para controle de erros em transmissões em canais ruidosos. No entanto, foi apenas na década de 1990 que sistemas de codificação práticos capazes de atingir a capacidade prevista por Shannon foram desenvolvidos com o advento dos códigos turbo por Berrou, Glavieux e Thitimajshima [4] e com a redescoberta dos códigos com matrizes de verificação de paridade esparsas propostos por Gallager em [5]. Esses dois esquemas possuem em comum o fato de que seus algoritmos de decodificação mais utilizados são baseados em técnicas iterativas e, em conjunto com códigos *Luby transform* (LT) [6], são partes fundamentais desta tese, onde investigamos suas aplicações para proteção desigual contra erros e codificação conjunta fonte-canal.

Proteção desigual contra erros (UEP - *unequal error protection*) é uma característica desejável para sistemas de comunicação onde símbolos com diferentes sensibilidades a ruídos são transmitidos, e é um desperdício de recursos computacionais, ou mesmo inviável, proporcionar uma proteção uniforme para todo o fluxo de dados. Existem basicamente três estratégias para se obter proteção desigual contra erros em sistemas de transmissão: *bit-loading*, modulação com codificação multinível e codificação de canal [7]. No presente trabalho, vamos nos concentrar nesta última estratégia, especificamente em códigos com matrizes de verificação de paridade esparsas (LDPC - *low-density parity-check*) e códigos LT. Além disso, apresentamos alguns resultados aplicáveis ao projeto de esquemas de codificação concatenados utilizados no contexto UEP.

Por último, mas não menos importante, abordamos o problema da codificação conjunta fonte-canal. A ideia principal quando se lida com o problema da codificação conjunta fonte-canal consiste em fazer uso da redundância resídual resultante de uma compressão incompleta de dados a fim de diminuir a taxa de erro do sistema de comunicação. Nesta tese, a abordagem escolhida para codificação conjunta fonte-canal baseia-se no uso de códigos LDPC e na compressão de fontes por meio de formação de síndromes.

Esta tese está organizada da seguinte maneira. Em primeiro lugar, no Capítulo 2, introduzimos o sistema de comunicação e modelo de transmissão que iremos considerar. Além disso, apresentamos alguns conceitos básicos sobre codificação que são essenciais para uma compreensão completa dos capítulos subsequentes. O Capítulo 3 descreve a relação entre os dois diferentes tipos de gráficos de transferência de informação extrínseca (EXIT - *extrinsic information transfer*) que surgem na análise de um esquema de codificação turbo com concatenação híbrida usado para se obter proteção desigual contra erros. A partir desta análise, mostramos que ambos os tipos de gráficos podem ser utilizados para analisar a decodificação iterativa de tais códigos hibridamente concatenados. Finalmente, mostramos que a análise de códigos turbo híbridos pode ser reduzida ao estudo das concatenações em série que compõem o sistema.

O Capítulo 4 aborda códigos LDPC com proteção desigual contra erros. Sabe-se que códigos LDPC irregulares podem ser particularmente bem adaptados para esquemas de transmissão que requerem proteção desigual do dados transmitidos devido aos diferentes graus de ligação dos nós de variável em sua representação por meio de grafos bipartidos. No entanto, esta capacidade depende fortemente do perfil de conexão entre as classes de proteção definidas dentro de um palavra-código. Nós derivamos uma análise tipo multirramos de códigos LDPC a fim de otimizar tais perfis de conexão de acordo com os requisitos de desempenho de cada classe de proteção. Isto permite a construção de códigos com matrizes de verificação de paridade esparsas nos quais a diferença entre o desempenho das classes de proteção pode ser ajustada e que possuem uma capacidade de proteção desigual contra erros que não diminui à medida que o número de iterações de decodificação cresce.

No Capítulo 5, desenvolvemos para códigos LT uma análise multirramos semelhante à apresentada no Capítulo 4. Utilizando esta análise, duas técnicas existentes para a construção de códigos LT com proteção desigual podem ser avaliadas e explicadas de uma maneira unificada. Além disso, propomos uma terceira metodologia para a construção de códigos LT com proteção desigual que apresenta um desempenho melhor do que as técnicas de construção já presentes na literatura.

O contexto multirramos aplicado nos capítulos 4 e 5 é então utilizado no Capítulo 6 para o problema da codificação conjunta fonte-canal. A abordagem seguida nesse capítulo baseia-se em um modelo gráfico onde as estruturas dos códigos de fonte e de canal são conjuntamente exploradas. Especificamente, estamos interessados na otimização de sistemas conjuntos que executam uma codificação linear da saída da fonte e da entrada do canal por meio de códigos LDPC. Nós apresentamos um novo sistema no qual a quantidade de informação disponível no decodificador acerca da saída da fonte é aumentada através de uma melhora na ligação entre os grafos-fator dos códigos de fonte e de canal que compõem o sistema conjunto. Nós propomos ainda a aplicação de códigos LDPC para a codificação de fonte baseada em formação de síndrome. Além disso, desenvolvemos uma estratégia de otimização para os códigos componentes com base em uma otimização conjunta tipo multirramos.

Por fim, o Capítulo 7 resume os resultados da tese e faz considerações sobre possíveis trabalhos futuros.

C.2 Objetivos, Metodologia e Resultados

A seguir, descrevemos sucintamente os objetivos, metodologia e resultados contidos nos capítulos 3, 4, 5 e 6 da presente tese.

Capítulo 3 - Análise assintótica de códigos turbo híbridos

Códigos turbo [4] foram originalmente definidos como códigos paralelamente concatenados, i.e., uma concatenação paralela de dois códigos convolucionais binários com os ramos paralelos separados por um entrelaçador de tamanho apropriado, decodificados por meio de um algoritmo iterativo. Posteriormente, Benedetto *et al.* [33] apresentaram uma concatenação serial de códigos com entrelaçamento. Em geral, estes códigos serialmente concatenados exibem um *error floor* mais baixo do que os códigos paralelamente concatenados. No entanto, códigos serialmente concatenados geralmente apresentam um comportamento assintótico mais distante da capacidade assintótica

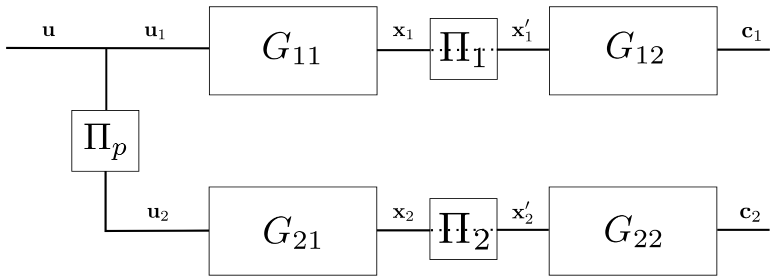

Figura C.1: Diagrama de blocos representando o codificador de um código turbo híbrido.

quando comparados com concatenações paralelas com os mesmos parâmetros. Um outro tipo de concatenação de códigos, definida como concatenação híbrida, consiste em uma combinação de concatenações seriais e paralelas, possibilitando assim explorar vantagens de ambas as formas de concatenação. Existem vários exemplos de concatenações híbridas na literatura, e.g., [34, 35]. Neste capítulo, nós investigamos o esquema híbrido proposto em [32], o qual é mostrado na Fig. C.1 e é formado pela concatenação em paralelo de dois códigos serialmente concatenados separados por um entrelaçador. Esta forma de concatenação emerge no contexto de códigos turbo com proteção desigual contra erros.

A característica UEP para códigos turbo híbridos é obtida através da eliminação (ou poda) de ramos na treliça de decodificação correspondente ao sistema. Isto foi demonstrado em [36], onde os autores mostraram que um procedimento de poda pode ser empregado para ajustar a taxa para diferentes classes de proteção em códigos turbo com proteção desigual. A poda pode ser feita por meio da concatenação em série de um código-mãe (*mother code*) com um código de poda (*pruning code*), o que leva à seleção de apenas alguns caminhos na treliça de decodificação. Na Fig. C.1, os códigos externos G_{11} e G_{21} correspondem aos códigos de poda, e os códigos internos G_{12} e G_{22} aos códigos-mãe. Para investigar o comportamento desta concatenação híbrida quando submetida a uma decodificação iterativa, nós fazemos uso dos gráficos EXIT propostos por ten Brink [28]. Contudo, pode-se definir dois gráficos EXIT diferentes para a concatenação híbrida investigada. O primeiro, ao qual chamamos de gráfico EXIT local, examina o comportamento dos códigos serialmente concatenados quando submetidos a uma decodificação iterativa. O segundo, ao qual chamamos de gráfico EXIT global, investiga a troca de informação extrínseca entre os ramos paralelos durante a decodificação.

Nosso objetivo é determinar a relação entre esses dois tipos de gráfico EXIT e mostrar que, por meio dos gráficos EXIT locais, o projeto de códigos turbo híbridos pode ser reduzido ao projeto de códigos convolucionais serialmente concatenados. Para isso, primeiramente fazemos uma recapitulação da decodificação iterativa de concatenações em paralelo e em série. De posse destes conceitos, nós apresentamos o esquema de decodificação iterativa de códigos turbo híbridos, o qual é composto por um conjunto de iterações locais (nas quais apenas um dos ramos paralelos é decodificado), e um conjunto de iterações globais (nas quais os ramos paralelos trocam informação extrínseca).

Definido o esquema iterativo para decodificação de códigos híbridos, introduzimos os conceitos envolvidos na construção dos gráficos EXIT local e global. A construção dos gráficos EXIT locais consiste em exibir em um único gráfico as curvas de transferência de informação mútua de um esquema de codificação serialmente concatenado [37]. A diferença consiste no fato de que para sistemas híbridos, a quantidade de informação *a priori* acerca dos símbolos sistemáticos não é mantida constante ao longo da decodificação global, i.e., após a decodificação local, um ramo paralelo envia informação extrínseca acerca dos símbolos sistemáticos para o ramo paralelo adjacente, a qual será utilizada como informação *a priori.* Dessa forma, o gráfico EXIT local será formado pela curva de transferência do decodificador interno, e um conjunto de curvas relativas às curvas de transferência do decodificador externo para diferentes valores de informação *a priori* recebida acerca dos símbolos sistemáticos.

Visto que os decodificadores internos só têm acesso à informação *a priori* vinda do canal de transmissão, a qual permanece constante durante toda a decodificação, a curva de transferência dos mesmos permanece inalterada durante toda a decodificação iterativa do sistema híbrido. Já para os decodificadores externos, a curva de transferência parte do ponto de abscissa zero (primeira decodificação local) e é deslocada para a direita no começo de cada decodificação local adicional. Este deslocamento ocorre devido ao fato de que ao fim de cada iteração local em um dos ramos paralelos, uma quantidade adicional de informação *a priori* a respeito dos símbolos sistemáticos será recebida pelo ramo paralelo adjacente. A esse conjunto formado pela curva de transferência do decodificador interno e pelas curvas de transferência do decodificador externo, nós demos o nome de gráficos EXIT locais.

Uma vez definidos os gráficos EXIT local e global, e tendo em vista que ambos podem ser usados para analisar a convergência do código turbo híbrido quando submetido a um procedimento de decodificação iterativa, nós descortinamos a relação entre estes dois tipos de gráfico. Isso é feito a partir da observação que nos pontos de abscissa zero dos gráficos EXIT locais, toda a informação que o decodificador externo possui acerca dos seus símbolos de saída provém da informação concernente aos símbolos sistemáticos fornecida pelo ramo paralelo adjacente. Esta observação, em conjunto com o fato de que todos os códigos envolvidos são sistemáticos, nos permite relacionar os pontos de abscissa zero dos gráficos locais com a trajetória de decodificação obtida no gráfico EXIT global.

Por fim, nós mostramos como construir os gráficos EXIT locais a partir das curvas de transferência dos códigos interno e externo da concatenação serial correspondente a cada um dos ramos paralelos. Isso permite reduzir o projeto de códigos turbo híbridos ao bem-conhecido projeto de códigos convolucionais serialmente concatenados [37]. Para isso, nós mostramos como calcular a informação extrínseca concernente aos símbolos sistemáticos ($I_{\mathrm{e},j}(\hat{\mathbf{u}})$) a partir da informação *a priori* acerca desses mesmos símbolos ($I_{\mathrm{a},j}(\hat{\mathbf{u}})$) e das informações extrínseca ($I_{\mathrm{e,o}}(\hat{\mathbf{x}})$) e *a priori* ($I_{\mathrm{a,o}}(\hat{\mathbf{x}})$) a respeito dos símbolos de saída do codificador externo. Em outras palavras, a partir das curvas de transferência dos decodificadores interno e externo, nós mostramos como calcular o deslocamento para a direita da curva de transferência do decodificador externo.

Assim, por meio apenas do gráfico EXIT local, pode-se estudar a convergência do sistema híbrido como um todo, i.e., a convergência do sistema global pode ser analisada sem a necessidade de construir o gráfico EXIT global. Isto reduz a análise do sistema híbrido global ao estudo de uma concatenação serial.

Capítulo 4 - Análise tipo multirramos de códigos LDPC com proteção desigual contra erros

Uma das abordagens possíveis para a construção de sistemas com capacidades UEP consiste no projeto de códigos corretores de erros nos quais os símbolos que compõem uma palavra-código não possuem o mesmo nível de proteção. Entre estes, códigos LDPC irregulares são comumente apontados como inerentemente capazes de fornecer proteção desigual contra erros, uma vez que é amplamente observado que o grau dos nós de variável de códigos LDPC irregulares afeta a taxa de erro do símbolo representado por tal nó.

Este fato foi usado em trabalhos anteriores [39,40,41] para otimizar a distribuição de graus dos nós de variável de códigos LDPC irregulares usando *density evolution* [24]. No entanto, uma vez que o critério de desempenho assintótico para a otimização de famílias de códigos LDPC é a diferença entre o limiar de convergência e o limite de Shannon, os autores em [40] argumentaram que códigos LDPC longos não são capazes de fornecer qualquer proteção desigual contra erros devido ao fato de que o objetivo da otimização é não possuir nenhum erro ao longo palavra código ao fim da decodificação. Para contornar este problema, eles propuseram o conceito de proteção desigual contra erros como diferentes velocidades de convergência locais, ou seja, visto que os símbolos que compõem uma palavra-código são divididos em várias classes, a ideia é definir a classe mais sensível como aquela que converge mais rapidamente a fim de tê-la livre de erros com o número mínimo de iterações. Nós afirmamos que uma análise mais adequada dos códigos LDPC com proteção desigual contra erros códigos deve aplicar um modelo mais geral do que o utilizado para a representação de códigos LDPC sem classes distintas de proteção (como feito até agora na literatura). Um modelo especialmente adequado é fornecido pelos códigos LDPC multirramos [24].

Diferentemente de códigos LDPC usuais, nos quais a conectividade nos grafos-fator é determinada apenas pelos graus dos nós, na configuração multirramos, várias classes de ramos são definidas e cada nó é caracterizado pelo número de conexões com ramos de cada classe. Por meio da técnica de *density evolution* para a configuração multirramos, pode-se obter um controle mais preciso do desempenho das diferentes classes de proteção. Isto deve-se ao fato de que as densidades das mensagens enviadas através de cada tipo de ramo (consequentemente, dentro de cada classe de proteção) podem ser controladas durante o processo iterativo de decodificação.

A importância de se obter um controle mais preciso da conexão entre as diferentes classe é ilustrado em [50], onde foram investigadas as capacidades UEP de vários algoritmos bem conhecidos de construção de códigos LDPC, como a construção aleatória, *progressive-egde-growth* (PEG), construção em zigue-zague, ciclo extrínseco aproximado (ACE), e um algoritmo PEG-ACE. Esta investigação levou à conclusão de que as propriedades do grafo-fator relevantes para o comportamento UEP não são nem ciclos, nem *trapping sets*, mas a conectividade entre as classes de proteção. Por exemplo, a construção de códigos LDPC por meio do algoritmo PEG-ACE produz nós de variáveis que são geralmente bem conectados a todas as classes de proteção através dos nós de paridade. Em contrapartida, os códigos LDPC com matrizes de verificação de paridade construídas pela técnica ACE apresentam nós de variáveis que não apresentam grande conectividade entre as diferentes classes de proteção.

Estas observações explicam a razão por trás do fato de que, ao contrário dos códigos construídos por meio de PEG, os códigos construídos através do algoritmo ACE possuem boas capacidades UEP para um grande número de iterações de decodificação. Uma explicação intuitiva consiste no fato de que, quando nós de variável de diferentes classes são bem conectados, mensagens confiáveis e não-confiáveis emitidas por diferentes classes podem avançar para as outras com mais facilidade, afetando seu desempenho. Isto irá diminuir a diferença entre o desempenho das classes de proteção à medida que o número de iterações aumenta. Assim, eventualmente, a diferença no desempenho irá desaparecer com um número infinito de iterações como observado em [40]. O objetivo principal do Capítulo 4 consiste em desenvolver uma otimização de tal perfil de conectividade a partir da abordagem multirramos.

Tendo em vista tal objetivo, o primeiro passo para o desenvolvimento do algoritmo de otimização apresentado no Capítulo 4 consiste na introdução de uma descrição multirramos de códigos LDPC com proteção desigual. Uma forma natural de incluir códigos LDPC com proteção desigual em tal contexto é distinguir os ramos ligados a diferentes classes de proteção dentro de uma palavra-código. De acordo com essa estratégia, os ramos conectados a nós de variável pertencentes a uma mesma classe de proteção são definidos como sendo do mesmo tipo. Por exemplo, considere o grafo fator da Fig. C.2, no qual as setas representam a informação recebida do canal e os nós de variável são divididos em 2 classes de proteção disjuntas C_1 e C_2 representadas pelos círculos cinzas e brancos, respectivamente. Uma descrição multirramos pode ser feita definindo os ramos conectados aos nós de variável da classe C_1 e C_2 como tipo-1 (linhas cheias) e tipo-2 (linhas tracejadas), respectivamente.

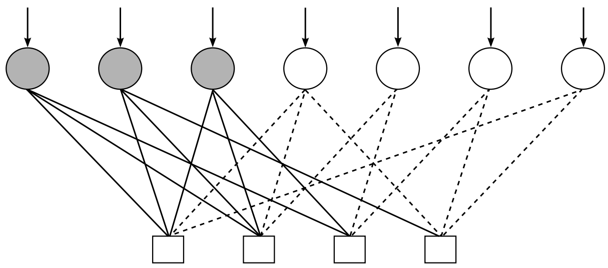

Figura C.2: Grafo fator tipo multirramos com 2 classes de proteção.

Uma vez obtida tal descrição, nós procedemos à análise assintótica de códigos LDPC com proteção desigual dentro do contexto multirramos. Nós assumimos que os códigos LDPC em questão são decodificados iterativamente por meio do algoritmo soma-produto, no qual as mensagens trocadas entre os nós de variável e de verificação são razões de verossimilhança (*log-likelihood ratios*) consideradas independentes e com uma distribuição de probabilidade gaussiana simétrica. Por meio destas hipóteses, derivamos as equações que descrevem a evolução da decodificação iterativa por meio da informação mútua entre as mensagens trocadas pelos nós de variável e verificação e os valores dos símbolos correspondentes. Tais equações são então aplicadas como uma das restrições do problema de otimização, i.e., a otimização deve ser obtida observando a restrição de que a probabilidade de erro na decodificação de um nó de variável deve tender a zero quando o comprimento das palavras-código e número de iterações tendem a infinito.

De posse das equações de evolução da informação mútua entre os valores representados pelos nós de variável e as mensagens recebidas por estes, nós propomos uma otimização iterativa que considera cada classe de proteção isoladamente, partindo da menos para a mais protegida. O algoritmo de otimização desenvolvido pode ser resumido da seguinte maneira: dado um código LDPC com proteção desigual, minimize o grau médio dos nós de verificação com respeito à classe de proteção em consideração. Essa minimização tem como objetivo diminuir a quantidade de mensagens com pouca confiabilidade (i.e., aquelas que são emitidas por uma classe de proteção menos protegida) que são processadas por um nó de verificação. Além disso, com o objetivo de controlar o nível de conexão entre as diferentes classes de proteção, nós introduzimos um novo parâmetro de otimização, ao qual chamamos de “vetor de conexão interclasse”. Os elementos de tal vetor definem o maior número permitido de ramos de cada tipo ligados a um nó de verificação.

Para demonstrar a aplicação do algoritmo de otimização desenvolvido, nós primeiramente construímos um código LDPC com proteção desigual e 3 classes de proteção seguindo a abordagem introduzida em [40]. Em seguida, nós otimizamos as conexões entre as classes de proteção para diferentes vetores de conexão interclasse. Isto resultou em 5 códigos referenciados como códigos I, II, III, IV e V. Para mostrar como o vetor de conexão interclasse influencia o desempenho de cada uma das classes de proteção, nós realizamos dois conjuntos de simulações. O primeiro conjunto faz uso de um pequeno número de iterações do algoritmo soma-produto (7 iterações) para investigar o desempenho em sistemas com recursos computacionais e tempo de decodificação limitados (como ocorre com a maior parte dos sistemas práticos). O segundo conjunto de simulações utiliza um número alto de iterações (50 iterações) para verificar o comportamento assintótico dos códigos projetados.

Os resultados obtidos nas simulações com um pequeno número de iterações mostram que, nesse cenário, é desejável manter a classe mais protegida o mais isolada possível da classe mais vulnerável. No entanto, se um grande número de iterações é aplicado, é possível obter um ganho significativo no desempenho quando a classe de proteção composta pelos símbolos de paridade é igualmente bem conectada às classes de proteção formadas pelos símbolos sistemáticos. Adicionalmente, as simulações sugerem que para ambos os casos, quanto maior o isolamento entre as classes sistemáticas, maior será a diferença entre os desempenhos das mesmas. Por fim, os códigos otimizados por meio do algoritmo que propusemos apresentam um desempenho superior aos melhores códigos LDPC com proteção desigual previamente publicados, tanto para um pequeno como para um grande número de iterações do algoritmo soma-produto.

Capítulo 5 - Códigos de taxa flexível com proteção desigual

Códigos de taxa flexível (*rateless codes*) formam uma classe de códigos originalmente projetados para a transmissão de dados através de um canal com apagamentos. Contudo, diferentemente de códigos tradicionais para canais com apagamentos, não há necessidade de conhecimento das condições do canal antes da transmissão, uma vez que os símbolos codificados podem ser gerados em tempo real até o decodificador ser capaz de recuperar os dados originalmente transmitidos. Esta característica torna esses códigos particularmente adequados para transmissões do tipo *multicast* em canais com apagamentos, canais não-uniformes e canais variantes no tempo. Esta classe de códigos é formada pelos códigos LT [6], códigos Raptor [26] e códigos *Online* [27].

Generalizações dos códigos de taxa flexível desenvolvidas a fim de lhes proporcionar a capacidade de proteção desigual contra erros foram expostas em [51] e [52]. As ideias contidas em tais referências são melhor entendidas ao interpretar os símbolos de entrada e saída como nós de um grafo bipartido, onde os símbolos de entrada são representados pelos nós de variável e os símbolos de saída são representados pelos nós de paridade.

Seja k e n o número de símbolos de entrada e de saída, respectivamente. A forma proposta pelos autores em [51] para gerar códigos de taxa flexível com proteção desigual consiste em dividir os nós de variável em conjuntos (classes de proteção) $s_1, s_2, \ldots, s_r$ de tamanhos $\alpha_1 n, \alpha_2 n, \ldots, \alpha_r n$ tal que $\sum_{j=1}^{r} \alpha_j = 1$, e associar a cada conjunto uma probabilidade p_j, que determina a probabilidade de um ramo estar ligado a um nó de variável contido em s_j, para $j = 1, \ldots, r$. Dessa maneira, diferentes valores de p_j dão origem a taxa de erros com diferentes desempenhos.

Por sua vez, os autores em [52] propuseram a construção de códigos de taxa flexível com proteção desigual através da definição de “janelas”. Cada janela é formada pelos primeiros $k_i = \sum_{j=1}^{i} s_j$ símbolos de entrada, e assim, os símbolos mais importantes formam a primeira janela, enquanto a última janela é formada por todos os símbolos de entrada. Nesse esquema, o codificador primeiro define aleatoriamente uma janela e procede à codificação considerando apenas os símbolos contidos na mesma. Todos os esquemas supracitados partilham a ideia de usar a distribuição de nós de paridade desenvolvida em [26] e [6] e induzir uma nova distribuição aos nós de variável a fim de obter uma proteção desigual dos símbolos de entrada.

A literatura existente carece de uma abordagem mais geral acerca de códigos de taxa flexível com proteção desigual. O objetivo do Capítulo 5 é fornecer essa análise mais geral a fim de proporcionar um quadro comum para a literatura existente. Para isso, introduzimos uma descrição multirramos para códigos de taxa flexível, particularmente para códigos LT. Tal descrição segue o mesmo princípio utilizado para códigos LDPC no capítulo anterior, i.e., partindo do grafo bipartido que representa as relações entre símbolos de entrada e de saída, nós definimos diferentes classes de símbolos de entrada. Os ramos são então classificados de acordo com a classe de símbolos de entrada à qual eles estão conectados.

Por exemplo, na Fig. C.3, nós dividimos os símbolos de entrada (nós de variável) em duas classes. Os três primeiros nós de variável pertencem à primeira classe. Consequentemente, todos os ramos conectados a estes símbolos são definidos como ramos tipo-1 (linhas cheias). Além disso, os últimos 5 nós de variável formam outra classe de proteção, e todos os ramos conectados aos mesmos são definidos como ramos tipo-2 (linhas tracejadas).

A descrição multirramos de códigos LT possibilita incluir as técnicas previamente publicadas para a geração de códigos de taxa flexível com proteção desigual em um contexto unificado. A importância deste contexto unificado reside no fato do mesmo fornecer uma base comum para a análise e comparação entre os esquemas presentes na literatura. Assim sendo, introduzimos uma descrição multirramos das técnicas apresentadas em [51] e [52]. Além disso, a partir da observação de que ambos os esquemas existentes baseiam-se na modificação dos coeficientes da distribuição de graus tipo multirramos dos nós de verificação, nós propusemos um novo algoritmo para a

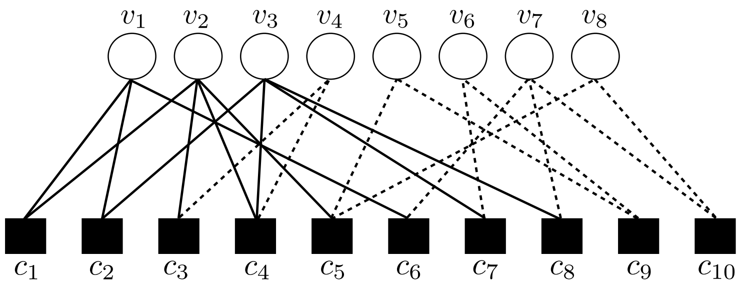

Figura C.3: Grafo multirramos com duas classes de proteção diferentes para um código LT com $k = 8$ e $n = 10$.

construção de códigos LT com proteção desigual ao qual chamamos de construção flexível.

A ideia da construção flexível consiste em aumentar a probabilidade de seleção dos símbolos de entrada considerados mais importantes, i.e., os símbolos pertencentes à classe de proteção mais sensível à ocorrência de erros na transmissão. Contudo, esse viés em prol da seleção de determinada classe de símbolos é feito sem alterar a distribuição de probabilidades geral do código LT em questão. Entre as vantagens do esquema proposto está a possibilidade de generalização do algoritmo de codificação dos códigos de taxa flexível com proteção desigual para qualquer número de classes de proteção.

Finalmente, de forma a avaliar o desempenho de códigos de taxa flexível no contexto multirramos, desenvolvemos uma análise assintótica baseada no cálculo da probabilidade de erro na recuperação dos símbolos de entrada de códigos LT decodificados iterativamente. Esta análise é desenvolvida a partir da observação de que a decodificação iterativa de códigos LT é análoga à decodificação por meio do algoritmo soma-produto de símbolos transmitidos através de um canal com apagamentos. Essa analogia nos permitiu usar resultados derivados para o canal aditivo binário com apagamentos para calcular a probabilidade de um símbolo de entrada não ser recuperado.

Usando a análise assintótica desenvolvida, nós comparamos as técnicas previamente existentes com a construção flexível proposta para diferentes parâmetros. Adicionalmente, realizamos simulações para um número finito de símbolos de entrada ($k = 5000$). Os resultados indicam que, para regimes de transmissão com alto *overhead*, o esquema flexível atinge taxas de erro mais baixas do que as construções introduzidas em [51] e [52]. Além disso, para aplicações que necessitam de uma pré-codificação dos dados por códigos tradicionais para canais com apagamentos (e.g., códigos Raptor), o nosso esquema apresenta a vantagem de necessitar de apenas uma pré-codificação, enquanto o esquema baseado em janelas deve pré-codificar cada classe de proteção individualmente.

Capítulo 6 - Sistema de codificação conjunta fonte-canal baseado em códigos LDPC

A ideia de um sistema de codificação conjunta fonte-canal utilizando códigos LDPC tanto para compressão de fonte como para codificação de canal foi descrita pela primeira vez em [64], tendo os autores proposto duas configurações possíveis. A primeira estrutura proposta baseia-se numa concatenação em série de dois códigos LDPC, onde os códigos externo e interno realizam a compressão da fonte e codificação de canal, respectivamente. A segunda estrutura baseia-se em um único código LDPC sistemático, no qual a saída da fonte compõe a parte sistemática da palavra-código, a qual é puncionada antes da transmissão, de modo que apenas a parte não-sistemática seja enviada através do canal de comunicação.

Apesar de sua introdução em [64], foi em [65] que um sistema de codificação conjunta de fonte e canal baseado em códigos LDPC foi estudado pela primeira vez. Seguindo a abordagem concatenada, uma palavra código $\mathbf{c}$ foi definida como

$$\mathbf{c} = \mathbf{s} \cdot \mathbf{G}_{cc} = \mathbf{u} \cdot \mathbf{H}_{sc}^{T} \cdot \mathbf{G}_{cc} \ ,$$

onde $\mathbf{G}_{cc}$ (de dimensões $l \times m$) representa a matriz geradora do código LDPC correspondente ao código de canal, $\mathbf{H}_{sc}$ (de dimensões $l \times n$) representa a matriz de verificação de paridade do código LDPC correspondente ao codificador de fonte, $\mathbf{s}$ corresponde à sequência comprimida pelo codificador de fonte, e $\mathbf{u}$ corresponde à saída da fonte de informação.

Considerando uma fonte binária sem memória e utilizando o algoritmo soma-produto para a decodificação, o resultados das simulações em [66] mostram a presença de *error floors* nas curvas de desempenho, os quais são uma consequência do fato de que algumas sequências emitidas pela fonte correspondem a padrões de erro que não podem ser corrigidos pelo código LDPC utilizado como codificador de fonte (compressor). A solução proposta em [66] para lidar com este erro residual foi reduzir a taxa de compressão, ou aumentar o comprimento da saída da fonte. No entanto, estas soluções não são muito atraentes, uma vez que o aumento do tamanho da saída da fonte eliminaria uma das vantagens do sistema proposto que é a possibilidade de um melhor desempenho em um cenário não-assintótico, e reduzir a taxa de compressão desloca o desempenho do sistema para longe da sua capacidade assintótica.

Nossa ideia para lidar com este problema e assim gerar um sistema de codificação conjunta fonte-canal com desempenho competitivo mesmo para sequências de informação de comprimento curto, mantendo a simplicidade da compressão baseada no cálculo de síndromes, consiste em aumentar a quantidade de informação disponível na decodificação acerca dos símbolos emitidos pela fonte. Isto pode ser feito através da inserção de novos ramos ligando os nós de verificação de paridade do código de canal aos nós de variável do código de fonte no grafo-fator correspondente ao sistema conjunto apresentado em [65]. O raciocínio por trás dessa estratégia é que essa inserção adicional de ramos fornecerá aos nós de variável do código de fonte uma quantidade de informação extrínseca que irá diminuir significativamente o *error floor* devido a padrões de erro que não podem ser corrigidos pelo código LDPC utilizado como compressor.

Para alcançar o objetivo proposto, desenvolvemos um novo algoritmo de codificação a partir da abordagem em série apresentada em [65]. A diferença entre as codificações do sistema em série e do sistema aqui exposto reside no fato de que neste, não só o símbolos comprimidos, mas também a saída da fonte deve ser considerada para a codificação de canal. Note que em nosso sistema, a taxa efetiva pode ser mantida constante quando comparado com os sistemas conjuntos anteriormente propostos, uma vez que os símbolos correspondentes à saída da fonte podem ser puncionados antes da transmissão.

Após definir o algoritmo de codificação, nós utilizamos a notação multirramos para proceder à análise assintótica e posterior otimização do sistema de codificação conjunta fonte-canal proposto. O contexto multirramos é utilizado com o objetivo de investigar a evolução da decodificação iterativa dos decodificadores de fonte e canal conjuntamente. A descrição multirramos do sistema de codificação conjunta fonte-canal proposto é feita através da definição de quatro tipos de ramos.

O primeiro tipo é composto pelos ramos conectados somente a nós de variável e de verificação de paridade pertencentes ao grafo-fator do código de fonte. Analogamente, o segundo tipo de ramos é formado pelos ramos ligados somente aos nós pertencentes ao grafo-fator do código de canal. O terceiro tipo de ramos corresponde àqueles ligados a um nó de variável do canal e a um nó de verificação da fonte. Finalmente, os ramos que conectam os nós de variável do código LDPC usado como compressor aos nós de verificação de paridade do grafo-fator do código de canal constituem o quarto tipo de ramos. A Fig. C.4 ilustra os quatro tipos de ramos e as informações correspondentes à estatística da fonte (setas tracejadas) e à informação do canal (setas cheias) recebidas pelos nós de variável.

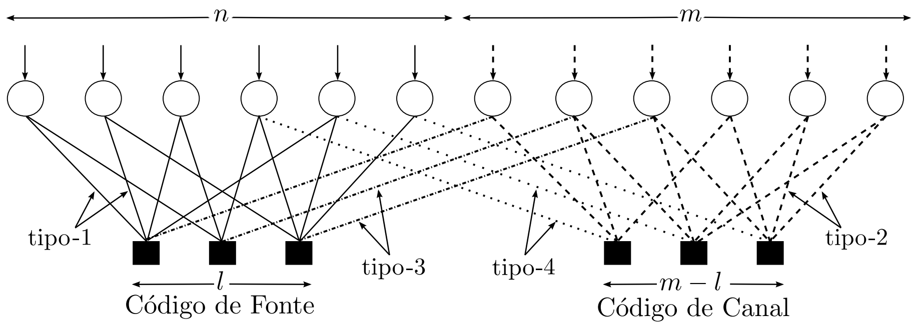

Figura C.4: Grafo fator conjunto para os códigos LDPC de fonte e canal com conectividade adicional.

Uma vez definido o grafo tipo multirramos, derivamos o conjunto de equações que descreve a evolução da informação mútua entre as mensagens trocadas através de cada tipo de ramo e os valores dos nós de variável aos quais as mesmas se destinam. Tais equações permitem analisar o processo de decodificação iterativa e avaliar se os nós de variável convergem para o valor correto ao fim de um determinado número de iterações. Tendo derivado tal conjunto de equações, nós pudemos então propor um algoritmo de otimização para o sistema de codificação conjunta fonte-canal.

No algoritmo proposto, nós primeiramente otimizamos um código LDPC para transmissão em um canal com ruído aditivo gaussiano branco. Esta otimização inicial é a mesma utilizada para códigos LDPC irregulares [16] e é feita sem considerar nenhuma conexão com o codificador de fonte. Após otimizar o código LDPC de canal, nós associamos os nós de variável de maior grau à parte sistemática das palavras-código do mesmo. Isso é feito com o objetivo de fornecer mais proteção à mensagem comprimida visto que, conforme citamos no Capítulo 4, quanto maior o grau de um nó de variável, maior sua resiliência a erros de transmissão. Por fim, otimizamos o código LDPC de fonte considerando todas as conexões do mesmo com o código de canal, i.e., levando em consideração as mensagens recebidas através dos ramos tipo 3 e tipo 4.

Como último conjunto de resultados apresentados neste trabalho, nós construímos um sistema de codificação conjunta fonte-canal com códigos de fonte e canal otimizados pelo algoritmo descrito acima. Considerando uma fonte binária sem memória e transmissão através de um canal com ruído aditivo gaussiano branco, nós obtivemos um código de fonte com taxa de compressão $R_{sc} = 0.2361$ e um código de canal de taxa $R_{cc} = 0.4805$, o que gera um sistema conjunto com taxa total $R_{over} = R_{cc}/R_{sc} \simeq 2.03$.

A fim de ilustrar os méritos da otimização proposta, nós comparamos o desempenho de nosso sistema conjunto com o esquema de codificação conjunta baseado em códigos LDPC com mesma taxa total $R_{over} = 2.03$ introduzidos por Caire *et al.* em [64] e investigados em [65]. Este sistema possui os ramos que conectam os nós de variável do código de canal aos nós de verificação de paridade dos códigos de fonte como única ligação entre os grafos-fator dos códigos componentes. Os resultados de nossas simulações mostram que o sistema otimizado aqui proposto é capaz de, mantendo a taxa total constante, reduzir significativamente o *error floor* resultante da compressão de mensagens da fonte que correspondem a padrões incorrigíveis pelo código LDPC utilizado como compressor.

C.3 Conclusão

Nós investigamos a análise assintótica e projeto de esquemas de codificação modernos para sistemas com requisitos de proteção desigual contra erros e aplicações com codificação conjunta fonte-canal. Em relação à proteção desigual contra erros, iniciamos nosso estudo com uma análise assintótica de códigos turbo híbridos. Em seguida, por meio de uma análise multirramos, investigamos códigos LDPC e LT, propondo algoritmos de otimização para melhorar o desempenho de ambos os códigos em aplicações com proteção desigual. Por fim, um sistema de codificação conjunta fonte-canal baseado em códigos LDPC foi estudado. A seguir, resumimos as contribuições da tese e apontamos possíveis tópicos futuros de pesquisa.

Como primeira contribuição, desenvolvemos a construção de gráficos EXIT locais para uma concatenação híbrida de códigos convolucionais a qual nós chamamos de códigos turbo híbridos, e mostramos a relação entre os gráficos EXIT local e global. Além disso, destacamos que por meio da relação encontrada entre a convergência local e global, a análise do sistema global pode ser reduzida ao estudo de um código formado por uma concatenação serial, uma vez que a convergência do sistema global pode ser prevista a partir do gráfico EXIT local.

Posteriormente, realizamos uma análise tipo multirramos de códigos LDPC com proteção desigual contra erros. Por meio de tal análise, nós derivamos um algoritmo de otimização que visa otimizar a ligação entre as classes de proteção definidas dentro de uma palavra-código de um determinado código LDPC com características UEP. Essa otimização permitiu-nos não só controlar as diferenças entre os desempenhos das classes de proteção por meio de um único parâmetro, mas também possibilitou a construção de códigos LDPC com uma capacidade UEP não-evanescente quando um número moderado ou elevado de iterações de decodificação é aplicado. Por fim, o algoritmo de otimização introduzido possui a capacidade de gerar códigos LDPC com proteção desigual com desempenho superior ao dos códigos existentes na literatura, independentemente do número de iterações necessário ao algoritmo de decodificação.

Um possível tópico para pesquisas futuras acerca deste tema é a investigação de códigos LDPC com proteção desigual com mais de três classes de proteção definidas dentro de uma palavra-código. Além disso, nós restringimos nossas otimizações a códigos LDPC com nós de verificação regulares. Seria interessante considerar códigos onde os nós de verificação são irregulares a fim de verificar se é possível obter ganhos extras no desempenho da taxa de erro aplicando a otimização proposta.

Como último tópico de investigação sobre sistemas com proteção desigual, introduzimos uma análise tipo multirramos para códigos LT com proteção desigual. Além disso, derivamos as equações de evolução da probabilidade de erros na decodificação de códigos LT com proteção desigual, analisamos duas das técnicas existentes para gerar os mesmos, e propomos um terceiro esquema ao qual chamamos de construção flexível. Finalmente, mostramos por meio de simulações que, para altos *overheads*, os códigos que propusemos apresentam um desempenho melhor do que os sistemas existentes e têm vantagens para aplicações onde uma pré-codificação é necessária (e.g., códigos Raptor), uma vez que utiliza apenas uma pré-codificação para todo o conjunto de dados evitando assim efeitos de comprimento finito que podem surgir devido à codificação separada de classes de proteção com um baixo número de símbolos.

Uma possível extensão do trabalho realizado sobre códigos LT com proteção desigual seria modificar o algoritmo de otimização a fim de otimizar os códigos de acordo com os requisitos de proteção de cada classe de proteção, isto é, a taxa de erro de símbolo requerida para cada classe seria um parâmetro da otimização.

Por fim, propusemos um esquema de codificação conjunta fonte-canal baseado em códigos LDPC e, por meio da análise multirramos previamente desenvolvida para códigos LDPC, propusemos um algoritmo de otimização para esses sistemas. Tendo por base a ideia de codificação de fonte por meio de síndromes, apresentamos um novo sistema, no qual a quantidade de informação disponível no decodificador sobre os símbolos emitidos pela fonte é aumentada por meio de um acréscimo do número de conexões entre os grafos-fator correspondentes aos códigos de fonte e aos códigos de canal que compõem o sistema conjunto. Em comparação com esquemas existentes de codificação conjunta fonte-canal baseado em códigos LDPC, os resultados das simulações apresentados mostram uma redução significativa do *error floor* causado pela codificação de mensagens que correspondem a padrões de erro incorrigíveis pelo código LDPC utilizado como codificador de fonte.

Este tópico oferece interessantes possibilidades de investigação. Uma delas é a construção de um sistema conjunto com características UEP por meio da aplicação de códigos LDPC com proteção desigual como codificadores de fonte baseados em formação de síndromes. Outro tópico de investigação possível é a melhoria do desempenho do nosso sistema proposto através da diminuição da probabilidade de ocorrência de um padrão incorrigível usando encurtamento (*shortening*), que consiste em colocar uma confiabilidade infinita em alguns nós de variável do código LDPC usado como código de fonte. Usando essa estratégia, os nós com confiabilidade infinita devem ser puncionados antes da transmissão a fim de manter a taxa de compressão inalterada, e têm suas posições conhecidas tanto pelo codificador como pelo decodificador.

Finalmente, uma otimização iterativa dos códigos componentes do sistema de codificação conjunta fonte-canal aqui proposto pode ser desenvolvida. Para tal, pode-se considerar que, na iteração inicial, um dos códigos LDPC é fixo (podemos por exemplo fixar o código LDPC de canal na primeira iteração) e em seguida otimizar o outro código componente seguindo a abordagem padrão baseada em gráficos EXIT. Na iteração seguinte, o código otimizado previamente é fixado e a otimização do outro código componente implementada.

Own Publications

- H. V. Beltrão Neto, W. Henkel, and V. C. da Rocha Jr., "Multi-edge-type unequal error protecting low-density parity-check codes," in *Proc. IEEE Information Theory Workshop*, Paraty, Brazil, Oct. 2011.
- H. V. Beltrão Neto, W. Henkel, and V. C. da Rocha Jr., "Multi-edge framework for unequal error protecting LT codes," in *Proc. IEEE Information Theory Workshop*, Paraty, Brazil, Oct. 2011.
- A. Wakeel, D. Kronmueller, W. Henkel, and H. V. Beltrão Neto, "Leaking interleavers for UEP turbo codes," in *Proc. 6th International Symposium on Turbo Codes & Iterative Information Processing*, Brest, France, Aug. 2010.
- H. V. Beltrão Neto and W. Henkel, "Relation between local and global EXIT charts in hybrid turbo codes," in *Proc. International ITG Conference on Source and Channel Coding*, Siegen, Germany, Jan. 2010.
- H. V. Beltrão Neto and V. C. da Rocha Jr., "Iterative decoding results for the Gaussian multiuser binary adder channel," in *Proc. 10th International Symposium on Communication Theory and Applications*, Ambleside, England, July 2009.
- H. V. Beltrão Neto, W. Henkel, and V. C. da Rocha Jr., "Multi-Edge-Type Optimization of Unequal Error Protecting Low-Density Parity-Check Codes," submitted to IEEE Transactions on Communications.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–427 and pp. 623–656, Jul. and Oct. 1948.

[2] M. J. E. Golay, "Notes on digital coding," *Proc. IRE*, vol. 37, p. 657, 1949.

[3] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, Apr. 1950.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: turbo codes," in *Proc. IEEE International Conference on Communication (ICC)*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[5] R. G. Gallager, "Low-density parity-check codes," Ph.D. dissertation, MIT, 1963.

[6] M. Luby, "LT codes," in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, Nov. 2002, pp. 271–282.

[7] W. Henkel, K. Hassan, N. von Deetzen, S. Sandberg, L. Sassatelli, and D. Declercq, "UEP concepts in modulation and coding," *Hindawi, Advances in Multimedia*, Vol. 2010, Article ID 416797, 14 pages, 2010. doi:10.1155/2010/416797.

[8] J. Hagenauer, "Source-controlled channel decoding," *IEEE Transactions on Communications*, vol. 43, no. 9, pp. 2449–2457, Sep. 1995.

[9] T. Cover and J. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006.

[10] P. Elias, "Coding for two noisy channels," in *Information Theory, The 3rd London Symposium*, 1955, pp. 61–76.

[11] S. Lin and D. Costello, *Error Control Coding*, 2nd ed. Prentice Hall, 2004.

[12] P. Elias, "Coding for noisy channels," in *IRE Convention Record, Pt. 4*, 1955, pp. 37–46.

[13] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. IEEE Press, 1999.

[14] A. Viterbi, "Error bound for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[15] L. Bahl, J. Cocke, F. Jelineka, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 248–287, Mar. 1974.

[16] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[18] M. M. Vasconcelos, "Decodificação iterativa de códigos baseados em matrizes de verificação de paridade esparsas," Master's thesis, Federal University of Pernambuco, Brazil, Apr. 2006.

[19] S. ten Brink, "Designing iterative decoding schemes with the extrinsic information transfer chart," *International Journal of Electronics and Communications*, vol. 54, no. 6, pp. 389–398, 2000.

[20] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2657–2673, Nov. 2004.

[21] S. Y. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[22] S. ten Brink, “Convergence behavior of iteratively decoded parallel concatenated codes,” *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[23] F. Brännström, “Convergence analysis and design of multiple concatenated codes,” Ph.D. dissertation, Chalmers Univ. Technology, Mar. 2004.

[24] T. Richardson and R. Urbanke, *Modern Coding Theory.* Cambridge University Press, 2008.

[25] ——, “Multi-Edge Type LDPC Codes,” Tech. Rep., 2004, submitted to IEEE Transaction on Information Theory.

[26] A. Shokrollahi, “Raptor codes,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[27] P. Maymounkov, “Online codes,” NYU, Tech. Rep. TR2003-883, Nov. 2002.

[28] S. ten Brink, “Convergence of iterative decoding,” *Electronics Letters*, vol. 35, no. 10, pp. 806–808, May 1999.

[29] ——, “Convergence of multi-dimensional iterative decoding schemes,” in *Proc. Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, Nov. 2001, pp. 270–274.

[30] S. ten Brink, G. Kramer, and A. Ashikhmin, “Design of low-density parity-check codes for modulation and detection,” *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, Apr. 2004.

[31] W. E. Ryan and S. Lin, *Channel Codes, Classical and Modern.* Cambridge, 2009.

[32] N. von Deetzen and W. Henkel, “Decoder scheduling of hybrid turbo codes,” in *IEEE International Symposium on Information Theory*, Seattle, USA, Jul. 2006.

[33] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, “Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding,” *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[34] D. Divsalar and F. Pollara, “Serial and hybrid concatenated codes with applications,” in *Proc. Int. Symp. on Turbo Codes and Related Topics*, Brest, France, Sep. 1997, pp. 80–87.

[35] H. Gonzalez, C. Berrou, and S. Kerouédan, "Serial/parallel turbo codes for low error rates," in *IEEE Military Commun. Conf.*, 2004, pp. 346–349.

[36] W. Henkel and N. von Deetzen, "Path pruning for unequal error protection," in *International Zurich Seminar on Communications*, Zurich, Switzerland, Feb. 2006.

[37] S. ten Brink, "Code characteristic matching for iterative decoding of serially concatenated codes," *Annals of Telecommunications*, vol. 56, no. 7-8, pp. 394–408, Jul. 2001.

[38] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[39] C. Poulliat, D. Declercq, and I. Fijalkow, "Optimization of LDPC codes for UEP channels," in *Proc. IEEE International Symposium on Information Theory (ISIT '04)*, Jun. 2004.

[40] ——, "Enhancement of unequal error protection properties of LDPC codes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, Article ID 92659, 9 pages, doi:10.115/2007/92659.

[41] N. Rahnavard, H. Pishro-Nik, and F. Fekri, "Unequal error protection using partially regular LDPC codes," *IEEE Transactions on Communications*, vol. 55, no. 3, pp. 387–391, Mar. 2007.

[42] L. Sassatelli, W. Henkel, and D. Declercq, "Check-irregular LDPC codes for unequal error protection under iterative decoding," in *Proc. 4th International Symposium on Turbo Codes & Related Topics*, Apr. 2006.

[43] V. Kumar and O. Milenkovic, "On unequal error protection LDPC codes based on Plotkin-type constructions," *IEEE Transactions on Communications*, vol. 54, no. 6, pp. 994–1005, 2006.

[44] N. von Deetzen and S. Sandberg, "On the UEP capabilities of several LDPC construction algorithms," *IEEE Transactions on Communications*, vol. 58, no. 11, pp. 3041–3046, Nov. 2010.

[45] T. Tian, C. Jones, D. Villasenor, and R. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Transactions on Communications*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.

[46] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[47] D. Vukobratović and V. Šenk, "Generalized ACE constrained progressive edge growth LDPC code design," *IEEE Communications Letters*, vol. 12, no. 1, pp. 32–34, Jan. 2008.

[48] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," *JPL INP Progress Report 42-154*, Aug. 15 2003.

[49] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. 50th Annual IEEE Global Telecommunications Conference*, Nov. 2007, pp. 3250–3254.

[50] N. von Deetzen, "Modern coding schemes for unequal error protection," Ph.D. dissertation, Jacobs University Bremen, School of Engineering and Science, 2009.

[51] N. Rahnavard, B. N. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Transactions on Information Theory*, vol. 53, no. 4, pp. 1521–1532, Apr. 2007.

[52] D. Sejdinović, D. Vukobratović, A. Doufexi, V. Šenk, and R. Piechocki, "Expanding window fountain codes for unequal error protection," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2510–2516, Sep. 2009.

[53] M. Luby, M. Mitzenmacher, and A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. 9th SIAM Symposium on Discrete Algorithms*, Jan. 1998, pp. 364–373.

[54] J. L. Massey, "Joint source and channel coding," *Communications Systems and Random Process Theory*, vol. 11, pp. 279–293, Sijthoff and Nordhoff, 1978.

[55] P. E. Allard and A. W. Bridgewater, "A source encoding technique using algebraic codes," in *Proc. Canadian Computer Conference*, 1972, pp. 201–213.

[56] K. C. Fung, S. Tavares, and J. M. Stein, "A comparison of data compression schemes using block codes," in *Proc. IEEE Int. Electrical Electronics Conf.*, Oct. 1973, pp. 60–61.

[57] T. C. Ancheta, “Syndrome-source-coding and its universal generalization,” *IEEE Transactions on Information Theory*, vol. 22, no. 4, pp. 432–436, Jul. 1976.

[58] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Transactions on Information Theory,*, vol. 23, no. 3, pp. 337–343, May 1977.

[59] ——, “Compression of individual sequences via variable-rate coding,” *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, Sep. 1978.

[60] G. Caire, S. Shamai, and S. Verdú, “A new data compression algorithm for sources with memory based on error correcting codes,” in *IEEE Workshop on Information Theory*, Paris, France, Mar. 30 - Apr. 4 2003, pp. 291–295.

[61] J. Garcia-Frias and Y. Zhao, “Compression of binary memoryless sources using punctured turbo codes,” *IEEE Communications Letters*, vol. 6, no. 9, pp. 394–396, Sep. 2002.

[62] J. Hagenauer, J. Barros, and A. Schaefer, “Lossless turbo source coding with decremental redundancy,” in *Proc. International ITG Conference on Source and Channel Coding*, Erlangen, Germany, Jan. 2004, pp. 333–339.

[63] N. Dütsch and J. Hagenauer, “Combined incremental and decremental redundancy in joint source-channel coding,” in *Proc. Int. Symposium on Information Theory and its Applications*, Parma, Italy, Oct. 2004.

[64] G. Caire, S. Shamai, and S. Verdú, “Almost-noiseless joint source-channel coding-decoding of sources with memory,” in *Proc. 5th International ITG Conference on Source and Channel Coding*, Jan. 2004, pp. 295–304.

[65] M. Fresia, F. Pérez-Cruz, and H. V. Poor, “Optimized concatenated LDPC codes for joint source-channel coding,” in *Proc. IEEE Int. Symposium on Information Theory*, Seoul, South Korea, 2009.

[66] M. Fresia, F. Pérez-Cruz, H. V. Poor, and S. Verdú, “Joint source/channel coding with low density parity check matrices,” *IEEE Signal Processing Magazine*, vol. 27, no. 6, pp. 104–113, Nov. 2010.

[67] G. Caire, S. Shamai, and S. Verdú, “Noiseless data compression with low-density parity-check codes,” *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 66, pp. 263–284, American Mathematical Society, 2004.

[68] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Tech. Rep. SRC 124, Tech. Rep., May 1994.

[69] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 638–656, Feb. 2001.

[70] C. Poulliat, D. Declercq, C. Lamy-Bergot, and I. Fijalkow, "Analysis and optimization of irregular LDPC codes for joint source-channel decoding," *IEEE Communications Letters*, vol. 9, no. 12, pp. 1064–1066, Dec. 2005.

[71] M. Beermann, T. Breddermann, and P. Vary, "Rate-compatible LDPC codes using optimized dummy bit insertion," in *Proc. 8th International Symposium on Wireless Communication Systems (ISWCS)*, Nov. 2011, pp. 447 –451.

[72] T. Tian and C. R. Jones, "Construction of rate-compatible LDPC codes utilizing information shortening and parity puncturing," *EURASIP Journal on Wireless Communications and Networking*, vol. 5, pp. 789–795, Oct. 2005.