Unequal Error Protection LDPC Codes

Diploma Thesis

Lucile SASSATELLI

ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

UNIVERSITÉ
de Cergy-Pontoise

IUB
International
University
Bremen

Issuance date:	04.04.2005
Submission date:	03.09.2005
Supervisor:	Prof. Dr.-Ing. W. Henkel

September 13th, 2005

Contents

1 Introduction **1**

2 General Presentation of LDPC Codes **3**

2.1 Definition, Parameterization, and Usual Notations 4

2.1.1 Definitions 4

2.1.1.1 Regular LDPC Codes 4

2.1.1.2 Irregular LDPC Codes 5

2.1.1.3 Systematic coding of LDPC codes 6

2.1.2 Decoding LDPC Codes by Belief Propagation 7

2.1.3 Density evolution and Gaussian Approximation 9

2.1.3.1 Density evolution 9

2.1.3.2 Gaussian Approximation: Mutual Information Evolution 12

2.1.3.3 Mutual Information for a Gaussian Consistent Channel . 13

2.1.3.4 Evolution Equations 14

2.1.3.5 Stability Condition 17

3 State of the Art **18**

3.1 Optimization of LDPC codes over some channels 18

3.1.1 Optimisation over AWGN Channel 18

3.1.2 Optimization over other channels 20

3.2 UEP Optimization: An Introduction 21

4 UEP LDPC Codes 23

4.1 UEP properties created by irregularities over bit nodes 23

4.1.1 Parameterization and Asymptotic Study of such UEP LDPC Codes 24

4.1.2 Cost Function for such UEP 25

4.2 Short LDPC: UEP linear block code optimization 26

4.2.1 Algebraic Properties and Unequal Error Protection 27

4.2.2 The Derived Algorithm 27

4.2.3 Results under Maximum likelihood and Belief Propagation Decoding 28

4.3 Optimization of the Check-Node Profile 33

4.3.1 Parametrization of UEP LDPC codes 33

4.3.2 Density evolution for the detailed representation and UEP properties 34

4.3.3 Set of Possible Good Codes 42

4.3.4 A particular choice to realize UEP over check nodes 44

4.3.4.1 The chosen coding scheme 44

4.3.4.2 Case we don't need P: only info columns of H are pruned and protected. 46

4.3.4.3 Case we prune away redundancy in H or choose protected column among it, and then need P 46

Existence of $\mathbf{P}$ 46

Code rate of the subcode 48

Computation of the preprocessing matrix 50

4.3.4.4 Hierarchical optimization algorithm 52

4.3.4.5 Results 54

4.4 Puncturing 56

5 Conclusions 62

A 64

A.1 Transition from proportions of edges to proportions of nodes 64

A.2 Transition from proportions of nodes to proportions of edges 65

A.3 Expression of the code rate in terms of proportions 65

Bibliography **67**

Chapter 1

Introduction

Nowadays, high rate digital communications are very important. It allows increased rate and number of users, achieving good performances. In order to position our work, we first present the global transmission scheme in Fig. 1.1.

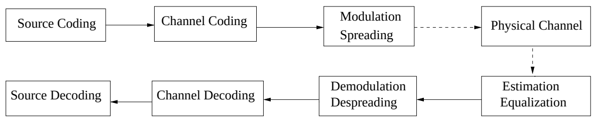

Figure 1.1: Usual communication chain.

To illustrate our problem, let us describe the elements of this chain. We begin in the transmitter with the following blocks:

- Source coding: in this block, source data are compressed and reshaped. Some parts of the source signal are more vulnerable than others.
- Channel coding: structured redundancy is added to make data robust to errors caused by the channel. The code rate R is defined as the ratio between transmitted information bits and the number of transmitted bits. The source can be encoded in an uniform way or in a heterogeneous one in order to take into account the properties of the source signal (unequal error protection (UEP) techniques).
- Modulation: the order and the type of the modulation are managed here (PSK, ASK, QAM, multicarrier modulation), the power of transmitted symbols, or the spreading when we have to deal with a multiple access system by code spreading.
- The physical channel: disturbances are introduced. The channel leads to inter symbol interferences (ISI caused by fading channel) due to multi-paths, multiuser inteferences (MUI), and adds (e.g. white gaussian or impulse) noise.

The receiver is composed of corresponding blocks, that can be described as follows:

- Estimation and Equalization: the transmission channel is estimated (multiple paths). Equalization allows to reduce or cancel ISI.
- Demodulation/Despreading: used to find bits from received symbols and to separate users in a multiple access system.
- Channel decoding: corrects remaining errors in the previous obtained binary sequence.
- Source decoding: reconstruction of the emitted data by decompression of the sequence going out from channel decoder.

In this work, we will focus on the optimization of channel codes for UEP, which has to minimize the effect of errors on media reconstruction by adapting the channel code to the structure of the source data (errors should rather affect less important bits, considering the properties of the source). Here we will only consider Low-Density Parity-Check (LDPC) codes, which are very flexible and have very good performances. Studies of other UEP methods have already been done especially for UEP Turbo Codes [24].

Within the framework of channel coding optimization for UEP using LDPC codes, analysis of UEP properties of LDPC codes thanks to usual or detailed parameterization permits to express the bit error probability for each class of sensibility of the codeword. An optimization method is then proposed. In [17] a construction method is developped in order to achieve UEP for a given code rate and given UEP constraints (proportions of classes), only by modifying the irregularity profile of bit nodes, and considering the check node profile to be fixed.

The goal of this work was to focus on UEP properties of LDPC codes achieved by pruning and puncturing. By pruning some bits in a codeword, i.e. to fix (e.g. to 0) and then not transmit them, or equivalently replacing the corresponding columns by zero columns in the parity check matrix, we directly modify the irregularity profile of check nodes, and can achieve some UEP configuration. The resulting code, of lower code rate, is a subcode derived from a mother code. Thus we study consequences of modifying check nodes profile. The idea behind is achieving different UEP configurations with different pruning schemes and the same decoder.

By changing the check-nodes profile, we will see that we construct codes that converge faster on a part of the codeword, but also achieve UEP at infinite number of iterations.

This thesis is structured as follows. In Chapter 2 we present an overall study of LDPC codes and specially the density evolution, the analytic tool to analyze these codes and their behavior. An overview of usual optimizations of LDPC codes is given in Chapter 3, as well as an introduction to Unequal Error Protection (UEP) methods. In Chapter 4 is explained the work of this thesis, i.e. the developped theory for UEP LDPC codes, a practical system to realize it, and expected experimental results. Finally, conclusions are given in Chapter 5.

Chapter 2

General Presentation of LDPC Codes

LDPC (Low-Density Parity-Check) codes are known as a family of high-performance codes, that are capacity-achieving on some standard channels like the binary erasure channel. They stand for an alternative to Turbo Codes, which achieve very good performance codes on some standard channels too. The following aspects guided us to consider this family of channel codes in UEP applications:

- Performances fit with UEP target multimedia communications: like Turbo Codes, LDPC codes have iterative decoding, that allows to reach bit error probabilities of $10^{-5} - 10^{-6}$, for a wide range of signal to noise ratios. These are the required orders for sensible applications such as fixed picture or video transmissions. A delay caused by the interleaver must be tolerated. Therefore LDPC codes can be an alternative to Turbo Codes for UEP target multimedia transmissions.
- Systematic code: For our appications, a systematic code is very useful. In such a code, information bits are integrally copied into the codeword. The first interest is that even in case of decoding errors (convergence to a word that is not a codeword), we should be able to find the systematic part, although with errors. Moreover if the receiver has no channel decoder, the source decoder has only to ignore the redundancy part. Building a systematic code from any code is very easy.
- Easy to parameterize and then to optimize: one of the biggest advantages of this family of codes is the possibility to optimize according to the channel and the application, thanks to the explicit (analytical) characterization of the state of the decoder during iterations in an asymptotic approach (infinite codeword length), that is function of channel and receiver parameters. This represents a huge advantage compared to Turbo constructions for which no analytical description exists, only simulation based on EXITcharts.
- A not completely explored domain: the optimization of LDPC codes irregularity for standard and non-standard channels has already been studied, especially by Poulliat

in [17] for UEP applications, but only considering bit nodes. Publications on the use of pruning on LDPC codes do not seem numerous (e.g. some to avoid stopping sets cite18), and almost non-existent when pruning is considered to achieve UEP inside a codeword.

2.1 Definition, Parameterization, and Usual Notations

LDPC codes are low density linear block codes, introduced by Gallager [8] in 1963. An LDPC code in $GF(Q)$ (with $Q = 2^q$) is represented by its sparse parity matrix **H** of size $(N-K)\times N$ whose non zero elements belong to the Galois field $GF(Q)$. N is the codeword length, K the number of information bits related to a codeword, $M \geq N-K$ the number of redundancy bits, and the code rate $R = K/N \geq 1 - M/N$, with equality if **H** is fullrank. The code is the set of words **c** $\in GF(Q)^K$ such that $\mathbf{H}.\mathbf{c} = \mathbf{0}$. When $Q = 2$, this the case binary LDPC codes and their description is done by parity equations. When $Q \geq 2$, this is the case of non binary LDPC codes. In this work, we consider only binary LDPC codes. In each case, the structure the parity matrix can be regular or not. A code is regular if the number of non zero elements in every rows (respectively columns) is constant. Irregular if these numbers are not constant.

2.1.1 Definitions

2.1.1.1 Regular LDPC Codes

Definition 2.1 *A regular LDPC code with its three parameters (N, t_c, t_r) is defined by a matrix with exactly t_c and t_r ones per column and row, respectively.*

The code rate is $R = K/N \geq 1 - t_c/t_r$, with equality if **H** is fullrank. Those three parameters define a family of regular codes, and one code among this family is given by a particular realization of the parity-check matrix. In an equivalent way, an LDPC code can be represented by a bipartite graph, called factor graph [12], or Tanner graph, made of two kinds of nodes: variable nodes representing bits of codeword, and check nodes associated to parity-check functions. Those two kinds of vertices are linked with each other by edges indicating to which parity equation variable nodes, i.e. the associated bits, take part in.

The ith bit node and the lth check node are connected if $H_{l,i} = 1$. The degree of connection of a bit node (the same for a check node) is the number of edges linked to this node. A node is said i connected or of degree i if it is connected to i edges. Figure (2.1) shows the representation of a regular code parametrized by $(N = 8, t_c = 2, t_r = 4)$. One code corresopnds to one particular realization of the interleaver.

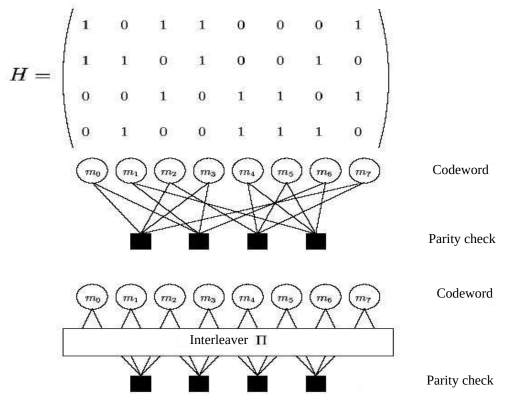
$H = \begin{pmatrix} 1&0&1&1&0&0&0&1 \\ 1&1&0&1&0&0&1&0 \\ 0&0&1&0&1&1&0&1 \\ 0&1&0&0&1&1&1&0 \end{pmatrix}$
m_0 m_1 m_2 m_3 m_4 m_5 m_6 m_7
Codeword
Parity check
Codeword
Interleaver Π
Parity check

Figure 2.1: Representation of regular code ($N = 8, t_c = 2, t_r = 4$). The upper scheme is a realization of the code, the bottom the family.

2.1.1.2 Irregular LDPC Codes

A code is irregular if it is not regular. The usual parameterization of irregular LDPC codes is done by means of polynomials:

- Polynomial associated to variable nodes:

$$\lambda(x) = \sum_{i=2}^{t_{cmax}} \lambda_i x^{i-1}$$

  where λ_i is the proportion of edges of the graph connected to bit nodes of degree i, and t_{cmax} is the maximum number of edges linked to a bit node.

- Polynomial associated to check nodes:

$$\rho(x) = \sum_{j=2}^{t_{rmax}} \rho_j x^{j-1}$$

  where ρ_j is the proportion of edges of the graph connected to check nodes of degree j, and t_{rmax} is the maximum number of edges linked to a check node.

Those two quantities are linked by the code rate:

$$R = 1 - \frac{\sum_{j=2}^{t_{rmax}} \rho_j/j}{\sum_{i=2}^{t_{cmax}} \lambda_i/i}$$

There is also a dual parameterization of the previous one:

- Polynomial associated to data nodes:

$$\tilde{\lambda}(x) = \sum_{i=2}^{t_{cmax}} \tilde{\lambda}_i x^{i-1}$$

  where $\tilde{\lambda}_i$ is the proportion of bit nodes of degree i.

- Polynomial associated to check nodes:

$$\tilde{\rho}(x) = \sum_{j=2}^{t_{rmax}} \tilde{\rho}_j x^{j-1}$$

  where $\tilde{\rho}_j$ is the proportion of check nodes of degree j.

The transition from one paramatrization to another is done by:

$\tilde{\lambda}_i = \frac{\lambda_i/i}{\sum_k \lambda_k/k} \qquad \tilde{\rho}_j = \frac{\rho_j/j}{\sum_k \rho_k/k}$
$\lambda_i = \frac{i\tilde{\lambda}_i}{\sum_k k\tilde{\lambda}_k} \qquad \rho_j = \frac{j\tilde{\rho}_j}{\sum_k k\tilde{\rho}_k}$

Thus a family of irregular codes is parametrized by $(N, \lambda(x), \rho(x))$. The regular case is a particular case of this parameterization where $\lambda(x)$ and $\rho(x)$ are monomials. Figure (2.2) is a graphical representation for this kind of code. $(\lambda(x), \rho(x))$ defines the irregularity profile of the code according to columns and rows.

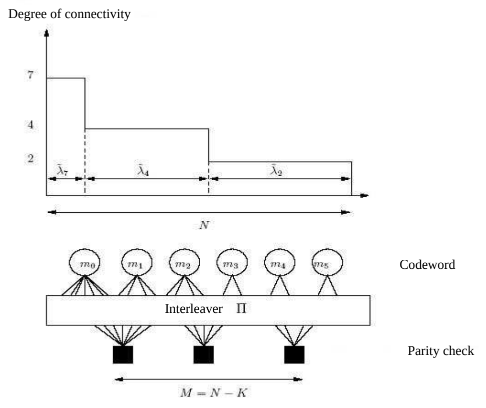

Figure 2.2: Representation of a family of irregular codes.

2.1.1.3 Systematic coding of LDPC codes

For practical reasons, codes should be systematic: information bits are directly copied into the codeword. In general building the generator matrix G from H is not too easy. Nevertheless it is possible to encode using the parity matrix. Several methods exist to make the encoding to be systematic. However they are out of our scope. We will only consider the simplest method using an upper triangular matrix for H with ones on the main diagonal: info bits are associated to the non triangular part, and redundancy bits are computed recursively from parity equations, and associated to the triangular part. The sparcity of the matrix is more or less preserved. We will thus only consider systematic matrices.

2.1.2 Decoding LDPC Codes by Belief Propagation

Although a maximum likelihood decoding of LDPC codes is possible, the complexity increases too much as soon as we consider sufficiently long codes, which is important to obtain decent performances. A sub-optimum decoding algorithm, known as Sum-Product algorithm or Belief Propagation (BP) algorithm is used instead. It spreads along edges messages forwarding probabilities or logarithmic likelihood ratios (LLR). To each branch two messages are associated, one for each direction. The principle of BP is Bayes rule applied locally (on every bit of the codeword) and iteratively to estimate a posteriori probabilities (APP) of every bit. It has been shown that over a cycle-free graph (tree case), local factorization of Bayes rules leads to exact computation of APP of bit nodes. In this case, messages over the whole graph are independent from each other. However, in a non cycle free graph (which is the case for any finite length LDPC code), messages are not independent, and then APP are not computed exactly, that says that the algorithm is not optimal anymore. The sparcer the graph, that says H, will be, the less numerous cycles will be, and the less important the dependency between messages will be.

Definition 2.2 *The messages over edges are one dimensionnal and called LLR, for logarithm likelihood ratio. LLR of a message coming from a variable node will be denoted by v, and u will denote a message coming from a variable node. They are respectively defined by*

$$v = \log \frac{p(y|c=0)}{p(y|c=1)} \tag{2.1}$$

$$u = \log \frac{p(y^{'}|c^{'}=0)}{p(y^{'}|c^{'}=1)} \tag{2.2}$$

(2.3)

where c is the bit value of the node and y denotes all the information available to the node up to the present iteration obtained from edges other than the one carrying v. $c^{'}$ is the bit value of the variable node that gets the message from the check node up to the present iteration obtained from edges other than the one carrying u.

Let us now present the BP algorithm, where messages over edges are one dimensionnal (LLR), with $v = \log \frac{p(y|c=0)}{p(y|c=1)}$, the outcoming message from a bit node and $u = \log \frac{p(y^{'}|c^{'}=0)}{p(y^{'}|c^{'}=1)}$ the outcoming message from a check node. Each decoding iteration is consists of two stages:

- Update of a bit node of degree i (notations on Fig. (2.3))

  $$v_m^{(l)} = u_0 + \sum_{k=1,k\neq m}^{i} u_k^{(l-1)}, \forall m = 1...i$$

  v_m is the message (LLR) over the mth edge coming out of a bit node. The messages u_k are the LLR coming out of a check node and u_0 is the LLR of the channel observation. At the first iteration, every messages u_k are equal to zero.

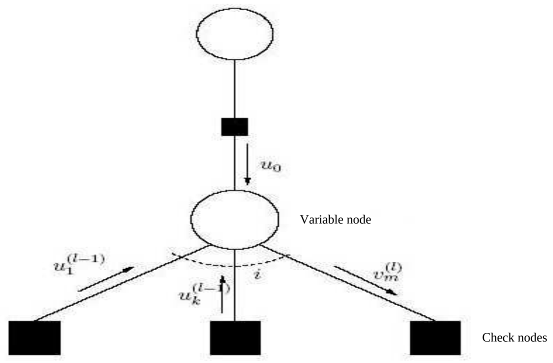

Figure 2.3: Update of variable nodes.

- Update of a check node of degree j (notations on Fig. (2.4))

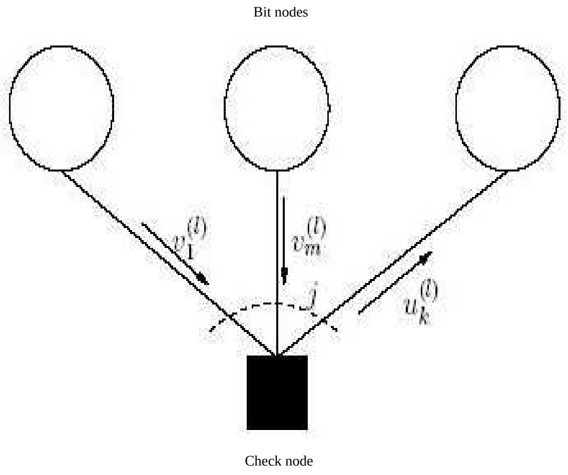

Figure 2.4: Update of check nodes.

$$\tanh \frac{u_k^{(l)}}{2} = \prod_{m=1, m\neq k}^{j} \tanh \frac{v_m^{(l)}}{2}, \forall k = 1...j$$

u_k is the message (LLR) over the kth edge coming out of a check node. The messages v_m are the LLR coming out of a bit node.

One interation of the BP algorithm is accomplished when all messages of the graph have been computed once by the previous equations. After L iterations, we can compute the aposteriori ratio for each bit node given by:

$$v_{app,n} = u_0 + \sum_{k=1}^{i} u_k^{(L)}, \forall n = 1, ..., N$$

And the final decision on binary values of data nodes is taken by:

$$\hat{m}_n = \frac{1 - \text{sign}(v_{app,n})}{2}, \forall n = 1, ..., N$$

2.1.3 Density evolution and Gaussian Approximation

Here the analysis and optimization of LDPC codes presented in an asymptotic context under BP decoding.

2.1.3.1 Density evolution

In [19] and [20], a general method that permits to predict asymptotic preformances is presented: the authors of [20] proved a so-called concentration theorem according to which decoding performances over any random graph converge to its average performance when the codeword length is large enough. Thus, relevant evaluation of performances of LDPC codes is possible only in the limit case of infinite codeword lengths. The infinite graph can then be considered as a tree (cycle-free graph), which allows to consider every messages independent from each other. The method called **Density Evolution**, proposed in [19, 20], follows the evolution of probability densities of messages spreading over the whole graph when using belief propagation. Messages are assumed to be independent and identically distributed (iid). Let express this evolution of densities of messages [19].

Let α denote one bit node to which is associated the observation $u_\alpha = \log\left(\frac{p_{Y|X}(y_\alpha|x_\alpha=1)}{p_{Y|X}(y_\alpha|x_\alpha=-1)}\right)$.
Update of bit node α of degree i:

$$m_{0,\alpha} = u_\alpha + m_1 + \ldots + m_{i-1} \tag{2.4}$$

Update of check node β of degree j:

$$m_{0,\beta} = \gamma^{-1}\left(\gamma(m_1) + \ldots + \gamma(m_{j-1})\right) \tag{2.5}$$

where

$$\gamma : \overline{\mathbb{R}} \to GF(2) \times \mathbb{R}$$
$$z \to \gamma(z) = \left(sign(z), -\log\left(\tanh\frac{|z|}{2}\right)\right)$$

with

$$sign(z) = \begin{bmatrix} 0 & \text{if} & z > 0 \\ 0 & \text{with proba 0.5 if} & z = 0 \\ 1 & \text{with proba 0.5 if} & z = 1 \\ 1 & \text{if} & z < 0 \end{bmatrix}$$

Figure (2.5) shows what density evolution computes: the densities P_{l+1} and Q_{l+1} of the two kinds of messages in function of P_l and Q_l, respectively. Let us do this computation.
Hypothesis:

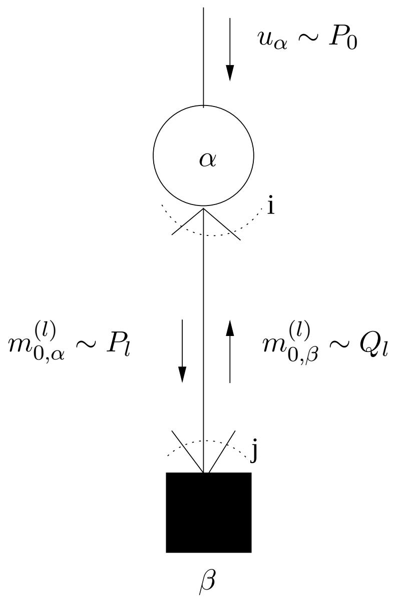

Figure 2.5: Densities of messages over edges.

- The zero codeword for being transmit: $\underline{x} = \underline{1}$.
- Cycle-free graph (all messages are independent) (due to the concentration theorem to cycle-free case for infinite code length).

$P_l(z)$ is the average probability of the codes of the family, such that sub-jacent graph be a tree.

$$P_l(z) = P(m_{0,\alpha}^{(l)} \leq z | x_\alpha = 1)$$

Let $\otimes$ denote the convolution.

- At a bit node :
  By the independence assumption, random variables that are summed up in Eq. (2.4) are independent. So, the density of their sum is the convolution of their densities.

  $$P_{l+1}^{(i)} = P_0 \otimes Q_l^{\otimes(i-1)} \qquad (2.6)$$

  So, for an irregular graph:

  $$P_{l+1} = \sum_i P_{l+1}^{(i-1)}.P(\text{edge connected to bit node of degree } i)$$

  $$P_{l+1} = \sum_i \lambda_i P_0 \otimes Q_l^{\otimes(i-1)}$$
  $$P_{l+1} = P_0 \otimes \lambda(Q_l) \qquad (2.7)$$

  With $\lambda(.) = \sum_i \lambda_i (.)^{\otimes(i-1)}$.

- At a check node :
  Assuming that if
  $$X \sim F_X(z)$$
  Then
  $$\gamma(X) \sim \Gamma(F_X(z))$$
  Using Eq. (2.5), we obtain in the same way:
  $$Q_{l+1}^{(j-1)} = \Gamma^{-1}\left(\Gamma(P_{l+1})^{\otimes(j-1)}\right) \tag{2.8}$$
  So
  $$Q_{l+1} = \sum_j \rho_j Q_{l+1}^{(j-1)}$$
  And by linearity of Γ^{-1}
  $$Q_{l+1} = \Gamma^{-1}\left(\sum_j \rho_j \Gamma(P_{l+1})^{\otimes(j-1)}\right)$$
  $$Q_{l+1} = \Gamma^{-1}\left(\rho(\Gamma(P_{l+1}))\right) \tag{2.9}$$
  With $\rho(.) = \sum_j \rho_j(.)^{\otimes(j-1)}$.
- By combining Eq. (2.7) and Eq. (2.9) we obtain the desired recursion for P_{l+1} in terms of P_l:

  Definition 2.3 *Density evolution presented in [19] is expressed by*
  $$P_{l+1} = P_0 \otimes \lambda\left(\Gamma^{-1}\left(\rho(\Gamma(P_l))\right)\right)$$

The zero codeword for being transmit, they computed equations describing density evolution along the iterations, and this analysis led to the following main results, when binary inputs and symetric outputs channels are considered (remember that a binary input $x = 0, 1$ channel is said to be output-symmetric if and only if the conditionnal probability for the output fulfills $p(y|x = 0) = p(-y|x = 1)$):

- Consistence

  Definition 2.4 *A density of probability $f(x)$ is said to be consistent (i.e. with exponential symmetry) iff*
  $$f(x) = e^x f(-x), \quad \forall x \in \mathbb{R}$$

  According to [19], if the channel is a binary input output-symmetric channel the initial densities of messages are consistent in the sense of Def. (2.4) (Proposition 1 of [19] page 629), and this property is kept along the iterations of decoding (Theorem 3 of [19] page 628).

- Convergence
  According to the consistency conservation properties, (Theorem 7 and 8 of [19]) show that the error probability $P_e^{(l)}$ is a non increasing function of l and converges to zero as l tends to infinity, or equivalently as the message densities tends to Δ_∞.

- Stability condition
  Analysing convergence by density evolution, (Theorem 5 of [19] page 630) shows that studying the stability in the neighborhood of the fixed point allows to determine a necessary condition on the parameters of the code to ensure the convergence of the error probability to zero.

  Theorem 2.1 *Let $S = \int_{\mathbb{R}} f_0(x) \exp^{-x^2/2} dx$, where f_0 is the consistent initial density of messages, and let $\lambda^{'}(x)$ and $\rho^{'}(x)$ denote the derivatives of irregularity polynomials $\lambda(x) = \sum_{i=2}^{t_{cmax}} \lambda_i(x)x^{i-1}$ and $\rho(x) = \sum_{j=2}^{t_{rmax}} \rho_j(x)x^{j-1}$. If $\lambda^{'}(0)\rho^{'}(1) \leq S^{-1}$, then the error probability will converge to zero, otherwise it a non zero value will minimize it.*

  This condition gives an upper bound on the value λ_2.

Under BP decoding, density evolution permits to show that LDPC code have a threshold behavior: there is an optimal threshold $\delta^\star$ of signal to noise ratio beyond which the block error probability converges to zero for an infinite codeword length. In the case of AWGN (Additive White Gaussian Noise) channel, the optimal threshold is given by the signal to noise ratio $\delta^\star = (E_b/N_0)^\star$. Asymptotic performances of BP decoded LDPC codes can be compared with the Shannon limit, and we can then determine what could be the best family of codes for a given channel. This has been done in [19] to find the degree distribution pairs $(\lambda(x), \rho(x))$ that have the lowest threshold at a given code rates.

2.1.3.2 Gaussian Approximation: Mutual Information Evolution

We now present the asymptotic study od LDPC codes over the AWGN channel, what we are going to use from now. Since the previous introduced density evolution method is too complex to be easily applied, a simpler version has been introduced by Chung [6] for the particular case of the transmission over Gaussian channel. Densities of messages are modeled by Gaussian density or mixture of Gaussian densities for a regular and irregular codes, respectively. Thanks to the consistency of messages along stages of BP decoding, this approximation allows to boil down the asymptotic decoding study to study of only one parameter along the iterations. This kind of density approximation has first been introduced by ten Brink for the analysis of the convergence behavior of the iterative decoding of parallel concatenated codes [22] using EXIT charts. This approach has then been used for a lot of concatenated system, like turboequalization, by modeling the input extrinsic information by a Gaussian density $N(m, 2m)$. Due to the difficulty to express density update rules in many of the systems, input-output relation are computed by Monte-Carlo simulations, which is equivalent to an asymptotic study when considering infinite codeword lengths.

The huge advantage of LDPC codes is that their parameters and parameters of the channel allow to find an analytical relation between the value of the study parameter (which can be m for example) between iteration l and $l+1$. In [6] densities of messages u coming from check nodes and v from bit nodes are modeled by consistent Gaussian densities. This seems realistic for messages v, but debatable for messages u (Figure 4 and 5 in [6]). We assume transmission using BPSK modulation over an AWGN channel whose noise variance is σ^2. The zero codeword is transmitted (since for a channel and the BP decoder fulfilling symmetry condition, error probabilities at the lth iteration do not depend on the transmitted codeword). Therfore u_0 is Gaussian $N(2/\sigma^2, 4/\sigma^2)$ which is consistent according to Def. (2.4). Theorem 3 of [19] (p.628) asserts that consistency is kept along iterations for a given binary-input memoryless output-symmetric channel. Then, in order to be able to model message densities to be Gaussian, they must fulfil the consistency condition: $N(m, \sigma^2)$ is consistent if and only if $\sigma^2 = 2m$. That is the reason why the density evolution can be expressed as a one parameter evolution. [6] chose the mean of messages, but we can chose, still under the Gaussian approximation of the message densities, to follow the mutual informations of a virtual AWGN channel, whose output would be messages v or u coming out from bit nodes or check node, respectively.

2.1.3.3 Mutual Information for a Gaussian Consistent Channel

Let v be a message such that $v \sim N(\pm m, 2m)$ that is the output of a binary-input Gaussian channel. The mutual information between v and input c of the virtual channel is given by:

$$x_v = I(v,c) = H(v) - H(v|c)$$

$$x_v = -\int_{\mathbb{R}} (\sum_{c=\pm 1} f_c(c) f_{v|c}(v|c)) \log_2 (\sum_{c=\pm 1} f_c(c) f_{v|c}(v|c)) dv$$
$$+ \int_{\mathbb{R}} (\sum_{c,v} f_c(c) f_{v|c}(v|c) \log_2 (f_{v|c}(v|c))) dv$$

$$x_v = -\frac{1}{2} \sum_{c=\pm 1} f_{v|c}(v|c) \log_2 (\frac{1}{2}(f_{v|c}(v|c=1) + f_{v|c}(v|c=-1))) + \frac{1}{2} \sum_{c=\pm 1} \int_{\mathbb{R}} f_{v|c}(v|c) \log_2 (f_{v|c}(v|c)) dv$$

So

$$x_v = \frac{1}{2} \sum_{c=\pm 1} \int_{\mathbb{R}} f_{v|c}(v|c) \log_2 (\frac{2 f_{v|c}(v|c)}{f_{v|c}(v|c=+1) + f_{v|c}(v|c=-1)}) dv$$

We now shorten $f_{v|c}(v|c)$ by $f(v|c)$. Since the channel is symmetric, we have $f(v|c=-1) = f(-v|c=1)$, and by consistency $f(-v|c=1) = f(v|c=1)\exp^{-v}$, this implies:

$$x_v = \int_{\mathbb{R}} f(v|c=+1) \log_2 (\frac{2 f(v|c=+1)}{f(v|c=+1) + f(v|c=-1)}) dv$$

$$x_v = \int_{\mathbb{R}} f(v|c=+1)\log_2\left(\frac{2f(v|c=+1)}{f(v|c=+1)(1+e^{-v})}\right)dv$$

$$x_v = 1 - \int_{\mathbb{R}} f(v|c=+1)\log_2(1+e^{-v})dv$$

$$x_v = 1 - \frac{1}{\sqrt{4\pi m}}\int_{\mathbb{R}} \log_2(1+e^{-v})\exp\left(-\frac{(v-m)^2}{4m}\right)dv = J(m) \qquad (2.10)$$

J is called the mutual information function, linking mutual information of a Gaussian consistent channel to the mean of messages whose density is $N(m, 2m)$. Equation (2.10) is rewritten as:

$$J(m) = 1 - \mathbb{E}_x(\log_2(1+e^{-x})), \quad x \sim N(m, 2m) \qquad (2.11)$$

J is continuous and a strictly monotonous function, so J^{-1} exists and permits to compute the mean of messages from the mutual information.

2.1.3.4 Evolution Equations

Let $x_{cv}^{(l)}$ and $x_{vc}^{(l)}$ be the mutual information associated to messages coming from check nodes to variable nodes and from variable nodes to check nodes, respectively. From [7], we have the following update relation:

- Update at variable nodes

$$x_{vc}^{(l)} = \sum_{i=2}^{t_{cmax}} \lambda_i J\left(\frac{2}{\sigma^2} + iJ^{-1}(x_{cv}^{(l-1)})\right) \qquad (2.12)$$

- Update at check nodes

$$x_{cv}^{(l)} = 1 - \sum_{j=2}^{t_{rmax}} \rho_j J((j-1)J^{-1}(1 - x_{vc}^{(l)})) \qquad (2.13)$$

Proof: Let α denote a bit node, and $\overline{m}$ the mean of message m. From Eq. (2.4) we have

$$\overline{m}_{0,\alpha}^{(l)} = \overline{u}_\alpha + \overline{m}_1^{(l-1)} + \ldots + \overline{m}_{i-1}^{(l-1)}$$

Assuming that $\overline{m}$ is $\overline{m_{cv}}$ the average over the whole graph of messages coming from check nodes, and by symmetry of the channel:

$$\overline{m}_{0,\alpha}^{(l)} = \frac{2}{\sigma^2} + (\text{degree of } \alpha).\overline{m}_{cv}^{(l-1)} \qquad (2.14)$$

And using the J function, we have $\overline{m_{cv}} = J^{-1}(x_{cv})$. So, in the same way as for general density evolution, making the average over the whole graph of messages coming from variable

nodes leads to Eq. (2.12).
Proof of Eq. (2.13):
Now let β denote a check node. We can rewrite :

$$-\log\left(\tanh(\frac{m_{0,\alpha}}{2})\right) = -\log\left(\frac{e^{m_{0,\alpha}}-1}{e^{m_{0,\alpha}}+1})\right)$$

Now remember that, by Bayes rule:

$$m_{0,\alpha} = \log\left(\frac{P(x=1|y)}{P(x=-1|y)}\right) = \log\left(\frac{P(y|x=1)}{P(y|x=-1)}\right)$$

where x is the random variable describing the codeword bit value associated to the variable node α, and y is the random variable describing all the information incorporated into this message. Let $v[0] = P(y|x=1)$ and $v[1] = P(y|x=-1)$. A message going out of a check node will be generally called u. So

$$-\log\left(\tanh(\frac{m_{0,\alpha}}{2})\right) = -\log\left(\frac{v[0]/v[1]-1}{v[0]/v[1]+1})\right)$$

$$-\log\left(\tanh(\frac{m_{0,\alpha}}{2})\right) = -\log\left(\frac{v[0]-v[1]}{v[0]+v[1]})\right)$$

Using the Discrete Fourier Transform, we have:

$$\begin{bmatrix} DFTv[0] \\ DFTv[1] \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} . \begin{bmatrix} v[0] \\ v[1] \end{bmatrix} = \begin{bmatrix} v[0]+v[1] \\ v[0]-v[1] \end{bmatrix}$$

Finally we have

$$-\log\left(\tanh(\frac{m_{0,\alpha}}{2})\right) = DFTv$$

Defining $DFTv = \log\left(\frac{DFTv[0]}{DFTv[1]}\right)$. This means that a check node sums up the messages, but in the frequential domain. Moreover according to Eq. (2.10) we express the mutual information of $DFTv$:

$$x_{DFTv} = 1 - \mathbb{E}\left(\log_2(1+e^{-DFTv})\right)$$

$$x_{DFTv} = 1 - \mathbb{E}\left(\log_2(1+\frac{e^v-1}{e^v+1})\right) = 1 - \mathbb{E}\left(\log_2(\frac{2e^v}{e^v+1})\right)$$

$$x_{DFTv} = 1 - \mathbb{E}\left(1-\log_2(1+e^{-v})\right) = \mathbb{E}\left(\log_2(1+e^{-v})\right)$$

Which is exactly:

$$x_{DFTv} = 1 - x_v \tag{2.15}$$

We can then describe the evolution of x_{DFTu} exactly in the same way as x_{vc} since the update of $DFTu$ is exactly the same as $m_{0,\alpha}$:

$$DFT\overline{m}_{0,\beta}^{(l)} = DFT\overline{m}_1^{(l)} + \ldots + DFT\overline{m}_{j-1}^{(l)}$$

That says

$$DFT\overline{m}_{0,\beta}^{(l)} = (j-1)DFT\overline{m}_{vc}^{(l)}$$

With $DFT\overline{m}_{vc}^{(l)} = J^{-1}(DFTx_{vc}) = J^{-1}(1-x_{vc})$, we obtain:

$$x_{DFTu} = J\left((j-1)J^{-1}(1-x_{vc})\right)$$

Finally over the whole graph:

$$x_{DFTu} = \sum_j \rho_j J\left((j-1)J^{-1}(1-x_{vc})\right)$$

Applying Eq. (2.15), we proved Eq. (2.13):

$$x_{cv}^{(l)} = 1 - \sum_{j=2}^{t_{rmax}} \rho_j J((j-1)J^{-1}(1-x_{vc}^{(l)}))$$

□

Note that we can also express the evolution by following the mean of messages over the graph when decoding. To do so, we need the function ϕ.

Definition 2.5

$$\phi(x) = 1 - \frac{1}{\sqrt{1\pi x}} \int_{\mathbb{R}} \tanh\frac{u}{2} e^{\frac{-(u-x)^2}{4x}} du, \text{ if } x > 0 \tag{2.16}$$

$$\phi(x) = 1, \text{ if } x = 0 \tag{2.17}$$

$$\tag{2.18}$$

We then express the evolution of the mean, computed exactly in the same way as the mutual information.

$$m_u^{(l)} = \phi^{-1}(1 - [1 - \sum_{i=2}^{t_{cmax}} \lambda_i \phi(m_v^{(l)})]^{j-1}) \tag{2.19}$$

$$m_v^{(l)} = s + (b-1)\sum_{j=2}^{t_{rmax}} \rho_j m_u^{(l-1)} \tag{2.20}$$

$$\tag{2.21}$$

The combination of Eq. (2.12) and (2.13) yields the EXIT chart of the code defined by $(\lambda(x), \rho(x))$. It is an explicit non-linear function of $(\lambda(x), \rho(x))$ and of parameters of the channel (σ^2 in our case):

$$x_{vc}^{(l)} = F(\underline{\lambda}, \underline{\rho}, x_{vc}^{(l-1)}, \sigma^2) \tag{2.22}$$

where $\underline{\lambda} = [\lambda_2, \ldots, \lambda_{t_{cmax}}]^T$ and $\underline{\rho} = [\rho_2, \ldots, \rho_{t_{rmax}}]^T$. Several optimization methods exist, which can consider $\rho(x)$ as fixed in order to make F linear in $\underline{\lambda}$. The convergence threshold is defined by the smallest E_b/N_0 beyond which $x_{vc}^{(l)} \to 1$ when $l \to \infty$.

2.1.3.5 Stability Condition

Under the Gaussian approximation, [6] provides a looser stability condition in the neighborhood of the fixed point:

$$\lambda_2 \leq \lambda_2^\star = \frac{e^{1/2\sigma^2}}{\prod_{j=2}^{t_{rmax}} (j-1)^{\rho_j}} \tag{2.23}$$

whereas the general condition given by density evolution is:

$$\lambda_2 \leq \lambda_2^\star = \frac{e^{1/2\sigma^2}}{\sum_{j=2}^{t_{rmax}} \rho_j (j-1)} \tag{2.24}$$

Jensen’s inequality shows that the first condition is looser than the secund one:

$$\prod_{j=2}^{t_{rmax}} (j-1)^{\rho_j} \leq \sum_{j=2}^{t_{rmax}} \rho_j (j-1)$$

The stability condition is very important because it controls the mutual information behavior at very low error probabilities (or equivalently, when the mutual information is near to 1).

Chapter 3

State of the Art

This chapter is meant to first introduce principal possibilities for optimizations using the characteristics of LDPC codes, and then we focus on already studied UEP optimization strategies, before in next chapter presenting our own strategies.

3.1 Optimization of LDPC codes over some channels

3.1.1 Optimisation over AWGN Channel

One of the most common criteria is to minimize the convergence threshold of LDPC code of code rate R:

$$\delta^{\star} = \arg\min_{\sigma^2}\left(\frac{1}{2R\sigma^2} | F(\underline{\lambda}, x, \sigma^2) \geq x, \forall x \in [0,1]\right) \tag{3.1}$$

In order to simplify the optimization, we can first only optimize $\lambda(x)$ for a fixed $\rho(x)$. Optimization is carried out in two stages:

- Maximization of code rate for fixed $\rho(x)$ and σ^2:
  The most used cost function consists in optimizes the code rate R. The problem becomes linear since the cost function and the constraints become linear in $\lambda(x)$, and thus can be solved by linear programming. For given $\rho(x)$ and σ^2, we determine $\lambda(x)$ that maximizes R. Let $\underline{\lambda} = [\lambda_2, \ldots, \lambda_{t_{cmax}}]^T$ and $\underline{1/t_c} = [1/2, \ldots, 1/t_{cmax}]^T$. Remembering the relation
  $$R = 1 - \frac{\sum_{j=2}^{t_{rmax}} \rho_j / j}{\sum_{i=2}^{t_{cmax}} \lambda_i / i}$$
  the optimization problem can be expressed as: $\underline{\lambda}_{opt} = \arg\max_{\underline{\lambda}}(\underline{1/t_c}^T \underline{\lambda})$ under the constraints:

  - Mixture constraint:
$$\underline{\lambda}^T\underline{1} = 1$$
  - Proportion constraint:
$$\forall i = 2 \ldots t_{cmax}, \lambda_i \in [0, 1]$$
  - Convergence constraint:
$$\forall x \in [0, 1], F(\underline{\lambda}, x, \frac{2}{\sigma^2}) \geq x$$
  - Stability constraint ;
$$\lambda_2 \leq \frac{e^{1/2\sigma^2}}{\sum_{j=2}^{t_{rmax}} \rho_j(j-1)}$$

- Threshold minimization

Definition 3.1 *A concentrated degree distribution over check nodes is defined by the polynomial*
$$\rho(x) = \rho x^{k-1} + (1-\rho)x^k$$

From here we will call improperly **concentrated code** a code with such a concentrated ρ polynomial.

Once R is determined, we increase σ^2 as long as R can be reached. The resulting threshold (E_b/N_0) is $\delta = \frac{1}{2R\sigma^2}$. $\rho(x)$ can now be optimized. From Theorem 2 of [6] (page 665), we learn that a concentrated degree distribution optimizes the speed of convergence. That's why, assuming $\rho(x)$ in this form, this optimization means optimizing one parameter $\overline{\rho} = k - \rho$ which is the average degree of connectivity of check nodes. Finally, the degree distribution pair $(\lambda(x), \rho(x))$ that minimizes the threshold is chosen.

Hence, the evolution of mutual information in the form of an asymptotic study for infinite codeword length shows that an optimal value $\overline{\rho}_{opt}$ exists for each t_{cmax}, that allows to reach a minimum convergence threshold. This threshold approaches Shannon capacity when t_{cmax} increases, and then $\overline{\rho}$ too (these two parameters are strongly linked together), see Fig. (3.1) extracted from [7].

This is similar to Gallager's result under maximum likelyhood decoding, according to which densest codes are the best regarding the convergence threshold. Two important remarks must be made:

- This asymptotic result is verified only for large codeword length (e.g. $N = 30000$), but for short ones (e.g. $N = 1000$), the tree assumption is not valid enough. Cycles in

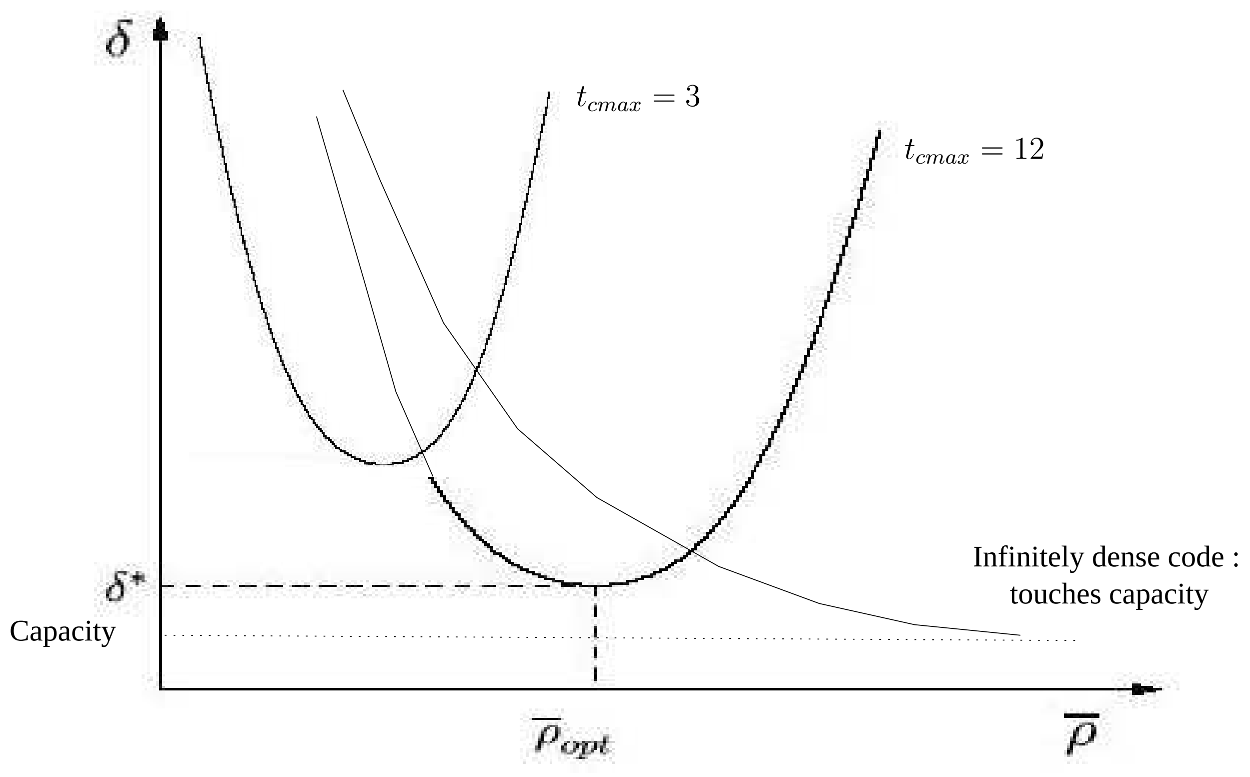

Figure 3.1: Gap to the capacity, for given code rate.

the graph of densest codes worsen them, breaking message independency and thus BP optimality. Less dense codes have higher girth (the girth is the length of the smallest cycle), which ensures best efficiency of BP decoding. Hence the code hierarchy for ”short” code length is the contrary of the expected one.

- A second remark, important for our work on UEP, is to highlight that the convergence threshold deals with the global behavior of the code, and we will see that a lower BER than the global one can be achieved for most protection classes, even if the global threshold is increased. Differences between classes will depend on which offset on global threshold is allowed, but this will be seen in detail in next chapter.

3.1.2 Optimization over other channels

Similar optimizations can be performed for other channels (BEC, BSC, Laplace, AWGN [19, 20], Rayleigh), among which optmization strategies on multiple access channels, multicarriers channels, memory channel or high spectrum efficiency channels. We restricted ourselves to an introduction into the usual LDPC optimization for AWGN channels. This offers some insights and links to our UEP optimizations.

3.2 UEP Optimization: An Introduction

Let us consider the transmission of media like voice, fixed image, or video, whose characteristics are heterogeneous sensibility to errors in the data stream. The code stream of source-encoded blocks is hierarchically structured and contains:

- Headers to describe the type and parameters of compression.
- Structure control data that are indicators of code stream synchronization, position, or indexing.
- Compressed data delivered from the source coder: e.g. speech encoder coefficients, image texture, or movement vectors.

This constitutes a very logical ensemble, and it is obvious that errors on headers is a disaster since true reconstruction parameters of the compression are not known at all. The final result at the receiver will then be completely different according to error localization. Sensibility classes can be distinguished inside compressed data, according to the compression system used. For video, errors on movement vectors are more disturbing than errors on texture. The same holds for low frequency coefficients in fixed images, whereas high frequency ones are generally associated to details. Actually uniformly protecting such a code stream would be sub-optimal. This highlights the interest of realizing unequal error protection by modifying on the irregularity of a code.

When speaking about irregularity for UEP, we distinguish systems with irregularity caused by puncturing and/or pruning, and those with intrinsic irregularity:

- Irregular punctured/pruned systems: Puncturing consists of not emitting some bits of the codeword, thereby decreasing the initial code rate R. The receiver knows the puncturing pattern, and considers not transmitted bits as erasures. This technique worsens the performance of the code allowing to obtain a wide range of rates. Unequal error protection can then be achieved by applying different code rates to each part of the source data, according to the required robustness. Another way of adding irregularity is using a pre-processing block before the code, in order to prune it. Puncturing and pruning will be the chosen method to realize UEP in our work, and has been further studied in [24] for Turbo Codes.
- Intrinsic irregularity systems:

  One can think of systems without a posteriori added irregularity block, but with intrinsic unequal protection properties.

  - [4,16] presented unequal error protection linear codes. Linear codes can achieve different protection inside a codeword, i.e., averaged error probability is not uniform inside the codeword. These properties are due to algebraic characteristics of the parity-check matrix, considering maximum likelihood decoding (MLD, syndrome decoding). The protection level of the ith bit is associated to its local minimum distance, which is exactly the minimum codeword weight with a one in the ith position. This is also the degree of independence of the ith column in the parity-check matrix: the minimum number of columns that are included in linear combination that leads to zero, with coefficient one at the ith column. The local minimum distance associated to each bit of the codeword determines the maximum number of errors in the whole codeword, still allowing this bit to be corrected. The local minimum distance can be greater than the global one, which means that the ith bit can be corrected even if the whole codeword is not recovered by MLD. That explains the interest of such codes under MLD, when considering in the previously mentioned JPG transmission, for example. Construction methods of such codes have been presented, but a big problem is the poor control that we can have over the proportions of the classes, which can be very disturbing for the latest application.

- Another family of such irregular coding systems is multi-level coded modulation. Each bit of a symbol is associated to a given code, which differs from others by its code rate. Then the protection level of bits depends on the code, and on the position in constellation labelling, which means that two kinds of irregularities can be exploited.
- LDPC codes can be punctured [9] in order to create average irregularity. Puncturing influences the code rate: average performances differ between two codewords encoded with different puncturing patterns. Nevertheless it is more suited to make use of an irregularity that leads to unequal error protection of bits inside a codeword: most connected bits will have lower error probability. This has been highlighted in [6], and applied for optimization for several transmit channels. The optimization for AWGN done by Poulliat in [17] will be presented in the next chapter.

In the following we will present our work that concentrated on two approaches. The first considers LDPC code as a linear block code and optimizes the code according to the local minimum distances [4, 16]. The second approach is an asymptotic optimization for BP decoding and is based on pruning and puncturing of a mother code.

Chapter 4

UEP LDPC Codes

In the previous chapter, we presented the family of LDPC codes, its parameterization and the asymptotic study of belief propagation (BP) decoding: density evolution under gaussian approximation. We are now going to focus on the possibilities that both dimensions of their irregularity profile provide, to achieve unequal error protection inside a codeword.

In the first section of this chapter the usual UEP optimization of LDPC codes is presented, which allocates most important bits to the most connected variable nodes. We then develop a pruning method for linear block codes, completely derived from [4, 16], that has no real practical interest. Finally check optimization is carried out. This is not usual method due to the fact that check profile must be concentrated. But this will be seen in detail. We then look at the pruning method to optimize the check irregularity, and finally analyse briefly what an optimal puncturing of such an UEP code could be.

4.1 UEP properties created by irregularities over bit nodes

We now present the optimization realized by C. Poulliat in [17, 18]. This was a quite unusual optimization when it was presented because known methods were focusing on the global average performances, such as the convergence threshold on a given channel or puncturing pattern. The global convergence of the error probability to zero is usually the only one cost considered, because the parameterization, and the evolution equations of LDPC codes do not distinguish information and redundancy inside codeword and consider an infinite code length and an infinite number of iterations. We try to see how to specialize those equations for UEP created by irregularities on bit and check nodes. We are interested here in local convergence of a part of codeword, associated to sensitive data, for finite number of iterations, which determines the following kind of optimization.

4.1.1 Parameterization and Asymptotic Study of such UEP LDPC Codes

Let us assume the proportion of each class C_k of sensibility defined by the source. We use from here

$$\underline{\alpha} = \{\alpha_k | k = 1..N_c - 1\}$$

where N_c is the number of classes over the whole codeword, information bits are spread over the $N_c - 1$ first classes, the N_cth class containing the whole redundancy. We have

$$\sum_{k=1}^{N_c-1} \alpha_k = 1$$

The proportions of bits in codeword inside classes are

$$\underline{p} = (\alpha_1 R, \dots, \alpha_{N_c-1} R, 1 - R)$$

We still have $\rho(x) = \sum_{j=2}^{t_{rmax}} \rho_j x^{j-1}$, but define

$$\lambda^{(C_k)}(x) = \sum_{i=2}^{t_{cmax}^{(k)}} \lambda_i^{(C_k)} x^{i-1}$$

and

$$\tilde{\lambda}^{(C_k)}(x) = \sum_{i=2}^{t_{cmax}^{(k)}} \tilde{\lambda}_i^{(C_k)} x^{i-1}$$

which are polynomials of proportion of edges linked to degree i bit nodes belonging to the kth class, and proportion of degree i bit nodes belonging to C_k.
Specified evolution equation of mutual information can then be derived from Eq. (2.12) and Eq. (2.13):

- Update check nodes

$$x_{cv}^{(l)} = 1 - \sum_{j=2}^{t_{rmax}} \rho_j J\left((j-1)J^{-1}(1 - x_{vc}^{(l)})\right) \tag{4.1}$$

- Update variable nodes

$$x_{vc}^{(l)} = \sum_{k=1}^{N_c} \sum_{i=2}^{t_{cmax}} \lambda_i^{(C_k)} J\left(\frac{2}{\sigma^2} + J^{-1}(x_{cv}^{(l-1)})\right) \tag{4.2}$$

Equation (4.2) is obtained by adding the mutual information coming into each class of bitnodes since there is no overlap between the classes. We then can derive convergence and stability conditions from the fact that $\lambda_2 = \sum_{k=1}^{N_c} \lambda_2^{(C_k)}$.

4.1.2 Cost Function for such UEP

Under the Gaussian approximation, [6] gives the error probability associated to degree i bit node at the lth iteration:

$$P_i^{(l)} = Q\left(\sqrt{\frac{\frac{2}{\sigma^2} + iJ^{-1}(x_{cv}^{(l)})}{2}}\right) \tag{4.3}$$

Proof: Let X be the random variable that denotes the a posteriori probability of one bit. Let us express the error probability of one bit:

$$\begin{aligned} P_e(bit) &= P(X \leq 0|bit=0).\frac{1}{2} + P(X \geq 0|bit=1).\frac{1}{2} \\ &= P(X \leq 0|bit=0).\frac{1}{2} + P(-X \leq 0|bit=1).\frac{1}{2} \end{aligned} \tag{4.4}$$

Thanks to the symmetry of the channel:

$$P_e(bit) = P(X \leq 0|bit=0) \tag{4.5}$$

Let X_u be the random variable whose distribution is $N(0,1)$. Under Gaussian approximation, X is Gaussian consistent for any iteration according to the symmetry of the channel and the conservation of the consistence along the iterations: $X|bit=0 \sim N(m, \sigma^2 = 2m)$. Let now X denote improperly $X|bit=0$, but this will make the expressions clearer. We then have:

$$\begin{aligned} X_u &= \frac{X-m}{\sigma} \\ X_u &= \frac{X}{\sigma} - \frac{\sigma}{2} \end{aligned}$$

Which yields

$$X = \sigma X_u + \frac{\sigma^2}{2}$$

Inserted in Eq. (4.5), we finally have

$$\begin{aligned} P_e(bit) &= P(\sigma X_u + \frac{\sigma^2}{2} \leq 0) \\ &= P(X_u \leq -\frac{\sigma}{2}) \\ &= P(X_u \geq \frac{\sigma}{2}) \end{aligned}$$

Finally

$$P_e(bit) = Q\left(\frac{\sigma}{2}\right)$$

Now we remember that

$$\frac{\sigma}{2} = \sqrt{\frac{\sigma^2}{4}} = \sqrt{\frac{m}{2}}$$

We conclude that

$$P_e(bit) = Q\left(\sqrt{\frac{m}{2}}\right) \tag{4.6}$$

By replacing the mean m of the sum of messages coming into that bit by the its expression of Eq. (2.14), we obtain Eq. (4.3). □

Beyond the asymptotic convergence threshold, $J^{-1}(x_{cv}^{(l)})$ is an increasing function of l. Since Q is a decreasing function, 4.3 shows that at a given number of iterations, the more a bit is connected, the more it is protected, considering the associated error probability (the convergence of this node is faster).

The protection of one class can then be expressed as

$$P_l^{(C_k)} = \frac{1}{\alpha_k R} \sum_{i=2}^{t_{cmax}} \tilde{\lambda}_i^{(C_k)} Q\left(\sqrt{\frac{\frac{2}{\sigma^2} + iJ^{-1}(x_{cv}^{(l)})}{2}}\right) \tag{4.7}$$

and then be bounded by

$$Q(\sqrt{\frac{\frac{2}{\sigma^2} + \overline{\tilde{\lambda}^{(C_k)}} J^{-1}(x_{cv}^{(l)})}{2}}) \le P_l^{(C_k)} \le Q(\sqrt{\frac{\frac{2}{\sigma^2} + t_{cmin}^{(k)} J^{-1}(x_{cv}^{(l)})}{2}}) \tag{4.8}$$

with

$$\overline{\tilde{\lambda}^{(C_k)}} = \frac{1}{\alpha_k R} \sum_{i=2}^{t_{cmax}} \tilde{\lambda}_i^{(C_k)} i$$

which is the average degree of a bit node in C_k. The minimum bound is directly obtained by the convexity of the $Q(.)$ function, and the maximum bound by the decreasing of the $Q(.)$ function.

The derived linear programming algorithm is meant to achieve a joint optimization of $\overline{\tilde{\lambda}^{(C_k)}}$ and $t_{cmin}^{(k)}$, under the constraints of proportion, code rate, convergence, stability, and hierarchical constraints (since the optimization is sequential, the irregularity profile of already optimized classes must not be modified by the current optimization).

4.2 Short LDPC: UEP linear block code optimization

We are now going to present some results regarding UEP optimization of LDPC code considering it as linear block code under maximum likelihood decoding. Our optimization algorithm is based on computating of degree of independence of columns of the **H** matrix. This approach has a huge drawback: due to computation time, it can be applied only to very short codes ($N < 50$), and thus excludes required practical approach. However this way still serve as a first step towards understanding UEP created by irregularities over check nodes.

4.2.1 Algebraic Properties and Unequal Error Protection

Masnick in [16] and then Boyarinov in [4] presented linear unequal error protection codes under MLD.

Definition 4.1 *Inside a codeword, the local minimum distance d_i of the ith bit is exactly the minimum codeword weight with one in the ith position.*

Definition 4.2 *The degree of independence of the ith column of the parity-check matrix of the code is the minimum number of columns that are included in a linear combination that equals zero, with a coefficient one at the ith column.*

Lemma 4.1 *The local minimum distance d_i of the ith bit of the codeword is the the degree of independence of the ith column of the parity-check matrix of the code.*

Definition 4.3 *The protection level f_i of the ith bit of a codeword is the maximum number of errors in the codeword that still allows the correction of this bit.*

$$f_i = \lfloor \frac{d_i - 1}{2} \rfloor$$

Thus, the local minimum distance associated to each bit of the codeword determines the maximum number of errors in the whole codeword that still allows the correction of this bit. The local minimum distance can be greater than the global one, which means that the ith bit can be corrected even if the whole codeword cannot be restored by MLD.
Those algebraic properties can be linked to *Majority Logic Decoding* presented in [3] which works on a poorer difinition of local minimal distance to simplify the decoding.

4.2.2 The Derived Algorithm

Classes are not defined by their proportions at the beginning, which is another drawback of the linear coding approach. Actually, we do not intend to construct an arbitrary linear block code, but a subcode of a mother code from which we choose the right columns to be removed in the parity-check matrix in order the resulting parity-check matrix be the matrix defining a code with the required properties.
Here are the parameters of the optimization:

- Let us denote the set of initial (i.e. mother code) local minimum distances by $\underline{w_1} = [w_1(1), ..., w_1(N_0)]$, which has to evolve to $\underline{w_2}$ along the optimization.
- Let c_i be the number of different zero linear combinations of w_i columns (we have $c_i \geq w_i$), and $C_i$the set of corresponding indices ($card(C_i) = c_i$).

In our example:

- We start from a regular mother code with parameters ($N_0 = 20, t_c = 3, t_r = 6$), the number of info bits of the subcode $K_1 = 3$. This defines a subcode length of $N_1 = N_0 - (K_0 - K_1) = 13$.
- Let the required UEP profile be $\underline{w_2} = [w_2(1), ..., w_2(K_1)]$, which are the required local minimum distances on info bits of the subcode.
- U is the vector where indexes of columns of **H** to be pruned away are stored (length $K_0 - K_1$).
- $\underline{w_{1init}}$ is the initial $\underline{w_1}$ vector, ordered in decreasing order, before optimization of a selected column.

We sequentially choose the best column to be optimized by running in w_{1init} from left to right Fig. (4.1).

4.2.3 Results under Maximum likelihood and Belief Propagation Decoding

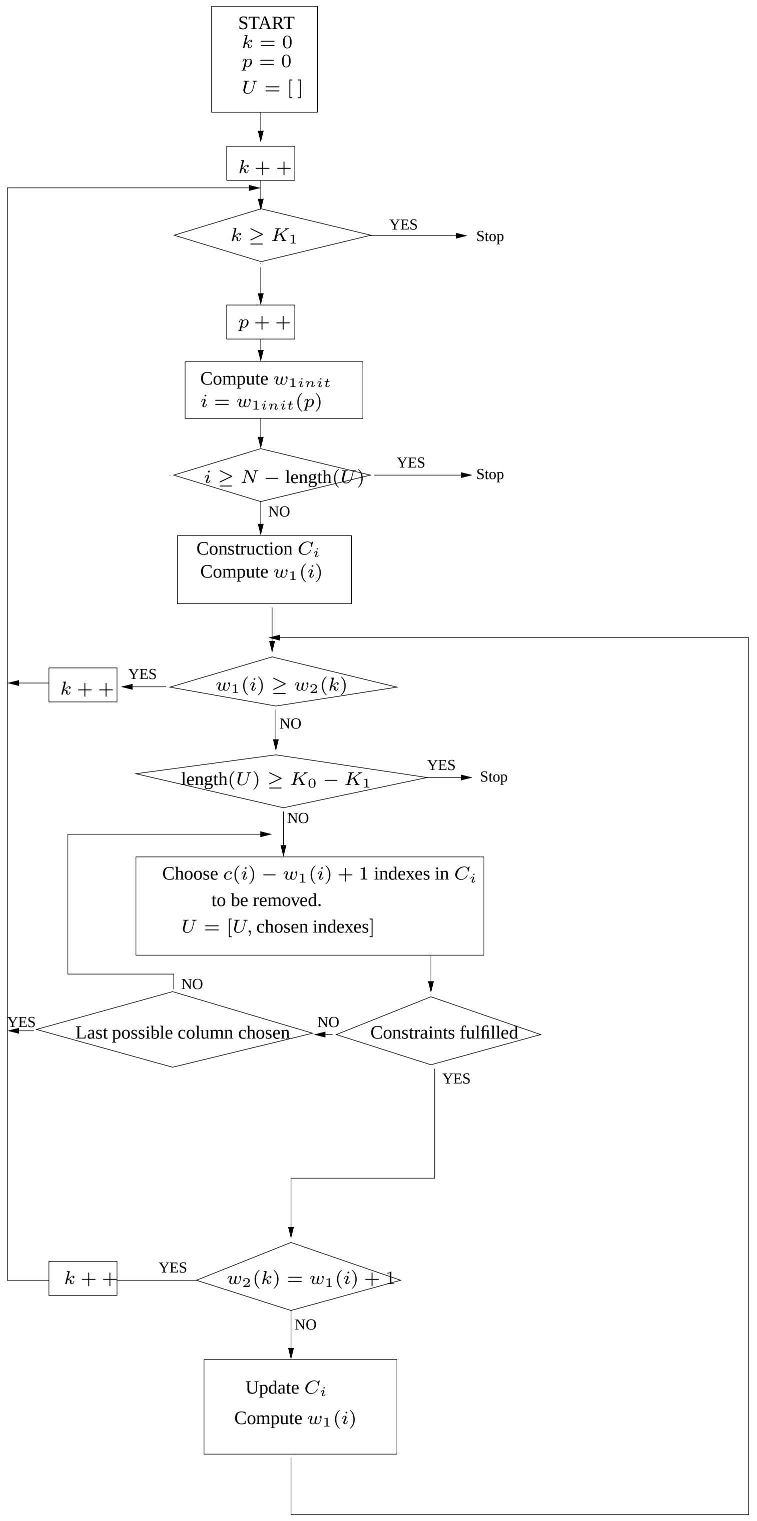

Figure 4.1: Sheme of the encoder of the subcode

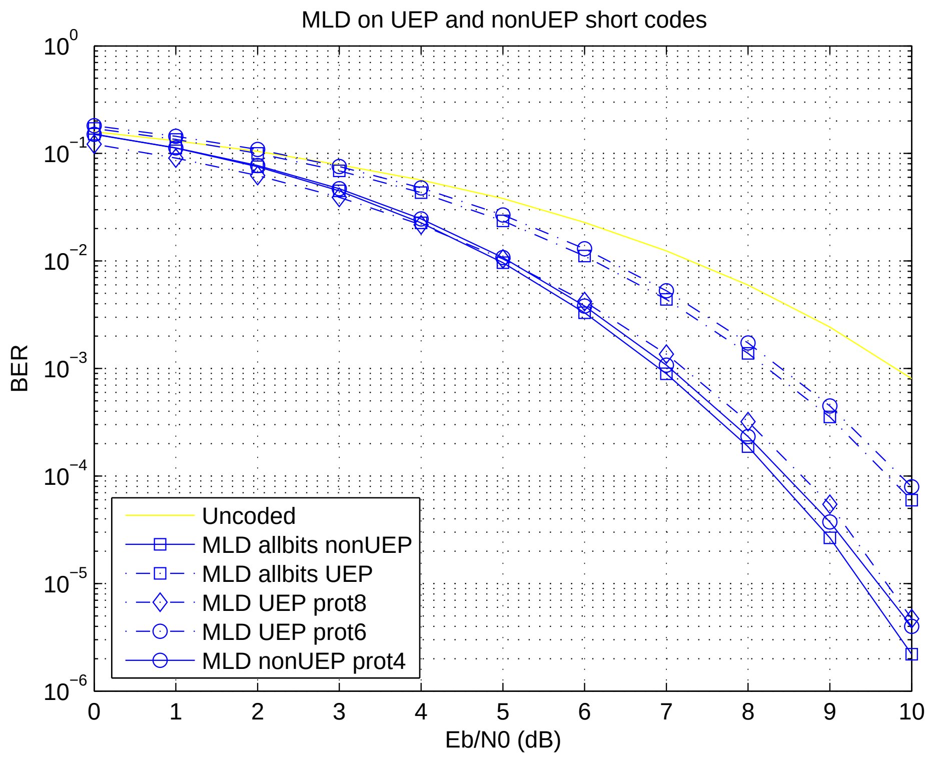

Figure 4.2: MLD on UEP and non UEP short codes

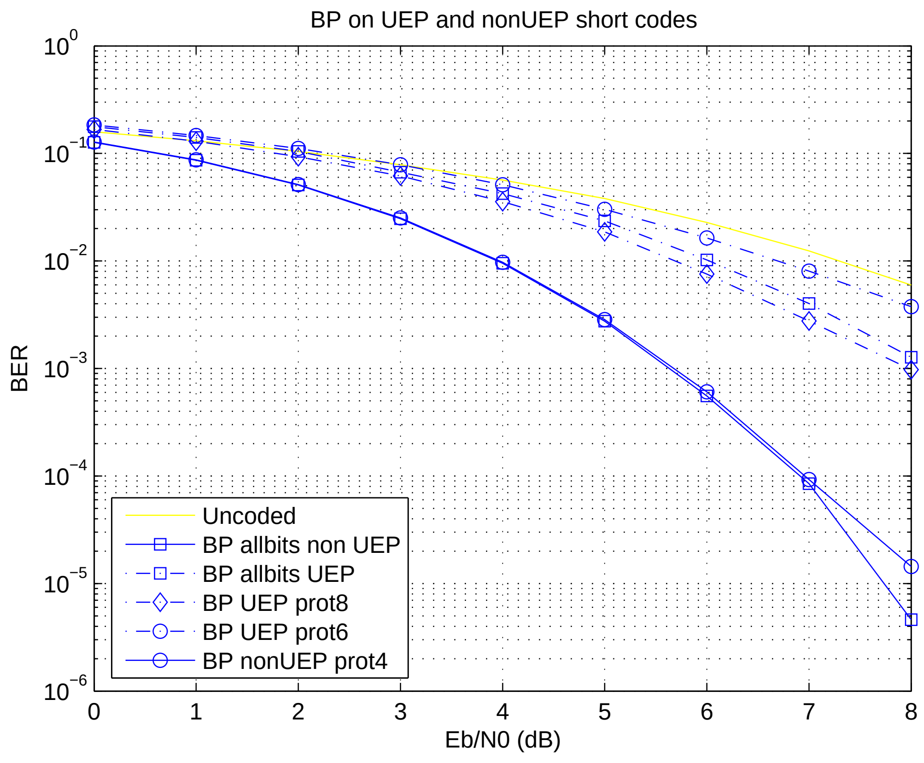

Figure 4.3: BP on UEP and non UEP short codes

Figures (4.2) and (4.3) show that the code of length 20 seems better than the UEP length 13 one in any case, although biggest differences between classes can be seen in the last code. This could have two possible reasons:

- The Gallager's result: the densest code is the best under MLD (and even BP because it is longer in our case)

- The rescaling due to the huge differencies between the two code rates ($1/2$ and $3/13$)

Moreover, considering the mother code as an LDPC code under BP decoding, we would say that it is not UEP at all since it is completely regular (on bit and check nodes). On the contrary, considering it as a linear block code under syndrome decoding, it has some UEP properties, since local minimum distances are either 4 or 6. This specificity of unequal error protection that depends on the code and on the means of decoding, has to be further explained.
All the essential ingredients for the explanation are already available from [23], where we extract some required definitions.

Definition 4.4 ***(Cycle)*** *A cycle of length $2d$ is a set of d variable nodes and d constraint nodes connected by edges such that a path exists that travels through every node in the set and connects each node to itself without traversing an edge twice.*

Definition 4.5 *(**C_d Cycle set**) A set of variable nodes in a bipartite graph is a C_d set if (1) it has d elements, and (2) one or more cycles are formed between this set and its neighboring constraint set. A set of d variable nodes does not form a C_d set only if no cycles exist between these variables and their constraint neighbors.*

Definition 4.6 *(**S_d Stopping set**) A variable node set is called an S_d set if it has d elements and all its neighbors are connected to it at least twice.*

Definition 4.7 *(**L_d Linearly dependent set**) A variable node set is called an L_d set if it is comprised of exactly d elements whose columns are linearly dependent but any subset of these columns is linearly independent.*

According to Lemma 1 and Theorem 2 of [23], we can summarize the relationship between C_d, S_d and L_d in Fig (4.4).

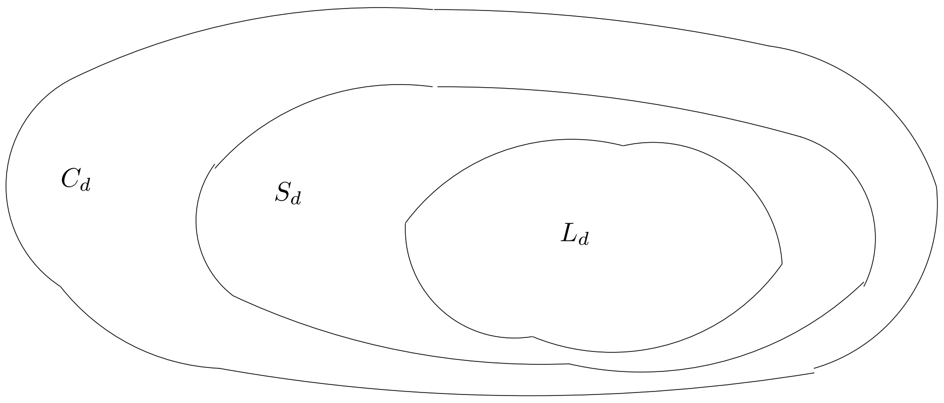

Figure 4.4: Venn diagramm showing relationship of C_d, S_d and L_d.

That says that the local minimum distance of each variable node is associated to a cycle of length $2d_{min}$, but the converse is not true (the variable nodes that form a cycle are not necessarily dependent). So an ML decoder can successfully decode an erased stopping set because the associated local minimum distances of the nodes are not necessarily low, whereas such a stopping set can never be overcome by an iterative decoder.

Thus, the local minimum distance of one bit, defined in Def. (4.1), is an upper bound on the depth plus one of the biggest local tree that starts at the considered variable node. Let us then analyse the case of a regular LDPC code:

- For a finite code length N:
  - Under Maximun Likelihood Decoding, the code is decoded in an optimal way, in the sense of the minimum distance. The code can have UEP properties due to its local minimum distances, associated to some cycles in the graph, that

can be different from each other. The UEP properties are then dependent on the realization of the **H** matrix. The local properties of the code are taken into account by the MLD.

  - The Belief Propagation is sub-optimum decoding, and quite ”global” in the sense that it does not take into account local properties randomly created with the **H** matrix. Local differences will be created by the local sub-optimalities of BP decoding at finite code length, and some of these sub-optimalities are associated to small local minimum distances.

- For an infinite code length: Belief Propagation decoding is the Maximum Likelihood Decoding. The minimum distance tends to infinity and the length of the smallest cycle, called the girth, too. Therefore, all local minimum distances tend to infinity too and UEP properties defined by the two means of decoding tend to be the same.

Thus, UEP properties depend on the code and also on the way that it is decoded: the optimization must be done as a function of the chosen decoding method. This is, of course, practically determined by the code length since at low N ($N < 500$), MLD will be used, otherwise BP.

4.3 Optimization of the Check-Node Profile

We first describe the specific parameterization of UEP LDPC codes, then the density evolution for this *detailed representation*, and finally our optimization algorithm. It is based on the optimization of the check node irregularity profile, which still considers local performances, but not only for a finite number of iterations anymore.

4.3.1 Parametrization of UEP LDPC codes

A very useful parameterization for our work is the *detailed representation* of irregular LDPC codes presented by Kasai in [11]. They constructed new families of LDPC codes which are sub-ensembles of conventional irregular LDPC code ensembles. The detailed representation they adopted allows to design optimal codes more accurately by restricting choices for the interleaver.

Definition 4.8 *Let B and D be two sets or irregularity degrees. A function* $\pi : B\times D \to [0,1]$ *is said to be the* joint degree distribution *of* (B,D) *if* $\sum_{b\in B}\sum_{d\in D}\pi(b,d) = 1$*. This function describing the connections between the different degrees of the code is called the* ***detailed representation*** *of the code.*

Definition 4.9 *We also define* marginal degree distributions of variable and check blocks with respect *to* π *by*

$$\hat{\lambda}(x) := \sum_{b \in B} \hat{\lambda}_b x^{b-1}$$

and

$$\hat{\rho}(x) := \sum_{d \in D} \hat{\rho}_d x^{d-1}$$

with

$$\hat{\lambda}_b := \sum_{d \in D} \pi(b,d) \qquad , \qquad \hat{\rho}_d := \sum_{b \in B} \pi(b,d)$$

Definition 4.10 *For* $\pi(b,d)$*, we define two fractions*

$$\lambda(b,d) := \frac{\pi(b,d)}{\hat{\rho}_d} \qquad , \qquad \rho(b,d) := \frac{\pi(b,d)}{\hat{\lambda}_b}$$

It can be verified that $\rho(b,d)$ equals the fraction of edges connecting nodes of degree b and d among all edges of degree b.

This *detailed representation* can be used to describe the methods for different Poisson constructions explored by MacKay et al. in [15]. They distinguish a Poisson, a super-Poisson and a sub-Poisson construction, which differ from each other by the variance of the distribution of the high weight columns per row.
In this work, we focus only on the influence of the check node irregularities on the UEP properties, i.e., on the distribution of rows weight. We consider codes with a regular bit nodes profile (all of the same degree).

4.3.2 Density evolution for the detailed representation and UEP properties

Theorem 3 in [11] states that under local tree assumption of depth 2T and some other constraints, for any $1 \leq l \leq T$, the distribution functions $Q_l(d)$ of messages originating from check nodes of degree d and $P_l(b)$ of messages originating from bit nodes of degree b are equal to

$$Q_l(d) = \Gamma^{-1}(\Gamma(\sum_{b \in B} \lambda(b,d) P_l(b))^{\otimes(d-1)}) \tag{4.9}$$

$$P_l(b) = P_0(b) \otimes (\sum_{d \in D} \rho(b,d) Q_{l-1}(d))^{\otimes(b-1)}) \tag{4.10}$$

Note that $P_l(b)$ and $Q_l(d)$ do not depend on d and b, respectively. These expressions are directly derived from equations (2.8) and (2.6), respectively.
Let s still denote $2/\sigma^2$. From Eq. (4.9), we can derive a Gaussian approximation for the detailed representation:

$$m_u^{(l)}(d) = \phi^{-1}(1 - [1 - \sum_{b \in B} \lambda(b,d)\phi(m_v^{(l)}(b))]^{d-1}) \tag{4.11}$$

$$m_v^{(l)}(b) = s + (b-1) \sum_{d \in D} \rho(b,d) m_u^{(l-1)}(d) \tag{4.12}$$

and

$$x_{cv}^{(l)}(d) = 1 - J\left((d-1)J^{-1}\left(1 - \sum_{b \in B} \lambda(b,d) x_{vc}^{(l)}(b)\right)\right) \tag{4.13}$$

$$x_{vc}^{(l)}(b) = J\left(s + (b-1)\sum_{d \in D} \rho(b,d) J^{-1}\left(x_{cv}^{(l-1)}(d)\right)\right)$$

We may mention the equality between $m_u(d)$ in (4.11) and f_j in [6] in the case that B denotes the degrees of irregularity over the whole graph. d and j denote the same thing: the connectivity degree of one check node. Let t be the mean over the whole graph of messages coming out of check nodes. $f_j(s,t)$ is the mean of messages coming out of a check node of degree j in terms of $s = 2/\sigma^2$ and t the mean at the previous iteration. In [6] we find

$$f_j(s,t) = \phi^{-1}\left(1 - \left(1 - \sum_{i=2}^{t_{cmax}} \lambda_i \phi(s + (i-1)t)\right)^{j-1}\right)$$

From this equation we observe that the lower is j (or equivalently d in our work, the higher are the messages coming out of check node of degree j (d), i.e. the faster is the local convergence of these. Figure (4.5) extracted from [6] shows the gaps $f_j(s,t)-t$ that represent this local convergence. The curves are parameterized by j. t increases as the number of iterations increases, from left to right. We clearly see the previous quoted effect of j on the local convergence. We must also notice that the difference between messages coming from check nodes of different degrees seems not to decrease when decoding.
b and i denote the same thing: the connectivity degree of one bit node. Let r denote the function $\phi(.)$ of the mean over the whole graph of messages coming out of bit nodes. $h_i(s,r)$ is the function $\phi(.)$ (see Def. (2.5)) of the mean of messages coming out of bit nodes of degree i in terms of $s = 2/\sigma^2$ and r the mean at the previous iteration. In [6], we find

$$h_i(s,r) = \phi\left(s + (i-1)\sum_{j=2}^{t_{rmax}} \rho_j \phi^{-1}(1 - (1-r)^{j-1})\right)$$

We see from this equation that the higher is i (or equivalently b in our work), the lower is $\phi(.)$ of the messages coming out of bit node of degree i (b), i.e. the faster is the local convergence of these messages. Figure (4.6) extracted from [6] shows the gaps $h_i(s,r) - r$ that represent this local convergence. The curves are parameterized by i. r decreases as the number of iterations increases, from right to left. We clearly see the previous quoted effect of i on the local convergence and the flattening out of differences between messages coming from bit nodes of different degrees at high enough number of iterations, in contrast to the behavior at check nodes side.

This check nodes behavior is a very interesting, and we will elaborate on more.

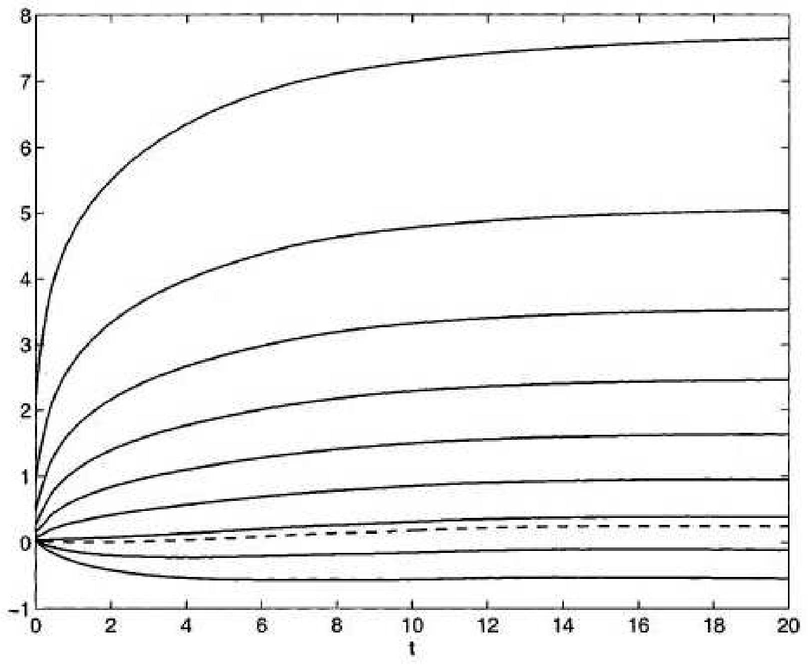

Figure 4.5: $f_j(s,t) - t$ for $j = 2, .., 10$ (top to bottom)

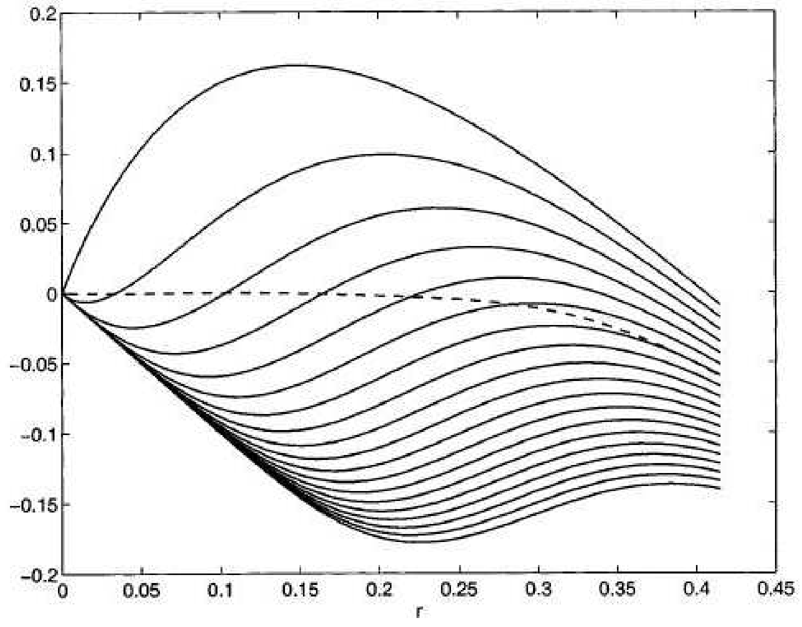

Figure 4.6: $h_i(s,r) - r$ for $i = 2, .., 20$ (top to bottom)

The behaviors of different degrees check nodes seem to remain different despite of the increasing number of iterations. This behavior is directly linked with the erasure-correction capability of a check node. In the sequel, we do not provide rigorous argumentation, but we try to give an intuition on what is happening at both kinds of nodes.

An erasure message corresponds to the LLR $L = 0$ since we have absolutely no information from the channel if the bit was 0 or 1.

- At a bit node LLRs are summed up. Then it is sufficient to have at least one message not erased to ensure that all the messages but one coming from this bit node are different from 0. So the probability that at least one message is not erased grows with the connectivity of the bit node, which explains that the more a bit node is connected, the more it is protected.
- At a check node LLR are multiplied (improper but equivalent since $(\tanh L = 0) \Leftrightarrow (L = 0))$. Then, it is sufficient that two messages are erased to have all the messages coming from this check node equal to 0. So the probability that at least two messages are erased decreases with the connectivity of the check node, which can explain the less a check node is connected, the more it can correct incoming erasures.

Remember that $LLR \in \mathbb{R}$, and $0 \leq \tanh(LLR) \leq 1$. At a high number of iterations, many LLRs are high.

- At a bit node, the important LLRs are the highest because they are summed up. At a given high number l of iterations, we decide that a message coming out of a bit node is of bad quality if the corresponding LLR is below a fixed threshold that does not depend on the considered bit node or on the number of iterations. At a high enough number l of iterations, a bit node produces bad message (i.e. a low LLR) if the number of incoming high LLRs is below a fixed number that we choose in terms of the number of iterations, i.e. in terms of the order of the current LLRs that can be added to reach the fixed threshold, but not in terms of the degree of the bit since the quality criterion for the messages is the same over the whole graph. Let $fix(l)$ denote this maximum number of high LLRs that produce bad messages at the lth iteration. Let p_l denote the probability that a LLR be considered as high (i.e. message of good quality) at the lth iteration.
Then we can write

$$\begin{aligned} P_{bad}^{(l)}(i) &= \mathrm{P}(\text{a message coming from a bit node of degree } i \text{ be of bad quality} | \text{high number } l \text{ of iter}) \\ &= \sum_{k=0}^{\min(fix(l),i)} C_{\min(fix(l),i)}^{k} p_l^k (1-p_l)^{\min(fix(l),i)-k} \end{aligned}$$

Since
$$\lim_{l\to\infty} LLR = \infty$$
we have
$$\lim_{l\to\infty} fix(l) = 0$$
whereas the connectivity degree i of a bitnode is, of course, fixed when decoding. Therefore,
$$\lim_{l\to\infty} \min(fix(l), i) = fix(l)$$

Hence, at high enough number of iterations, we obtain

$$\lim_{l\to\infty} P_{bad}^{(l)}(i) = \sum_{k=0}^{fix(l)} C_{fix(l)}^{k} p_l^k (1-p_l)^{fix(l)-k}$$

and then can state that the probability $P_{bad}^{(l)}(i)$ that a bit node of degree i produces a bad message tends to be independent of its degree i when the number of iterations increases sufficiently.
We conclude that when the number of iterations tends to infinity, the probabilities $P_{bad}^{(l)}(i_1)$ and $P_{bad}^{(l)}(i_2)$ of two bit nodes of different degrees i_1 and i_2 to generate bad messages tend to be the same. This means that all the variable nodes of the graph of any degree have the same behavior at a high number of iterations.
This explains that the UEP created by irregularities over bit nodes disappears at a high number of iterations.

- Whereas at check node side, high LLR have no influence since they are translated by ones by the hyperbolic tangent and then multiplied. The most important LLRs, which determine the quality of outgoing messages, are the smallest ones. A message coming out of a check node is of bad quality if at least one of the incoming LLRs is small. Let q_l denote the probability that a LLR entering into a check node be considered as small at a given high number of iterations l. Then we can write:

$$\begin{aligned} P_{bad}^{(l)}(j) &= \text{P(a message coming from check node of degree } j \text{ be of bad quality } \big| \text{high number } l \text{ of iter)} \\ &= \text{P(at least one of incoming } LLR \text{ be small } \big| \text{high number } l \text{ of iter)} \\ &= \sum_{k=1}^{j-1} C_{j-1}^{k} q_l^k (1-q_l)^{j-1-k} \end{aligned}$$

Since we are at a given high number of iterations, this can be approximated by

$$P_{bad}^{(l)}(j) = (j-1)\cdot q_l$$

Let us now express the ratio between the probabilities of outgoing LLRs of a check node to be small for two check nodes of different degrees j_1 and j_2:

$$\frac{P_{bad}^{(l)}(j_1)}{P_{bad}^{(l)}(j_2)} = \frac{(j_1-1)q_l}{(j_2-1)q_l}$$

$$\frac{P_{bad}^{(l)}(j_1)}{P_{bad}^{(l)}(j_2)} = \frac{j_1-1}{j_2-1} \tag{4.14}$$

This ratio is a constant. It does not depend on q_l, i.e., on the number of iterations, for high enough number of iterations. The behaviors of different check nodes remains different even at high number of iterations, i.e., at a low bit-error rate.
This explains that the UEP created by irregularities over check nodes remains at a high number of iterations which we exploit in this work.

We continue by switching from mean of messages to their mutual information, in order to be able to plot EXIT charts (Extrinsic Information Transfer charts) for UEP codes and express the error probability.

B and *D* can either be the degrees contained in the whole graph, and then Eq. (4.13) desbribe the usual Gaussian approximation of density evolution or the degrees inside one class of protection. We have seen at the very beginning of this chapter that a class is defined by the bit nodes that belong to it. The check nodes belonging to a class will be the ones linked to the bit nodes of this class.
The averaged mutual information of messages going from the check nodes of this class to the bit nodes of this class can be expressed as

$$x_{cv}^{(l)^{(C_k)}} = \sum_{b \in C_k} \lambda_b^{(C_k)} \sum_{d \in C_k} \rho^{(C_k)}(b,d) x_{cv}^{(l)}(d)$$

with $\rho^{(C_k)}(b,d) := \frac{\pi(b,d)}{\lambda_b^{(C_k)}}$ and $\lambda_b^{(C_k)} := \sum_{d \in C_k} \pi(b,d)$, then $\sum_{d \in C_k} \rho^{(C_k)}(b,d) = 1$.

Together with Eq. (4.13), we obtain

$$x_{cv}^{(l)^{(C_k)}} = 1 - \sum_{b \in C_k} \lambda_b^{(C_k)} \sum_{d \in C_k} \rho^{(C_k)}(b,d) J\left((d-1)J^{-1}\left(1 - \sum_{b \in \text{graph}} \lambda(b,d) x_{vc}^{(l)}(b)\right)\right)$$

And so we can express the difference defining the convergence of one classe, i.e., the medium quality arriving to its bit nodes, compared to the medium quality of all messages of the graph at the previous iteration.

$$x_{cv}^{(l)^{(C_k)}} - x_{cv}^{(l-1)} = 1 - \sum_{b \in C_k} \lambda_b^{(C_k)} \sum_{d \in C_k} \rho^{(C_k)}(b,d) J\left((d-1)J^{-1}\left(1 - \sum_{b \in \text{graph}} \lambda(b,d) x_{vc}^{(l)}(b)\right)\right) - x_{cv}^{(l-1)} \tag{4.15}$$

which is in our particular case of regularity over bit nodes ($\lambda(i) = \delta(i-3)$):

$$x_{cv}^{(l)^{(C_k)}} - x_{cv}^{(l-1)} = 1 - \sum_{d \in C_k} \rho^{(C_k)}(d) J\left((d-1)J^{-1}(1 - J(\frac{2}{\sigma^2} + (3-1)J^{-1}(x_{cv}^{(l-1)})))\right) - x_{cv}^{(l-1)}$$

We can rewrite this as

$$x_{cv}^{(l)^{(C_k)}} - x_{cv}^{(l-1)} = 1 - \sum_{d \in C_k} \rho^{(C_k)}(d) J((d-1)J^{-1}(1 - x_{vc}^{(l)})) - x_{cv}^{(l-1)} \tag{4.16}$$

In the following we will keep the last expression of this gap since the relation between $x_{vc}^{(l)}$ and $x_{cv}^{(l-1)}$ does not depend on parameters included in the optimization.

According to Eq. (4.6), we can express the error probability associated with a bit at the lth iteration by

$$P_l(bit) = Q\left(\sqrt{\frac{\text{mean of messages coming into bit}}{2}}\right) \tag{4.17}$$

In our particular case of aregular degree distribution over bit nodes, the difference between bits is due anymore to their degree i of connection, but caused by the degree of connection of the check nodes linked to this bit (called $|d_1(bit)|$ later) Equation (4.3) (extracted from [6]) is

$$P_{graph}^{(l+1)} = \sum_{k=0}^{N_c} \alpha_k Q(\sqrt{\frac{J^{-1}(x_{vc}^{(l)^{(C_k)}})}{2}})$$

And the error probability of bits inside class C_k is:

$$P_{l+1}^{(C_k)} = Q\left(\sqrt{\frac{J^{-1}(x_{vc}^{(l)^{(C_k)}})}{2}}\right)$$

$$P_{l+1}^{(C_k)} = Q\left(\sqrt{\frac{s+(3-1)J^{-1}(\sum_{d \in C_k} \rho^{(C_k)}(d)(1-J((d-1)J^{-1}(1-x_{vc}^{(l)}(3)))))}{2}}\right)$$

Then we can then derive bounds on the error probability of class C_k, where $d_{min}^{(C_k)}$ is the minimum degree of checks belonging to the class C_k:

$$Q\left(\sqrt{\frac{s+(3-1)\sum_{d \in C_k} J^{-1}\left(\rho^{(C_k)}(d)(1-J((d-1)J^{-1}(1-x_{vc}^{(l)}(3))))\right)}{2}}\right) \leq P_{l+1}^{(C_k)} \leq \qquad (4.18)$$
$$Q\left(\sqrt{\frac{s+(3-1)J^{-1}\left(1-J((d_{max}^{(C_k)}-1)J^{-1}(1-x_{vc}^{(l)}(3)))\right)}{2}}\right)$$

However more interesting bounds seem to be the ones directly on the difference between mutual informations, defining the convergence of one classe, expressed in our particular case as:

$$\boxed{\text{1-J}\left((\sum_{d \in C_k} \rho^{(C_k)}(d)d-1)J^{-1}(1-x_{vc}^{(l)})\right) - x_{cv}^{(l-1)} \leq x_{cv}^{(l)^{(C_k)}} - x_{cv}^{(l-1)} \leq 1-J\left((d_{min}^{(C_k)}-1)J^{-1}(1-x_{vc}^{(l)})\right) - x_{cv}^{(l-1)} \text{(4.19)}}$$

We see a dependency on the average check connection degree of the class C_k:

$$\overline{\rho}^{(C_k)} = \sum_{d \in C_k} \rho^{(C_k)}(d)d$$

Our algorithm is directly derived from the given bounds in Eq. (4.19). It is first meant to speed up the convergence, but we will see that UEP properties also remain at high number of iterations. At given al, then at given $x_{cv}^{(l-1)}$, we try to maximize the bounds of Eq. (4.19) by optimizing jointly both parameters $\overline{\rho}^{(C_k)}$ and $d_{min}^{(C_k)}$.

Therefore, the most protected classes, at a given number of iterations, are the ones linked with check nodes of lowest degrees, and even at high number of iterations according to Eq. (4.14). We should highlight two results of the asymptotic approach that appear to be contradictory to the first section of Chapter 3:

- The correction capability of a check node increases when its connection degree decreases, whereas
- The convergence threshold decreases when the connection degree of check nodes increases (with variable nodes degree).

We may think that at low SNR (bad quality of messages), and low number of iteration (increases the risk that poor $|d_1|$ bit receives only bad messages), the hierarchy of classes is reversed. This is not the case when looking at simulations: at any SNR, at any number of iterations, the class with the smallest $\overline{\rho}$ has the lowest BER. We should rather think that improving convergence of some classes, still acting on check nodes, implies worsening some others (see Fig. (4.10)), and worsening the overall convergence threshold of the code. That's why we should define the set of possible good codes, considering practical code length and expected performances.

4.3.3 Set of Possible Good Codes

Irregularity on the check profile leads to two problems:

- **Concentration problem**: influences the speed of convergence of the code. Indeed Chung has shown in [6] that a concentrated $\rho(x)$ polynomial $\rho(x) = \rho x^d + (1-\rho)x^{d-1}$ defined in Def. (3.1) maximises the speed of convergence of the whole code. To achieve UEP properties by irregularity on checks profile, little tolerance on concentration, and then on the global speed of convergence has to be defined. The global code will converge slower, but its most protected class will converge faster that the ones of concentrated code Fig. (4.9). However, the problem is not so extremely important, since the complexity is not so much increased with the number of iterations due to intelligent scheduling algorithms [5].
- **Density problem**: according to Gallager's result, densest codes have the lowest gap to capacity. At given code rate, there is one optimum average connectivity of check nodes $\overline{\rho}$ that minimizes the gap to the capacity Fig. (3.1) (for infinite code length and infinite number of iterations). This key parameter of the code, linked with t_{cmax}, determines the density of the code. The denser is the graph, the higher have to be the connectivity ratio. If the value of $\overline{\rho}$ is moved from the optimum, the value of t_{cmax}

must be changed too. Thus, at a given t_{cmax}, the required $\overline{\rho}$ can be achieved wether with a concentrated degree distribution at check node side, or with an unconcentrated one. But obtaining UEP by reducing the degree of some check nodes needs to adapt t_{cmax}. If one does not do so, the global convergence threshold of the code, expressed by E_b/N_0 will increase. That is the main problem in the chosen optimization scheme hat we present later. Our optimization, by removing bit nodes, decreases $\overline{\rho}$ while t_{cmax} is kept. The UEP less dense code must have higher threshold. However if we consider finite (quite short) code length N, a reducing $\overline{\rho}$ approach could be relevant due to the reversed hierarchy (chapter 3 first section). For $N = 1500$, both codes have quite same global threshold (at infinite number of iterations). Although thresholds of these found UEP codes are the same, UEP properties are quite different Fig. (4.10). Possible approach for long codes would be to first choose a tolerance offset ϵ on global threshold Fig. (4.7) in order to fix a trade-off between differences of classes and degradation of the threshold, but this work is not able to ensure that most protected classes of unconcentrated code with degraded threshold will have lower error probability than the global more concentrated code with lower gap to capacity, for long code length. We should quantify, for high number of iterations, the gain of local thresholds of most protected classes in function of the amplitude over check degrees. Somehow for short enough code length, the chosen optimization seems to be relevant.

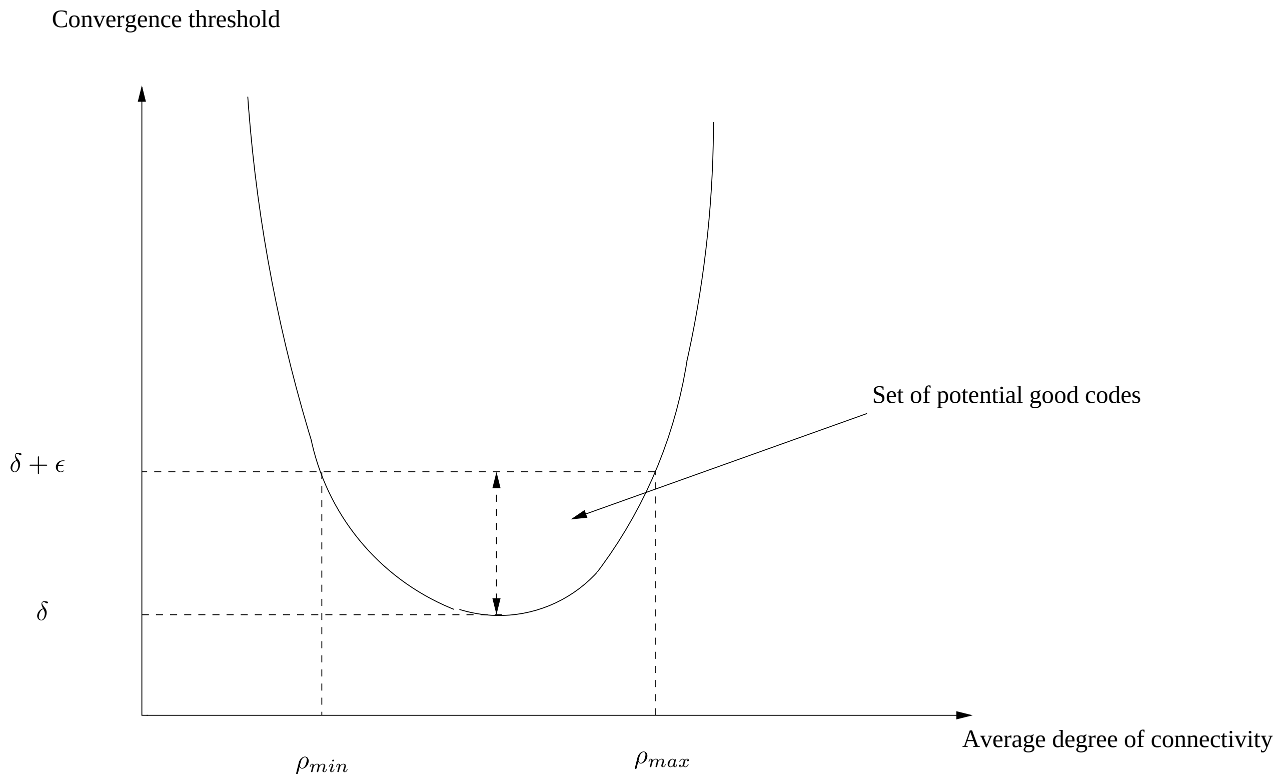

Figure 4.7: Set of potentially good codes.

Thus, at short enough code length, and low or high number of iterations, such approach that reduces $\overline{\rho}$ in our chosen optimization system is quite flexible:

- If the transmission has to be achieved, even with poor quality, we allow big amplitude on degrees of check nodes. For example if one wants to transmit a JPG picture even with bad quality, putting headers and very low frequency DCT coefficients in most protected classes ensures the transmission, even if the resulting picture is quite fuzzy.
- If initial global threshold must be kept, this UEP method, allowing a ρ polynomial almost concentrated (three consecutive degrees), can be seen as a kind of patch, or second stage method after UEP optimization over bit nodes.

Remember that the spreading of degrees of check nodes should not be a problem if the maximum degree of connection of bit nodes is adapted, i.e. in a joint optimization. It could raise a problem if the check optimization is proceeded after the bit nodes optimization, as a second stage. If it is done before, a constraint on t_{cmax} should be added in the optimization of bit node profile if one wants to keep the best convergence threshold.

4.3.4 A particular choice to realize UEP over check nodes

The goal of this work was to focus on UEP properties led by pruning and puncturing methods. Actually by pruning some bits of the codeword, it means to fix (e.g. to 0) and then not to transmit them, or equivalently, replace the corresponding columns in the **H** matrix by 0, we directly modify the irregularity profile of the check nodes, and can achieve some UEP configuration. The resulting code is a subcode derived from a mother code. By doing so, we intend to reach different UEP configurations, with different pruning schemes, with the same mother code and the same decoder.
We assume the number of information components of the subcode to be given. Then the code rate of the subcode is given too. We will see that the amount of redundancy is the same in the mother code and in the subcode.

4.3.4.1 The chosen coding scheme

Figure (4.8) shows the coding scheme that we use as a starting point:

Let **H** and **G** be the parity-check (size $M_0 \times N_0$) and generator (size $K_0 \times N_0$) matrices of the mother code and assume that they are in a systematic form (i.e. full rank). Let R_0 be the code rate of the mother code. The subcode has a given number of info bits: K_1. Then we are able to prune away $K_0 - K_1$ columns of the **H** matrix and the subcode would have a length of $N_1 = N_0 - (K_0 - K_1)$. We introduce a preprocessing generator matrix, called **P** (size $K_1 \times K_0$), which is used to fix the desired bits of the codewords of the subcode.
Let u be the number of pruned bits, then the code rate of the subcode is

$$\begin{aligned} R_1 &= \frac{K_0 - u}{N_0 - u} \\ &= \frac{K_1}{N_0 - (K_0 - K_1)} \end{aligned} \tag{4.20}$$

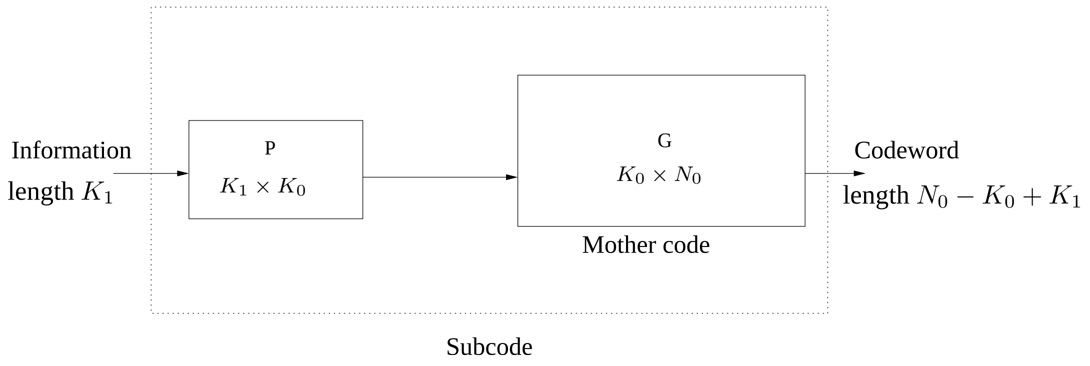

Figure 4.8: Scheme of the subcode encoder

Then we can write R_1 as a function of $v = u/N_0$:

$$R_1(v) = \frac{R_0 - v}{1 - v}$$

which is a decreasing function of v. Whatever the number of bits we prune away, pruning decreases the code rate.

This preprocessing matrix is not needed if we prune away only columns of information of the **H** matrix, and choose the K_1 best protected columns among the information columns of the **H** matrix, which reduces a lot the possible UEP configurations.

Let $\mathbf{G}^{'}$ of size $K_0 \times K_0$ be: $\mathbf{G}^{'} = [$protected columns of $\mathbf{G},$ columns of $\mathbf{G}$ to be pruned away$]$, and **B** of size $K_1 \times K_0$ be: $\mathbf{B} = [\mathbf{I}_{K_1}, \mathbf{0}_{K_1 \times (K_0 - K_1)}]$.
We are going to choose some columns of $\mathbf{H}$ to be pruned away. This means that the corresponding bits of the codeword of the subcode must equals zero, i.e. fixed deterministically. Then the corresponding columns of the generator matrix of the subcode have to be made of only zeros. Once we determined the K_1 best protected columns of the $\mathbf{H}$ matrix of the mother code, the corresponding columns in the generator matrix of the subcode have to be columns of the identity matrix, since the UEP code must be systematic to be able to control UEP over information bits.
Then the preprocessing matrix **P**, which is the tool to achieve the UEP we chose, is designed such that

$$\mathbf{P} \cdot \mathbf{G}^{'} = \mathbf{B} \tag{4.21}$$

We are going to verify that we can choose totally freely the $K_0 - K_1$ bits to prune away and the K_1 best protected among the N_0 bits of the mother code by discussing different choices of columns to prune away and to protect, and show that $\mathbf{P}$ permits to reach the expected and desired code rate.

4.3.4.2 Case we don't need P: only info columns of H are pruned and protected.

Then $\mathbf{H_s}$ and $\mathbf{G_s}$ are the parity-check and generator matrices of the subcode, are obtained by removing columns to prune away in $\mathbf{H}$, and the corresponding ones, which are columns of the identity, in $\mathbf{G}$ where we remove also corresponding rows (i.e. the row where there was the one). Since the best protected columns are chosen as being information columns, they are already made of the identity. Then $\mathbf{H_s}$ and $\mathbf{G_s}$ are of size $M_0 \times N_0 - (K_0 - K_1)$ and $K_1 \times N_0 - (K_0 - K_1)$, respectively. They are both of full rank (because $\mathbf{G_s}$ is made of the identity $\mathbf{I}_{K_1}$ and $\mathbf{H_s}$ of $\mathbf{I}_{M_0}$ since pruned columns are not identity columns which are associated to the redundancy). Then the code rate of the subcode would be:

$$R_1 = 1 - \frac{\text{rank}(\mathbf{H_s})}{N - (K_0 - K_1)} = \frac{K_1}{N_0 - (K_0 - K_1)}$$

The obtained rate is the desired one.

4.3.4.3 Case we prune away redundancy in H or choose protected column among it, and then need P

We prune non identity columns of $\mathbf{G}$, and then may have rank$(\mathbf{G}^{'}) < K_0$, which can raise a problem on the existence of a $\mathbf{P}$ matrix that fulfills Eq. (4.21) because $(\mathbf{G}^{'}$ can be not invertible anymore. We first prove that we can find a full rank $\mathbf{P}$ matrix, and then will see that the code rate of the subcode is the one desired.

Existence of $\mathbf{P}$

Definition 4.11 *A matrix is in a **reduced row echelon form** if it is made of a triangular upper part of size the rank of the matrix, after linear combinations of its rows, and then permutation of the columns.*

Definition 4.12 *A matrix will be said in a **reduced row form** if the previous manipulations on its rows have been made, but without permutting its columns at the end.*

So we may have rank$(\mathbf{G}^{'}) < K_0$. We want to find a condition on $\mathbf{G}^{'}$ such that we can compute $\mathbf{P}$ that fulfills Eq. (4.21).

Theorem 4.1 *A necessary and sufficient condition on $\mathbf{G}^{'}$ that allows to compute $\mathbf{P}$ that fulfills Eq. (4.21) is:*

$$\textit{rank}(\mathbf{G}^{'}) \geq K_1 \tag{4.22}$$

Proof: Let $\mathbf{G_2}$ be $\mathbf{G}^{'}$ in a reduced row form (i.e. without any manipulation on the columns of $\mathbf{G}^{'}$). We prove that such linear combinations on the rows of $\mathbf{G}^{'}$ still allow to find a $\mathbf{P_2}$ matrix such that

$$\mathbf{P_2} \cdot \mathbf{G_2} = \mathbf{B} \tag{4.23}$$

Let $\mathbf{g_{2k}}$ denote the kth row of the $\mathbf{G_2}$ matrix. Then we have the linear combination $\mathbf{g_{2k}} = \sum_{l=1}^{K_0} \alpha_l^{(k)} \mathbf{g_l}$. Let $\mathbf{A}$ be the matrix where at any column l and row k its element is equal to $\alpha_l^{(k)}$.

Remember that for any matrices

$$\text{rank}(\mathbf{A} \cdot \mathbf{B}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})) \tag{4.24}$$

- Necessity
  Using Eq. (4.24) in Eq. (4.23) we obtain

  $$\text{rank}(B) = K_1 \leq \min(\text{rank}(\mathbf{P_2}), \text{rank}(\mathbf{G_2}))$$

  A necessary condition is then

  $$\text{rank}(\mathbf{G_2}) \geq K_1$$

  since $\text{rank}(\mathbf{G_2}) = \text{rank}(\mathbf{G}^{'})$ by construction of $\mathbf{G_2}$, this condition is equivalent to

  $$\text{rank}(\mathbf{G}^{'}) \geq K_1$$

- Sufficient
  Assuming that $\text{rank}(\mathbf{G_2}) \geq K_1$, the computed $\mathbf{P_2}$ matrix from Eq. (4.23) will be of rank greater or equal to K_1 by construction. Since

  $$\mathbf{G_2} = \mathbf{A}.\mathbf{G}^{'}$$

  we translate Eq. (4.23) by

  $$\mathbf{P_2} \cdot \mathbf{A} \cdot \mathbf{G}^{'} = \mathbf{B} \tag{4.25}$$

  which means that

  $$\mathbf{P} = \mathbf{P_2}.\mathbf{A} \tag{4.26}$$

  and from which we can infer

  $$\text{rank}(\mathbf{P_2} \cdot \mathbf{A} = \mathbf{P}) \geq K_1$$

  The $\mathbf{P}$ matrix will be computed by using Eq. (4.26).

The condition

$$\text{rank}(\mathbf{G}^{'}) \geq K_1$$

is then necessary and sufficient to ensure the existence of a matrix $\mathbf{P}$ that fulfills Eq. (4.21), and since $\mathbf{P}$ is of size $K_1 \times K_0$, $\mathbf{P}$ will be of full rank according to Eq. (4.25). $\square$

Let rg denote rank$(\mathbf{G}^{'})$. Equation (4.23) can be represented by

$$\mathbf{P_2}.\left[\begin{array}{ccccccc} \text{non} & \text{zero} & \text{part} & \text{of} & \text{reduc} & \text{row} & \text{form} \\ & & & \mathbf{G}^{'}_{rg\times K_0} & & & \\ - & - & - & - & - & - & - \\ & & & \mathbf{0}_{(K_0-rg)\times K_0} & & & \end{array}\right] = \left[\begin{array}{ccc} 1 & & \\ & \ddots & \mathbf{0}_{K_1\times(K_0-K_1)} \\ & 1 & \end{array}\right] \tag{4.27}$$

Provided $\mathbf{G}^{'}$ fulfills Condition (4.22), a solution for $\mathbf{P_2}$ exists, and if rank$(\mathbf{G}^{'}) \leq K_0$, then we have degrees of freedom for $\mathbf{P_2}$, and then also for $\mathbf{P}$.

Code rate of the subcode Let us now compute the rate of the subcode. For this, we consider the decoding. We have two possibility for the decoding:

- Either we use the decoder of the mother code without adding anything, the parity-check matrix of the subcode will exactly $\mathbf{H_mother}$ with pruned columns removed. This allows to save memory and complexity, but does not exploit all the available parity-check equations since the ones of the preprocessing code $\mathbf{P}$ are not used, which limits the performances.
  Then we have
  $$R_1 = 1 - \frac{\text{rank}(\mathbf{H_{motherpruned}})}{N_0 - (K_0 - K_1)}$$
  A constraint, called*Code rate constraint* in the optimization algorithm, ensures that the parity-check matrix of the subcode, i.e. the matrix of the mother code without the pruned columns, will have a code rate of $\frac{K_1}{N_0-(K_0-K_1)}$, or that equivalently rank($\mathbf{H_{motherpruned}}$).
- Or we use all the available parity-check equations to have better performances. Let us study this case in what follows.

So in this last case, let us **forget pruning and consider the subcode as an usual serial concatenation** (without any interleaver, discussed later) of the two codes $\mathbf{P}$ and $\mathbf{G}$ (i.e. the mother code).

$\mathbf{H}_{pruned}$ is not anymore the parity matrix of the subcode since another parity equations are added. The subcode is defined by:

$$\mathbf{G}_s = \mathbf{P}.\mathbf{G} : K_1 \times N_0$$

$$\mathbf{H}_s : (N_0 - K_1) \times N_0$$

$\mathbf{H}_s$ is made of $\mathbf{H}_{mothercode}$ and the $\mathbf{H}_p$ parity matrix of the generator preprocessing matrix $\mathbf{P}$. $\mathbf{H}_p$ is of size $(K_0 - K_1) \times K_0$:

$$\mathbf{H}_p = \left[\begin{array}{cc} \mathbf{I}_{K_0-K_1)} & \mathbf{R}_{(K_0-K_1)\times K_1} \end{array}\right] \tag{4.28}$$

$\mathbf{I}_{K_0-K_1)}$ is the identity associated to redundancy columns of the precode $\mathbf{P}$, and $\mathbf{R}_{(K_0-K_1)\times K_1}$ are associated to information bits of the subcode.
The same form for $\mathbf{H}$ of the mother code:

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{N_0-K_0)} & \mathbf{T}_{(N_0-K_0)\times K_0} \end{bmatrix} \tag{4.29}$$

The K_0 bits of the codeword of the precode $\mathbf{P}$ are directly copied into the K_0 information bits of the mother code. The have $\mathbf{H_s}$ in this form:

$$\mathbf{H}_s = \begin{bmatrix} & \mathbf{H_{mother}} & \\ - & - & - \\ \mathbf{0}_{(K_0-K_1)\times(N_0-K_0)} & & \mathbf{H}_{p(K_0-K_1)\times K_0} \end{bmatrix} \tag{4.30}$$

That can be rewritten as

$$\mathbf{H}_s = \begin{bmatrix} \mathbf{I}_{N_0-K_0} & \mathbf{T}_{(N_0-K_0)\times K_0} \\ \mathbf{0}_{(K_0-K_1)\times(N_0-K_0)} & \mathbf{H}_{p(K_0-K_1)\times K_0} \end{bmatrix} \tag{4.31}$$

$\mathbf{G}$ is in a systematic form but $\mathbf{P}$ is not, i.e. $\mathbf{H}_{mother}$ is in a systematic form but $\mathbf{H_p}$ is not. We are only sure that the bits of the whole codeword that fulfill parity-check equations of the precode $\mathbf{P}$ are the information bits of the mother code. The parity-check matrix $\mathbf{H_s}$ of the subcode **is not in a systematic form** in Eq. (4.31), and then we cannot distinguish columns of redundancy and columns of information of the subcode in this form. To put $\mathbf{H_s}$ in a systematic form, i.e. in a reduced row echelon form, the permutations on its columns that we would have to do will show that the information of the subcode can correspond to redundancy of the mother code (be careful to not confuse subcode and precode).
We now want to show that after having pruned any $K_0 - K_1$ columns of the $\mathbf{H_s}$ matrix in the given non systematic form, we have

$$\text{rank}(\mathbf{H_s}) = N_0 - K_0$$

in order the code rate to be

$$R_1 = 1 - \frac{\text{rank}(\mathbf{H_s})}{N_0 - (K_0 - K_1)} = \frac{K_1}{N_0 - (K_0 - K_1)}$$

Proof: By doing linear combinations on the rows of the matrix $\mathbf{H_p}$, only the $K_0 - K_1$ last rows of the matrix $\mathbf{H_s}$ are manipulated. Then to put $\mathbf{H_p}$ in a systematic form, only the K_0 last columns of the matrix $\mathbf{H_s}$ are permuted. We then obtain the following form of $\mathbf{H_s}$ (Eq. (4.32)) called $\mathbf{H_s}_{sys}$, where the last K_1 columns are associated to the K_1 information bits of the subcode, and the $K_0 - K_1$ pruned columns are taken among the $N_0 - K_1$ columns, which are the columns of a squarred upper triangular matrix.

$$\mathbf{H_s}_{sys} = \begin{bmatrix} \mathbf{I}_{N_0-K_0} & \multicolumn{2}{c}{\mathbf{T}_{(N_0-K_0)\times K_0}} \\ \mathbf{0}_{(K_0-K_1)\times(N_0-K_0)} & \mathbf{I}_{K_0-K_1} & \mathbf{R}_{(K_0-K_1)\times K_1} \end{bmatrix} \tag{4.32}$$

Equation (4.32) shows that at least the $N_0 - K_1$ first columns are independent from each other, then if we prune $K_0 - K_1$ of these columns we have

$$\text{rank}(\mathbf{H_s}) = \text{rank}(\mathbf{H_s}_{sys}) \geq N_0 - K_0$$

But what we prune is redundancy of the subcode (be aware to not confuse with redundancy of the precode or the mother code), by construction of $\mathbf{H_s}_{sys}$. Therefore, since the number of rows of the parity-check matrix is exactly the number of redundancy bits, we must remove the row corresponding to the pruned column (of same indice as the column, where there is the one on the main diagonal). Then at the end of pruning, $\mathbf{H_s}_{sys}$ is of size $N_0 - K_1 - (K_0 - K_1) \times N_0 - (K_0 - K_1)$ i.e. $N_0 - K_0 \times N_0 - (K_0 - K_1)$, then we have

$$\text{rank}(\mathbf{H_s}) = \text{rank}(\mathbf{H_s}_{sys}) \leq N_0 - K_0$$

We conclude

$$\text{rank}(\mathbf{H_s}) = N_0 - K_0$$

□

What ensures that whatever the columns we choose to prune away, the code rate of the subcode will be $R_1 = 1 - \frac{\text{rank}(\mathbf{H_s})}{N_0-(K_0-K_1)} = \frac{K_1}{N_0-(K_0-K_1)}$. Note that The resulting $\mathbf{H_s}_{sysprun}$ after pruning will be equal to $\mathbf{H_s}_{motherprun}$ if we chose to prune only information bits of the mother code, otherwise different.

Then, it is sufficient that the condition (4.22) be fulfilled to be able to compute the $\mathbf{P}$ matrix and have a code rate of the subcode equal to the one desired, even if we choose columns to be pruned away and best protected columns among redundancy of the mother code.

Computation of the preprocessing matrix After having verified that we can choose the $K_0 - K_1$ bits to be pruned away and the K_1 best protected among the N_0 bits of the mother code, we are going to explain how the $\mathbf{P}$ matrix is computed.

Let us describe the solution of the system:

$$\mathbf{A_{sys}}.\mathbf{C_{sys}} = \mathbf{B_{sys}}$$

where $\mathbf{A_{sys}}$ (size $K_0 \times K_0$)is $\mathbf{G_2}$ after permutting its columns to transfer it into an echelon form, and then transposing. $\mathbf{B_{sys}}$ (size $K_0 \times K_1$) is B after permutting columns in the same way as $\mathbf{G_2}$, and then transpose, and $\mathbf{C_{sys}}$ (size $K_0 \times K_1$) is $\mathbf{P_2}$ transpose. Let rg still denote the rank of $\mathbf{G}'$.

$$\begin{bmatrix} 1 & & & \vdots & \vdots \\ & \ddots & & & \\ & & 1 & & \\ - & - & - & - & - \\ & \mathbf{0}_{(K_0-rg)\times K_0} & & & \end{bmatrix} .\mathbf{C_{sys}} = \mathbf{B_{sys}} \tag{4.33}$$

We can note that provided $rg \leq K_0$, we have $(K_0 - rg)$ degrees of freedom for the coefficients of $\mathbf{C_{sys}}$.

$\forall j \in [1, K_1]$

$$c_{sys}(1,j) = \sum_{l=rg+1}^{K_0} a_{sys}(1,l).c_{sys}(l,j) + b_{sys}(1,j)$$

$$\vdots$$

$$c_{sys}(rg,j) = \sum_{l=rg+1}^{K_0} a_{sys}(rg,l).c_{sys}(l,j) + b_{sys}(rg,j)$$

These $(K_1 \times rg)$ equations determine the elements of $\mathbf{P}$ that can be chosen arbitrarily, and the way to compute the remaining elements from the chosen ones.

We are now proposing a method to fix these degrees of freedom. The fact to fix them adds information that we could use by considering the preprocessing matrix simply as a precoder. A first possibility would be to think of it in terms of a serial concatenation of two codes (the outer code of generator matrix $\mathbf{P}$, and the inner one the mother code), and could decode this in an iterative way, for example, if we find $\mathbf{P}$ to be an LDPC code (due to its size), and adding an interleaver. However the serial concatenation of two LDPC codes does not improve too much the decoding, even if the girth is improved. Another possibility is to consider the $\mathbf{P}$ matrix as some additional parity-check equations, as showed in expression of $\mathbf{H}_s$. Let us choose an arbitrary $\mathbf{H}_p$, for example such that it improves the UEP properties of interesting bits by choosing its irregularity accordingly, or as a part of $\mathbf{H}_{mother}$ to decrease the required memory. Note that the user will have to choose the constraints on the optimization and so the strength of UEP, according to his available memory and processing power.

Once $\mathbf{H}_p$ is chosen, we are now describing how to compute $\mathbf{C_{sys}}(rg+1 : K_0, 1 : K_1)$.
$\mathbf{H}_p : (K_0 - K_1) \times K_0$ whose elements are h(i,j) and $\mathbf{P}^T : K_0 \times K_1$ whose elements are d(i,j)

$$\mathbf{H}_p \cdot \mathbf{P}^T = \mathbf{0}_{(K_0-K_1)\times K_1}$$

is rephrased as

$\forall (i,j) \in [1, K_0 - K_1] \times [1, K_1]$

$$\sum_{l=1}^{K_0} h(i,l) c_{sys}(l,j) = 0$$

$\Leftrightarrow$

$$\sum_{l=1}^{rg} h(i,l)[\sum_{m=rg+1}^{K_0} a_{sys}(l,m) c_{sys}(m,j) + b_{sys}(l,j)] + \sum_{l=rg+1}^{K_0} h(i,l) c_{sys}(l,j) = 0$$

Finally,

$$\sum_{m=rg+1}^{K_0} c_{sys}(m,j)[h(i,m)+\sum_{l=1}^{rg} h(i,l)a_{sys}(l,m)] = \sum_{l=1}^{rg} h(i,l)b_{sys}(l,j)$$

Rewritten in matrix form, this reads:

$$\mathbf{H}_p \cdot \begin{bmatrix} \mathbf{A_{sys}}(1:rg, rg+1:K_0) \\ I_{(K_0-rg)} \end{bmatrix} . \begin{bmatrix} \mathbf{C_{sys}}(rg+1:K_0, 1:K_1) \end{bmatrix} = \begin{bmatrix} \mathbf{B_{sys}}(1:rg, 1:K_1) \\ 0_{(K_0-rg)\times K_1} \end{bmatrix} \tag{4.34}$$

Let E denote the matrix resulting from the multiplication of the two first terms. In order to ensure the existence of a solution, it is sufficient that rank$(E) \leq K_0 - rg$. However Eq. (4.22) yields rank$(E) \leq \min(K_0 - K_1, K_0 - rg)$, assuming that $\mathbf{H}_p$ is chosen to be of full rank. Thus, we are sure to have a unique solution on $\mathbf{C_{sys}}(rg+1:K_0, 1:K_1)$, provided the previous condition rank$(\mathbf{G}') \geq K_1$ holds.

4.3.4.4 Hierarchical optimization algorithm

Let us remember the bounds of Eq. (4.19) on which our optization is based:

$$1-J\left((\overline{\rho}^{(C_k)}-1)J^{-1}(1-x_{vc}^{(l)})\right)-x_{cv}^{(l-1)} \leq x_{cv}^{(l)^{(C_k)}}-x_{cv}^{(l-1)} \leq 1-J\left((d_{min}^{(C_k)}-1)J^{-1}(1-x_{vc}^{(l)})\right)-x_{cv}^{(l-1)}$$

with

$$\overline{\rho}^{(C_k)} = \sum_{d \in C_k} \rho^{(C_k)}(d)d$$

Due to the chosen coding scheme, K_1 and the mother code are fixed at the beginning of the optimization, therefore the code rate is fixed to $R = \frac{K_1}{N_0-K_0+K_1}$, and the optimization does not consider it at all. Let us denote the minimum degree of check nodes of the whole graph by j_{min} (j and d are used to denote the same thing), and their average by $\overline{\rho}$. The optimization focuses on the two important quantities of bounds (4.19) : $\overline{\rho}^{(C_k)}$ and $d_{min}^{(C_k)}$, and is composed of two main stages, for given class:

- We choose the $(\alpha_k K_1)$ most protected bitnodes.
- At given d_{min}, we try to put a maximum number of check nodes linked to these bit nodes to d_{min} in order to decrease $\overline{\rho}^{C_k}$.
- We check, if the following constraints are fulfilled. If yes:
- We decrease d_{min} by one if the tolerance that we fixed regarding the concentration is not yet reached, and start over again.

Note that we work with d_{min} and not $d_{min}^{(C_k)}$ because the composition of the C_k class is updayed at the beginning of each iteration of the optimization algorithm, which allows to

take advantage of all the possibilities of pruning of every check nodes. We determine the composition of the clesses only at the very end of our algorithm.
In a more detailed way:

Definition 4.13 $N_0(bit)$ *denotes the set of check nodes linked to variable node* bit*.*
$N_1(bit)$ *is the set of bit nodes linked to each check node belonging to* $N_0(bit)$*.*

Definition 4.14 $\bar{d}_1(bit)$ *denotes the average of degrees of checks linked to that certain variable node* bit*, and* $|set|$ *be the cardinal of the set* set*. Then we have:*

$$\bar{d}_1(bit) = \frac{|N_1(bit)|}{|N_0(bit)|}$$

Then, the adopted algorithm can be described as it follows:

For each class C_k.

- while (the constraints are fulfilled and the tolerance over the break of concentration is not reached)
  - $\bar{d}_1$ over the whole graph are arranged in an increasing order
    - for each check node in C_k, we search for a bit to pruned away, such that $bit_{pruned} = \arg\max_{bit}(\bar{d}_1(bit))$ under:
    - hierarchical constraints:
      - $b_{pruned} \notin C_i, \quad \forall i \leq k$
      - b_{pruned} must not be linked with a check node of degree greater or equal to the concentration constraint
      - avoid unvoluntary pruning (a column of $\mathbf{H}$ can become independent from all the others and then does not define a code anymore)
    - usual constraints (described in Chapter 2)
      - proportion constraint

$$\sum_{k=1}^{N_c} \alpha_k \sum_{j=2}^{t_{rmax}} \tilde{\rho}_j^{(C_k)} = 1$$

Where $\tilde{\rho}_j^{(C_k)}$ is the proportion of check nodes of degree j belonging to the C_k class.

- code rate constraint
  Let us denote the number of pruned columns at the current iteration of the optimization procedure by u, then the code rate at this iteration has to be $R = \frac{K_0-u}{N_0-u}$. We then must have

$$(1-R)\sum_{i=2}^{t_{cmax}} \frac{\lambda_i}{i} = \sum_{j=2}^{t_{rmax}} \frac{\rho_j}{j}$$

- convergence constraint (see Eq. (2.22))

$$x_{vc}^{(l)} = F(\underline{\lambda}, \underline{\rho}, x_{vc}^{(l-1)}, \sigma^2)$$

- stability constraint (see Eq. (2.24))

$$\lambda_2 \leq \lambda_2^{\star} = \frac{e^{1/2\sigma^2}}{\sum_{j=2}^{t_{rmax}} \rho_j (j-1)}$$

  This condition is automatically fulfilled in the case of a regular mother code.

At the end of the optimization. Constraint that ensures the existence of a $\mathbf{P}$ matrix (see Condition (4.22)):

$$\text{rank}(\mathbf{G}^{'}) \geq K_1$$

4.3.4.5 Results

Curves correspond to a regular LDPC mother code of length $N_0 = 2000$ and code rate $R_0 = 1/2$. The subcode has a length of $N_1 = 1000$ and code rate $R_1 = 1/3$. The N_c classes to be optimized are defined by the proportions $\alpha(k)$ for $k \leq N_c - 1$ (the number of info bits in the class C_k is $\alpha(k) \cdot R_1 \cdot N_1$ if $k \leq N_c - 1$, and $\sum_{k=1}^{N_c-1} \alpha(k) = 1$, and $(1-R_1).N_1 = (1-R_0).N_0$ in the last one which then contains the whole redundancy). The optimization is done for $N_c = 3$ classes with $\alpha(1) = 0.1$, $\alpha(2) = 0.9$. The mother code has parameters (2000,3,6).
Optimizations to obtain unconcentrated (degrees for checks between 2 and 6) and almost concentrated (degrees for checks between 4 and 6) degrees codes are done to compare the performances.
The decoding is done bu using only the pruned parity-check matrix of the mother code.

Check profile of the almost concentrated code					
j	2	3	4	5	6
Class1	0.000000e+00	0.000000e+00	9.038095e-01	9.619048e-02	0.000000e+00
Class2	0.000000e+00	0.000000e+00	6.666667e-01	3.333333e-01	0.000000e+00
Class3	0.000000e+00	0.000000e+00	3.556667e-01	4.863333e-01	1.580000e-01

Check profile of the unconcentrated code					
j	2	3	4	5	6
Class1	1.590476e-01	1.971429e-01	3.314286e-01	2.695238e-01	4.285714e-02
Class2	1.111111e-02	4.888889e-02	4.066667e-01	4.600000e-01	7.333333e-02
Class3	1.333333e-03	8.666667e-03	1.603333e-01	4.816667e-01	3.480000e-01

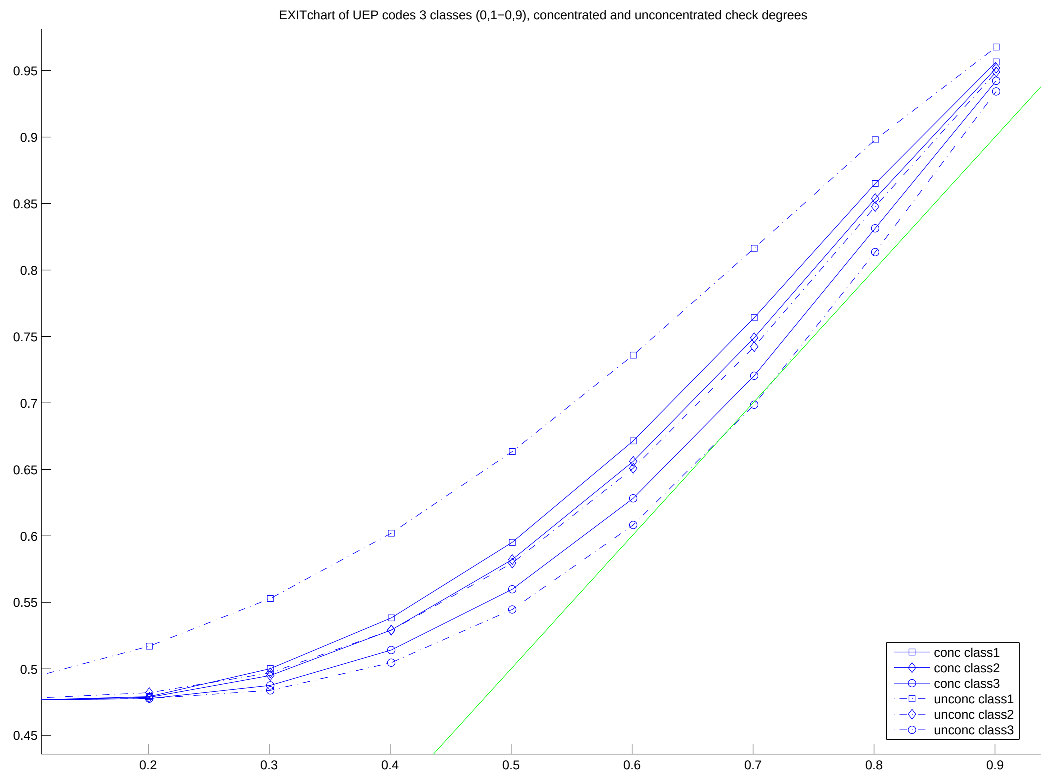

Figure 4.9: EXIT curves of classes of almost concentrated and unconcentrated check irregularity codes.

Fig. (4.9) shows EXIT curves defined in Eq. (4.16) for each class of almost concentrated and unconcentrated check irregularity codes. The more the first class is protected, the more the less protected ones are degraded: the best protected class has a faster convergence in the unconcentrated code than the corresponding one in the concentrated code. The intermediate classes are quite equivalent whereas the last class of the unconcentrated code has a slower convergence than the corresponding one in the concentrated one.

Figure (4.10) shows the behavior at low bit-error rates, which cannot be seen from an EXIT curve. This would be near the $(1, 1)$ point in the EXIT chart, i.e. at a high number of iterations. Here for 30 iterations. We clearly see that UEP properties remain also at a high number of iterations, which constitutes a huge difference from UEP properties generated by irregularities over bit nodes, which induces convergence speed differences. The check optimization would be a means to achieve UEP at low number of iterations (accelerating the convergence), and at a high number. This behavior can be explained by Fig. (4.5) and the comments following it in the first section. As well we still have better performance at 30 iterations for the first class of the unconcentrated code than for the concentrated one, equivalent performance for the middle class, and poorer performances for the last one.

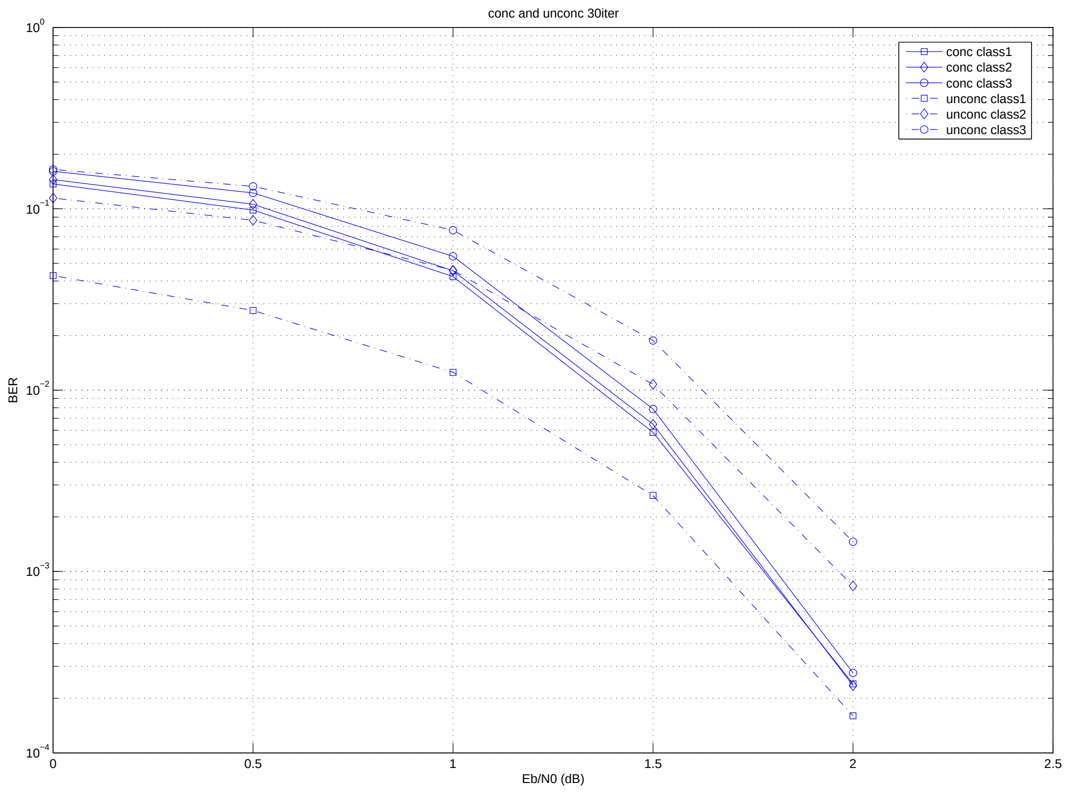

Figure 4.10: BER of almost concentrated and uncocentrated check irregularity codes

These created UEP properties that remain even at a high number of iterations might be very interesting since techniques to improve a lot the number of iterations without increasing too much complexity exists [5].

4.4 Puncturing

The puncturing could be a method to realize UEP by increasing the code rate and worsening certain bits, but without the possibility to improve some others. In a punctured code, the quality of the messages coming to interesting checks (i.e. belonging to one class) would be more important than the degrees of these checks. At a check node we add up erasure messages (i.e. with LLR, defined in Def. (2.2), that equals zero) instead of making them deterministic by pruning (i.e. LLR equals infinity that makes the bitnode and the linked edges disappear from the graph). Then instead of achieving UEP only by puncturing the code, we can be more interested in puncturing the code whose UEP is created by irregularities over check nodes and bit nodes. The puncturing must then be compatible with the UEP properties. In order to match the definitions of [9], we have to define and redefine some variables.

The old π of the Def. (4.8) that defines the detailed representation of LDPC codes turns to Pi.

Definition 4.15 $G_{i,j}$ *is the set of bit nodes of degree* i *linked to check nodes of degree* j.

Definition 4.16 $\pi_{i,j}^{(0)}$ *is the proportion of puntured symbols in* $G_{i,j}$ *before decoding.*

Remember the useful following definition (4.10)

Definition 4.17 $\lambda(i,j)$ *and* $\rho(i,j)$ *are the proportion of bit nodes of degree* i *among bit nodes linked to check nodes of degree* j*, and the proportion of check nodes of degree* j *among check nodes linked to bit nodes of degree* i*, respectively.*

Thus we define

Definition 4.18 *The total puncturing fraction* $p^{(0)}$ *is the proportion of punctured variable nodes over the whole graph:*

$$p^{(0)} = \sum_i \sum_j Pi(i,j)\pi_{i,j}^{(0)}$$

Proof:

$$\begin{aligned}
p^{(0)} &= \sum_i \text{Proba(bitnode be of degree } i \text{ and be punctured)} \\
p^{(0)} &= \sum_i \sum_j \text{Proba(bitnode be of degree } i \text{ and linked to check of degree } j \text{ and be punctured)} \\
p^{(0)} &= \sum_i \sum_j \text{Proba(bitnode be punctured} | \text{bitnode is of degree } i \text{ and linked to check of degree } j\text{)} \\
p^{(0)} &= \sum_i \sum_j Pi(i,j)\pi_{i,j}^{(0)}
\end{aligned}$$

□

With an analysis with Gaussian approximation, we can follow the evolution of the proportion of punctured symbols when decoding. To do so, we need some other definitions.

Definition 4.19 $\varepsilon_j^{(k)}$ *is the probability for the message coming from a check node of degree* j *to be zero at the* k*th iteration.*
$e_i^{(k)}$ *is the probability that the message of a variable node of degree* i *is zero.*

These quantities are easily computed as

$$\varepsilon_j^{(k)} = 1 - \sum_i \lambda(i,j)(1-e_i^{(k)})^{j-1}$$

and

$$e_i^{(k)} = \sum_j \rho(i,j)\pi_{i,j}^{(0)} \left(\varepsilon_j^{(k-1)}\right)^{i-1}$$

Proof:

$$\begin{aligned}
1-\varepsilon_j^{(k)} &= \text{Proba}(u \text{ coming from a check node of degree } j \text{ be non zero}) \\
&= \sum_i \text{Proba}((j-1)\ v \text{ coming from bit of degree } i \text{ be non zero}|\text{bit is linked to check of degree } j) \\
&= \sum_i \text{Proba}(v \text{ comes from bit of degree } i|\text{bit is linked to check of degree } j) \\
&\quad \text{Proba}(v \text{ be non zero}|\text{bit of degree } i)^{j-1} \\
&= \sum_i \lambda(i,j)(1-e_i^{(k)})^{j-1} \qquad (4.35)
\end{aligned}$$

$$\begin{aligned}
e_i^{(k)} &= \text{Proba}(v \text{ coming from a bit node of degree } i \text{ be zero}) \\
&= \sum_j \text{Proba}((i-1)\ u \text{ coming from check of degree } j \text{ be zero}|\text{check is linked to bit of degree } i) \\
&\quad \text{P}(u_0 \text{ be zero}) \\
&= \sum_j \text{Proba}(u \text{ comes from check of degree } j|\text{check is linked to bit of degree } i)\text{P}(u_0 \text{ be zero}|\text{bit} \in G_{i,j}) \\
&\quad \text{Proba}(u \text{ be zero}|\text{check of degree } j)^{i-1} \\
&= \sum_j \rho(i,j)\pi_{i,j}^{(0)} \left(\varepsilon_j^{(k-1)}\right)^{i-1}
\end{aligned}$$

□

These quantities are used to compute the residual puncturing proportion $\pi_{i,j}^{(k)}$ and the proportion $p^{(k)}$ of punctured symbols at the kth iteration:

$$\begin{aligned}
\pi_{i,j}^{(k)} &= \pi_{i,j}^{(0)}(\sum_{j=2}^{t_{rmax}} \rho(i,j)\varepsilon_j^{(k-1)})^i \\
p^{(k)} &= \sum_i \sum_j Pi(i,j)\pi_{i,j}^{(k)} \qquad (4.36)
\end{aligned}$$

Since we are interested in UEP and our criterion is the difference between the evolution of messages, let us express the mean of messages coming from check nodes in terms of the puncturing pattern. To do so, we need some other definitions.

First in order to shorten the notations, let us define

$$\begin{aligned}\lambda_{i,j}^{\pi} &= \lambda(i,j)\pi_{i,j}^{(0)} \\ \lambda_{i,j}^{1-\pi} &= \lambda(i,j)(1-\pi_{i,j}^{(0)}) \\ \lambda_{i}^{\pi} &= \sum_j Pi(i,j)\pi_{i,j}^{(0)}\end{aligned}$$

which are the initial proportion of punctured bits of degree i among all the bits linked to check nodes of degree j, the initial proportion of unpunctured bits of degree i among all the bits linked to check nodes of degree j and the initial proportion of punctured bits of degree i, respectively.

Definition 4.20 *Let $\chi_{n,m}^{(k)}$ be the probability that exactly m messages coming into a bit node of degree m are not erased at the kth iteration. If C_n^m denotes the binomial coefficient, we have*

$$\chi_{n,m}^{(k)} = C_n^m (1-\varepsilon^{(k-1)})^m \varepsilon^{(k-1)^{n-m}}$$

Thus, we can express the updated mean of a check node of degree j as

$$m_u^{(k)}(j) = \phi^{-1}\left(1 - \frac{1}{(1-e^{(k)})^{j-1}}\left[1 - \sum_i [\lambda_{i,j}^{\pi} \sum_l \chi_{i-1,l}^{(k)} \phi(l m_u^{(k-1)}) + \lambda_{i,j}^{1-\pi} \sum_l \chi_{i-1,l}^{(k)} \phi(l m_u^{(k-1)} + m_{u_0})]\right]^{j-1}\right) \tag{4.37}$$

The term in the squared brackets is composed of mean of messages coming from bit nodes punctured (there is no observation from the channel so $u_0 = 0$) at the kth iteration and the mean of messages coming from bit nodes unpunctured at the kth iteration.
We have

$$\begin{aligned}m_u^{(k)} &= \sum_j \rho_j m_u^{(k)}(j) \\ &= \sum_i \sum_j Pi(i,j) m_u^{(k)}(j)\end{aligned}$$

The evolution of the puncturing fraction in Eq. (4.36) indicates that the residual puncturing fraction while decoding does not depend on SNR but only on the detailed distribution pair $(Pi(x,y) = \sum_i \sum_j Pi(i,j)x^{i-1}y^{j-1}, \pi^{(0)}(x,y) = \sum_i \sum_j \pi(i,j)x^{i-1}y^{j-1})$. Thus, as long as the degree distribution satisfies $e^{(k+1)} < e^{(k)}$ for any $k \geq 0$, we can reduce the residual puncturing fraction to any small value, regardless of the SNR. After enough iterations, $e^{(k)}$ and $\varepsilon^{(k)}$ converge to zero and $\chi_{i,l}$ becomes δ_{il}, which simplifies Eq. (4.37).

$$m_u^{(k)}(j) = \phi^{-1}\left(1 - \left[1 - \sum_i [\lambda_{i,j}^{\pi} \phi((i-1)m_u^{(k-1)}) + \lambda_{i,j}^{1-\pi} \phi((i-1)m_u^{(k-1)} + m_{u_0})]\right]^{j-1}\right) \tag{4.38}$$

We abbreviate the sum by $r_j^{(k-1)}$ and define a function H such that

$$\begin{aligned}r_j^{(k-1)} &= \sum_i \left[\lambda_{i,j}^{\pi} \phi((i-1)m_u^{(k-1)}) + \lambda_{i,j}^{1-\pi} \phi((i-1)m_u^{(k-1)} + m_{u_0})\right] \\ r_j^{(k)} &= H(m_{u_0}, \lambda_{i,j}^{\pi}, r^{(k-1)})\end{aligned}$$

and

$$\begin{aligned}
m_u^{(k)} &= \sum_j \rho_j m_u^{(k)}(j) \\
m_u^{(k)} &= \sum_j \rho_j \phi^{-1}\left(1 - \left[1 - r^{(k-1)}\right]^{j-1}\right) \\
&= \sum_j \rho_j \phi^{-1}\left(1 - \left[1 - \sum_i [\lambda_i^{\pi}\phi((i-1)m_u^{(k-1)}) + \lambda_i^{1-\pi}\phi((i-1)m_u^{(k-1)} + m_{u0})]\right]^{j-1}\right)
\end{aligned}$$

where

$$r^{(k-1)} = \sum_i [\lambda_i^{\pi}\phi((i-1)m_u^{(k-1)}) + \lambda_i^{1-\pi}\phi((i-1)m_u^{(k-1)} + m_{u0})]$$

with

Definition 4.21 λ_i^{π} *denotes the proportion of punctured bits of degree i that equals, according to Bayes rule,*

$$\lambda_i^{\pi} = \lambda_i \pi_i = \sum_j Pi(i,j)\pi_{i,j}$$

Let us now show the following relation between $r_j^{(k)}$ and $r^{(k)}$.

$$r^{(k)} = \sum_j \rho_j r_j^{(k)} \tag{4.39}$$

Proof:

$$\sum_j \rho_j r_j^{(k)} = \sum_j \sum_i \rho_j [\lambda_i^{\pi}\phi((i-1)m_u^{(k-1)}) + \lambda_i^{1-\pi}\phi((i-1)m_u^{(k-1)} + m_{u0})]$$

By definition

$$\rho_j \lambda_{i,j} = Pi(i,j)$$

Then

$$\sum_j \rho_j r_j^{(k)} = \sum_i \sum_j [Pi(i,j)\pi_{i,j}^{(0)}\phi((i-1)m_u^{(k-1)}) + Pi(i,j)\pi_{i,j}^{(0)}\phi((i-1)m_u^{(k-1)} + m_{u0})]$$

According to Def. (4.21)

$$\sum_j \rho_j r_j^{(k)} = \sum_i [\lambda_i^{\pi}\phi((i-1)m_u^{(k-1)}) + \lambda_i^{1-\pi}\phi((i-1)m_u^{(k-1)} + m_{u0})]$$

Which is exactly

$$r^{(k)} = \sum_j \rho_j r_j^{(k)}$$

□

Then we have

$$\begin{aligned} m_u^{(k)} &= \sum_j \rho_j m_u^{(k)}(j) \\ m_u^{(k)} &= \sum_j \rho_j \phi^{-1}\left(1 - \left[1 - r^{(k-1)}\right]^{j-1}\right) \end{aligned}$$

For error-free decoding, this last recursive equation must grow to infinity, which is $m_u^{(k+1)} > m_u^{(k)}$ for any $k > 0$, or equivalently, with

$$r^{(k)} = \sum_j \rho_j r_j^{(k)}$$

which leads to another form for the condition of the convergence of the decoding

$$r^{(k)} > r^{(k-1)} \tag{4.40}$$

The design goal optimal puncturing defined in [9] is to maximize the puncturing fraction $p^{(0)}$ for a given E_b/N_0, such that Eq. (4.40) is fulfilled.
In our case of UEP code, the UEP using checks irregularity is ”defined” by the comparison between the gaps

$$r_j^{(k)} - r^{(k-1)} = H(m_{u_0}, \lambda_{i,j}^{\pi}, r_j^{(k-1)}) \tag{4.41}$$
$$r^{(k)} - r^{(k-1)} = H(m_{u_0}, \lambda^{\pi}, r^{(k-1)}) - r^{(k-1)}$$

Finally, puncturing such a UEP code requires to define a tolerance limiting how much the gap (4.41) can be decreased (it can not be increased) in order to not destroy the UEP properties more than we are allowed. These constraints on local gaps defining UEP must be included in the design of the detailed puncturing distribution $\pi_{i,j}^{(0)}$. The design of the detailed puncturing distribution $\pi_{i,j}^{(0)}$ could be done with the same means as used in [9], i.e. *discretized density evolution*, but this has not been studied further in this work.

Chapter 5

Conclusions

In this work we have proposed a method to optimize the unequal error protection properties of LDPC Codes. We have shown that it is possible to adapt the two kinds of irregularities in order to speed up the local convergence. We first discussed the definition of UEP properties, and highlighted the fact that an LDPC code can have UEP properties if decoded by maximum-likelihood, but none if decoded by belief propagation. UEP properties must then be defined depending on the used decoding.
We have adopted a detailed representation of LDPC codes allowing to describe subsets of possible interleavers that fit the UEP requirements, to define local convergence and to find a cost function. Since the irregularities of the bit node profile have already been studied, we especially focused on the check node profile optimization, keeping the bit node profile set regular. We found that the irregularities over check nodes does not only influence the speed of a local convergence, but also generates different behaviors at different parts of the codeword at high number of iterations, in contrast to irregularities over bit nodes; we tried to explain these two behaviors formally. This fact that UEP properties remain at high number of iterations is very interesting if we consider recent work in [5] which reduces the complexity of decoding, and then allows a higher number of iterations with the same resources. However, acting on check irregularities implied sub-optimality of the overall code in the case when the maximum degree of bit nodes is not adapted, and we then had to define a validity domain for our optimization, that then can be considered and achieved whether as a second stage in the optimization of the whole code, i.e. after bit nodes optimization, or as a first stage that would add a constraint on the following optimization. We would then have to keep all the parameters in the cost function, and optimize the check node profile in terms of the fixed bit nodes profile.
On a practical point of view, we tried to optimize a so-called mother code by pruning, i.e. by making some bits deterministic, in order to construct a subcode, with lower code rate, that fulfils the UEP requirements, and that can be decoded by the same decoder as the mother code, or a better one according to the available memory of the receiver. Finally we tried to briefly analyze what the optimal puncturing of such UEP codes should be, still using the detailed representation of LDPC codes, in order to compensate the code rate loss due to

pruning.
Such an optimization provides flexibility in selecting the appropriate scheme from performance, computational-complexity and memory-requirements perspectives.

As further tasks, testing the robustness to variations of proportions of classes should be useful considering practical applications of such codes. Another work would be to optimize both kinds of irregularities in a joint way, and not sequentially anymore, by properly describing the cost function, and still considering the required performance and the constraints of the target system. The difficulty of such an approach would lie in the non-linearity of the optimization.

Appendix A

A.1 Transition from proportions of edges to proportions of nodes

$$\begin{aligned}
\tilde{\lambda}_i &= \text{proportion of bit nodes of degree } i \\
&= \frac{\text{number of bit nodes of degree } i}{\text{number of bit nodes in the whole graph}} \\
&= \frac{\frac{\text{number of edges linked to bit nodes of degree } i}{i}}{\sum_k \frac{\text{number of edges linked to bit nodes of degree } k}{k}} \\
&= \frac{\frac{\text{total number of edges } \lambda_i}{i}}{\sum_k \frac{\text{total number of edges } \lambda_k}{k}}
\end{aligned}$$

Which is translated by

$$\tilde{\lambda}_i = \frac{\frac{\lambda_i}{i}}{\sum_k \frac{\lambda_k}{k}}$$

The same arguing is carried out to obtain the similar expression for $\tilde{\rho}_j$.

A.2 Transition from proportions of nodes to proportions of edges

$$\begin{aligned}
\lambda_i &= \text{proportion edges linked to bit nodes of degree } i \\
&= \frac{\text{number edges linked to bit nodes of degree } i}{\text{number of edges in the whole graph}} \\
&= \frac{\text{number of bit nodes of degree } i.i}{\sum_k \text{number of to bit nodes of degree } k.k} \\
&= \frac{\text{total number of bit nodes } \lambda_i.i}{\sum_k \text{total number of bit nodes } \lambda_k.k}
\end{aligned}$$

Which is translated by

$$\lambda_i = \frac{\lambda_i.i}{\sum_k \lambda_k.k}$$

The same arguing is carried out to obtain the similar expression for ρ_j.

A.3 Expression of the code rate in terms of proportions

$$R = 1 - \frac{\text{total number of check nodes}}{\text{total number of bitnodes}}$$

According to the previous proofs, we easily obtain :

$$R = 1 - \frac{\sum_j \frac{\text{total number of edges } \rho_j}{j}}{\sum_i \frac{\text{total number of edges } \lambda_i}{i}}$$

Finally

$$R = 1 - \frac{\sum_j \frac{\rho_j}{j}}{\sum_i \frac{\lambda_i}{i}}$$

Bibliography

[1] A. Amraoui. *LDPCopt*, http://lthcwww.epfl.ch/research/ldpcopt/.

[2] A. Ashikhmin, G. Kramer, and S. ten Brink. Extrinsic Information Transfer Functions : Model and Erasure Channel Properties. *IEEE Trans. on Inform. Theory*, 50(11):2657–2673, November 2004.

[3] R.E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley, 1984.

[4] I.M. Boyarinov and G.L. Katsman. Linear Unequal Error Protection Codes. *IEEE Trans. on Inform. Theory*, 27(2):168–175, March 1981.

[5] J.C. Chen, A. Dholakia, E. Eleftheriou, M.P.C Fossorier, and X-Y Hu. Reduced-Complexity Decoding of LDPC Codes. *IEEE Trans. on Communications*, 53(8):1288–1299, August 2005.

[6] S.Y. Chung, T. Richardson, and R. Urbanke. Analysis of Sum-Product Decoding Low-Density Parity-Check Codes using a Gaussian Approximation. *IEEE Trans. on Inform. Theory*, 47(2):657–670, February 2001.

[7] D. Declercq. Optimisation et performances des codes LDPC pour des canaux non-standards. In *Habilitation à diriger des recherches*, Université de Cergy-Pontoise, December 2003.

[8] R.G. Gallager. Low-Density Parity-Check Codes. *IRE Trans. on Inform. Theory*, pages 21–28, 1962.

[9] J. Ha and S.W McLaughlin. Optimal Puncturing of Irregular Low-Density Parity-Check Codes. In *IEEE International Conference on Communications*, pages 3110–3114, Anchorage, Alaska, USA, May 2003.

[10] S. Haykin. *Communication Systems*. J. Wiley and Sons, Inc., 4th edition, 2001.

[11] K. Kasai, T. Shibuya, and K. Sakaniwa. Detailedly Represented Irregular Low-Density Parity-Check Codes. *IEICE Trans. Fundamentals*, E86-A(10):2435–2443, October 2003.

[12] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor Graphs and the Sum-Product Algorithm. In *IEEE Trans. on Inform. Theory*, volume to see, page to see, July 1998.

[13] G. Lechner. Convergence of the Sum-Product Algorithm for Short Low-Density Parity-Check Codes. Master's thesis, Vienna University of Technology, April 2003.

[14] D.J.C. MacKay and R.M. Neal. Near Shannon Limit Performance of Low-Density Parity-Check Codes. *Electronics Letters*, 33(6):457, March 1997.

[15] D.J.C. MacKay, S.T. Wilson, and M.C. Davey. Comparison of Constructions of Irregular Low-Density Parity-Check Codes. *IEEE Trans. on Communications*, 47(10):1449–1453, October 1999.

[16] B. Masnick and J. Wolf. On Linear Unequal Error Protection Codes. *IEEE Trans. on Inform. Theory*, 3, no.4:600–607, October 1967.

[17] C. Poulliat. *Allocation et Optimisation de Ressources pour la Transmission de Donnees Multimedia*. PhD thesis, Université de Cergy-Pontoise, 2004.

[18] C. Poulliat, D. Declercq, and I. Fijalkow. Optimization of LDPC Codes for UEP Channels. In *ISIT 2004*, Chicago, USA, June 2004.

[19] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes. *IEEE Transactions on Communications*, 47(2):619–637, February 2001.

[20] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under massage-passing decoding. *IEEE Trans. on Inform. Theory*, 47:599–618, February 2000.

[21] A. Roumy. *Lecture Notes*, DEA TIS, 2005.

[22] S. ten Brink. Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

[23] T. Tian, C. Jones, J.D. Villasenor, and R.D. Wesel. Construction of Irregular LDPC Codes with Low Error Floors. In *ICC2003*, Anchorage, Alaska, USA, 2003.

[24] N. von Deetzen. Unequal Error Protection Turbo Codes. Master's thesis, Universität Bremen, February 2005.

[25] X. Yang and D. Yuan. New Research on Unequal Error Protection Property of Irregular LDPC Codes. In *IEEE Consummer Communications and Networking Conference*, pages 361–363, January 2004.