HAL
open science

Codes LDPC multi-binaires hybrides et méthodes de décodage itératif

Lucile Sassatelli

To cite this version:

Lucile Sassatelli. Codes LDPC multi-binaires hybrides et méthodes de décodage itératif. Théorie de l'information [cs.IT]. Université de Cergy Pontoise, 2008. Français. NNT : . tel-00819413

HAL Id: tel-00819413

https://theses.hal.science/tel-00819413

Submitted on 1 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ECOLE DOCTORALE SCIENCES ET INGENIERIE
De l'Université de Cergy-Pontoise

THÈSE

Présentée pour obtenir le grade de docteur de l'université de Cergy-Pontoise

Spécialité : Traitement des Images et du Signal

Codes LDPC multi-binaires hybrides et méthodes de décodage itératif

par
Lucile SASSATELLI

Laboratoire ETIS - UMR CNRS 8051

3 octobre 2008

Devant le jury composé de :

M. E. Biglieri,	Examinateur
M. J. Blanc-Talon,	Examinateur
M. B. Vasic,	Examinateur
M. G. Zémor,	Examinateur
M. J. Boutros,	Rapporteur
M. J.-P. Tillich,	Rapporteur
M. D. Declercq,	Directeur de thèse

Résumé

Cette thèse porte sur l'analyse et le design de codes de canal définis par des graphes creux. Le but est de construire des codes ayant de très bonnes performances sur de larges plages de rapports signal à bruit lorsqu'ils sont décodés itérativement.

Dans la première partie est introduite une nouvelle classe de codes LDPC, nommés code LDPC hybrides. L'analyse de cette classe pour des canaux symétriques sans mémoire est réalisée, conduisant à l'optimisation des paramètres, pour le canal gaussien à entrée binaire. Les codes LDPC hybrides résultants ont non seulement de bonnes propriétés de convergence, mais également un plancher d'erreur très bas pour des longueurs de mot de code inférieures à trois mille bits, concurrençant ainsi les codes LDPC multi-edge. Les codes LDPC hybrides permettent donc de réaliser un compromis intéressant entre région de convergence et plancher d'erreur avec des techniques de codage non-binaires.

La seconde partie de la thèse a été consacrée à étudier quel pourrait être l'apport de méthodes d'apprentissage artificiel pour le design de bons codes et de bons décodeurs itératifs, pour des petites tailles de mot de code. Nous avons d'abord cherché comment construire un code en enlevant des branches du graphe de Tanner d'un code mère, selon un algorithme d'apprentissage, dans le but d'optimiser la distance minimale. Nous nous sommes ensuite penchés sur le design d'un décodeur itératif par apprentissage artificiel, dans l'optique d'avoir de meilleurs résultats qu'avec le décodeur BP, qui devient sous-optimal dès qu'il y a des cycles dans le graphe du code.

Dans la troisième partie de la thèse, nous nous sommes intéressés au décodage quantifié dans le même but que précédemment : trouver des règles de décodage capables de décoder des configurations d'erreur difficiles. Nous avons proposé une classe de décodeurs utilisant deux bits de quantification pour les messages du décodeur. Nous avons prouvé des conditions suffisantes pour qu'un code LDPC, avec un poids de colonnes égal à quatre, et dont le plus petit cycle du graphe est de taille au moins six, corrige n'importe quel triplet d'erreurs. Ces conditions montrent que décoder avec cette règle à deux bits permet d'assurer une capacité de correction de trois erreurs pour des codes de rendements plus élevés qu'avec une règle de décodage à un bit.

Abstract

This thesis is dedicated to the analysis and the design of sparse-graph codes for channel coding. The aim is to construct coding schemes having high performance both in the waterfall and in the error-floor regions under iterative decoding.

In the first part, a new class of LDPC codes, named hybrid LDPC codes, is introduced. Their asymptotic analysis for memoryless symmetric channel is performed, and leads to code parameter optimization for the binary input Gaussian channel. Additionally to a better waterfall region, the resulting codes have a very low error-floor for code rate one-half and codeword length lower than three thousands bits, thereby competing with multi-edge type LDPC. Thus, hybrid LDPC codes allow to achieve an interesting trade-off between good error-floor performance and good waterfall region with non-binary coding techniques.

In the second part of the thesis, we have tried to determine which kind of machine learning methods would be useful to design better LDPC codes and better decoders in the short code length case. We have first investigated how to build the Tanner graph of a code by removing edges from the Tanner graph of a mother code, using a machine learning algorithm, in order to optimize the minimum distance. We have also investigated decoder design by machine learning methods in order to perform better than BP which is suboptimal as soon as there are cycles in the graph.

In the third part of the thesis, we have moved towards quantized decoding in order to address the same problem: finding rules to decode difficult error configurations. We have proposed a class of two-bit decoders. We have derived sufficient conditions for a column-weight four code with Tanner graph of girth six to correct any three errors. These conditions show that decoding with the two-bit rule allows to ensure weight-three error correction capability for higher rate codes than the decoding with one bit.

Remerciements

Je tiens à remercier les membres de mon jury d'avoir accepté d'évaluer cette thèse, et notamment Jean-Pierre Tillich et Joseph Boutros d'en avoir été les rapporteurs. Leurs compétences et leur générosité ont grandement contribué à l'amélioration du manuscript.

J'exprime toute ma gratitude à Bane Vasic, pour le temps qu'il m'a consacré, pour les échanges scientifiques particulièrement enrichissants, pour les grandes discussions culturelles, pour son soutien et sa sympathie. Je remercie également Shashi Kiran Chilappagari avec qui j'ai pris un réel plaisir à travailler.

Je remercie très chaleureusement Charly Poulliat, pour nos échanges quotidiens pendant ces trois années, son implication, ses conseils avisés et déterminants dans les moments de doute.

Un grand merci également à Adrian pour son soutien pendant ces trois années à partager le même bureau, à Auguste, dont les codes m'ont permis de gagner un temps précieux, pour sa perpétuelle bonne humeur et son enthousiasme débordant. Merci à Ayman, Sonia, David, Dimitri, Heykel, Abdel-Nasser pour la bonne ambiance au labo.

Enfin merci à mes parents qui m'ont fait confiance, à mon frère qui m'a toujours encouragée, et à celui qui a été mon soutien indéfectible pendant ces trois années.

Contents

Résumé **3**

Abstract **5**

Remerciements **7**

Introduction **17**
Context 17
Motivation 18
Outline 19
Contributions 21

1 Introduction to binary and non-binary LDPC codes **23**
1.1 Linear block error-correcting codes 23
1.2 Definition and parametrization of LDPC codes 25
1.3 General notation 27
1.4 Decoding of LDPC codes by Belief Propagation algorithm 28
1.5 Analysis of LDPC codes 32
1.5.1 Additional notation 32
1.5.2 Channel and message symmetry 33
1.5.3 Density evolution for $GF(q)$ LDPC codes 34
1.5.4 Approximation of message densities by only one scalar parameter 36
1.5.5 The stability condition 40
1.5.6 Design example of $GF(q)$ LDPC code ensemble on BIAWGN channel 40
1.6 Other design techniques 41
1.6.1 Finite length design of LDPC codes 41
1.6.2 Structured ensembles 42
1.7 Proof of theorems in Chapter 1 42

2 Hybrid LDPC Codes **45**
2.1 The class of hybrid LDPC codes 45
2.1.1 General hybrid parity-check equations 45
2.1.2 Hybrid LDPC code ensemble 47
2.1.3 Different sub-classes of hybrid LDPC codes 47

2.1.4 Hybrid LDPC codes with linear maps 48
2.1.5 Parametrization of hybrid LDPC ensemble 49
2.1.6 Encoding of hybrid LDPC codes 52
2.1.7 Decoding algorithm for hybrid LDPC codes 52
2.2 Asymptotic analysis of hybrid LDPC code ensembles 54
2.2.1 Symmetry of the messages 54
2.2.2 Density evolution 55
2.2.3 Invariance induced by linear maps (LM-invariance) 56
2.2.4 The Stability condition for hybrid LDPC Codes 57
2.2.5 EXIT charts and accuracy of the approximation for hybrid LDPC codes 60
2.3 Distributions optimization 65
2.3.1 Context of the optimization 65
2.3.2 Optimization with multi-dimensional EXIT charts 66
2.3.3 Optimization with mono-dimensional EXIT charts 69
2.4 Finite length optimization 72
2.4.1 Row optimization 73
2.4.2 Avoiding low weight codewords 73
2.5 Results 74
2.5.1 Rate one-half codes 74
2.5.2 Rate one-sixth codes 77
2.6 Conclusions 80
2.7 Proofs of theorems in Chapter 2 81
2.7.1 Symmetry 83
2.7.2 A useful lemma 85
2.7.3 LM-invariance 86
2.7.4 Proof of Theorem 3 88
2.7.5 Information content Through Linear Maps 91
2.7.6 Mutual information of a probability vector and its Fourier Transform 92

3 Machine Learning Methods for Code and Decoder Design 95
3.1 Previous works 95
3.1.1 Information-theoretic models of artificial neural networks 95
3.1.2 Learning methods and error-correcting codes 96
3.2 Machine Learning Methods for Code Design 96
3.2.1 Problem 96
3.2.2 Neural networks and codes Tanner graphs 97
3.2.3 Cost function for pruning 100
3.2.4 Pruning methods 100
3.3 Machine Learning Methods for Decoder Design 102
3.3.1 Decoding is a classification problem 102
3.3.2 Modelization of BP decoding 104
3.3.3 Cost function 104
3.3.4 Solving the minimization problem 106

3.3.5 Estimating the mutual information 107
3.3.6 Some other methods 109
3.4 Conclusion 112

4 Two-Bit Message Passing Decoders for LDPC Codes Over the Binary Symmetric Channel 115
4.1 Introduction 115
4.2 The class of two-bit decoders 117
4.3 Guaranteed weight-three error correction 119
4.3.1 Sufficient condition for correction of three errors 119
4.4 Asymptotic analysis 128
4.4.1 Density evolution 128
4.4.2 Thresholds of quantized decoders 129
4.5 Numerical results 131
4.6 Conclusion 131

Conclusions and Perspectives 133
Conclusions 133
Perspectives 134

List of Tables

1.1 Thresholds of $GF(q)$ LDPC code ensembles with constant check degree d_c and code rate one half, optimized with EXIT charts on the BIAWGN channel. The maximum variable degree allowed in the optimization procedure is $d_{v_{max}} = 30$. Thresholds are given in term of the SNR $\frac{E_b}{N_0}$ in dB, and are obtained using the Gaussian approximation. 41

2.1 Distribution $\Pi(i, j, k, l)$ of a hybrid LDPC code ensemble with code rate one-half and threshold 0.1864 dB under Gaussian approximation. The marginals $\tilde{\Pi}(i, k)$ and $\tilde{\Pi}(j, l)$ correspond to the proportions of variable nodes of type (i, k) and check nodes of type (j, l), respectively. When a proportion is forced to zero by the sorting constraint, $\times$ is put in the box. . 68

2.2 Nodewise distributions of the hybrid LDPC codes used for the finite length simulations. 74

2.3 Nodewise distribution of the rate $\frac{1}{6}$ and $\frac{1}{12}$ hybrid LDPC codes 78

4.1 Examples of message update for a column-weight-four code, when $C = 2$, $S = 2$ and $W = 1$. 119

4.2 Decision rule for the two-bit decoder defined by $(C, S, W) = (2, 2, 1)$. . . 121

4.3 Update rule for the two-bit decoder defined by $(C, S, W) = (2, 2, 1)$. . . . 122

4.4 Thresholds of different decoders for column-weight-four codes with row degree ρ. 130

List of figures

1.1 Parity-check matrix of a non-binary LDPC code and its bipartite graph. . 26
1.2 Representation of a ensemble of irregular LDPC codes. 28
1.3 Variable node update 30
1.4 Check node update 31
1.5 EXIT curves of $(2, 4)$ $GF(2)$, $GF(8)$ and $GF(256)$ regular codes. The SNR is 0.7dB. 42

2.1 Factor graph of parity-check of an hybrid LDPC code. 46
2.2 Message transform through linear map. 48
2.3 Parametrization of a hybrid LDPC code ensemble 51
2.4 Quantities Ω for hybrid and non-hybrid LDPC codes in terms of maximum symbol order q_{max}. These figures show that a hybrid LDPC code can be stable when a non-binary code is not. 59
2.5 FER versus $\frac{E_b}{N_o}$: code rate one-half. $K = 1024$ information bits except for the multi-edge type LDPC code for which $K = 1280$ information bits. No finite length optimization has been applied. $N_{iter} = 500$ except for quasi-cyclic LDPC code (from [1]) for which $N_{iter} = 50$. 76
2.6 FER versus $\frac{E_b}{N_o}$ (in dB): code rate one-half. $N_{bit} = 2048$ coded bits except for the multi-edge type LDPC code for which $N_{bit} = 2560$ coded bits. $N_{iter} = 500$ decoding iterations are performed. 77
2.7 Comparison of hybrid LDPC code with Turbo Hadamard codes (TH) taken from [2] and Zigzag Hadamard (ZH) codes taken from [3], for an information block length of $K_{bit} \simeq 200$. $N_{iter} = 30$ for Turbo Hadamard codes, and $N_{iter} = 200$ for the hybrid LDPC codes. 79
2.8 Comparison of hybrid LDPC code with punctured Turbo Hadamard (PTH) taken from [4] and other powerful codes, for code rate $1/6$. The PTH code has $K_{bit} = 999$ information bits, and the other codes have $K_{bit} = 1024$ information bits. $N_{iter} = 50$ for the PTH code, and $N_{iter} = 200$ for the other codes. 80

3.1 General definition of a formal neuron 97
3.2 An artificial neuron which computes the weighted sum of the inputs, and the apply the activation function f. 98
3.3 A polynomial neuron. 98

3.4 A factor graph and its corresponding neural network. Each neuron corresponds to an edge of the factor graph, hence there are $2.N_{edge}.N_{iter}$ neurons in the network. 99
3.5 Voronoi diagram (or Dirichlet tessellation): the partitioning of a plane with n points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other. 103
3.6 Evolution of the mutual information of variable to check messages along iteration of BP decoding of various codes. Transmission on AWGN channel with $\frac{E_b}{N_o} = 2$dB. The upper hashed dotted curve corresponds to the EXIT function of a cycle-free (3,6) LDPC code. The steps correspond to BP decoding of various finite-length (3,6) LDPC codes. 105
3.7 Flow chart of the optimization procedure using a genetic algorithm to find the best weights minimizing the cost function, for each iteration. N_{iter} is the number of decoding iterations for which we look for the correcting weights. 108

4.1 All possible subgraphs subtended by three erroneous variable nodes. . . . 121
4.2 Errors configuration for Case 2. 121
4.3 Errors configuration for Case 4. 124
4.4 Errors configuration for Case 5. 127
4.5 FER versus the crossover probability α for regular column-weight-four MacKay code. The code rate is 0.89 and the code length is $n = 1998$. . . 131

Introduction

Context

In 1948, Claude Shannon published a paper [5] in which he laid down the foundations of information theory. Shannon's original work on information theory was in direct response to the need to design communication systems that are both efficient and reliable. Reliable means that no loss of information occurs during transmission. In particular, information theory addresses both the limitations and the possibilities of reliable transmission of information over a communication channel. The noisy channel coding theorem asserts both that reliable communication at any rate beyond the channel capacity is impossible, and that reliable communication at all rates up to channel capacity is possible.

The central problem of communication theory is to construct an encoding and a decoding system to communicate reliably over a noisy channel.

During the 1990s, remarkable progress was made towards the Shannon limit, using codes that are defined in terms of sparse random graphs, and which are decoded by a simple probability-based message-passing algorithm. In a sparse-graph code, the nodes in the graph represent the transmitted bits and the constraints they satisfy. Hence, there are two kinds of nodes in the graph, which is therefore called bipartite graph. For a linear code which encodes K information bits into a codeword of N bits, the rate is $R = \frac{K}{N}$ and the number of constraints is of order $M = N - K$. Any linear code can be described by a graph, but what makes a sparse-graph code special is that each constraint involves only a small number of variables in the graph. The edges of the graph define a permutation, and that is why a sparse-graph code is said to rely on a random permutation. These codes are very interesting because they can be decoded by a local message-passing algorithm on the graph, and, while this algorithm is not a perfect maximum likelihood decoder, the empirical results are record-breaking.

We can mention two ensembles of sparse-graph codes which have excellent error-correction capability: Low-Density Parity-Check (LDPC) codes, and Turbo Codes. The class of LDPC codes was first proposed in [6] in 1963, and rediscovered thirty years later [7, 8, 9, 10, 11], after the invention of Turbo Codes [12]. This thesis investigates channel coding schemes based on LDPC codes. LDPC codes are decoded through the iterative local message-passing algorithm based on the *Belief Propagation* (BP) principle [13]. These codes have been shown to exhibit very good performance under iterative BP decoding over a wide range of communication channels, approaching channel capacity

with moderate decoding complexity.

Asymptotically in the codeword length, LDPC codes exhibit a threshold phenomenon. In other words, if the noise level is smaller than a certain decoding threshold (which depends on the bipartite graph properties) then it is possible to achieve an arbitrarily small bit error probability under iterative decoding, as the codeword length tends to infinity. On the contrary, for noise level larger than the threshold, the bit error probability is always larger than a positive constant, for any codeword length [10, 11]. On the Binary-input Additive White Gaussian Noise (BIAWGN) channel, this threshold value is defined in terms of Signal-to-Noise Ratio (SNR), on the Binary Symmetric Channel (BSC) in terms of error probability, on the Binary Erasure Channel (BEC) in terms of erasure probability. There are two main tools for asymptotic analysis of LDPC codes, i.e. for evaluating the decoding threshold associated to a given degree distribution: density evolution [10] and EXtrinsic Information Transfer (EXIT) charts [14]. One of the features that makes LDPC codes very attractive is the possibility to design, for several transmission channels, the degree distribution of the bipartite graph which provides a decoding threshold extremely close to the channel capacity [15]. For given code rate and node degrees, the threshold optimization is usually performed by means of numerical optimization tools, like differential evolution [16]. In the particular case of the BEC, where the transmitted bits are either correctly received or lost independently with some erasure probability, it was also shown that it is possible to design sequences of degree distributions, known as capacity-achieving sequences [17], whose threshold converges to the channel capacity.

Motivation

While the asymptotic design and analysis of LDPC codes is mostly understood, the design of finite length LDPC codes still remains an open question.

Indeed, the local message-passing algorithm, which is the BP decoder for LDPC codes, corresponds to the exact computation of *a posteriori* probabilities of variable values only if the graph is cycle-free, i.e., when the probability messages going into a node along the decoding iterations can be assumed independent. In that case, the BP decoder is exactly the Maximum-Likelihood (ML) decoder because it finds the global maximum of the ML criterion. This assumption is made for asymptotic study, when the codeword length is assumed to be infinite. In the finite length case, cycles appear in the graph [18]. In that case, the BP decoder does not compute anymore the *a posteriori* probabilities of variable values, thereby turning into suboptimal in the sense it does not correspond anymore to ML decoding. However, the BP decoding of LDPC code is based on this assumption thanks to the property of the graph of the code, which is sparse by definition of this class of codes. Many works [19, 20] have characterized the phenomenon which arises when BP decoder is used on loopy graphs, and which points out the difference between ML decoding and BP decoding. ML decoding is always able to find the codeword closest to the observation (even though it makes errors because this closest codeword is not the one which has been sent), whereas BP decoder may converge to fixed points which are not codewords. These points are usually called *pseudo-codewords*, and it has been shown

[19] that they are of first importance in the loss of performance of BP decoding, compared to ML decoding, and particularly in the *error floor* region. When the LDPC code is decoded by message passing algorithms, the frame error rate (FER) curve of has two regions: as the channel parameter decreases, the slope of the FER curve first increases, and then sharply decreases. This region of low slope for small channel parameter is called the *error floor* region.

Moreover, finite length LDPC codes with a degree distribution associated to a decoding threshold close to capacity, though characterized by very good waterfall performance, usually exhibit a bad error floor performance, due to poor minimum distance [21, 22]. Indeed, the capacity-approaching sequences of LDPC codes have a large fraction of degree two variable nodes [17, 10], which gives rise to low-weight codewords. Such codewords correspond to cycles in the subgraph of the Tanner graph which contain only degree two variable nodes.

To construct code ensembles with iterative decoding performance close to channel capacity and having a low error-floor, one needs to choose the random permutations, which make LDPC codes pseudo-random codes, in a structured way to avoid short cycles. The code ensembles with a structured choice of permutations are called structured. Hence, the design of finite length LDPC codes mostly relies on finding the best trade-off between the waterfall and error-floor regions, by carefully constructing the bipartite graph of the code. One of the most popular technique to design the graph, i.e., the permutations, of a code, is the Progressive-Edge-Growth (PEG) construction [23]. Code ensembles that have been studied in order to well perform in the finite-length case are those based on finite geometries [8] and on circulant permutation matrices [24]. More particularly, some structured code ensembles have been under the scope of many studies these last years: Irregular Repeat-Accumulate (IRA) codes [25], protograph-based LDPC codes [26] and multi-edge type LDPC [27]. These techniques, or their combinations, lead to codes with good code properties in terms, for instance, of girth of the bipartite graph and possibility to perform the encoding procedure efficiently.

The attempt to improve the trade-off between waterfall performance and error floor has recently inspired the study of more powerful, and somewhat more complex, coding schemes. This is the case of non-binary LDPC codes, Generalized LDPC (GLDPC) codes [28], Doubly-Generalized LDPC (D-GLDPC) codes [29] or Tail-biting LDPC (TLDPC) codes [30]. Non-binary LDPC codes have been introduced by Davey in [31]. The main interest of non-binary LDPC codes actually lies in the decoder: good non-binary LDPC codes have much sparser factor graphs (or Tanner graphs) than binary LDPC codes [32], and the BP decoder is closer to optimal decoding since the small cycles can be avoided with a proper graph construction, as proposed in [33].

Outline

This thesis encompasses three distinct chapters, in which three different methods are investigated with the same aim: designing new coding schemes in order to improve the

trade-off between waterfall performance and error floor.

The first chapter is dedicated to introduce the useful notions about binary and non-binary LDPC codes, as well as the existing tools for their analysis.

In the second chapter, we introduce and study a new class of LDPC codes that we call *multi-binary hybrid LDPC codes*. The class of hybrid LDPC codes is a generalization of existing classes of LDPC codes, both binary and non-binary. For hybrid LDPC codes, we allow the connectivity profile to be irregular and the orders of the symbols in the codeword to be heterogeneous. The asymptotic analysis of this class of codes is performed with a given detailed representation to derive stability condition and EXIT charts analysis. The study is performed on the BIAWGN channel, whereas studies of generalized LDPC codes usually consider the BEC [30, 29] where the one parameter approximation of message densities is straightforward, unlike for the BIAWGN channel. Thus, for the EXIT chart analysis, we have tried to provide an as complete as possible analysis of the accuracy of the projection of message densities on only one scalar parameter. Distributions are optimized and some thresholds computed. We show how the finite length optimization method of [34] can be adapted and applied to get very low error floor. We finally present experimental results for code rate one half, as well as for code rate one sixth.

The third chapter reviews the investigation done on the initial topic of this thesis: how some machine learning methods might be applied to the bipartite graph of a code for finite length optimization purpose? The final goal was to use hybrid LDPC codes as a tool for building codes with good finite length properties by means of a learning algorithm to be determined.

First, we are interested in code design. We look for a way to build the Tanner graph of a code by means of a supervised learning process applied to the graph of a mother code in order to decide which edges should be pruned away in order to lower the sub-optimality of the BP decoder.

Then, we move towards decoder design for a given LDPC code. We investigate how to modify the BP decoder by adapting it to the graph of a given code, in order to lower its sensibility to graph cycles. For this purpose, the BP decoder has been considered as a classifier with room for improvement.

The fourth chapter also aims at finding good decoders well performing on finite length LDPC codes, but with good asymptotic behavior too. In this chapter, we switch from continuous BP decoding to quantized decoding. The idea is still to find a decoding rule adapted to topologies hard to decode, like trapping sets [35]. To do so, a class of two-bit message passing decoders is proposed for the binary symmetric channel. The thresholds for various decoders in this class are derived using density evolution. For a specific decoder, the sufficient conditions for a column-weight-four LDPC code to correct all patterns up to three errors are derived. A code satisfying the conditions is constructed and numerical assessment of the code performance is provided via simulation results.

Contributions

In the present thesis, we proposed the following contributions:

- A new class of non-binary LDPC codes, named hybrid LDPC codes, is studied.
  - The asymptotic analysis is presented: the property of Linear-Application invariance is exhibited for the code ensemble, leading to a stability condition and an EXIT charts analysis for AWGN channels. Two kinds of EXIT charts of hybrid LDPC codes are studied: multi-dimensional and mono-dimensional EXIT charts.
  - Study of the condition allows to conclude that there exist many cases where any fixed point of density evolution for hybrid LDPC codes can be stable at lower SNR than for non-binary codes.
  - For the EXIT chart analysis, a detailed analysis of the accuracy of the approximation of message densities by one scalar parameter is provided.
  - Distribution optimization are performed to get finite-length codes with very low connection degrees and better waterfall region than protograph or multi-edge type LDPC codes.
  - A cycle cancellation technique is applied to hybrid LDPC codes, which are well fitted to such a technique, thanks to their specific structure.
  - The resulting codes appear to have, additionally to a better waterfall region, a very low error-floor for code rate one-half and codeword length lower than three thousands bits, thereby competing with multi-edge type LDPC. Thus, hybrid LDPC codes allow to achieve an interesting trade-off between good error-floor performance and good waterfall region with non-binary coding techniques.
- An investigation on how machine learning methods could be used for finite length optimization of LDPC coding schemes has been led:
  - It has been shown that no learning algorithm can be used to build a code from pruning the Tanner graph of a mother code, when the aim is simultaneously to have a high minimum distance and to exploit the value of the messages during the iterative decoding.
  - Decoder design, with machine learning methods, has been investigated. The decoding has been defined as a classification problem to which a better decoder than BP may be found, in order to handle message statistical dependencies. The neural network corresponding to the BP decoding has been expressed. To determine optimal synaptic weights to perform better than BP on a finite length code, we proposed a cost function based on the difference between an estimated mutual information and the EXIT chart. The reason why this approach fails has been detailed.

- Several classification methods have been studied to see whether they might advantageously substitute the BP decoder. The fundamental reason why this is not possible is exhibited: those methods are non-parametric machine learning algorithms where the elements to be classified, must be highly non-uniformly distributed. However, the channel coding problem corresponds to the opposite case.

- A class of two-bit message passing decoders for decoding column-weight-four LDPC codes over the binary symmetric channel is proposed and analyzed.
  - Thresholds are derived for various decoders in this class.
  - We consider a specific decoder in this class, and prove sufficient conditions for a code with Tanner graph of girth six to correct three errors.
  - A code satisfying the conditions is constructed and numerical assessment of the code performance is provided via simulation results.

Chapter 1

Introduction to binary and non-binary LDPC codes

This chapter introduces the binary and non-binary LDPC codes. The general channel coding problem is shortly explained, notations and definitions are given, and a non-extensive review of analysis tools necessary for the following is done.

1.1 Linear block error-correcting codes

A linear block code is a linear map which associates to K information symbols, N coded symbols, by adding $N - K$ redundancy symbols in order to lower the error probability when the transmission occurs over a noisy channel.

The linear map is described by G in the reminder, and the codewords set is denoted by $\mathcal{C}$ and called the code. The bold notation $\mathbf{G}$ is used to denote the matrix associated with the linear map G. When the code is defined over $GF(2)$, the codeword set corresponds to the image of $\{0,1\}^K$ by the linear map, and it is denoted by $\mathcal{C}$:

$$G: \quad \{0,1\}^K \rightarrow \mathcal{C} \subseteq \{0,1\}^N$$

To shorten the notations, we write: $\mathcal{C} = Im(G)$. This means that for any codeword $\mathbf{c} \in \{0,1\}^N$ of size $N \times 1$, there exists one unique information vector $\mathbf{v} \in \{0,1\}^K$ of size $K \times 1$ such that $\mathbf{c} = \mathbf{G}\mathbf{v}$, where the size of $\mathbf{G}$ is $N \times K$. Thus, a linear block code is determined by $\mathbf{G}$, which is called the generator matrix, but it can be also determined by $\mathbf{H}$ of size $(N-K) \times N$, which is called the parity-check matrix. Indeed, $\mathbf{H}$ is the matrix of the linear map whose image is the kernel of the application G. Hence, the following property allows us to determine whether a vector in $\{0,1\}^N$ belongs to the code $\mathcal{C}$:

$$\forall \mathbf{c} \in \mathcal{C}, \quad \mathbf{H} \cdot \mathbf{c} = \mathbf{0}$$

which is also equivalent to

$$\forall \mathbf{v} \in \{0,1\}^K, \quad \mathbf{HG} \cdot \mathbf{v} = \mathbf{0}$$

Consider a transmission over a noisy channel. Let $\mathbf{X}$ be the input random vector and let $\mathbf{Y}$ be the output random vector. We assume that $\mathbf{Y}$ depends on $\mathbf{X}$ via a conditional probability density function $\mathbf{P}_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y})$. Given a received vector $\mathbf{y} = (y_0, \ldots, y_{N-1})$, the most likely transmitted codeword is the one that maximizes $\mathbf{P}_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y})$ [36]. If the channel is memoryless and each of the codewords are equally likely, then this reduces to the codeword $\mathbf{x} = (x_0, \ldots, x_{N-1})$ which maximizes $\mathbf{P}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$. This is known as maximum likelihood (ML) estimate of the transmitted codeword and is written as follows [36]:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{C}} \mathbf{P}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$$

where the maximization is done over the input alphabet of the channel.

Now we discuss the correction capability of a linear block code. The correction ability of a code is determined by its minimum distance d_{min}, which is the smallest Hamming distance between two codewords [37]. From an algebraic perspective, the received vector is the sent codeword with some components corrupted. The error correction, i.e. the decoding process, consists in finding the nearest codeword to the received vector. All the vectors in $\{0,1\}^N$ whose nearest codeword is $\mathbf{x}$ are such that, for all $i \in 1, \ldots, N$, if the ith bit of the vector is different from the ith bit of the codeword $\mathbf{x}$, then the Hamming distance between $\mathbf{x}$ and the vector must be lower than $\frac{d_{min}^{loc}(i)}{2}$, with $d_{min}^{loc}(i)$ being the local minimum distance of bit i in the code, as defined in [38]. The local minimum distance on the ith digit corresponds to the minimum Hamming distance between two codewords whose the ith digits are different [38]. Hence, the maximum number of errors that a code can detect is $d_{min} - 1$, whatever the location of the errors in the codeword. Similarly, if the error correction is achieved according to the ML principle, the maximum number of errors that the code is able to correct is $\lfloor \frac{d_{min}}{2} \rfloor$. The maximum number of correctable errors is hence $\lfloor \frac{d_{min}-1}{2} \rfloor$, whatever the location of the errors in the codeword.

ML decoding corresponds to solve the nearest neighbor problem. Looking for the nearest neighbor in a high-dimensional space is an algorithmic problem which does not have a better solution than an exhaustive search when the space elements are not sorted. Thus, the decoding process can be very complex ($\mathcal{O}(2^K)$) [37]. This is brute force approach is reasonable only for short length codes. Faster sub-optimal solutions have been developed. The first one is applied to block codes like BCH [39] and Reed-Solomon codes [40]. In these approaches, the code is built with the *a priori* knowledge of the minimum distance, and built so as the nearest neighbor search can be performed in reduced subspaces. The second coding scheme which allows to have good minimum distance with acceptable decoding speed is based on convolutional codes. Encoding is done thanks to linear feedback shift registers fed by information bits. This technique generates a set $\mathcal{C}$ of codewords sorted according to the correlation between the bits of the codeword. Viterbi algorithm [41] takes advantage of this construction by modeling the encoder as a finite state machine whose transitions between possible states are considered as a Markov chain and form a convolutional trellis, or state graph. Each path in this state graph corresponds to a codeword, and looking for the most likely codeword results in finding the path which minimizes the distance with the received vector. The complexity is linear in the informa-

tion length ($\mathcal{O}(K)$) [41].

An important breakthrough has been performed in 1993 by Berrou et al. [12] who invented the Turbo Codes, which have been the first codes to exhibit simulation results close to the channel capacity. This coding scheme uses two different component codes in parallel, originally being convolutional codes. The result of decoding of one code is fed as *a priori* to the other code in an iterative way. In the sequel, we explain how the decoding complexity is dramatically reduced in the specific case of LDPC codes.

1.2 Definition and parametrization of LDPC codes

LDPC codes are low density linear block codes, introduced by Gallager [6] in 1963, and soon after their non-binary counterparts by Davey [31]. A binary LDPC code is defined on the finite Galois field of order 2, $GF(2)$, while a non-binary LDPC code is defined on the Galois field of order q, $GF(q)$. We consider in this work only field characteristics which are power of two: $q = 2^p$. An LDPC code is represented by its sparse parity-check matrix $\mathbf{H}$ of size $(N - K) \times N$. As previously, the codeword length is denoted by N and the number of information symbols by K. The number of redundancy symbols is $M = N - K$, and the code rate is given by $R = K/N \geq 1 - M/N$, with equality if $\mathbf{H}$ is full-rank (i.e., its row rank is equal to M). The structure of the parity-check matrix can be regular or not. A code is regular (resp. irregular) if the number of non zero elements in every rows and in every columns of $\mathbf{H}$ is (resp. is not) constant. In the reminder of this section, *LDPC codes* is used when the distinction between binary and non-binary LDPC codes is not relevant. The field order in which the code lies will be specified otherwise.

Definition 1 *[6] A regular LDPC code with its two parameters (d_v, d_c) is defined by a matrix with exactly d_v and d_c ones per column and row, respectively.*

The code rate is $R = K/N \geq 1 - d_v/d_c$, with equality if $\mathbf{H}$ is full-rank. Those two parameters (d_v, d_c) define a *ensemble* of regular codes. A ensemble of LDPC codes defined by (d_v, d_c), is made of all the possible parity-check matrices with these connection parameters. One code among this ensemble is given by a particular realization of the parity-check matrix. In the non-binary case, the non-zero values of the parity-check matrices are chosen uniformly at random in $GF(q)\backslash\{0\}$.

In a similar way, an LDPC code can be represented by a bipartite graph, called factor graph [42], or Tanner graph [43], made of two kinds of nodes: variable nodes representing bits of a 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-check equation variable nodes participate. For binary LDPC, the non-zero values of the parity-check matrix $\mathbf{H}$ belong to $GF(2)\backslash 0$, i.e., they can be equal only to 1. For non-binary LDPC codes, the non-zero values of the parity-check matrix $\mathbf{H}$ belong to $GF(q)\backslash 0$. The element of $\mathbf{H}$ on row i column j is denoted h_{ij}. The jth variable node and the ith check node are connected if $h_{ij} \neq 0$. For instance, if x_j denotes the variable node j symbol value, the ith

parity-check equation is fulfilled if

$$\sum_{j=0}^{N-1} h_{ij} x_j = 0 \tag{1.1}$$

where additions and multiplications are performed over $GF(q)$. The degree of connection of a variable 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 (1.1) sums up these notions.

$$H=\begin{bmatrix} h_{00} & h_{01} & h_{02} & 0 & 0 & 0 \\ 0 & h_{11} & 0 & h_{13} & h_{14} & 0 \\ h_{20} & 0 & 0 & h_{23} & 0 & h_{25} \\ 0 & 0 & h_{32} & 0 & h_{34} & h_{35} \end{bmatrix}$$

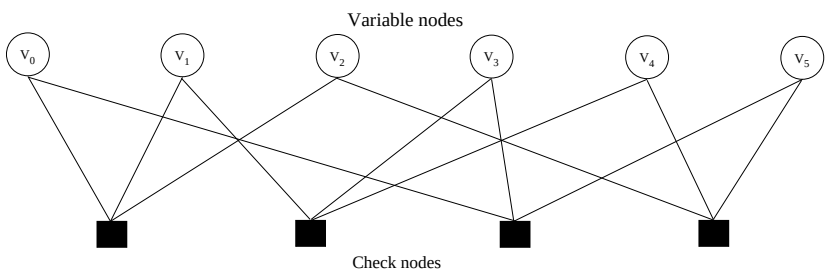


Figure 1.1 : Parity-check matrix of a non-binary LDPC code and its bipartite graph.

A code is irregular if it is not regular. The usual parametrization of irregular LDPC codes is done by means of polynomials [10], sometimes referred to as edgewise parametrization:

- Polynomial associated to variable nodes:

$$\lambda(x) = \sum_{i=2}^{d_{v_{max}}} \lambda_i x^{i-1}$$

  where λ_i is the proportion of edges of the graph connected to degree i variable nodes, and $d_{v_{max}}$ is the maximum degree of a variable node.

- Polynomial associated to check nodes:

$$\rho(x) = \sum_{j=2}^{d_{c_{max}}} \rho_j x^{j-1}$$

  where ρ_j is the proportion of edges of the graph connected to degree j check nodes, and $d_{c_{max}}$ is the maximum degree of a check node.

When the parity-check matrix of the code, whose graph parameters are $\lambda(x)$ and $\rho(x)$, is full rank, then those two quantities are related to the code rate by:

$$R = 1 - \frac{\sum_{j=2}^{d_{c_{max}}} \rho_j / j}{\sum_{i=2}^{d_{v_{max}}} \lambda_i / i} \tag{1.2}$$

There is also a dual parametrization of the previous one, referred to as nodewise parametrization [10]:

- Polynomial associated to data nodes:

$$\tilde{\lambda}(x) = \sum_{i=2}^{d_{v_{max}}} \tilde{\lambda}_i x^{i-1}$$

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

- Polynomial associated to check nodes:

$$\tilde{\rho}(x) = \sum_{j=2}^{d_{c_{max}}} \tilde{\rho}_j x^{j-1}$$

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

The transitions from one parametrization to another are given by:

$$\begin{aligned} \tilde{\lambda}_i &= \frac{\lambda_i / i}{\sum_k \lambda_k / k} \quad , \quad \tilde{\rho}_j = \frac{\rho_j / j}{\sum_k \rho_k / k} \\ \lambda_i &= \frac{i \tilde{\lambda}_i}{\sum_k k \tilde{\lambda}_k} \quad , \quad \rho_j = \frac{j \tilde{\rho}_j}{\sum_k k \tilde{\rho}_k} \end{aligned} \tag{1.3}$$

Thus, a ensemble of irregular LDPC codes is parametrized by $(N, \lambda(x), \rho(x))$. The regular case is a particular case of this parametrization where $\lambda(x)$ and $\rho(x)$ are monomials. Figure 1.2 is a graphical representation for this kind of code.

1.3 General notation

Throughout the thesis, vectors are denoted by boldface notations, e.g. $\mathbf{x}$. Random variables are denoted by upper-case letters, e.g. X and their instantiations in lower-case, e.g. x. The characterization and the optimization of non-binary LDPC codes are based on DE equations, assuming that the codes are decoded using iterative BP [31]. An important difference between non-binary and binary BP decoders is that the former uses multidimensional vectors as messages, rather than scalar values. There are two possible representations for the messages: plain-density probability vectors or Log-Density-Ratio

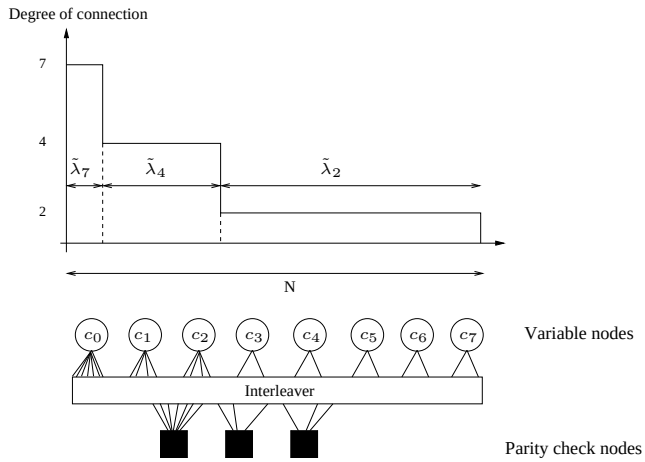


Figure 1.2 : Representation of a ensemble of irregular LDPC codes.

(LDR) vectors. We denote the q elements of the finite group $GF(q)$, or the finite field $GF(q)$, of order q by $(0, \alpha, \ldots, \alpha_{q-1})$. In the thesis, $P(X = x)$ denotes the probability that the random variable X takes the value x.

A q-dimensional probability vector is a vector $\mathbf{x} = (x_0, \ldots, x_{q-1})$ of real numbers such that $x_i = P(X = \alpha_i)$ for all i, and $\sum_{i=0}^{q-1} x_i = 1$.

Given a probability vector $\mathbf{x}$, the components of the Logarithmic Density Ratio (LDR) vector, associated with $\mathbf{x}$, are defined as

$$w_i = \log\left(\frac{x_0}{x_i}\right), \; i = 0, \ldots, q-1 \; . \tag{1.4}$$

Note that for all $\mathbf{x}$, $w_0 = 0$. We define the LDR-vector representation of $\mathbf{x}$ as the $q-1$ dimensional vector $\mathbf{w} = (w1, \ldots, w_{q-1})$. The observation of the channel under LDR form is a Logarithmic Likelihood Ratio (LLR). For convenience, in the derivation of the messages properties and the corresponding proofs reported in section 2.7, the value $w_0 = 0$ is not defined as belonging to $\mathbf{w}$. Given an LDR-vector $\mathbf{w}$, the components of the corresponding probability vector (the probability vector from which $\mathbf{w}$ was produced) can be obtained by

$$x_i = \frac{e^{-w_i}}{1 + \sum_{k=1}^{q-1} e^{-w_k}}, \; i = 0, \ldots, q-1 \tag{1.5}$$

A probability vector random variable is defined to be a q-dimensional random variable $\mathbf{X} = (X_0, ..., X_{q-1})$. An LDR-vector random variable is a $(q-1)$-dimensional random variable $\mathbf{W} = (W_1, ..., W_{q-1})$.

1.4 Decoding of LDPC codes by Belief Propagation algorithm

Depending on the transmission context (like channel type and computational power at the receiver), there are two kinds of decoding algorithms: hard decision algorithms and soft

decoding. The former will be studied in the last chapter, while the latter is the decoding algorithm that we use, unless the contrary is specified.

A priori probabilities on the value of each symbol of the codeword are first computed thanks to the channel outputs. For non-binary LDPC codes, these probabilities correspond to the probability that the symbol be equal to $\{\alpha_0, \ldots, \alpha_{q-1}\}$.

Although a maximum likelihood decoding of LDPC codes is possible [6], the complexity increases too much as soon as enough long binary codes are considered, and it is reasonable to expect that the complexity will not be lower for high order fields. That is why [6] then [43] proposed a sub-optimum decoding algorithm, finally revised by [44] and [42] for the case of factor graphs. This algorithm is known as Sum-Product [42] or BP [13] algorithm, and it spreads along edges messages forwarding probabilities or LDR. To each edge, two messages are associated, one for each direction. The principle of BP is Bayes rule applied locally and iteratively to estimate *a posteriori* probabilities (APP) of each codeword symbol. It has been shown that over a cycle-free graph (tree case), local factorization of Bayes rules leads to exact computation of APP of variable nodes because messages going into a node are independent from each other. However, in [18], it has been shown that the linear codes which have a cycle free Tanner graph have either a minimum distance lower or equal to 2 when the code rate R is greater than one half, or a minimum distance upper-bounded by $\frac{2}{R}$ otherwise. It is therefore impossible to consider such codes because the minimum distance that cannot grow with the codeword length, which is a desirable property. Hence, any finite length LDPC code has a cycle Tanner graph, then messages going into a node are not independent. Thus, APP are not computed exactly, and the algorithm is not optimal anymore in the sense it does not correspond anymore to ML decoding. However, the BP decoding of LDPC code is based on the cycle-free assumption thanks to the property of the graph of the code, which is sparse by definition of this class of codes.

Decoding principles apply similarly on $GF(q)$ codes, for $q > 2$, as for $GF(2)$ codes. This section describes only the non-binary case. Since non-binary codeword symbols are considered as random variables in $GF(q)$, messages on the edges of the graph are q sized vectors. BP algorithm intends to compute the APP of each codeword symbol. For instance, for the symbol corresponding to variable node v_i, the algorithm handles conditional probability vector $\mathbf{p}_i = (P(v_i = \alpha_0|y_i, S_i), \ldots, P(v_i = \alpha_{q-1}|y_i, S_i))$, where $P(v_i = \alpha_0|y_i, S_i)$ is the probability that the sent codeword symbol i is equal to α_0, given that the channel output for the ith symbol is y_i and given S_i the event that all parity-check equations connected to variable node v_i are fulfilled. The computation of $\mathbf{p}_i$ depends on the structure of the code factor graph through events S_i for all i. If input messages y_i are independent, the probabilities on the graph are computed exactly up to $\frac{g}{4}$ iterations, if g is the length of the shortest cycle in the graph, also called the *girth* of the graph.

To describe the BP decoding, $\{\mathbf{l}^{(t)}_{p_i v}\}_{i\in\{1,\ldots,d_v\}}$ denotes the set of messages getting in a degree d_v variable node v at the tth iteration, and $\{\mathbf{r}^{(t)}_{vp_i}\}_{i\in\{1,\ldots,d_v\}}$ the set of messages going out of this variable node. Index pv denotes the direction of message propagation (permutation node $\rightarrow$ variable node), vp denotes the opposite direction. Messages getting in (resp. out) a parity-check node c are similarly denoted by $\{\mathbf{r}^{(t)}_{p_i c}\}_{i\in\{1,\ldots,d_c\}}$ (resp.

$\{\mathbf{l}^{(t)}_{cp_i}\}_{i\in\{1,\ldots,d_c\}}$).

The decoding algorithm is composed of six stages:

- **Initialization:** All messages $\mathbf{r}^{(0)}$ going out of variable nodes to check nodes are initialized with *a priori* information computed at channel output $\{p_{ch_i}[a]\}_{i=\{0,\ldots,N-1\}}$, with

$$p_{ch_i}[a] = P(y_i|c_i = \alpha_a), \quad \alpha_a \in GF(q). \tag{1.6}$$

- **Variable node update:** The variable node v sends to check node c the probability for the symbol corresponding to v to be equal to $\alpha_a \in GF(q)$ (Fig.1.3). Messages going out of variable nodes are updated thanks to equation(1.7)

$$r^{(t+1)}_{vp_c}[a] = \mu_{vc} p_{ch_v}[a] \prod_{j=1, j\neq c}^{d_v} l^{(t)}_{p_j v}[a] \tag{1.7}$$

where $c \in \{1, \ldots, d_v\}$ and μ_{vc} is a normalization factor such that $\sum_{a=0}^{q-1} r^{P}_{vp_c}[a] = 1$.

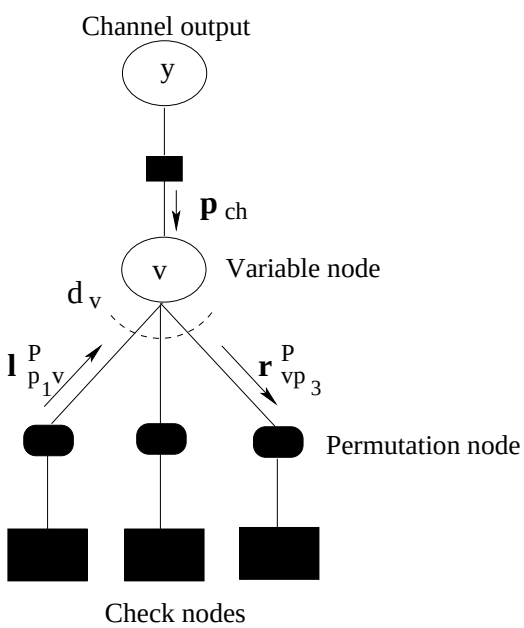


Figure 1.3 : Variable node update

- **Permutation nodes update:** This stage is a consequence of the parity equation (1.1). Indeed, the permutation function node on each edge corresponds to the multiplication of the non-zero value with the symbol value. Since these two values belong to $GF(q)$, this multiplication actually corresponds to a cyclic permutation of the vector messages.

$$r_{pc}[h_{ij} \times \alpha_k] = r_{vp}[\alpha_k] \quad k = \{0, \ldots, q-1\} \tag{1.8}$$

For message going from check nodes to variable nodes ($l_{cp} \rightarrow l_{pv}$), the inverse transform is achieved thanks to the inverse symbol h^{-1}_{ij} permutation.

- **Check nodes update:** Each check node processes its incoming vectors $\mathbf{r}^{(t)}$ and sends out updated messages $\mathbf{l}^{(t)}$ to all its neighbors (figure 1.4). The check node sends, to its neighboring variable nodes, the probability that the parity-check equation is fulfilled, given its incoming messages. Equation (1.10) is the update of the component a of the output vector $\mathbf{l}^{(t)}_{cp_v}$.

$$l^P_{cp_v}[a] = \sum_{\substack{\alpha_1,\ldots,\alpha_{v-1},\alpha_{v+1},\ldots,\alpha_{d_c}:\, j=1, j\neq v \\ \bigoplus_{i=1,i\neq v}^{d_c} \alpha_i = \alpha_a}} \prod^{d_c} r^{(t)}_{p_j c}[\alpha_j] \tag{1.9}$$

where the $\bigoplus$ operator explicits that the addition is performed over $GF(q)$. Elsewhere in the document, this operation is noted by common $+$, the addition is performed over $GF(q)$ if elements of $GF(q)$ are summed up. One can also express $\mathbf{l}^{(t)}_{cp_v}$ directly in terms of $\mathbf{r}^{(t+1)}_{vp_c}$

$$l^P_{cp_v}[a] = \sum_{\substack{\alpha_1,\ldots,\alpha_{v-1},\alpha_{v+1},\ldots,\alpha_{d_c},\, j=1, j\neq v \\ \bigoplus_{i=1,i\neq v}^{d_c} (g_i \times \alpha_i) = \alpha_a}} \prod^{d_c} r^{(t)}_{v_j p_j}[\alpha_j] \tag{1.10}$$

Figure 1.4 depicts equation(1.10): element $l^{(t)}_{cp3}[a]$ update consists in computing the sum of all products $r^{(t)}_{p_1 c}[a_1] \cdot r^{(t)}_{p_2 c}[a_2]$ satisfying the condition $a_1 \times a_2 \times a = 0$ with $a_1, a_2, a \in \{0, \ldots, q-1\}$.

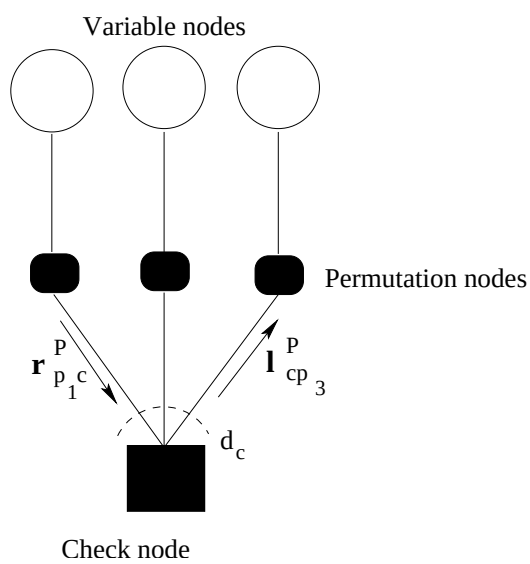


Figure 1.4 : Check node update

- **Stopping criterion:** Equation (1.11) corresponds to the decision rule on symbols values:

$$\hat{l}_n = \max_{\alpha_a} P_{ch}[a] \prod_{j=1}^{d_v} V^P_{p_j v}[a] \tag{1.11}$$

Updates of $\mathbf{r}^{(t)}$ and $\mathbf{l}^{(t)}$ messages is done iteratively until $\mathbf{H}\hat{\mathbf{l}} = \mathbf{0}$ (the decoder has converged to a codeword) or until the maximum number of iterations is reached (the decoder did not succeed in converging to a codeword).

Let us briefly mention available reduced-complexity techniques for decoding $GF(q)$ LDPC codes. The BP decoder implemented as above described has complexity $O(q^2)$. As introduced in an early paper of Richardson and Urbanke [11], the check node update can be performed using Fourier transforms translating the convolutional product, as soon as a group structure exists. Many works have then been done on this topic, like [45] that has shown that with Fourier transform decoding, the complexity scales as $O(q \log_2(q))$. Other low-complexity non-binary decoders, which have been presented recently in the literature [46, 47], implement approximated versions of the BP decoder.

1.5 Analysis of LDPC codes

This section first sets up the transmission context and explains why the error probability of non-binary LDPC codes can be assumed to be independent of the codeword sent. Then it is shown how the performance of $GF(q)$ LDPC code ensembles can be predicted by analyzing the densities of messages along iterative decoding. As this non-binary density evolution analysis is computationally intensive, only an approximation of message densities is given. By using a Gaussian approximation, one can design good irregularities for $GF(q)$ LDPC codes thanks to EXIT charts. Finally, the stability condition for non-binary LDPC codes is given, which ensures that the error probability is able to be arbitrary small, provided it has already dropped below some threshold. All the results presented in this section can be found in the literature [48], but note that they can be slightly modified because, unlike in [48], the considered channels are symmetric.

1.5.1 Additional notation

We give the definition of $+g$ operation, as introduced in [48]. Given a probability vector $\mathbf{x}$ and an element $g \in GF(q)$, $\mathbf{x}^{+g}$ is defined by

$$\mathbf{x}^{+g} = (x_g, x_{1+g}, \ldots, x_{(q-1)+g})$$

where addition is performed over $GF(q)$.
$\mathbf{x}^{\star}$ is defined as the set

$$\mathbf{x}^{\star} = \mathbf{x}, \mathbf{x}^1, \ldots, \mathbf{x}^{(q-1)}$$

Moreover, $n(\mathbf{x})$ is defined as the number of elements $g \in GF(q)$ satisfying $\mathbf{x}^{+g} = \mathbf{x}$.
Similarly, $\mathbf{x}^{\times g}$ is defined by [48]:

$$\mathbf{x}^{\times g} = (x_0, x_g, \ldots, x_{(q-1)\times g})$$

where multiplication $\times$ is performed over $GF(q)$.
The LDR vectors corresponding to $\mathbf{x}$ and $\mathbf{x}^{+g}$ are denoted by $\mathbf{w}$ and $\mathbf{w}^{+g}$, respectively.

Due to Definition 1.4 of the components of a LDR vector, the ith component of $\mathbf{w}^{+g}$ is w_i^{+g} which is defined by

$$w_i^{+g} = w_{g+i} - w_g, \quad \forall i = 0 \dots q-1$$

Unlike $\mathbf{w}^{+g}$, $\mathbf{w}^{\times g}$ is defined in the same way as $\mathbf{x}^{\times g}$:

$$w_i^{\times g} = w_{g \times i}, \quad \forall i = 0 \dots q-1$$

1.5.2 Channel and message symmetry

Only symmetric channels are considered in this work. Extension to arbitrary channel can be done by a coset approach, as detailed in [48]. In this section, we introduce classical results leading to asymptotic analysis, but we prove them in the specific case of the definition of channel symmetry we consider. These proofs are new, since the thorough study presented by Bennatan and Burshtein in [48] is done in the case of a coset approach.
The definitions of symmetric probability vector and LDR vector are given hereafter.

Definition 2 *[48] A probability vector random variable $\mathbf{Y}$ is symmetric if for any probability vector $\mathbf{y}$, the following expression holds:*

$$P(\mathbf{Y} = \mathbf{y} | \mathbf{Y} \in \mathbf{y}^\star) = y_0 \cdot n(\mathbf{y}) \tag{1.12}$$

where $\mathbf{y}^\star$ and $n(\mathbf{y})$ are as defined in Section 1.5.1.

Lemma 1 *[48] Let $\mathbf{W}$ be an LDR vector random variable. The random variable $\mathbf{Y} = LDR^{-1}(\mathbf{W})$ is symmetric if and only if $\mathbf{W}$ satisfies*

$$P(\mathbf{W} = \mathbf{w}) = e^{w_i} P(\mathbf{W} = \mathbf{w}^{+i}) \tag{1.13}$$

for all LDR vectors $\mathbf{w}$.

We refer the reader to the original article [48] for the proof of the equivalence between these two definitions.

The definition of channel symmetry we consider is the one of Li et al. [49].

Definition 3 *A channel is symmetric if and only if the density of the observation in probability form satisfies:*

$$P(\mathbf{Y} = \mathbf{y} | x = i) = P(\mathbf{Y} = \mathbf{y}^{+i} | x = 0)$$

Let us now prove that the channel symmetry implies that the error probability at any iteration of BP decoding of a $GF(q)$ code, is independent of the codeword that has been sent.

Lemma 2 *Let $P_e^{(t)}(\mathbf{x})$ denote the conditional error probability after the t-th BP decoding iteration of a $GF(q)$ LDPC code, assuming that codeword $\mathbf{x}$ was sent. If the channel is symmetric, then $P_e^{(t)}(\mathbf{x})$ is independent of $\mathbf{x}$.*

The proof of this lemma is provided in Section 1.7. This property allows to assume that the all-zero codeword has been transmitted, for the reminder of the asymptotic analysis of $GF(q)$ code ensemble performance.
Let us provide two additional properties that are usual for asymptotic analysis of LDPC codes,

Lemma 3 *If the channel is symmetric, then, under the all-zero codeword assumption, the initial message density P_0 in LDR form is symmetric:*

$$P_0(\mathbf{w}) = e^{w_i} P_0(\mathbf{w}^{+i})$$

The proof of this lemma is provided in Section 1.7. Furthermore, the following lemma is used in [48], and the proof is a direct extension of the proof of Lemma 1 in [11].

Lemma 4 *If the bipartite graph is cycle-free, then, under the all-zero codeword assumption, all the messages on the graph at any iteration, are symmetric.*

1.5.3 Density evolution for $GF(q)$ LDPC codes

This subsection presents density evolution for $GF(q)$ LDPC codes. The precise computation of the $GF(q)$ LDPC version of the algorithm is generally not possible in practice. The algorithm is however valuable as a reference for analysis purposes. The density evolution for $GF(q)$ LDPC codes is defined in Section 1.5.3, and the application of the concentration theorem of [11] is then given.

Since the density evolution analysis for non-binary LDPC codes is an extension of the binary case, we refer the reader to [11] and [10] for a complete and rigorous development of the density evolution for binary LDPC codes.

In [11] and [10], a general method that allows to predict asymptotic performance of binary LDPC codes is presented. The authors proved a so-called concentration theorem according to which decoding performance over any random graph converges, as the code length tends to infinity, to the performance when the graph is cycle-free. Thus, relevant evaluation of performance of binary LDPC codes is possible in the limit case of infinite codeword lengths. The proposed density-evolution method consists in following the evolution of probability densities of messages, spreading over the whole graph, when using belief propagation algorithm for decoding. Messages are assumed to be independent and identically distributed (iid).

Analogously to the binary case, density evolution for $GF(q)$ LDPC codes tracks the distributions of messages produced in belief-propagation, averaged over all possible neighborhood graphs on which they are based. The random space is comprised of random channel transitions, the random selection of the code from a (λ, ρ) $GF(q)$ LDPC ensemble (see section 1.2) and the random selection of an edge from the graph. The random space does not include the transmitted codeword, which is assumed to be fixed at the all-zero codeword (following the discussion of section 1.5.2). We denote by $\mathbf{R}^{(0)}$ any initial message across an edge, by $\mathbf{R}_t$ a variable to check message at iteration t, and by $\mathbf{L}_t$ a check to variable message at iteration t. The neighborhood graph associated with $\mathbf{R}_t$

and $\mathbf{L}_t$ is always assumed to be tree-like, and the case that it is not so is neglected. These notations are used when discussing plain-likelihood representation of density-evolution. When using LDR-vector representation, we let $\mathbf{R}'^{(0)}$, $\mathbf{R}'_t$ and $\mathbf{L}'_t$ denote the LLR-vector representations of $\mathbf{R}^{(0)}$, $\mathbf{R}_t$ and $\mathbf{L}_t$. To simplify the notations, it is assumed that all random variables are discrete and thus track their probability-functions rather than their densities. The following discussion focuses on the plain-likelihood representation. The translation to LDR representation is straightforward.

- **The initial message**. The probability function of $\mathbf{R}^{(0)}$ is computed in the following manner:
$$P(\mathbf{R}^{(0)} = \mathbf{x}) = \sum_{y \in \mathcal{Y}: \mathbf{r}^{(0)}(y) = \mathbf{x}} P(Y = y)$$
where Y is a random variable denoting the channel output, $\mathcal{Y}$ is the channel output alphabet and the components of $\mathbf{r}^{(0)}(y)$ are defined by equation (1.6), replacing y_i with y.

- **Check to variable node messages**. $\mathbf{L}_t$ is obtained from equation (1.10). The variable-to-check messages in equation (1.10) are replaced by independent random variables, distributed as $\mathbf{R}_{t-1}$. Similarly, the labels in equation (1.10) are also replaced by independent random variables uniformly distributed in $GF(q)\backslash\{0\}$. Formally, let d_c be the maximal check node degree. Then for each $d_j = 2, \ldots, d_c$ we first define,
$$P(\mathbf{L}_t^{(d_j)} = \mathbf{x}) = \sum_{\substack{\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(d_j-1)} \in \mathcal{P}, \\ g_1,\ldots,g_{d_j} \in GF(q): \\ \mathbf{l}(\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(d_j-1)},g_1,\ldots,g_{d_j}) = \mathbf{x}}} \prod_{n=1}^{d_j} P(G_n = g_n) \cdot \prod_{n=1}^{d_j-1} P(\mathbf{R}_{t-1} = \mathbf{r}^{(n)}) \tag{1.14}$$
where $\mathcal{P}$ is the set of all probability vectors, and the components of $\mathbf{l}(\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(d_j-1)} g_1, \ldots, g_{d_j})$ are defined as in equation (1.10). G_n is a random variable corresponding to the nth label, and thus $P(G_n = g) = \frac{1}{q-1}$ for all g. $P(\mathbf{R}_{t-1} = \mathbf{r}^{(n)})$ is obtained recursively from the previous iteration of belief propagation. The probability function of $\mathbf{L}_t$ is now obtained by
$$P(\mathbf{L}_t = \mathbf{x}) = \sum_{j=1}^{c} \rho_j P(\mathbf{L}_t^{(d_j)} = \mathbf{x}) \tag{1.15}$$

- **Variable to check node messages**. The probability function of $\mathbf{R}_0$ is equal to that of $\mathbf{R}^{(0)}$. For $t > 0$, $\mathbf{R}_t$ is is obtained from equation (1.7). The check-to-variable messages and initial messages in equation (1.7) are replaced by independent random variables, distributed as $\mathbf{L}_t$ and $\mathbf{R}^{(0)}$ respectively. Formally, let d_v be the maximal variable node degree. Then for each $d_i = 2, \ldots, d_v$ we first define,
$$P(\mathbf{R}^{(d_i)^{(t)}} = \mathbf{x}) = \sum_{\substack{\mathbf{r}^{(0)},\mathbf{l}^{(1)},\ldots,\mathbf{l}^{(d_i-1)} \in \mathcal{P}: \\ \mathbf{r}(\mathbf{r}^{(0)},\mathbf{l}^{(1)},\ldots,\mathbf{l}^{(d_i-1)}) = \mathbf{x}}} P(\mathbf{R}^{(0)} = \mathbf{r}^{(0)}) \prod_{n=1}^{d_i-1} P(\mathbf{L}_t = \mathbf{l}^{(n)})$$

where the components of $\mathbf{r}(\mathbf{r}^{(0)}, \mathbf{l}^{(1)}, ..., \mathbf{l}^{(d_i-1)})$ are defined as in equation (1.7). $P(\mathbf{R}^{(0)} = \mathbf{r}^{(0)})$ and $P(\mathbf{L}_t = \mathbf{l}^{(n)})$ are obtained recursively from the previous iterations of belief propagation. The probability function of $\mathbf{R}_t$ is now obtained by

$$P(\mathbf{R}_t = \mathbf{x}) = \sum_{i=1}^{v} \lambda_i P(\mathbf{R}_t^{(d_i)} = \mathbf{x}) \tag{1.16}$$

Theoretically, the above algorithm is sufficient to compute the desired densities. If one consider that the all-zero codeword has been sent, which is relevant in the context presented in section 1.5.2, it is easy to compute the probability of doing an error when applying decision rule (1.11) to a probability vector, e.g. $\mathbf{R}_t$.

As aforementioned, Richardson and Urbanke in [11] proved a concentration theorem that states that as the block length N approaches infinity, the bit error rate at iteration t converges to a similarly defined probability of error. It has been shown in [48] that the concentration theorem can be applied to frame error rate of $GF(q)$ LDPC code ensembles. In this way, the performance of correction of a $GF(q)$ LDPC ensemble, as defined in Section 1.2, can be exactly predicted. However, in practice, a major problem is the fact that the quantities of memory required to store the probability density of a q-dimensional message grows exponentially with q. That is why it is important to look for a computationally easier way to follow the message densities in order to be able to predict the code ensemble performance.

As mentioned in [48], if $P_e^t = P_e(\mathbf{R}_t)$ is a sequence of error probabilities produced by density evolution, then P_e^t is a non-increasing function of t. The demonstration is similar to the proof of theorem 7 in [10]. This non-increasing property ensures that the sequence corresponding to density evolution by iterating between equation (1.15) and equation (1.16) converges to a fixed point. Implementing the density evolution allows to check whether not this fixed point corresponds to the zero error probability, which means that the decoding in the infinite codeword length case has been successful. That is why $GF(q)$ LDPC codes, like binary LDPC codes, are said to have a *threshold behavior*.

In the sequel, it is explained why such an implementation is not possible for $GF(q)$ LDPC codes, unlike their binary counterparts. The proposed method from the literature aims at approximating the densities, thereby simplifying the recursion and making possible the study of its convergence. This method is presented in the next section.

1.5.4 Approximation of message densities by only one scalar parameter

Analogously to the binary case, a Gaussian approximation of the message densities is used to be able to practically track these densities and predict error probabilities of $GF(q)$ LDPC code ensembles. To reduce the densities to only one scalar parameter, things are a little more elaborated than in the binary case since messages are no more scalars but q-sized probability vectors, which entails that the densities are multi-variate densities.

Permutation-invariance

Permutation-invariance is a key property of $GF(q)$ LDPC codes that allows the approximation of their densities using one-dimensional functionals, thus greatly simplifying their analysis. It is only briefly described here, since more details can be found in [48]. The definition is based on the cyclic permutation of the elements of a probability vector message, when passing through the permutation nodes described in Section 1.4.

Definition 4 *[48] A probability vector random variable $\mathbf{X}$ is said to be permutation-invariant if for any fixed $g \in GF(q)\backslash\{0\}$, the random variable $\mathbf{X}^{\times g}$ is distributed identically with $\mathbf{X}$.*

This definition also holds for LDR vectors. It is shown in [48] that a message resulting from a random permutation is necessarily permutation-invariant. That is $\mathbf{X}^{\times g}$ is necessarily permutation-invariant when $\mathbf{X}$ is a random LDR or a probability vector and g is picked up uniformly at random in $GF(q)\backslash\{0\}$. Hence, this is the case for all messages on the graph of a given $GF(q)$ LDPC code ensemble, whose non-zero values are chosen uniformly in $GF(q)\backslash\{0\}$, except initial messages $\mathbf{R}^{(0)}$ and messages going out of variable nodes. Moreover, all the components of a permutation-invariant vector are identically distributed (lemma 8 in [48]). Combined with the symmetry and the Gaussian approximation, it allows the projection of message densities of $GF(q)$ LDPC code ensembles on only one parameter.

Gaussian approximation

For binary LDPC codes, Chung et al. [50] observed that the variable-to-check message densities well approximated by Gaussian random variables. Furthermore, the symmetry of Gaussian messages in binary LDPC decoding implies that the mean m and variance σ^2 of the random variable are related by $\sigma^2 = 2m$. Thus, the distribution of a symmetric Gaussian random variable may be described by a single parameter: m. This property was also observed by ten Brink et al. [14] and is essential to their development of EXIT charts. In the context of non-binary LDPC codes, Li et al. [49] obtained a description of the $q-1$ dimensional messages, under a Gaussian assumption, by $q-1$ parameters.

The following theorem explains how the mean vector and the covariance matrix of a symmetric LDR vector can be related to each other:

Theorem 1 *[48] Let $\mathbf{W}$ be an LDR-vector random variable, Gaussian distributed with a mean $\mathbf{m}$ and covariance matrix $\mathbf{\Sigma}$. If $\mathbf{\Sigma}$ is non-singular and $\mathbf{W}$ is symmetric, then*

$$\Sigma_{i,j} = m_i + m_j - m_{i\oplus j}, \quad \forall (i,j) \in [1, q-1]^2$$

If the LDR vector $\mathbf{W}$ distributed as $\mathcal{N}(\mathbf{m}, \mathbf{\Sigma})$ is additionally permutation-invariant, then all its components are identically distributed. Then the mean vector can be expressed as $m \cdot \mathbf{1}_{q-1}$ where $\mathbf{1}_{q-1}$ is the all one vector of size $q-1$. A Gaussian distributed, symmetric and permutation-invariant random variable is thus completely described by a single scalar parameter m.

EXIT charts for $GF(q)$ LDPC codes

Let us consider the binary input AWGN channel. This paragraph presents the tool for optimization of the irregularity of $GF(q)$ LDPC code ensemble thanks to EXIT charts.

First, let us discuss the accuracy of the Gaussian approximation of the channel output in symbolwise LLR form for $GF(q)$ LDPC code ensembles. The channel outputs are noisy observations of bits, from which we obtain bitwise LLR, all identically distributed as $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ [50]. Let $\mathbf{s}$ be the vector gathering the LLRs $b_1, \ldots, b_{p_k}$ of bits of which a symbol in $G(q_k)$ is made: $\mathbf{s} = (b_1, \ldots, b_{p_k})^T$. Each component of an input LLR random vector $\mathbf{l}$ of size $(q_k - 1)$ is then a linear combination of these bitwise LLRs:

$$\mathbf{l} = B_{q_k} \cdot \mathbf{s} \tag{1.17}$$

where B_{q_k} is the matrix of size $q_k \times \log_2(q_k)$ in which the ith row is the binary map of the ith element of $G(q_k)$. The distribution of initial messages is hence a mixture of one-dimensional Gaussian curves, but are not Gaussian distributed vectors. Indeed, it is easy to see that the covariance matrix of vector $\mathbf{l}$ is not invertible.

Formally, EXIT charts track the mutual information $I(C; \mathbf{W})$ between the transmitted code symbol C at a variable node and the message $\mathbf{W}$ transmitted across an edge emanating from it.

Definition 5 *[48] The mutual information between a symmetric LDR-vector message* $\mathbf{W}$ *of size* $q-1$ *and the codeword sent, under the all-zero codeword assumption, is defined by:*

$$I(C; \mathbf{W}) = 1 - \mathbb{E} \log_q \left(1 + \sum_{i=1}^{q-1} e^{-W_i} | C = 0 \right)$$

The equivalent definition for the probability vector $\mathbf{X} = LDR^{-1}(W)$ of size q is

$$I(C; \mathbf{X}) = 1 - \mathbb{E} \log_q \left(\frac{\sum_{i=0}^{q-1} X_i}{X_0} | C = 0 \right) . \tag{1.18}$$

In the following, the shortcut "mutual information of a LDR vector" is used instead of "mutual information between a LDR vector and the codeword sent". If this information is zero, then the message is independent of the transmitted code symbol and thus the probability of error is $\frac{q-1}{q}$. As the information approaches 1, the probability of error approaches zero. Note that we assume that the base of the log function in the mutual information is q, so as $0 \leq I(C; \mathbf{W}) \leq 1$. $I(C; \mathbf{W})$ is taken to represent the distribution of the message $\mathbf{W}$. That is, unlike density evolution, where the entire distribution of the message $\mathbf{W}$ at each iteration is recorded, with EXIT charts, $I(C; \mathbf{W})$ is assumed to be a faithful surrogate. In other words, since the densities are assumed to be dependent on only one scalar parameter, instead of tracking the mean of one component, one tracks the information content of the message. It is shown in [48] that, under the cycle free graph assumption:

$$I(C; \mathbf{W}) = 1 - \mathbb{E}_{\mathbf{W}} \left(\log_q (1 + \sum_{i=1}^{q-1} e^{-w_i}) | C = 0 \right)$$

The evolution of the mutual information of messages through the different steps of decoding is now given. We use dummy notations x_{in} and x_{out} for the mutual information evolution equations at each decoding step for ease of understanding. Then all steps are gathered into a single EXIT equation.

- Let $\mathbf{v}$ denote a probability vector, and $f(\mathbf{v})$ the corresponding Fourier Transform (FT) vector (see [49, 45] for use of multi-dimensional FFT on messages). The mutual information of the check node input is computed thanks to the following relation:

  $$x_{f(\mathbf{v})} = 1 - x_{\mathbf{v}}$$

  The demonstration of this relation is easy with direct calculus, provided in section 2.7.6.

  Thus, for the mutual information evolution through a check node with connection degree j, we have:

  $$x_{out} = 1 - J_c((j-1)J_c^{-1}(1 - x_{in}, q), q)$$

  with

  $$J_c(m,q) = 1 - \mathbb{E}_{\mathbf{v}}\left(\log_q(1 + \sum_{i=1}^{q-1} e^{-v_i})\right), \quad \text{with } \mathbf{v} \sim \mathcal{N}(m\mathbf{1}_{q-1}, \mathbf{\Sigma}) \tag{1.19}$$

- The mutual information of a variable node output is expressed thanks to the $J_v(\cdot,\cdot)$ function applied to σ^2 and to the sum of means, since symbol node update is the summation of LDRs. Here, x_{in} is the IC of truncation operator output. The IC x_{out} of the output of a symbol node with connection degree i, is given by:

  $$x_{out} = J_v(\sigma^2, (i-1)J_c^{-1}(x_{in}, q)\mathbf{1}_{q-1}, q)\ .$$

Finally, we get equation (1.20) that expresses the extrinsic transfer function of the non-binary BP decoder used on a BIAWGN channel from iteration number t to iteration number $t+1$. The information content of any check node incoming vector message at the $(t+1)^{th}$ iteration is denoted by $x_{vc}^{(t)}$.

The optimization method to find the best connectivity profile for a $GF(q)$ code is then the same as for binary LDPC codes.

$$x_{vc}^{(t+1)} = \sum_i \lambda_i J_v\left(\sigma^2, \mathbf{m}_{sc} + (i-1)J_c^{-1}\left(1 - \sum_j \rho_j J_c\left((j-1)J_c^{-1}(1 - x_{vc}^{(t)})\right)\right)\mathbf{1}_{q-1}\right) \tag{1.20}$$

1.5.5 The stability condition

Also obtained in [48], the stability condition, introduced in [10], is a necessary and sufficient condition for the error probability to converge to zero, provided it has already dropped below some value. This condition must be satisfied by the SNR corresponding to the threshold of the code ensemble. Therefore, ensuring this condition, when implementing an approximation of the exact density evolution, helps to have a more accurate approximation of the exact threshold.

Given a ensemble of $GF(q)$ LDPC codes defined by (λ, ρ), the following ensemble parameter is defined:

$$\Omega = \sum_j \rho_j (j-1) \tag{1.21}$$

For a given memoryless symmetric output channel with transition probabilities $p(y|x)$, the following channel parameter is also defined:

$$\Delta = \frac{1}{q-1} \sum_{i=1}^{q-1} \int \sqrt{p(y|i)p(y|0)}dy \tag{1.22}$$

Theorem 2 *[48] Consider a given $GF(q)$ LDPC ensemble parametrized by (λ, ρ). Let $P_e^t = P_e(\mathbf{R}_t)$ denotes the average error probability at iteration t under density evolution.*

- *If $\Omega \geq \frac{1}{\Delta}$, then there exists a positive constant $\xi = \xi(\lambda, \rho, P_0)$ such that $P_e^t > \xi$ for all iterations t.*
- *If $\Omega < \frac{1}{\Delta}$, then there exists a positive constant $\xi = \xi(\lambda, \rho, P_0)$ such that if $P_e^t < \xi$ at some iteration t, then P_e^t approaches zero as t approaches infinity.*

1.5.6 Design example of $GF(q)$ LDPC code ensemble on BIAWGN channel

Optimization is performed for the BIAWGN channel. The goal of the optimization with EXIT charts is to find a good ensemble of $GF(q)$ LDPC codes with the lowest convergence threshold, under a Gaussian approximation. This means that we look for the parameters $(\lambda(x), \rho(x))$ of the ensemble of $GF(q)$ LDPC codes with lowest convergence threshold.

Let us denote the code rate R, and the target code rate R_{target}. The optimization procedure [10, 50] consists in finding $(\lambda(x), \rho(x))$ which fulfills the following constraints at the lowest SNR:

Code rate constraint:	$R = R_{target}$ (see equation (1.2))
Proportion constraint:	$\sum_i \lambda_i = 1$ and $\sum_j \rho_j = 1$
Successful decoding condition:	$x_{vc}^{(t+1)} > x_{vc}^{(t)}$ (see equation (1.20))
Stability constraint:	$\Omega\Delta < 1$ (see equations (1.21) and (1.22))

	$d_c = 4$	$d_c = 5$	$d_c = 6$	$d_c = 7$	$d_c = 8$
$q = 4$	2.56	0.95	0.66	0.52	0.48
$q = 64$	0.76	0.53	0.51	0.58	0.90
$q = 256$	0.65	0.54	0.59	0.79	1.27

Table 1.1 : Thresholds of $GF(q)$ LDPC code ensembles with constant check degree d_c and code rate one half, optimized with EXIT charts on the BIAWGN channel. The maximum variable degree allowed in the optimization procedure is $d_{v_{max}} = 30$. Thresholds are given in term of the SNR $\frac{E_b}{N_0}$ in dB, and are obtained using the Gaussian approximation.

We briefly illustrate what can be the results of such an optimization, and how it allows to find again known results from the literature.

Table 1.1 gathers some thresholds obtained by optimization of the irregularities for various field order and check degrees. These thresholds are hence computed by EXIT charts, with a Gaussian approximation. The code rate is one-half. Since degree-1 variable nodes are not allowed in the optimization process, the code ensemble with $d_c = 4$ is regular with $d_v = 2$. In this case, we observe that the threshold is better for higher order field. This observation ca, be identified to the following claim of Hu and Eleftheriou in [33]. They considered $GF(q)$ random ensembles defined by the probability p that an element of the parity-check matrix be non-zero. When p is very low, the binary random ensemble defined by p is far away from the Shannon equiprobable random ensemble. In this case, they illustrated that the Hamming weight distribution of the $GF(q)$ random ensemble tends to the binomial distribution as q increases. As an additional example, EXIT curves of regular (2,4) codes in $GF(2)$, $GF(8)$ and $GF(256)$ are plotted on figure 1.5, confirming results of the first column of Table 1.1: the curve of $GF(256)$ is the only one for which the tunnel is open.

1.6 Other design techniques

1.6.1 Finite length design of LDPC codes

We do not detail the design techniques relative to finite length design of LDPC codes, but just mention some works on that. First, the PEG construction has been proposed in [23] to build the graph of codes, given the irregularities. This technique has recently been improved [51]. For non-binary LDPC codes, additionally to the PEG construction, Poulliat et al. [34] expressed a criterion and developed a technique for cancelling cycles of $GF(q)$ LDPC codes by an appropriate choice of the non-zero values.

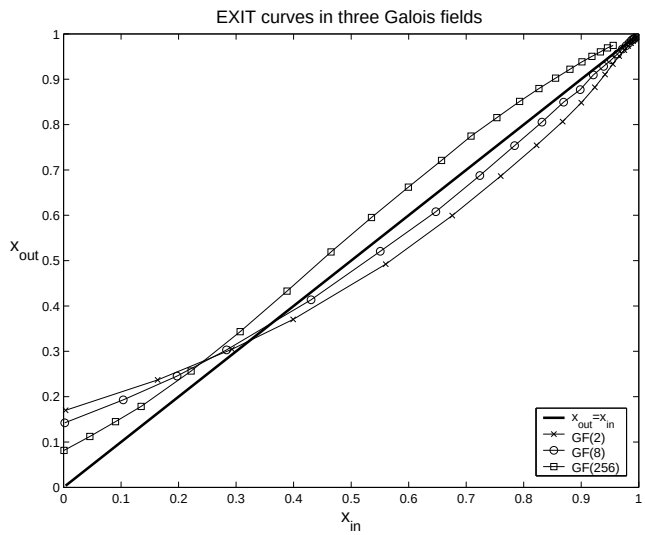


Figure 1.5 : EXIT curves of $(2,4)$ $GF(2)$, $GF(8)$ and $GF(256)$ regular codes. The SNR is 0.7dB.

1.6.2 Structured ensembles

As pointed out in the introduction, a very efficient way to design code ensembles with iterative decoding performance close to capacity and low error-floor, is to choose the permutations in a structured way. Indeed, the aforementioned representation of LDPC codes defines only the connection degrees of variable and check nodes, but any variable node can be connected to any check node. A structured code ensemble has a representation which defines to which type of check nodes each type of variable node can be connected. LDPC codes with a detailed representation have been introduced in [52]. Some structured code ensembles have been under the scope of many studies these last years: irregular repeat-accumulate (IRA) codes [25], protograph-based LDPC codes [26] and multi-edge type LDPC [27]. The design of good D-GLDPC codes have been addressed for the BEC in [29]. These techniques lead to codes with good code properties in terms, for instance, of girth of the bipartite graph and possibility to perform the encoding procedure efficiently. For a comprehensive survey of the design of those kinds of LDPC codes, we refer the reader to [53].

1.7 Proof of theorems in Chapter 1

Lemma 3.*Let $P_e^{(l)}(\mathbf{x})$ denote the conditional error probability after the l-th BP decoding iteration of a $GF(q)$ LDPC code, assuming that codeword $\mathbf{x}$ was sent. If the channel is symmetric, then $P_e^{(l)}(\mathbf{x})$ is independent of $\mathbf{x}$.*

Proof: The proof has the same structure as the proof of Lemma 1 in [11]. Thus, we do not detail it, but instead refer the reader to [11] and rather only give the key elements.

The notations are the same as in [11].

- Check node symmetry: For any sequence $(b_1, \ldots, b_{d_c-1})$ in $GF(q)$, we have

$$\Psi_c^{(l)}(\mathbf{m}_1^{+b_1}, \ldots, \mathbf{m}_{d_c-1}^{+b_{d_c-1}}) = \Psi_c^{(l)}(\mathbf{m}_1, \ldots, \mathbf{m}_{d_c-1})^{+b_1+\cdots+b_{d_c-1}}$$

- Variable node symmetry: We also have, for any $b \in GF(q)$:

$$\Psi_v^{(l)}(\mathbf{m}_0^{+b}, \mathbf{m}_1^{+b}, \ldots, \mathbf{m}_{d_v-1}^{+b}) = \Psi_v^{(l)}(\mathbf{m}_1, \ldots, \mathbf{m}_{d_c-1})^{+b}$$

With same notation as in [11], we define $\mathbf{y} = \mathbf{z}^{+\mathbf{x}}$, where $\mathbf{x}$ is a vector of size q, denoting an arbitrary codeword over $GF(q)$. $\mathbf{y}$ and $\mathbf{z}$ are sets of vectors, and each element $\mathbf{y}_t$ corresponds to $\mathbf{y}_t = \mathbf{z}_t^{+x_t}$.
Still with same notations as in [11], we easily prove that:

$$\mathbf{m}_{ij}^{(0)}(\mathbf{y}) = \mathbf{m}_{ij}^{(0)}(\mathbf{z})^{+x_i};.$$

We also prove that, since $\mathbf{x}$ is a codeword, then $\sum_{k:\exists e=(v_k,c_j)} x_k = 0$. Hence, as in [11], we conclude that

$$\mathbf{m}_{ji}^{(l+1)}(\mathbf{y}) = \mathbf{m}_{ji}^{(l+1)}(\mathbf{z})^{+x_i}$$

thanks to the check node symmetry, and

$$\mathbf{m}_{ij}^{(l+1)}(\mathbf{y}) = \mathbf{m}_{ij}^{(l+1)}(\mathbf{z})^{+x_i}$$

thanks to the variable node symmetry.

□

Lemma 3.*If the channel is symmetric, then, under the all-zero codeword assumption, the initial message density P_0 in LDR form is symmetric:*

$$P_0(\mathbf{W} = \mathbf{w}) = e^{w_i} P_0(\mathbf{W} = \mathbf{w}^{+i})$$

Proof: Let us define $\mathbf{y}$ by $\mathbf{y} = LDR^{-1}(\mathbf{w})$. If we call x_{noisy} the noisy observation of the sent symbol value, by following the notation of [10], we have $\mathbf{w} = L(x_{\text{noisy}})$. Hence, the ith component of $\mathbf{y}$ is $y_i = P(x_{\text{noisy}} \in L^{-1}(\mathbf{w})|x = i)$, and $w_i = \log\left(\frac{y_0}{y_i}\right) = \log\left(\frac{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w})|x=0)}{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w})|x=i)}\right)$ also.
Given the symmetry of the channel, let us prove that $P_0(\mathbf{W} = \mathbf{w})$ satisfies equation

(1.13):

$$
\begin{aligned}
e^{w_i} P_0(\mathbf{W} = \mathbf{w}^{+i}) &= e^{w_i} P(\mathbf{W} = \mathbf{w}^{+i} | x = 0) \\
&= \frac{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = 0)}{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = i)} P(\mathbf{Y} = \mathbf{y}^{+i} | x = 0) \\
&= \frac{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = 0)}{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = i)} P(\mathbf{Y} = \mathbf{y} | x = i) \\
&= \frac{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = 0)}{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = i)} P(\mathbf{W} = \mathbf{w} | x = i) \\
&= \frac{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = 0)}{P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = i)} P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = i) \\
&= P(x_{\text{noisy}} \in L^{-1}(\mathbf{w}) | x = 0) \\
&= P(\mathbf{W} = \mathbf{w} | x = 0) \\
&= P_0(\mathbf{W} = \mathbf{w})
\end{aligned}
$$

□

Chapter 2

Hybrid LDPC Codes

In this chapter, we introduce and study a new class of LDPC codes, named *hybrid LDPC codes*. The class of hybrid LDPC codes is a generalization of existing classes of LDPC codes, like non-binary or GLDPC codes. For hybrid LDPC codes, we allow the connectivity profile of the factor graph to be irregular, but also we allow the codeword symbols to be defined over different order sets. By adapting the work of [48], we show in particular that the class of hybrid LDPC codes can be asymptotically characterized and optimized using density evolution (DE) framework. All the proofs are gathered at the end of the chapter.

2.1 The class of hybrid LDPC codes

2.1.1 General hybrid parity-check equations

Classically, non-binary LDPC codes are described thanks to the local constraints given by parity-check equations involving some of the codeword symbols c_i. If a code is linear over a finite field $GF(q)$, the parity equation corresponding to the ith row of the parity-check matrix $\mathbf{H}$, is

$$\sum_j h_{ij} c_j = 0 \quad \text{in } GF(q) \tag{2.1}$$

The field $GF(2^p)$ can be represented using the vector space $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^p$ in a natural way. Multiplications in $GF(2^p)$ can be represented as matrix multiplications, after choosing a suitable representation. The set of matrices representing field elements then forms a field of invertible matrices. Thus, interpreting variables as elements of $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^p$ and using matrix multiplication to form linear constraints can be used to model LDPC over $GF(2^p)$.

We aim at generalizing the definition of the parity-check equation by allowing more general operations than multiplications by $h_{ij} \in GF(q)$, and moreover, by considering parity-checks where codeword symbols can belong to different finite sets: $c_k \in G(q_1)$. $G(q_1)$ is a finite set of order $q_1 = 2^{p_1}$ with a group structure. Indeed, we will only consider groups of the type $G(q_1) = \left(\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^{p_1}, +\right)$ with $p_1 = \log_2(q_1)$. Such a group corresponds

to an ensemble of p_1-sized vectors whose elements lie in $\frac{\mathbb{Z}}{2\mathbb{Z}}$. This is the reason why we adopt the fully denomination of these codes as being *multi-binary hybrid LDPC codes*. In the remainder, we use a shortcut and refer to them as *hybrid LDPC codes*.

Let q_1 and q_2, such that $q_1 < q_2$, denote the group orders of a column and of a row of $\mathbf{H}$, respectively. They will be similarly called variable and check order. Let $G(q_1)$ denote the group of variable j and $G(q_2)$ the group of parity-check i. The non-zero elements of the parity-check matrix are applications which have to map a value in the column group (variable node group), to a value in the row group (check node group, see figure 2.1). This is achieved thanks to functions named h_{ij} such that

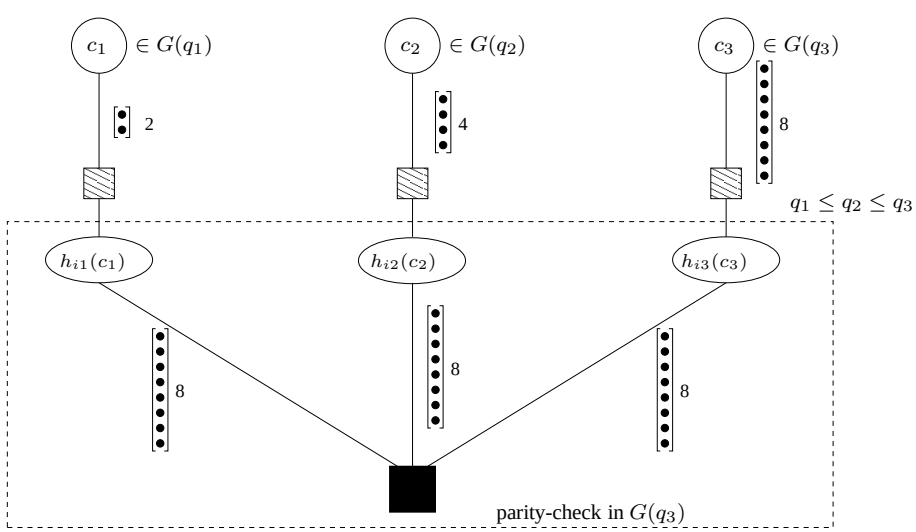
$$\begin{array}{rcl} h_{ij} : G(q_1) & \rightarrow & G(q_2) \\ c_j & \rightarrow & h_{ij}(c_j) \end{array}$$

Hence, an hybrid parity-check equation is given by

$$\sum_j h_{ij}(c_j) = 0 \quad \text{in } G(q_2) \tag{2.2}$$

We notice that, on equation (2.1) as well as on equation (2.2), the additive group structure defines the local constraints of the code. Moreover, as mentioned in [11], and deeply studied in, e.g., [45], the additive group structure possesses a Fourier transform, whose importance for the decoding is pointed out in section 2.1.7.

Since the mapping functions h_{ij} can be of any type, the class of hybrid LDPC codes is very general and includes classical non-binary and binary codes.



$$h_{i1}(c_1) + h_{i2}(c_2) + h_{i3}(c_3) = 0, \qquad h_{ij}(c_j) \in G(q_3)$$

defines a component code in the group $G = G(q_1) \times G(q_2) \times G(q_3)$

Figure 2.1 : Factor graph of parity-check of an hybrid LDPC code.

2.1.2 Hybrid LDPC code ensemble

By definition of $G(q_k)$, to each element of $G(q_k)$ corresponds a binary map of p_k bits. Let us call the minimum order of codeword symbols q_{min}, and the maximum order of codeword symbols q_{max}. The class of hybrid LDPC codes is defined on the product group $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^{p_{min}} \times \ldots \times \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^{p_{max}}$. Let us notice that this type of LDPC codes built on product groups has already been proposed in the literature [54][55], but no optimization of the code structure has been proposed and its application was restricted to the mapping of the codeword symbols to different modulation orders.

2.1.3 Different sub-classes of hybrid LDPC codes

Among the huge set of hybrid LDPC codes, we can distinguish as many classes as different types of non-zero elements of the parity-check matrix $\mathbf{H}$. Such a non-zero element is an application, that we denote by A, which maps the q_1 symbols of $G(q_1)$ into a subset of q_1 symbols that belongs to $G(q_2)$. It can be of any type. Let us consider the case where these applications are linear, i.e., represented by a matrix, with dimensions $p_2 \times p_1$. In that way, A actually connects the binary map vector of a symbol in $G(q_1)$ to the binary map vector of a symbol in $G(q_2)$. At this stage, it is quite straightforward to establish a connection between hybrid LDPC codes and doubly-generalized LDPC (D-GLDPC) codes, thoroughly studied in [29, 56]. Indeed, the linear map A can be seen as part of the generalized check and generalized variable. The code corresponding to the generalized variable v would have a number of information bits $K = p_1$ and length $N = \sum_l p_l$, where the sum is done over the groups of all the checks connected to v. The code of the generalized check c would have a number of redundancy bits $M = p_2$ and length $N = \sum_k p_k$, where the sum is done over the groups of all the variables connected to c. However, it is important to note that, if the idea is the same, hybrid LDPC codes are not exactly D-GLDPC codes because of the decoder. Indeed, with D-GLDPC codes, one considers that the generalized codes are at variable and check nodes sides, whereas with hybrid LDPC, we consider that the previous generalized codes for each node are split on each incoming edge. As detailed in section 2.3 on optimization, this difference allows us to affect different connection degrees on the nodes depending on their group order, i.e., depending on K for variables and on M for checks. In other words, we will be able to optimize the length of the codes, given the dimension. We distinguish different sub-classes of hybrid LDPC codes whose non-zero elements are linear maps:

(i) Applications that are not of rank p_1. This includes the case where the group order of a column can be higher than the group order of the row. From a D-GLDPC perspective, this allows to have generalized variables whose codes have $K > N$, that is to say the number of incoming bits is projected to a smaller one. This could be thought as puncturing, and, as a consequence, we get back the result that the rate of the graph can be lower than the actual code rate. This case is out of the scope of this thesis.

(ii) Applications that are of rank p_1. They are named full-rank applications, and corre-

spond to matrices of size $p_2 \times p_1$ with necessarily $p_1 \leq p_2$. Such an application is depicted on figure 2.2. We consider only these types of hybrid LDPC codes in this work, and details are given in the following section. This would correspond to a classical D-GLDPC code, where the rate of the graph is higher than the actual code rate, that is all generalized variables have necessarily codes with $N > K$. Indeed, no puncturing is done on bits.

2.1.4 Hybrid LDPC codes with linear maps

In this work, we consider only hybrid LDPC codes with the features described above, and whose non-zero elements are linear full-rank applications of rank equal to $\log_2(q_1)$ if the corresponding column is in $G(q_1)$.

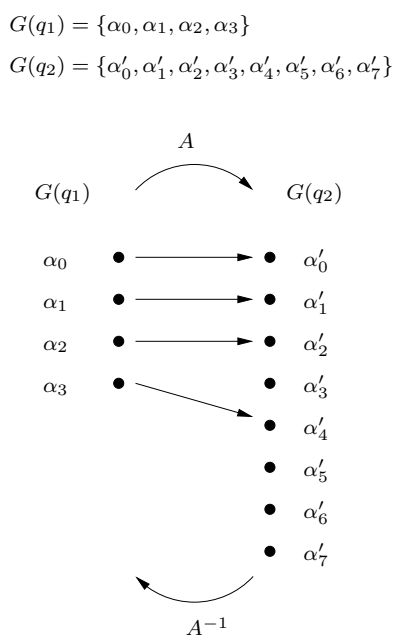


Figure 2.2 : Message transform through linear map.

In the sequel, we denote by q_1 and q_2 the group orders of given variable node and check node, respectively. With the assumption of section 2.1.2, we have $q_1 \leq q_2$. When looking at the factor graph of a hybrid LDPC code (see figure 2.1), we note that an edge of the graph carries two kinds of message probability vectors: messages of size q_1 and messages of size q_2. The function node corresponding to the linear map A (called h_{ij} on figure 2.1) is meant to make the components of the two types of message probability vectors correspond to each other. The transform of the probability vector is denoted *extension* from $G(q_1)$ to $G(q_2)$ when passing through A from variable node to check node, and the transform from $G(q_2)$ to $G(q_1)$ is denoted truncation from check node to variable node. We now give precise definitions of extension and truncation.

Let A be an element of the set of linear maps from $G(q_1)$ to $G(q_2)$ which are full-rank. Im(A) denotes the image of A (that is injective since dim(Im(A))=rank(A)=p_1).

The notations are the ones of figure 2.2.

$$\begin{array}{rrcl} A: & G(q_1) & \rightarrow & G(q_2) \\ & \alpha_i & \rightarrow & \alpha'_j = A(\alpha_i) \end{array}$$

Definition 6 *The extension* $\mathbf{y}$ *of the probability vector* $\mathbf{x}$ *by* A *is denoted by* $\mathbf{y} = \mathbf{x}^{\times A}$ *and defined by, for all* $j = 0, \ldots, q_2 - 1$*,*

$$\begin{array}{ll} \textit{if } \alpha'_j \notin \textit{Im}(A), & y_j = 0 \\ \textit{if } \alpha'_j \in \textit{Im}(A), & y_j = x_i \textit{ with } i \textit{ such that } \alpha'_j = A(\alpha_i) \end{array}$$

Although A is not bijective, we define A^{-1} the pseudo-inverse of A, by

$$\begin{array}{rrcl} A^{-1}: & \mathrm{Im}(A) & \rightarrow & G(q_2) \\ & \alpha'_j & \rightarrow & \alpha_i \text{ with } i \text{ such that } \alpha'_j = A(\alpha_i) \end{array}$$

Definition 7 *The truncation* $\mathbf{x}$ *of the probability vector* $\mathbf{y}$ *by* A^{-1} *is denoted by* $\mathbf{x} = \mathbf{y}^{\times A^{-1}}$ *and defined by, for all* $i = 0, \ldots, q_1 - 1$*,*

$$x_i = y_j \textit{ with } j \textit{ such that } \alpha'_j = A(\alpha_i)$$

In the sequel, we use a shortcut by calling the extension a linear map A, and by calling truncation its pseudo-inverse A^{-1}. Indeed, extension or truncation are generated by a linear map A and do not apply to group elements (i.e. symbol values), but on probability vectors. Additionally, we denote by $E_{k,l}$ the set of *extensions* from $G(q_k)$ to $G(q_l)$, and by $T_{k,l}$ the set of *truncation*s from $G(q_l)$ to $G(q_k)$.

2.1.5 Parametrization of hybrid LDPC ensemble

Classical LDPC codes are usually parametrized by two polynomials $(\lambda(x), \rho(x))$, whose each coefficient λ_i (resp. ρ_j) describes the proportions of edges connected to a variable node of degree i (resp. to a check node of degree j) [10]. Kasai et al. [52] introduced a detailed representation of LDPC codes, described by two-dimensional coefficients $\Pi(i, j)$, which are the proportion of edges connected to a variable node of degree i and also to a check node of degree j. Another important detailed and more general representation of LDPC codes is the multi-edge type [27], which we discuss at the end of this section.

In our case, an edge of the Tanner graph of an hybrid LDPC code has four parameters (i, q_k, j, q_l). An edge with these parameters is connected to a variable node in $G(q_k)$ of degree i, and is connected to a check node in $G(q_l)$ of degree j. We decide to extend the notation adopted by Kasai et al. in [52], and we denote by $\Pi(i, j, k, l)$ the proportion of edges connected to a variable node of degree i in $G(q_k)$ and to a check node of degree j in $G(q_l)$ (see figure 2.3).

Hence, $\Pi(i, j, k, l)$ is a joint probability which can be decomposed in several ways thanks to Bayes rule. For example, we have :

$$\Pi(i, j, k, l) = \Pi(i, j)\Pi(k, l|i, j)$$

where $\Pi(i,j)$ corresponds exactly to the definition adopted by Kasai, and $\Pi(k,l|i,j)$ describes the way the different group orders are allocated to degree i variable nodes and degree j check nodes.

A ensemble of hybrid LDPC codes is parametrized by Π and made of all the possible hybrid parity-check matrices whose parameters are those of the ensemble. The linear map of the parity-check matrices are chosen uniformly at random. Such a ensemble will be also called a Π hybrid LDPC code ensemble.

We denote by $\delta_{i,j}$ the Kronecker symbol ($\delta_{i,j} = 1$ if $i = j$, $\delta_{i,j} = 0$ otherwise). Here are some examples of specific parametrization of interest:

- When the code is on a single group (or field) $G(q)$ with uniform repartition of edges between the different degrees of connection, the four-dimensional representation reduces to: $\Pi(i) = \lambda_i$ and $\Pi(j) = \rho_j$ when $\Pi(k,l|i,j) = \delta_{q_k,q}\delta_{q_l,q}$. This is the description of irregular non-binary LDPC codes analyzed in [48].

- When the LDPC code is in $GF(2)$ and the repartition of edges between the different degrees of connection is non-uniform, the code is described by $\Pi(i,j)$ and $\Pi(k,l|i,j) = \delta_{q_k,2}\delta_{q_l,2}$. This corresponds to the detailed representation of irregular LDPC codes [52].

- When the hybrid LDPC codes has the check connection profile independent of the other parameters, and the connection profile of variable node depends on the proportion of each group order, the four-parameters representation reduces to:

$$\begin{aligned}\Pi(i,j,k,l) &= \Pi(i,k,l)\Pi(j)\\ &= \Pi(i,k)\Pi(l|i,k)\Pi(j)\\ &= \Pi(i|k)\Pi(j)\Pi(k)\Pi(l|k)\end{aligned}$$

- When the hybrid LDPC code has regular (d_v, d_c) connection profile:

$$\begin{aligned}\Pi(i,j,k,l) &= \Pi(j)\Pi(i,k|l)\Pi(l)\\ &= \delta(i,d_v)\delta(j,d_c)\Pi(k|l)\Pi(l)\end{aligned}$$

In the reminder, for more readable notations, we will write $\Pi(i,j,k)$ to denote the marginal distribution over l. The same with any other combinations of i,j,k,l, we will always use the same letters i,j,k,l to identify the parameters and the considered marginals.

Thus the very rich parametrization of hybrid LDPC codes, with four parameters, highlights the generality of this class of codes, which includes classical irregular binary and non-binary LPDC codes, and which allows more degrees of freedom. In particular, compared to D-GLDPC for example, we will be able to optimize the length of the generalized codes given their dimensions K or M, which are the group order characteristics. However, this representation is not as general as the one of multi-edge type LDPC codes [27] because, e.g., it cannot distinguish a check node connected to only one degree-1 variable, thereby preventing the use of degree one variable nodes in such described hybrid LDPC

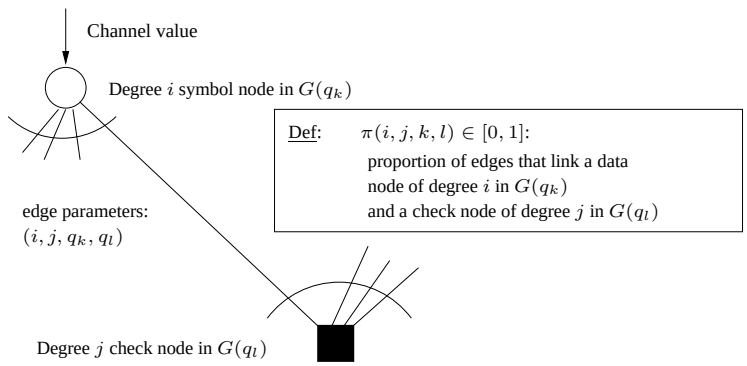


Figure 2.3 : Parametrization of a hybrid LDPC code ensemble

code ensembles.

We also define node wise proportions: $\tilde{\Pi}(i,k)$ and $\tilde{\Pi}(j,l)$ are the proportions of variable nodes of degree i in $G(q_k)$ and check nodes of degree j in $G(q_l)$, respectively. The connections between edgewise and node wise proportions are the following:

$$\begin{aligned}\tilde{\Pi}(i,k) &= \frac{\frac{\sum_{j,l}\Pi(i,j,k,l)}{i}}{\sum_{i,k}\frac{\sum_{j,l}\Pi(i,j,k,l)}{i}} \\ \tilde{\Pi}(j,l) &= \frac{\frac{\sum_{i,k}\Pi(i,j,k,l)}{j}}{\sum_{j,l}\frac{\sum_{i,k}\Pi(i,j,k,l)}{j}}\end{aligned} \tag{2.3}$$

The design code rate (i.e., the code rate when the parity-check matrix is full-rank) corresponding to the distribution Π is expressed by:

$$R = 1 - \frac{\sum_l \left(\sum_j \frac{\sum_{i,k}\Pi(i,j,k,l)}{j}\right)\log_2(q_l)}{\sum_k \left(\sum_i \frac{\sum_{j,l}\Pi(i,j,k,l)}{i}\right)\log_2(q_k)}$$

We define the *graph rate* as the rate of the binary code whose Tanner graph has parameters $\Pi(i,j)$. It is interesting to express the graph rate R_g in terms of Π, to compare it to the code rate of the hybrid code:

$$R_g = 1 - \frac{\sum_j \frac{\sum_i \Pi(i,j)}{j}}{\sum_i \frac{\sum_j \Pi(i,j)}{i}}$$

For the linear maps we consider, variable nodes are always in group of order lower than or equal to the group order of the check nodes to which they are connected. Hence the graph rate will be always higher than the code rate.

2.1.6 Encoding of hybrid LDPC codes

To encode hybrid LDPC codes whose non-zero elements are aforementioned full-rank linear maps, we consider upper-triangular parity-check matrices which are full-rank, i.e., without all-zero rows. The redundancy symbols are computed recursively, starting from the redundancy symbol depending only on information symbols. The images by the linear maps of the symbols involved in the parity-check equation but the redundancy symbol being computed, are summed up. the summation is performed in the group of the redundancy symbol, i.e., the group of the coresponding row. The redundancy symbol is set to the inverse of this sum by the linear map connected to it. This linear map is bijective from $G(q_l)$ to $G(q_l)$, if $G(q_l)$ is the group the redundancy symbol belongs to. Hence, information symbols satisfy that any assignment of values to them is valid, and the redundancy symbols are computed from them.

2.1.7 Decoding algorithm for hybrid LDPC codes

To describe the BP decoding, let $\mathbf{l}_{cv}^{(t)}$ denote the probability-vector message going into variable node v from check c at the tth iteration, and $\mathbf{r}_{vc}^{(t)}$ the probability-vector message going out of variable node v to check node c at the tth iteration. The connection degrees of v and c are denoted by d_v and d_c, respectively. Let A_{vc} denote the linear map on the edge connecting variable node v to check node c. The ath component of $\mathbf{l}_{cv}^{(t)}$ is denoted by $l_{cv}^{(t)}(a)$. The same holds for $r_{vc}^{(t)}(a)$. Let $\mathbf{x}$ be the sent codeword and N the number of codeword symbols. We recall that we simplify the notation as follows: for any group $G(q)$, for all $a \in [0, q-1]$, the element α_a is now denoted by a. Also, since A is a linear map, the matrix of the map is also denoted by A. Hence, for all linear map A from $G(q_1)$ to $G(q_2)$, $A(\alpha_i) = \alpha_j$ with $\alpha_i \in G(q_1)$ and $\alpha_j \in G(q_2)$, is translated into $Ai = j$ with $i \in [0, \ldots, q_1 - 1]$ and $j \in [0, \ldots, q_2 - 1]$.

- Initialization: Let $x_i \in G(q_i)$ be the ith sent symbol and y_i be the corresponding channel output, for $i = 0 \ldots N-1$. For each check node c connected to the vth variable node v, and for any $a \in [0, \ldots, q_k - 1]$:

$$r_{vc}^{(0)}(a) = r_v^{(0)}(a) = P(Y_v = y_v | X_v = a) \ ;$$

$$l_{vc}^{(0)}(a) = 1 \ .$$

- Variable node update: Consider a check node c and a variable node v. Let $\{c_1, \ldots, c_{d_v-1}\}$ be the set of all check nodes connected to v, except c. For all $a \in G(q_v)$

$$r_{vc}^{(t+1)}(a) = \mu_{vc} r_v^{(0)}(a) \prod_{n=1}^{d_v-1} l_{c_n v}^{(t)}(a) \tag{2.4}$$

where μ_{vc} is a normalization factor such that $\sum_{a=0}^{q_v-1} r_{vc}^{(t+1)}(a) = 1$.

- Check node update: Consider a check node c and a variable node v. Let $\{v_1, \ldots, v_{d_c-1}\}$ be the set of all variable nodes connected to c, except v. Let G be the Cartesian product group of the groups of the variable nodes in $\{v_1, \ldots, v_{d_c-1}\}$. For all $a \in G(q_v)$

$$l_{cv}^{(t)}(a) = \mu_{cv} \sum_{\substack{(b_1,\ldots,b_{d_c-1})\in G: \\ \bigoplus_{i=1}^{d_c-1} A_{v_i c} b_i = A_{vc} a}} \prod_{n=1}^{d_c-1} r_{v_n c}^{(t)}(b_n) \tag{2.5}$$

  where μ_{cv} is a normalization factor, and the $\bigoplus$ operator highlights that the addition is performed over $G(q_c)$, the group of the row corresponding to c, as defined in Section 2.1.4.

- Stopping criterion: Consider a variable node v. Let $\{c_1, \ldots, c_{d_v}\}$ be the set of all check nodes connected to v. Equation (2.6) corresponds to the decision rule on symbols values, at iteration t:

$$\hat{x}_v^{(t)} = \arg\max_a r_v^{(0)}(a) \prod_{n=1}^{d_v} l_{c_n v}^{(t)}(a) \ . \tag{2.6}$$

  Variable and check node updates are performed iteratively until the decoder has converged to a codeword, or until the maximum number of iterations is reached.

It is possible to have an efficient Belief propagation decoder for hybrid LDPC codes. As mentioned in [11][45], the additive group structure possesses a Fourier transform, so that efficient computation of the convolution can be done in the Fourier domain. One decoding iteration of BP algorithm for hybrid LDPC codes, in the probability domain with a flooding schedule, is composed of:

- Step 1 **Variable node update** in $G(q_j)$: pointwise product of incoming messages
- Step 2 **Message extension** $G(q_j) \rightarrow G(q_i)$ (see definition 6)
- Step 3 **Parity-Check update** in $G(q_i)$ in the Fourier domain
  - FFT of size q_i
  - Pointwise product of FFT vectors
  - IFFT of size q_i
- Step 4 **Message truncation** from $G(q_i) \rightarrow G(q_j)$ (see definition 7)

Although we do not focus on low-complexity decoders, it is important to note that hybrid LDPC codes are compliant with reduced complexity non-binary decoders which have been presented recently in the literature [46, 47]. In particular, [46] introduces simplified decoding of $GF(q)$ LDPC codes and shows that they can compete with binary LDPC codes even in terms of decoding complexity.

2.2 Asymptotic analysis of hybrid LDPC code ensembles

In this section, we describe the density evolution analysis for hybrid LDPC codes. Density evolution is a method for analyzing iterative decoding of code ensembles. In this section, we first prove that, on a binary input symmetric channel (BISC), we can assume that the all-zero codeword is transmitted because the hybrid decoder preserves the symmetry of messages, which entails that the probability of error is independent of the transmitted codeword.

We express the density evolution for hybrid LDPC codes, and mention the existence of fixed points, which can be used to determine whether or not the decoding of a given hybrid LDPC code ensemble is successful for a given SNR, in the infinite codeword length case. Thus, convergence thresholds of hybrid LDPC codes are similarly defined as for binary LDPC codes [10]. However, as for $GF(q)$ LDPC codes, the implementation of density evolution of hybrid LDPC codes is too computationally intensive, and an approximation is needed.

Thus, we derive a stability condition, as well as the EXIT functions of hybrid LDPC decoder under Gaussian approximation, with the goal of finding good parameters for having good convergence threshold. We restrict ourselves to binary input symmetric channels, but all the demonstrations can be extended to non-symmetric channels by using, e.g., a coset approach [48].

2.2.1 Symmetry of the messages

The definitions and properties induced by channel symmetry have been developed in section 1.5.2. All the lemmas carry unchanged over the hybrid LDPC ensemble.

Lemma 5 *Let $P_e^{(t)}(\mathbf{x})$ denote the conditional error probability after the tth BP decoding iteration of a hybrid LDPC code, assuming that codeword $\mathbf{x}$ was sent. If the channel is symmetric, then $P_e^{(t)}(\mathbf{x})$ is independent of $\mathbf{x}$.*

The proof of this lemma is provided in Section 2.7. For Lemma 4, we add the two following lemmas to the proof.

Lemma 6 *If $\mathbf{W}$ is a symmetric LDR random vector, then its extension $\mathbf{W}^{\times A}$, by any linear extension A with full rank, remains symmetric. The truncation of $\mathbf{W}$ by the inverse of A, denoted by $\mathbf{W}^{\times A^{-1}}$, is also symmetric.*

Proof of lemma 6 is given in section 2.7. The specificity of hybrid LDPC codes lies in function nodes on edges. Thus, when hybrid LDPC codes are decoded with BP, both data pass and check pass steps are the same as classical non-binary codes decoding steps. Since these steps preserve symmetry [10], lemma 6 ensures that the hybrid decoder preserves the symmetry property if the input messages from the channel are symmetric.

2.2.2 Density evolution

Analogously to the binary or non-binary cases, density evolution for hybrid LDPC codes tracks the distributions of messages produced by the BP algorithm, averaged over all possible neighborhood graphs on which they are based. The random space is comprised of random channel transitions, the random selection of the code from a hybrid LDPC ensemble parametrized by Π, and the random selection of an edge from the graph. The random space does not include the transmitted codeword, which is assumed to be set to the all-zero codeword (following Lemma 2). We denote by $\mathbf{R}^{(k)^{(0)}}$ the initial message across an edge connected to a variable in $G(q_k)$, by $\mathbf{R}^{(i,k)^{(t)}}$ the message going out of a variable node of degree i in $G(q_k)$ at iteration t. The message going out of a check node of degree j in $G(q_l)$ at iteration t is denoted by $\mathbf{L}^{(j,l)^{(t)}}$. We denote by $\mathbf{x}_l$ and $\mathbf{x}_k$ any two probability vectors of size q_l and q_k, respectively.

Let us denote by $\mathcal{P}_q$ the set of all probability vectors of size q. Let $\mathbf{r}_{q_k}(\mathbf{r}^{(0)}, \mathbf{l}^{(1)}, ..., \mathbf{l}^{(i-1)})$ denote the message map of a variable node of degree i in $G(q_k)$, as defined in equation (2.4): the input arguments are i probability vectors of size q_k. Let $\mathbf{l}_{q_l}(\mathbf{r}^{(1)}, ..., \mathbf{r}^{(j-1)})$ denote the message map of a check node of degree j in $G(q_l)$: the input arguments are $j-1$ probability vectors of size q_l.

$$P(\mathbf{L}^{(j,l)^{(t)}} = \mathbf{x}_l) = \sum_{\substack{\mathbf{r}^{(1)},...,\mathbf{r}^{(j-1)} \in \mathcal{P}_{q_l}: \\ \mathbf{l}_{q_l}(\mathbf{r}^{(1)},...,\mathbf{r}^{(j-1)}) = \mathbf{x}_l}} \prod_{n=1}^{j-1} \sum_{i,k} \Pi(i,k|j,l) \sum_{\substack{A \in E_{k,l}: \\ (\mathbf{r}^{(n)\times A^{-1}\times A}) = \mathbf{r}^{(n)}}} P(A) P(\mathbf{R}^{(i,k)^{(t)}} = \mathbf{r}^{(n)\times A^{-1}}) \; ; \quad (2.7)$$

$$P(\mathbf{R}^{(i,k)^{(t)}} = \mathbf{x}_k) = \sum_{\substack{\mathbf{r}^{(0)},\mathbf{l}^{(1)},...,\mathbf{l}^{(i-1)} \in \mathcal{P}_{q_k}: \\ \mathbf{r}_{q_k}(\mathbf{r}^{(0)},\mathbf{l}^{(1)},...,\mathbf{l}^{(i-1)}) = \mathbf{x}_k}} P(\mathbf{R}^{(k)^{(0)}} = \mathbf{r}^{(0)}) \prod_{n=1}^{i-1} \sum_{j,l} \Pi(j,l|i,k) \sum_{A \in E_{k,l}} P(A) \sum_{\substack{\mathbf{r} \in P_{q_l}: \\ \mathbf{r}^{\times A^{-1}} = \mathbf{l}^{(n)}}} P(\mathbf{L}^{(j,l)^{(t)}} = \mathbf{r}) \; . \quad (2.8)$$

Richardson and Urbanke [11] proved a *concentration theorem* that states that, as the block length N tends to infinity, the bit error rate at iteration t, of any graph of a given code ensemble, converges to the probability of error on a cycle-free graph in the same ensemble. The convergence is in probability, exponentially in N. As explained in [48] for classical non-binary LDPC codes, replacing bit- with symbol- error rate, this theorem carries over hybrid LDPC density-evolution unchanged.

Moreover, one can prove that the error-probability is a non-increasing function of the decoding iterations, in a similar way to the proof of Theorem 7 in [10]. This non-increasing property ensures that the sequence corresponding to density evolution, by iterating between equations (2.7) and (2.8), converges to a fixed point. Implementing the density evolution allows to check whether or not this fixed point corresponds to the zero error probability, which means that the decoding in the infinite codeword length case has been successful. Furthermore, Richardson and Urbanke proved in [11] the monotonicity

of error probability in terms of the channel parameter for physically degraded channels. Thus hybrid LDPC codes, like binary or non-binary LDPC codes, exhibit a threshold phenomenon.

Like for $GF(q)$ LDPC codes, implementing the density evolution for hybrid LDPC codes is too computationally intensive. Thus, in the sequel, we present a useful property of hybrid LDPC code ensembles, which allows to derive both a stability condition and an EXIT chart analysis for the purpose of approximating the exact density evolution for hybrid LDPC code ensembles.

2.2.3 Invariance induced by linear maps (LM-invariance)

Now we introduce a property that is specific to the hybrid LDPC code ensembles. Bennatan et al. in [48] used permutation-invariance to derive a stability condition for non-binary LDPC codes, and to approximate the densities of graph messages using one-dimensional functionals, for extrinsic information transfer (EXIT) charts analysis. The difference between non-binary and hybrid LDPC codes lies in the non-zeros elements of the parity-check matrix. Indeed, they do not correspond anymore to cyclic permutations, but to *extensions* or *truncations* which are linear maps (according to definitions 6 and 7). Our goal in this section is to prove that linear map-invariance (shortened by LM-invariance) of messages is induced by choosing uniformly the *extensions*. In particular, LM-invariance allows to characterize message densities with only one scalar parameter.

Until the end of the current section, we work with probability domain random vectors, but all the definitions and proofs also apply to LDR random vectors.

Definition 8 *A random vector $\mathbf{Y}$ of size q_l is LM-invariant if and only if for all k and $(A^{-1}, B^{-1}) \in T_{k,l} \times T_{k,l}$, the random vectors $\mathbf{Y}^{\times A^{-1}}$ and $\mathbf{Y}^{\times B^{-1}}$ are identically distributed.*

Lemma 7 *If a random vector $\mathbf{Y}$ of size q_l is LM-invariant, then all its components are identically distributed.*

Proof of lemma 7 is given in section 2.7.3.

Definition 9 *Let $\mathbf{X}$ be a random vector of size q_k, we define the random-extension of size q_l of $\mathbf{X}$, denoted $\tilde{\mathbf{X}}$, as the random vector $\mathbf{X}^{\times A}$, where A is uniformly chosen in $E_{k,l}$ and independent of $\mathbf{X}$.*

Lemma 8 *A random vector $\mathbf{Y}$ of size q_l is LM-invariant if and only if there exist q_k and a random vector $\mathbf{X}$ of size q_k such that $\mathbf{Y} = \tilde{\mathbf{X}}$.*

Proof of lemma 8 is given in section 2.7.3.

Thanks to lemma 6, the messages going into check nodes are LM-invariant in the ensemble of hybrid LDPC codes with uniformly chosen *extensions*. Moreover, random-*truncations*, at check node output, ensures LM-invariance of messages going into variable node (except the one from the channel).

2.2.4 The Stability condition for hybrid LDPC Codes

The stability condition, introduced in [10], is a necessary and sufficient condition for the error probability to converge to zero, provided it has already dropped below some value. This condition must be satisfied by the SNR corresponding to the threshold of the code ensemble. Therefore, ensuring this condition, when implementing an approximation of the exact density evolution, helps to have a more accurate approximation of the exact threshold.

In this paragraph, we generalize the stability condition to hybrid LDPC codes. Let $p(y|x)$ be the transition probabilities of the memoryless output symmetric channel and $c^{(k)}$ be defined by

$$c^{(k)} = \frac{1}{q_k - 1} \sum_{i=1}^{q_k - 1} \int \sqrt{p(y|i)p(y|0)}dy$$

Let $\mathbf{x}$ be a positive real-valued vector of size the number of different group orders. Let us define the g function by:

$$g(k, c^{(k)}, \Pi, \mathbf{x}) = c^{(k)} \Pi(i = 2|k) \sum_{j,l} \Pi(j, l|i, k)(j - 1) \sum_{k'} \Pi(k'|j, l) \frac{q_{k'} - 1}{q_l - 1} x_{k'}$$

For more readable notations, we also define the vector output function $\mathbf{G}(\mathbf{x})$ by:

$$\mathbf{G}(\mathbf{x}) = \{g(k, c^{(k)}, \Pi, \mathbf{x})\}_k$$

which means that the pth component of $\mathbf{G}(\mathbf{x})$ is $G_p(\mathbf{x}) = g(p, c^{(p)}, \Pi, \mathbf{x})$. Let $P_e^{(k)^t} = P_e(\mathbf{R}_t^{(k)})$ be the error probability when deciding the value of a symbol in $G(q_k)$ at iteration t. The global error probability of decision is $P_e^t = \sum_k \Pi(k) P_e^{(k)^t}$. Let us denote the convolution by $\otimes$. Then $\mathbf{x}^{\otimes n}$ corresponds to the convolution of vector $\mathbf{x}$ by itself n times.

Theorem 3 *Consider a given hybrid LDPC code ensemble parametrized by* $\Pi(i, j, k, l)$. *If there exists a vector* $\mathbf{x}$ *with all positive components, such that, for all* k, $\lim_{n \to \infty} g(k, c^{(k)}, \Pi, \mathbf{G}^{\otimes n}(\mathbf{x})) = 0$, *then there exist* t_0 *and* ϵ *such that, if* $P_e^{t_0} < \epsilon$, *then* P_e^t *converges to zero as* t *tends to infinity.*

Proof of theorem 3 is given in section 2.7.4.

This theorem only gives a sufficient condition for stability of the code ensemble. However, it may be possible to prove that this condition is also necessary by considering the actual transmission channel as a degraded version of an *erasurized* channel, as done in [48]. Indeed, all the necessary conditions to have such a proof, like, e.g., the cyclic-symmetry of a symmetric channel, the binary symmetry of LM-invariant symmetric messages or the equality between the random extended-truncated sum of messages and the sum of extended-truncated messages can be easily shown. To do such a proof, one must be careful to the fact that a node observes identically distributed messages, but different kinds of nodes do not observe identically distributed messages. By lake of time,

we have not completed this proof of necessity, and hence do not present the mentioned intermediate results. Although the necessity of stability condition has not been proved, it is sufficient for comparing to stability condition of classical binary and non-binary LDPC codes.

We first note that, for a usual non-binary $GF(q)$ LDPC code, the hybrid stability condition reduces to non-hybrid stability condition, given by [48], because

$$\lim_{n\to\infty} g(k, c^{(k)}, \Pi, \mathbf{G}^{\otimes n}(\mathbf{x})) = 0$$

is equivalent in this case to

$$\rho'(1)\lambda'(0)\frac{1}{q_k - 1}\sum_{i=1}^{q_k-1}\int \sqrt{p(y|i)p(y|0)}dy < 1$$

When the transmission channel is BIAWGN, we have

$$\int \sqrt{p(y|i)p(y|0)}dy = exp(-\frac{1}{2\sigma^2}n_i)$$

Let Δ_{nb} be defined by

$$\frac{1}{q_k - 1}\sum_{i=1}^{q_k-1} exp(-\frac{1}{2\sigma^2}n_i)$$

with n_i, the number of ones in the binary map of $\alpha_i \in G(q)$. Under this form, we can prove that Δ tends to zero as q goes to infinity on BIAWGN channel. This means that any fixed point of density evolution is stable as q tends to infinity for non-binary LDPC codes. This shows, in particular, that non-binary cycle-codes, that is with constant symbol degree $d_v = 2$, are stable if q tends to infinity, and can be used to design efficient coding schemes if q is large enough [33, 57].

As an illustration, we compare the stability conditions for hybrid LDPC codes with all variable nodes in $G(q)$ and all check nodes in $G(q_{max})$ and for non-binary LDPC codes defined on the highest order field $GF(q_{max})$. For hybrid codes of this kind, we have:

$$\lim_{n\to\infty} g(k, c^{(k)}, \Pi, \mathbf{G}^{\otimes n}(\mathbf{x})) = 0$$

is equivalent to

$$\left(\frac{1}{q-1}\sum_{i=1}^{q-1} exp(-\frac{1}{2\sigma^2}n_i)\right)\left(\Pi(i=2)\sum_j \Pi(j)(j-1)\frac{q-1}{q_{max}-1}\right) < 1$$

An advantage of hybrid LDPC codes over non-binary codes is that a hybrid LDPC code, with same maximum order group, can be stable at lower SNR.

On figure 2.4, we consider rate one-half non-binary LDPC codes on $GF(q)$, with $q = 2 \ldots 256$, and rate $R = 0.5$ hybrid LDPC codes of type $G(q) - G(q_{max})$, with all

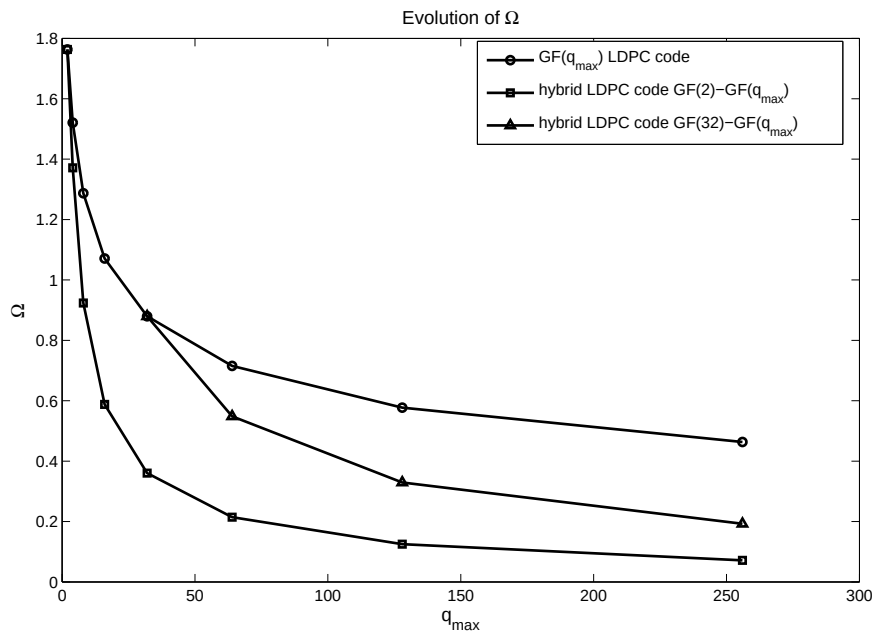


Figure 2.4 : Quantities Ω for hybrid and non-hybrid LDPC codes in terms of maximum symbol order q_{max}. These figures show that a hybrid LDPC code can be stable when a non-binary code is not.

variable nodes in $G(q)$ and check nodes in $G(q_{max})$. We assume regular Tanner graphs for all codes, with connection degree of variable nodes $d_v = 2$. Thus, the connection degree d_c of check nodes of non-binary LDPC codes is 4 for any $q_{max} = 2 \ldots 256$, while the one of hybrid LDPC codes varies with the graph rate:

$$d_c = \frac{d_v \log_2(q_{max})}{1-R} \frac{1}{\log_2(q)}$$

We consider BIAWGN channel whose noise parameter variance σ^2 is set to 0.97. We denote by Ω_{nb} and Ω_{hyb} the quantities of non-binary LDPC codes and hybrid LDPC codes, respectively, which must be strictly lower than one for stability. We observe, on figure 2.4, that $\Omega_{hyb} \leq \Omega_{nb}$. Hence, with the mentioned assumptions on chosen parameters values, a fixed point of density evolution is stable at lower SNR for hybrid LDPC codes than for classical $GF(q_{max})$ codes. It should be noted that the considered hybrid LDPC code ensemble corresponds to basic generalized LDPC codes [28]. Indeed, this is the only case where the general stability condition of theorem 3 can be simply expressed to be plotted. This result $\Omega_{hyb} \leq \Omega_{nb}$ is due to the fact that $\sum_{i=1}^{q-1} exp(-\frac{1}{2\sigma^2} n_i)$ is monotonically increasing with q. We compare the terms in both Ω_{hyb} and Ω_{nb}. In the hybrid case, $\sum_{i=1}^{q-1} exp(-\frac{1}{2\sigma^2} n_i)$ with $q < q_{max}$, while in the non-binary case it is $\sum_{i=1}^{q_{max}-1} exp(-\frac{1}{2\sigma^2} n_i)$. However in the hybrid case $d_c > 4$ (the graph rate is higher than the code rate). Since we obtained $\Omega_{hyb} \leq \Omega_{nb}$ for both $q = 2$ and $q = 32$, it can be conjectured that this result

holds for more elaborated hybrid LDPC codes, whose variable nodes belong to different group orders. However, we have not performed such a study. Note that the property according to which any fixed point of density evolution is stable as q tends to infinity for non-binary LDPC codes, also applies to hybrid LDPC codes of the above kind by inclusion. Those results indicate that there exist some cases where the optimization procedure to find good hybrid LDPC codes might be more efficient than for finding good non-binary LDPC codes, since the stability condition is less stringent.

2.2.5 EXIT charts and accuracy of the approximation for hybrid LDPC codes

Our goal is to find a method to measure the decoding threshold of a hybrid LDPC code ensemble with parameters Π, in such a way that it can be used in an optimization procedure, where the threshold will be used as the cost function. The decoding threshold is determined by tracking the densities of messages on an infinite cycle-free graph along the decoding iterations. For hybrid LDPC codes, the algorithm presented in section 2.2.2 is theoretically sufficient to compute the desired densities. However, in practice, a major problem is the fact that the quantities of memory required to store the probability density of a q-dimensional message grows exponentially with q. Exact density evolution is therefore too computationally intensive and we are going to look for a feasible and not too bad approximation of densities to track them. In [29], the authors analysed D-GLDPC on the BEC, which allowed to track only one parameter, the extrinsic information, instead of complete message densities. They used combinatorial calculus to express this extrinsic information.

We present the analysis for the BIAWGN channel. With binary LDPC codes, Chung et al. [50] observed that the variable-to-check messages are well approximated by Gaussian random variables, in particular when the variable node degree is high enough. The approximation is much less accurate for messages going out of check nodes. Furthermore, the symmetry of the messages in binary LDPC decoding implies that the mean m and variance σ^2 of the random variable are related by $\sigma^2 = 2m$. Thus, a symmetric Gaussian random variable may be described by a single parameter. This property was also observed by ten Brink et al. [14] and was essential to their development of EXIT charts for Turbo Codes. In the context of non-binary LDPC codes, Li et al. [49] obtained a description of $q-1$-dimensional Gaussian distributed messages by $q-1$ parameters. Bennatan et al. in [48] used symmetry and permutation-invariance to reduce the number of parameters from $q-1$ to one. This enabled the generalization of EXIT charts to $GF(q)$ LDPC codes. For hybrid LDPC codes, the Gaussian assumption for messages on the graph is not as straight forward as for classical binary or non-binary LDPC codes. This section discusses the accuracy of the Gaussian approximation for hybrid LDPC codes, and how we can handle it.

Projection of message densities on one scalar parameter

Our goal is to determine, for a given code ensemble parametrized by Π and a given SNR, when the decoding will be successful. Let us recall definition 5:
The mutual information between a symmetric LDR vector message $\mathbf{W}$ and the codeword sent, under the all-zero codeword assumption, is defined by:

$$I(C;\mathbf{W}) = 1 - \mathbb{E}\left(1 + \sum_{i=1}^{q-1} e^{-W_i} | C = 0\right)$$

The expectation is performed with respect to the density of $\mathbf{W}$.

We denote by $x_{APP}^{(t)}$ the average mutual information between a posteriori probability vectors and the channel input, computed at each variable node of the hybrid graph at iteration t. In the reminder of this part, we will shorten this expression by "the mutual information of a vector message". We state that the decoding is successful if and only if:

$$\lim_{t\to\infty} x_{APP}^{(t)} = 1 \tag{2.9}$$

In order to determine for which hybrid LDPC code ensemble defined by Π, equation (2.9) is satisfied at a given SNR, we have to track the message densities to evaluate $x_{APP}^{(t)}$ at each iteration t. Since tracking multi-variate densities of vector messages is prohibitive, we now present the approach we adopt to consider that these densities are determined by only one scalar parameter, that we are therefore going to track.

First, let us discuss the accuracy of the Gaussian approximation of the channel output in symbolwise LLR form for hybrid LDPC code ensembles. The channel outputs are noisy observations of bits, from which we obtain bitwise LLR, all identically distributed as $\mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ [50]. Let $\mathbf{s}$ be the vector gathering the LLRs $b_1, \ldots, b_{p_k}$ of bits of which a symbol in $G(q_k)$ is made: $\mathbf{s} = (b_1, \ldots, b_{p_k})^T$. Each component of an input LLR random vector $\mathbf{l}$ of size $(q_k - 1)$ is then a linear combination of these bitwise LLRs:

$$\mathbf{l} = B_{q_k} \cdot \mathbf{s}$$

where B_{q_k} is the matrix of size $q_k \times \log_2(q_k)$ in which the ith row is the binary map of the ith element of $G(q_k)$. The distribution of initial messages is hence a mixture of one-dimensional Gaussian curves, but is not Gaussian. Indeed, it is easy to see that the covariance matrix of vector $\mathbf{l}$ is not invertible.

Secondly, let us introduce a slight extension of Theorem 6 in [48].

Theorem 4 *Let $\mathbf{W}$ be an LDR random vector, Gaussian distributed with mean $\mathbf{m}$ and covariance matrix $\mathbf{\Sigma}$. Assume that the probability density function $f(\mathbf{w}$ of $\mathbf{W}$ exists and that $\mathbf{\Sigma}$ is nonsingular. Then $\mathbf{W}$ is both symmetric and LM-invariant if and only if there exits $\sigma > 0$ such that:*

$$\mathbf{m} = \begin{bmatrix} \sigma^2/2 \\ \sigma^2/2 \\ \vdots \\ \sigma^2/2 \end{bmatrix} \quad , \quad \mathbf{\Sigma} = \begin{bmatrix} \sigma^2 & & & \sigma^2/2 \\ & \sigma^2 & & \\ & & \ldots & \\ \sigma^2/2 & & & \sigma^2 \end{bmatrix}$$

The proof of Theorem 4 is the same as the proof of Theorem 6 in [48], because the permutation-invariance property [48] is used only through the fact that the components of a vector satisfying this property are identically distributed. This fact is ensured by a LM-invariant vector thanks to Lemma 7.

Thirdly, Lemma 7 ensures that, if a vector is LM-invariant, then its components are identically distributed. Hence, if we assume that a message is Gaussian distributed, symmetric and LM-invariant, its density depends on only one-scalar parameter. Let us now discuss the relevance of approximating the message densities of a hybrid LDPC code ensemble by Gaussian random vectors. Let $\mathbf{r}^{(t)}(\mathbf{x})$ be the density of a LDR message going out of a variable node in $G(q_k)$ after being extended by an extension chosen uniformly at random in $E_{k,k}$. Any component of such vector has density $r^{(t)}(x)$. Messages going out of variable nodes are extended when passing through the linear extension function nodes. As described in Section 2.1.4, the extension turns, e.g., a q_1-sized probability vector into a q_2-sized vector, with $q_2 \geq q_1$. This means that $q_2 - q_1$ of the resulting extended LDR message components are infinite, because these components of the corresponding probability vector are zero. Hence, the density of each component, of an extended message, is a mixture including a Dirac Δ_∞. Since this LDR vector is the random extension of the variable node output message, it is LM-invariant. From Lemma 7, each component is identically distributed.

Property 1 *The probability density function of any component of an LDR message after extension at iteration t, is expressed as*

$$d^{(t)}(x) = \beta r^{(t)}(x) + (1-\beta)\Delta_\infty$$

where the weight β is independent of t.

Proof: At any decoding iteration, $r^{(t)}(x)$ cannot have a Δ_∞ component because there exists no set of linear maps connected to the neighboring check nodes of v, such that there exists forbidden elements in $G(q_k)$ to which the symbol value associated to v cannot be equal. This is due to the fact that each check node (or the associated redundancy symbol) is in a group of order higher or equal to the group orders of its neighboring variable nodes. Hence, β is independent of the decoding iterations (it depends only on the groups of the codeword symbols).

□

It is therefore easy to show that any normalized moment, of order greater than 1, of the vector density (expectation of the product of a different number of its components) is equal to the same moment of the vector density $\mathbf{r}^{(t)}(\mathbf{x})$. Thus, if we assume that the vector density $\mathbf{r}^{(t)}(\mathbf{x})$, i.e., at variable node output, is dependent on only one scalar parameter, so is the whole density of the extended vector message. In other words, the density of vector message of a hybrid LDPC code cannot be approximated by a Gaussian density, due to the Δ_∞ component in the density, but is dependent on only one parameter if we assume that the density $\mathbf{r}^{(t)}(\mathbf{x})$ is Gaussian. The same property holds for messages before truncation, if we assume that messages going into variable nodes are Gaussian distributed.

Since the messages going into variable nodes are symmetric and LM-invariant, their sum done during the variable node update, is symmetric and LM-invariant by Lemma 18 in [48] and Lemma 11 (see Section 2.7). Hence, the one-scalar parameter approximation for hybrid LDPC codes is not less accurate than for $GF(q)$ LDPC codes [48].

The parameter, defining the message densities, we choose to track is the mutual information between a message and the codeword sent.

Since the connection between mutual information and the mean of a symmetric Gaussian distributed variable is easily obtained by interpolating simulation points, we consider means of Gaussian distributed vectors with same mutual information as the message vectors. That is we consider a projection of the message densities on Gaussian densities, based on Property 1 which ensures that densities of messages going out of or into check nodes are dependent on the same parameters as densities of messages going into or out of variable nodes. There are two models of messages handled by the hybrid decoder, and hence we define two functions to express the mutual information:

- Messages going out of variable nodes are not LM-invariant, and their mutual information is expressed thanks to a function called $J_v(\sigma^2, \mathbf{m}, q_k)$ in terms of the BIAWGN channel variance σ^2, a mean vector $\mathbf{m}$ and q_k, the group order of the variable node. The mean $\mathbf{m}$ is the mean of a Gaussian distributed vector.

- For a hybrid LDPC code ensemble with uniformly chosen linear maps, messages going into and out of check nodes are LM-invariant. If $G(q_l)$ denotes the group of the check node, the mutual information of messages is expressed by a function $J_c(m, q_l)$. m is the mean of a Gaussian random variable (any component of a Gaussian distributed vector with same mutual information as the graph message).

Let us now detail the evolution of mutual information of messages through BP decoding.

- The mutual information of a variable node output is expressed thanks to the $J_v(\cdot,\cdot,\cdot)$ function applied to sum of means, since variable node update is the summation of LDRs. Here, x_{in} is the mutual information of truncation operator output, and $\mathbf{1}_{q_k}$ is the all-one vector of size q_k. The mutual information x_{out} of the output of a variable node in $G(q_k)$ with connection degree i, is given by:

$$x_{out} = J_v(\sigma^2, (i-1)J_c^{-1}(x_{in}, q_k)\mathbf{1}_{q_k-1}, q_k) \ .$$

- The mutual information of extended message from $G(q_k)$ to $G(q_l)$ does not depend on which linear extension is used, but only on the group orders. Let x_{in} and x_{out} denote the mutual information of extension input and output, respectively. It follows from Definition 5

$$(1 - x_{out}) \log_2(q_l) = (1 - x_{in}) \log_2(q_k) \ .$$

- To express the mutual information of truncated message from $G(q_l)$ to $G(q_k)$, we use the LM-invariance property of input and output of the truncation operator. Let

x_{in} and x_{out} denote the mutual information of truncation input and output, respectively.

$$x_{out} = J_c(J_c^{-1}(x_{in}, q_l), q_k)$$

- Let $\mathbf{v}$ denote a probability vector, and $f(\mathbf{v})$ the corresponding Fourier transform (FT) vector. Let $x_{\mathbf{v}}$ be the mutual information of a probability vector $\mathbf{v}$, and $x_{f(\mathbf{v})}$ denote the function given in equation (1.18) applied to the vector $f(\mathbf{v})$.

  Lemma 9 *The connection between $x_{\mathbf{v}}$ and $x_{f(\mathbf{v})}$ is*

  $$x_{f(\mathbf{v})} = 1 - x_{\mathbf{v}} \ .$$

  The proof is provided in Section 2.7. Through a check node in $G(q_l)$ with connection degree j, the mutual information transform from the FT perspective is equivalent to the one given by the reciprocal channel approximation [58]:

  $$x_{out} = 1 - J_c((j-1)J_c^{-1}(1 - x_{in}, q_l), q_l) \ .$$

  The reciprocal channel approximation used for hybrid LDPC codes is not looser than when it is used with non-binary LDPC codes, since the message densities are considered as, or projected on, Gaussian densities in both cases. However, by computer experiment, the approximation is looser than for binary LDPC codes in the first decoding iterations when the check node degree is very low ($j = 3$ or 4).

We obtain the whole extrinsic transfer function of one iteration of the hybrid LDPC decoder (equation (2.12)). The mutual information of a message going out of a check node of degree j in $G(q_l)$ at the tth iteration and before truncation is denoted by $x_{cv}^{(j,l)^{(t)}}$. The same after truncation to become q_k sized is denoted $x_{cv,k}^{(j,l)^{(t)}}$. Analogously, the mutual information of a message going out of a variable node of degree i in $G(q_k)$ at the tth iteration and before extension is denoted by $x_{vc}^{(i,k)^{(t)}}$. The same after extension to become q_l-sized is denoted $x_{vc,l}^{(i,k)^{(t)}}$.

$$x_{vc,l}^{(i,k)^{(t)}} = 1 - \frac{\log_2(q_k)}{\log_2(q_l)}\left(1 - x_{vc}^{(i,k)^{(t)}}\right) \tag{2.10}$$

$$x_{cv}^{(j,l)^{(t)}} = 1 - J_c\left((j-1)J_c^{-1}(1 - \sum_{i,k}\Pi(i,k|j,l)x_{vc,l}^{(i,k)^{(t)}}, q_l), q_l\right) \tag{2.11}$$

$$x_{cv,k}^{(j,l)^{(t)}} = J_c\left(J_c^{-1}(x_{cv}^{(j,l)^{(t)}}, q_l), q_k\right)$$

$$x_{vc}^{(i,k)^{(t+1)}} = J_v\left(\sigma^2, (i-1)J_c^{-1}(\sum_{j,l}\Pi(j,l|i,k)x_{cv,k}^{(j,l)^{(t)}}, q_k), q_k\right) \tag{2.12}$$

We also define the a posteriori (or cumulative) mutual information for each kind of variable node at the tth iteration by

$$y^{(i,k)^{(t)}} = J_v\left(\sigma^2, i \cdot J_c^{-1}(\sum_{j,l}\Pi(j,l|i,k)x_{cv,k}^{(j,l)^{(t)}}, q_k), q_k\right) \ . \tag{2.13}$$

For any (i,k), $y^{(i,k)^{(t)}}$ is the quantity that must tend to 1 when t tends to infinity, for successful decoding. In the remainder, we refer to this mutual information evolution equation by using the notation $F(.)$ such that:

$$\{x_{vc}^{(i,k)^{(t+1)}}\}_{i,k} = F(\{x_{vc}^{(i,k)^{(t)}}\}_{i,k}, \Pi(i,j,k,l), \sigma^2) \ .$$

2.3 Distributions optimization

Let us recall that the condition we consider for successful decoding is

$$\lim_{t\to\infty} x_{APP}^{(t)} = 1$$

With classical unstructured LDPC codes, $x_{APP}^{(t+1)}$ can be expressed as a recursion in terms of $x_{APP}^{(t)}$. Hence, condition 2.9 is equivalent to $x_{APP}^{(t+1)} > x_{APP}^{(t)}$ $\forall t \geq 0$. With hybrid LDPC codes, we cannot write such a recursion because all nodes do not receive identically distributed messages. Thus, the usual condition $x_{APP}^{(t+1)} > x_{APP}^{(t)}$ is not the condition for successful decoding of hybrid LDPC code ensembles. We present two solutions to overcome this impediment to use classical EXIT charts. The first solution is to use *multi-dimensional EXIT charts*, following the idea of [53], though in a slightly different way. This method allows to handle all the degrees of freedom of the detailed representation for optimization of the code profile. The second solution consists in assuming parameters (j,l) of check nodes independent of parameter (i,k) of variable nodes. This will be done by assuming constant group order q_l for all check nodes, and degree of connection independent of the properties of the variable nodes to which they are connected. This method turns the optimization into a linear programming problem, hence much more quickly solved by computer than hill-climbing methods.

2.3.1 Context of the optimization

Optimization is performed for the BIAWGN channel. The goal of the optimization with EXIT charts is to find a good ensemble of hybrid LDPC codes with the lowest convergence threshold for a target code rate, under a Gaussian approximation. That means that we look for the parameters $\Pi(i,j,k,l)$ of the ensemble of hybrid LDPC codes with lowest convergence threshold. We decide not to explore group orders higher than $q_{max} = 256, p_{max} = 8$, nor connection degrees higher than $d_{v_{max}}$ and $d_{c_{max}}$, thus we look for $(i,j,k,l) \in [2, d_{v_{max}}] \times [2, d_{c_{max}}] \times [1,8] \times [1,8]$. Let us denote the code rate R, and the target code rate R_{target}. The optimization procedure consists in finding $\Pi(i,j,k,l)$ which

fulfills the following constraints at the lowest SNR:

$$\begin{aligned}
&\text{Code rate constraint:} && R = R_{target} \\
&\text{Proportion constraint:} && \sum_{i,j,k,l} \Pi(i,j,k,l) = 1 \\
&\text{Sorting constraint:} && \Pi(i,j,k,l) = 0, \quad \forall (i,j,k,l) \text{ such that } q_k > q_l \qquad (2.14) \\
&\text{Successful decoding condition:} && \lim_{t\to\infty} y^{(i,k)^{(t)}} = 1, \quad \forall (i,k) \qquad (2.15) \\
& && \text{with } \{x_{vc}^{(i,k)^{(t+1)}}\}_{(i,k)} = F(\{x_{vc}^{(i,k)^{(t)}}\}_{(i,k)}, \Pi(i,j,k,l), \sigma^2)
\end{aligned}$$

The threshold is the objective function. We do not include the stability condition in the optimization constraints because it is not easy to check it in the general case. However, as explained in section 2.2.4, we can assume it as non stringent for the optimization process. Let us recall the expression of the code rate, which is going to be used in the remainder:

$$R = 1 - \frac{\sum_l \left(\sum_j \frac{\sum_{i,k} \Pi(i,j,k,l)}{j} \right) \log_2(q_l)}{\sum_k \left(\sum_i \frac{\sum_{j,l} \Pi(i,j,k,l)}{i} \right) \log_2(q_k)} \qquad (2.16)$$

2.3.2 Optimization with multi-dimensional EXIT charts

The detailed representation $\Pi(i,j,k,l)$ turns hybrid LDPC code ensembles into structured code ensembles, which are characterized by sub-interleavers. In that case, the successful decoding condition $\lim_{t\to\infty} x_{APP}^{(t)} = 1$ is equivalent to $\lim_{t\to\infty} y^{(i,k)^{(t)}} = 1$ for all (i,k). The multi-dimensional EXIT algorithm can be presented as follows for hybrid LDPC codes:

1) Initialisation: t=0. Set $x_{cv}^{(j,l)^{(0)}} = 0$ for all (j,l).

2) Compute $x_{vc}^{(i,k)^{(t)}}$ for all (i,k) with equation (2.12).

3) Compute $x_{cv}^{(j,l)^{(t)}}$ for all (j,l) with equation (2.11).

4) Compute $y^{(i,k)^{(t)}}$ for all (i,k) with equation (2.13).

5) If $y^{(i,k)^{(t)}} = 1$ up to the desired precision for all (i,k) then stop; otherwise $t = t+1$ and go to step 2.

This algorithm converges only when the selected SNR is above the threshold. Thus, the threshold is the lowest value of SNR for which all $y^{(i,k)^{(t)}}$ converge to 1.

Letting the detailed representation $\Pi(i,j,k,l)$ fully general allows to have check nodes in different order groups. Indeed, allowing check nodes in different order groups has been inspired by the results obtained in [29] for D-GLDPC optimized on the BEC. In that article, the authors show that better thresholds and error-floors can be achieved by introducing only a small fraction of generalized codes at check and variable sides among classical single parity-check and repetition codes. In that case, the successful decoding condition constraint 2.15 cannot be expressed linearly in terms of $\Pi(i,j,k,l)$. That is

why we cannot use any linear programming tool for optimization, we need a hill-climbing method. As usually with LDPC codes optimization, we use the differential evolution algorithm [16]. The optimization problem has been expressed in the previous section. Several problems arise when optimizing hybrid LDPC codes with differential evolution:

- The parameter space. When there is no additional constraint on Π different of those above mentioned, the number of parameters, which are joint proportions, to be determined by the optimization method is $D = \frac{q_{max}(q_{max}+1)}{2} d_{v_{max}} d_{c_{max}}$. To get an idea on how many parameters DE algorithm is able to handle, the authors automatically limit the number of parameters to 35 in their code available from [16]. This limit is quickly reached in the case of optimization of hybrid LDPC codes, leading to an equivalent high number of population vectors and hence very slow convergence of DE. Therefore we have to make a heuristic reduction of the parameter space by, e.g., allowing only very small connection degrees for variable nodes ($d_{v_{max}} = 5$ to 10), only two different check degrees and two different group orders.

- The initialization problem. In spite of the reduction of the dimension of the parameter space, this space remains too big to allow to randomly initialize the population vectors, otherwise too few of them fulfill the code rate. That is why we need another method to well initialize the population vectors. We show that the initialization problem of finding vectors of proportions which correspond to code ensembles with target code rate R (see equation (2.16)) can be expressed by a convex combination problem [59]. This can be seen when one picks at random the marginal proportions $\Pi(j, l)$ for all (j, l), and looks for the conditional probabilities $\Pi(i, k|j, l)$ satisfying the code rate. To solve this problem, the solution we have used is the simplex method [60] with random cost function, which is used when the cost function and the problem constraints are linear in terms of the parameters to be optimized. However, the solutions found by the simplex algorithm always satisfy with equality some of the inequality constraints because the cost function is linear, therefore the solution to the maximization or minimization of the cost function is on facets of the constraint polytope which is a convex hull. This implies that a non-negligible part of proportion vector components will be set to zero or one by the brute simplex method. Thus, to use simplex for initialization of of the vector population of DE to non-trivial very bad components, we need to empirically adapt the lower and upper bounds of the vector components from $[0, 1]$ to, e.g., $[0.03, 0.95]$.

- Interpolations. Another difficulty in using DE to optimize hybrid LDPC distributions is the computation time entailed by $J_v(\cdot,\cdot,\cdot)$ and $J_c(\cdot,\cdot)$ functions. Indeed, the $J_v(\cdot,\cdot,\cdot)$ and $J_c(\cdot,\cdot)$ functions are evaluated by Monte-Carlo simulations offline, and then interpolated. For a given group order q_l, J_c is the function of only one parameter, which is the mean of any component of the LM-invariant vectors going into or out of the check node, and hence we use a mono-dimensional polynomial interpolation to get a functional approximation. For a given group order q_k, J_v is the function of three parameters, and hence we use a 2-dimensional spline surface to interpolate J_v. Since these functions are used in the multi-dimensional EXIT

(i, q_k) (j, q_l)	$(2,64)$	$(2,128)$	$(2,256)$	$(3,64)$	$(3,128)$	$(3,256)$	$(4,64)$	$(4,128)$	$(4,256)$	$\tilde{\Pi}(j,l)$
$(5,64)$	0.0073	×	×	0	×	×	0	×	×	0.0086
$(5,128)$	0	0.0089	×	0	0	×	0.0080	0.0175	×	0.0405
$(5,256)$	0.0003	0.0290	0.0001	0.0226	0	0	0	0.0001	0	0.0614
$(6,64)$	0.0087	×	×	0.0470	×	×	0.0554	×	×	0.1091
$(6,128)$	0.0367	0.0003	×	0.0521	0.0063	×	0.0218	0.0931	×	0.2065
$(6,256)$	0.4248	0.0197	0.0043	0.0851	0.0021	0.0101	0.0042	0.0151	0.0193	0.5739
$\tilde{\Pi}(i,k)$	0.5916	0.0717	0.0055	0.1707	0.0069	0.0083	0.0554	0.0779	0.0120	

Table 2.1 : Distribution $\Pi(i,j,k,l)$ of a hybrid LDPC code ensemble with code rate one-half and threshold 0.1864 dB under Gaussian approximation. The marginals $\tilde{\Pi}(i,k)$ and $\tilde{\Pi}(j,l)$ correspond to the proportions of variable nodes of type (i,k) and check nodes of type (j,l), respectively. When a proportion is forced to zero by the sorting constraint, $\times$ is put in the box.

charts, the computation time for the cost function, i.e., for the threshold, is much higher than in the binary case too.

Result of the optimization

It results from the optimization with DE that distributions with best thresholds are not obtained for a majority of binary variable and check nodes. It is worthy to recall that only small connection degrees are allowed for check nodes (5 or 6). Also, as mentioned in section 2.1.5, the detailed representation adopted in this work is less general than the multi-edge type representation [27]. Indeed, it is possible to consider proportions of different (i,k) type punctured symbols, but it is not possible to assume degree one variable nodes because we cannot describe check nodes with exactly one edge to such a variable. This is the reason why we logically do not get back the code distributions of multi-edge type LDPC codes [27], i.e., binary LDPC codes with low connection degrees and thresholds close to capacity. Instead, we obtain distributions Π with very low connection degrees (2 to 4) and very good thresholds under the above discussed Gaussian approximation, when only high order groups ($G(16)$ to $G(256)$) are allowed. This is in agreement with the results of [33].

An example of such a resulting distribution is given in table 2.1. Firstly, we see from this table that the optimization procedure puts a maximum of powerful component codes (or "generalized codes", see section 2.1.3), i.e. variable nodes in the smallest order group ($G(64)$) connected to check nodes in the highest order group ($G(256)$). Secondly, the variable nodes in a high order group tend to correspond to poor component codes, and hence, higher connection degrees are affected to this type of variable nodes in order to have a code length high enough to balance the high K, which is in turns $\log_2(q_k)$. This interpretation can also be made in terms of code doping [1, 61].

Graph construction

We now discuss the graph construction of such a hybrid LDPC code: how to build a graph satisfying the detailed representation, i.e., where all check nodes cannot be connected to

any variable nodes.

The first solution is to modify the PEG algorithm to take into account the structure specificity of such a hybrid LDPC ensemble where the global permutation is made of various sub-interleavers. However, we did not have enough time to do this.

Another way is to build the graph thanks to the protograph method [53], in the same way as multi-edge type LDPC codes are built. However, building the protograph of a hybrid LDPC code fulfilling the detailed representation resulting from the optimization, without additional restrictions on the detailed parametrization, can be quickly arduous. We did not have enough time to investigate this method.

Moreover, since the best thresholds resulting from DE have been observed for high order groups, this has been a hint to assume that we will not have an important loss in the achievable thresholds when restricting the detailed representation in these conditions. This restriction consists in considering all check nodes in the same group and with connection degrees independent of the variable nodes they are connected. This allows to switch from a non-linear optimization to a linear optimization, which is the topic of the following section.

Finally, it is worthy to note that all the presented tools, i.e. decoders and EXIT charts, may be used for optimization of hybrid protograph based LDPC codes by using equation presented in [53] with functions $J_v(\cdot,\cdot,\cdot)$ and $J_c(\cdot,\cdot)$, or hybrid multi-edge LDPC codes provided that the tools are adapted to the multi-edge type representation. However, some problems would have to be solved for the definition of such a code ensemble, e.g. can the linear maps be randomly chosen on each edge of the code graph resulting from lifting, or do they have to be the same as the one defining the protograph ?

2.3.3 Optimization with mono-dimensional EXIT charts

In this part, we consider the optimization of hybrid LDPC code ensembles with all check nodes in the same group $G(q_l)$ and with connection degrees independent of the variable nodes to which they are connected. We present how general equations (2.12) turns into mono-dimensional EXIT charts, and how this allows the use of linear programming for optimization. Let $x_e^{(t)}$ denote the averaged mutual information of extended messages. It is expressed in terms of the mutual information $x_{vc}^{(i,k)^{(t)}}$ of messages going out of variable nodes of degree i in $G(q_k)$, by simplification of equation (2.10):

$$x_e^{(t)} = 1 - \frac{1}{\log_2(q_l)} \sum_{i,k} \Pi(i,k) \log_2(q_k)(1 - x_{vc}^{(i,k)^{(t)}}) \ .$$

From equation (2.10), we can see that, for any (i,k,l):

$$\lim_{t\to\infty} x_{vc,l}^{(i,k)^{(t)}} = 1 \Leftrightarrow \lim_{t\to\infty} x_{vc}^{(i,k)^{(t)}} = 1$$

and then the successful decoding condition (2.15) reduces to

$$\lim_{t\to\infty} x_e^{(t)} = 1 \ .$$

By simplifying equation (2.12), $x_e^{(t+1)}$ can be expressed by a recursion in terms of $x_e^{(t)}$ as:

$$x_{cv,k}^{(j,l)\,(t)} = J_c\left(J_c^{-1}\left(1 - J_c\left((j-1)J_c^{-1}(1-x_e^{(t)}, q_l), q_l\right), q_l\right), q_k\right) \;;$$

$$x_e^{(t+1)} = \sum_{i,k} \Pi(i,k)\left(1 - \frac{log(q_k)}{log(q_l)}\left(1 - J_v\left(\sigma^2, (i-1)J_c^{-1}(\sum_j \Pi(j|i,k)x_{cv,k}^{(j,l)\,(t)}, q_k)\mathbf{1}_{q_k-1}, q_k\right)\right)\right) . \tag{2.17}$$

Thus, the condition for successful decoding of hybrid LDPC codes in that specific case is

$$\forall t \geq 0, \quad x_e^{(t+1)} > x_e^{(t)} \tag{2.18}$$

In that case, the optimization procedure aims at finding distribution $\Pi(i,k|j,l)$ for given $\Pi(j,l)$. We see on equation (2.17) that $x_e^{(t+1)}$ depends linearly on $\Pi(i,k)$, turning the optimization problem into a linear programming problem. We may jointly optimize th whole distribution $\Pi(i,k)$, but we rather prefer to present in the next sections two different methods. In each case, one of the two sets of parameters, $\Pi(i)$ or $\Pi(k)$, is set *a priori*.

Set group-order profile, open connexion profile

The first way to optimize $\Pi(i,k)$ is to set the different group orders, and then find the best connection profile of variable nodes for each group. Starting from $\Pi(i,j,k,l)$, the decomposition we use is the following:

$$\begin{aligned}\Pi(i,j,k,l) &= \Pi(i,k,l)\Pi(j)\\ &= \Pi(i,k)\Pi(l|i,k)\Pi(j)\\ &= \Pi(i|k)\Pi(k)\Pi(l|k)\Pi(j)\end{aligned}$$

Actually, we do not set the proportion of edges in $G(q_k)$ exactly, but the proportion of variable nodes in $G(q_k)$. We put the redundancy (check nodes and corresponding variable nodes) in the highest order group $G(q_{\text{red}}) = G(q_{max})$, corresponding to a proportion α_{red} of variable nodes, and the information variable nodes in two lower order groups $G(q_{\text{info1}})$ and $G(q_{\text{info2}})$, corresponding to proportions α_{info1} and α_{info2}.

Hence, the proportion which is optimized is $\Pi(i|k)$. This means that, for each group order k of variable node, we look for the best connection profile for these variable nodes in $G(q_k)$. Thus, we optimize as many connection profiles as the number of different group orders of variable nodes. This is performed in a single optimization procedure by concatenating $\Pi(i|k)$, $\forall(i,k)$ in a single vector. In this way, this vector of profiles will hence contain:

$$\begin{array}{rll}\text{First profile:} & \forall i = 2\ldots d_{v_{max}}, & \Pi(i,\,\text{red})\\ \text{Second profile:} & \forall i = 2\ldots d_{v_{max}}, & \Pi(i,\,\text{info1})\\ \text{Third profile:} & \forall i = 2\ldots d_{v_{max}}, & \Pi(i,\,\text{info2})\end{array} \tag{2.19}$$

Equation (2.17) reduces to:

$$x_e^{(t+1)} = F(x_e^{(t)}, \Pi(i,k), \sigma^2) \tag{2.20}$$

$$x_e^{(t+1)} = \sum_{\text{k=red,info1,info2}} \sum_i \Pi(i,k)\left(1 - \frac{log(q_k)}{log(q_{\text{red}})}\left(1 - J_v\left(\sigma^2, (i-1)J_c^{-1}(\sum_j \Pi(j)x_{cv,k}^{(j,\text{red})\,(t)}, q_k)\mathbf{1}_{q_k-1}, q_k\right)\right)\right)$$

Due to the fact that we a priori set the group orders of variable nodes necessarily equal or lower than check nodes group orders they are connected, the rate of the hybrid bipartite graph, whose nodes are in different order groups, is higher than the code rate (i.e., the actual rate of the transmission). Setting the proportion of variable nodes in $G(q_k)$ for all k also sets the rate of the graph, which becomes the target graph rate in the optimization procedure. From the target code rate R_{target}, we can compute the target graph rate, denoted by R_{graph} by:

$$R_{graph} = \frac{\dfrac{R_{target}}{\displaystyle\sum_{k=\text{info1,info2}} \alpha_k \log_2(q_k)}}{\dfrac{R_{target}}{\displaystyle\sum_{k=\text{info1,info2}} \alpha_k \log_2(q_k)} + \dfrac{1-R_{target}}{\alpha_{\text{red}} \log_2(q_{\text{red}})}} \tag{2.21}$$

The result of the optimization is finally the set of the three profiles $\Pi(i,k)$, $\forall(i,k) \in [1, d_{v_{max}}] \times$ [red,info1,info2], for which the following constraints are fulfilled at lowest SNR:

$$\begin{array}{rll}
\text{Proportion constraint:} & \forall i = 2 \ldots d_{v_{max}}, & \displaystyle\sum_i \Pi(i,\text{red}) + \Pi(i,\text{info1}) + \Pi(i,\text{info2}) = 1 \\
\text{Code rate constraint:} & \forall k = \text{red,info1,info2}, & \displaystyle\sum_i \frac{\Pi(i,k)}{i} = \frac{\alpha_k}{1-R_{graph}} \sum_j \frac{\Pi(j)}{j} \\
\text{Sorting constraint:} & \Pi(i,j,k,l) = 0, \quad \forall(i,j,k,l) \text{ such that } q_k > q_l & \\
\text{Successful decoding condition:} & x_e^{(t+1)} = F(x_e^{(t)}, \Pi(i,k), \sigma^2) > x_e^{(t)} &
\end{array} \tag{2.22}$$

Set connexion profile, open group-order profile

Another way to optimize hybrid LDPC ensembles is to set the connection profile and optimize the group orders of variable nodes. As in the previous section, we set the check node parameters (group order $G(q_{red})$ and connection profile), independently of the variable nodes parameters. This time, the decomposition of $\Pi(i,j,k,l)$ is:

$$\Pi(i,j,k,l) \quad = \quad \Pi(i,k|l)\Pi(j)\Pi(l)$$

Similarly to equation (2.19), we aim at optimizing several group order profiles, as many as the number of different variable node connection degrees. In a finite length performance purpose, we start from an ultra-sparse Tanner graph with a regular connection profile (e.g. ($d_v = 2, d_c = 3$)). Hence the previous decomposition falls into:

$$\Pi(i,j,k,l) \quad = \quad \delta(i,d_v)\delta(j,d_c)\Pi(k)\delta(l,\text{red})$$

Since the group order profile of the redundancy is set, the result of the optimization will be the group order profiles of information variable nodes. We denote by $\mathcal{I}$ the indexes of

the group order of information symbols. In other words, any information symbols is in $G(q_k)$ with $k \in \mathcal{I}$. Equation (2.17) reduces to:

$$\begin{aligned} x_e^{(t+1)} &= F(x_e^{(t)}, \Pi(k), \sigma^2) \\ x_e^{(t+1)} &= \sum_{k=red,\mathcal{I}} \Pi(k) \left(1 - \frac{log(q_k)}{log(q_{red})} \left(1 - J_v \left(\sigma^2, (d_v-1) J_c^{-1}(\sum_{j=d_c} \Pi(j) x_{cv,k}^{(j,red)^{(t)}}, q_k) \mathbf{1}_{q_k-1}, q_k\right)\right)\right) \end{aligned} \tag{2.23}$$

The graph rate R_{graph} is determined by $1 - \frac{d_v}{d_c}$, and the code rate R is hence:

$$R = \frac{R_{graph} \sum_{k \in \mathcal{I}} \Pi(k) \log_2(q_k)}{R_{graph} \sum_{k \in \mathcal{I}} \Pi(k) \log_2(q_k) + (1 - R_{graph}) \log_2(q_{red})} \tag{2.24}$$

R_{target} still denotes the target code rate, and the result of the optimization is hence the profile $\Pi(k)$, $\forall k \in \mathcal{I}$, for which the following constraints are fulfilled at lowest SNR:

Proportion constraint:	$\sum_k \Pi(k) = 1$
	$\forall k > red, \quad \Pi(k) = 0$
	$\Pi(red) >= 1 - R_{graph}$
Code rate constraint:	$R = R_{target}$ (see equation (2.24))
Opened EXIT chart:	$x_e^{(t+1)} = F\left(x_e^{(t)}, \Pi(k), \sigma^2\right) > x_{vc}^{(t)}$ (see equation (2.23))

Thresholds of distributions optimized in that ways are presented in section 2.5.1.

2.4 Finite length optimization

This section presents an extension of optimization methods that has been described in [34] for finite length non-binary LDPC codes with constant variable degree $d_v = 2$. We address the problem of the selection and the matching of the parity-check matrix $\mathbf{H}$ nonzero clusters. In this section, we assume that the connectivity profile and group order profile of the graph have been optimized, with constant variable degree $d_v = 2$. With the knowledge of the graph connectivity, we run a PEG algorithm [23] in order to build a graph with a high girth.

The method is based on the binary image representation of $\mathbf{H}$ and of its components, i.e. the non-zero clusters of the hybrid code in our case (cf. section 2.1). First, the optimization of the rows of $\mathbf{H}$ is addressed to ensure good waterfall properties. Then, by taking into account the algebraic properties of closed topologies in the Tanner graph, such as cycles or their combinations, an iterative method is used to increase the minimum distance of the binary image of the code by avoiding low weight codewords.

2.4.1 Row optimization

Based on the matrix representation of each nonzero entry, we give thereafter the equivalent vector representation of the parity-check equations associated with the rows of H.

Let $\mathbf{x} = [x_0 \dots x_{N-1}]$ be a codeword in $\mathcal{G} = G(q_{min}) \times \dots \times G(q_{max})$, and let p_j be the number of bits representing the binary map of symbol $x_j \in G(2^{p_j})$, $j = 0, \dots N-1$. For the ith parity equation of H in the group $G(2^{p_i})$, we have the following vector equation:

$$\sum_{j:H_{ij} \neq 0} H_{ij} \mathbf{x_j} = \mathbf{0} \tag{2.25}$$

where H_{ij} is the $p_i \times p_j$ binary matrix representation of the non-zero cluster, $\mathbf{x_j}$ is the vector representation (binary map) of the symbol x_j. The all zero component vector is noted $\mathbf{0}$.

Considering the i-th parity-check equation as a single component code, we define $\mathbf{H_i} = [H_{ij_0} \dots H_{ij_m} \dots H_{ij_{d_c-1}}]$ as its equivalent binary parity check matrix, with $\{j_m : m = 0 \dots d_c - 1\}$ the indexes of the nonzero elements of the i-th parity-check equation. The size of $\mathbf{H_i}$ is $p_i \times (p_{ij_0} + \dots + p_{ij_{d_c-1}})$, with p_i and p_{ij_k} the extension orders of the groups of the check node and the k-th connected variable node, respectively. Let $\mathbf{X_i} = [\mathbf{x_{j_0}} \dots \mathbf{x_{j_{d_c-1}}}]^t$ be the binary representation of the symbols of the codeword $\mathbf{x}$ involved in the ith parity-check equation. When using the binary representation, the i-th parity-check equation of $\mathbf{H}$ (2.25), can be written equivalently as $\mathbf{H_i}\mathbf{X_i}^t = \mathbf{0}^t$.

We define $d_{min}(i)$ as the minimum distance of the binary code associated with $\mathbf{H_i}$. As described in [34], a d_c-tuple of d_c linear maps is chosen in order to maximize the minimum distance $d_{min}(i)$ of the code corresponding to the ith row of $\mathbf{H}$, $i = 0, \dots, M-1$. For hybrid LDPC codes, we adopt the same strategy, and choose for $\mathbf{H_i}$ a binary linear component code with the highest minimum distance achievable with the dimensions of $\mathbf{H_i}$. For example, let $\mathbf{H_i}$ be obtained from a $d_c = 3$ check node with the three symbols belonging to $G(2^8) \times G(2^8) \times G(2^2)$, $\mathbf{H_i}$ has size (8×18) and the highest possible minimum distance is $d_{min}(i) = 5$ [62]. For hybrid LDPC codes, even if the connection degree is constant for all check nodes, the dimensions of the component code $\mathbf{H_i}$ could differ and depend on the symbols orders which appear in $\mathbf{X_i}$.

2.4.2 Avoiding low weight codewords

We now address the problem of designing codes with good minimum distance. It has been shown in [34] that the error floor of non-binary LDPC codes based on ultra-sparse ($d_v = 2$) graph is not uniquely due to pseudo-codewords, but also to low weight codewords. Here we consider hybrid LDPC codes with constant variable degree $d_v = 2$. We adopt for hybrid LDPC codes the same strategy that has been introduced in [34], which aims at avoiding the low weight codewords which are contained in the smallest cycles. In order to do so, we first extract and store the cycles of the Tanner graph with length belonging to $\{g, \dots, g + gap\}$, where g is the girth and gap is a small integer such that the number of cycles with size $g + gap$ is manageable.

As in the previous section, we consider the binary images of cycles as component codes. Let $\mathbf{Hc_k}$ be the binary image of the k-th stored cycle. Since we consider $(2, d_c)$ codes, if some columns of $\mathbf{Hc_k}$ are linearly dependent, so will be the columns of $\mathbf{H}$ (see [34] for more details). This means that a codeword of a cycle is also a codeword of the whole code. The proposed approach is hence to avoid low weight codewords by properly choosing the nonzero clusters implied in the cycles, so that no codeword of low-weight is contained in the cycles. This is achieved by ensuring that the binary matrices $\mathbf{Hc_k}$ corresponding to the cycles have full column rank. The iterative procedure that we use in this optimization step is essentially the same as the one depicted in [34]. In each step of the iterative procedure, we change the values of a limited number of non-zero clusters in order to maximize the number of cycle component codes $\mathbf{Hc_k}$ which are full rank. Thus, the matrix of a cycle should be full rank to cancel the cycle. Contrarily to classical non-binary LDPC codes for which the matrix of a cycle is squared, the matrix of a cycle of a hybrid LDPC code is rectangular, with more rows than columns. This means that we will have more degrees of freedom to cancel the cycles in hybrid LDPC codes. Hybrid LDPC codes are therefore well-suited to this kind of finite length optimization procedure.

2.5 Results

2.5.1 Rate one-half codes

Optimized distributions: thresholds and finite length performance

Table 2.2 : Nodewise distributions of the hybrid LDPC codes used for the finite length simulations.

	Hybrid LDPC code 1	Hybrid LDPC code 2
$\tilde{\Pi}(i=2, q_k=32)$		0.3950
$\tilde{\Pi}(i=2, q_k=64)$	0.4933	0.2050
$\tilde{\Pi}(i=2, q_k=256)$	0.4195	0.4000
$\tilde{\Pi}(i=6, q_k=64)$	0.0772	
$\tilde{\Pi}(i=6, q_k=256)$	0.0100	
$\tilde{\Pi}(j=5, q_l=256)$	0.5450	1
$\tilde{\Pi}(j=6, q_l=256)$	0.4550	
$\left(\frac{E_b}{N_o}\right)^{\star}$ (dB)	0.675	0.550

Based on the optimization methods presented in section 2.3.3, we first present some code distributions and corresponding thresholds for code rate one-half, as given in table 2.2. For all the presented results, the channel is the BIAWGN channel with BPSK modulation. Thresholds are computed by Monte-Carlo simulations. In table 2.2, $\left(\frac{E_b}{N_o}\right)^{\star}$ denotes the decoding convergence thresholds of the distributions in each column. The

hybrid LDPC code number 1 is obtained by the method presented in section 2.3.3, when setting the check node connection profile, all check nodes are in $G(256)$, putting all the redundancy variable nodes in $G(256)$ and information variables in $G(64)$. The connection profiles for these two groups are then optimized with $d_{v_{max}} = 10$. As already observed in section 2.3.2, variable nodes in the highest order group are affected with as much high connection degrees as possible, to balance the poor generalized component code. The hybrid LDPC code number 2 is obtained by the method presented in section 2.3.3, when setting the graph connections to be regular with constant variable degree $d_v = 2$ and constant check degree $d_c = 5$. Although these thresholds are not better than the one of a regular $(d_v = 2, d_c = 4)$ $GF(256)$ LDPC code, which is 0.5 dB [63], we can exhibit hybrid LDPC distributions with better thresholds than the one of a regular $(d_v = 2, d_c = 4)$ $GF(256)$ LDPC code, by allowing higher connection degrees. However, our purpose is to point out the good finite length performance of hybrid LDPC codes, and that is why we have focused on low connection degrees. For such low degrees, we are going to see that hybrid LDPC codes have very good finite length performances, but they do not approach the capacity as close as multi-edge type LDPC codes do. This is due to the adopted detailed representation Π which cannot handle degree one variable nodes. However, it would be an interesting perspective to switch from the detailed representation to a multi-edge type representation for LDPC codes. This will certainly enable to get capacity-approaching distributions with low connection degrees. Indeed, it has been shown in [30] that introducing degree-1 variable nodes in non-binary LDPC codes makes the decoding threshold getting closer to the theoretical limit. Modifying the representation of hybrid LDPC code ensemble is therefore very interesting for future work. We only present in table 2.2 the thresholds of the distributions which are used for the following finite length simulations.

Figure 2.5 represents some frame error rate (FER) curves for different codes, all with $K = 1024$ information bits and code rate one-half. Figure 2.5 shows the performance curves of hybrid LDPC codes number 1 and 2 compared with Quasi-cyclic Tanner codes from [1], irregular LDPC codes from [10], a $GF(256)$ LDPC code, a protograph based LDPC code from [26] and a multi-edge type LDPC code from [27] with code length $N = 2560$ bits ($K = 1280$ information bits). This code has been specially design for low error-floor. The graphs of the binary, non-binary and hybrid LDPC codes have been built with the random PEG algorithm described in [51].

We see that the hybrid LDPC code number 1 has performance very close to the protograph based LDPC code, while the hybrid LDPC code number 2 has better waterfall performance than the protograph based LDPC code but higher error floor. Also, the hybrid LDPC code number 2 has a worse waterfall region than a regular $(dv = 2, dc = 4)$ $GF(256)$ LDPC code, but a better error floor. These two observations are clues to investigate a finite length optimization of the hybrid LDPC code, in order to refine the structure of the graph to achieve better error floor performance.

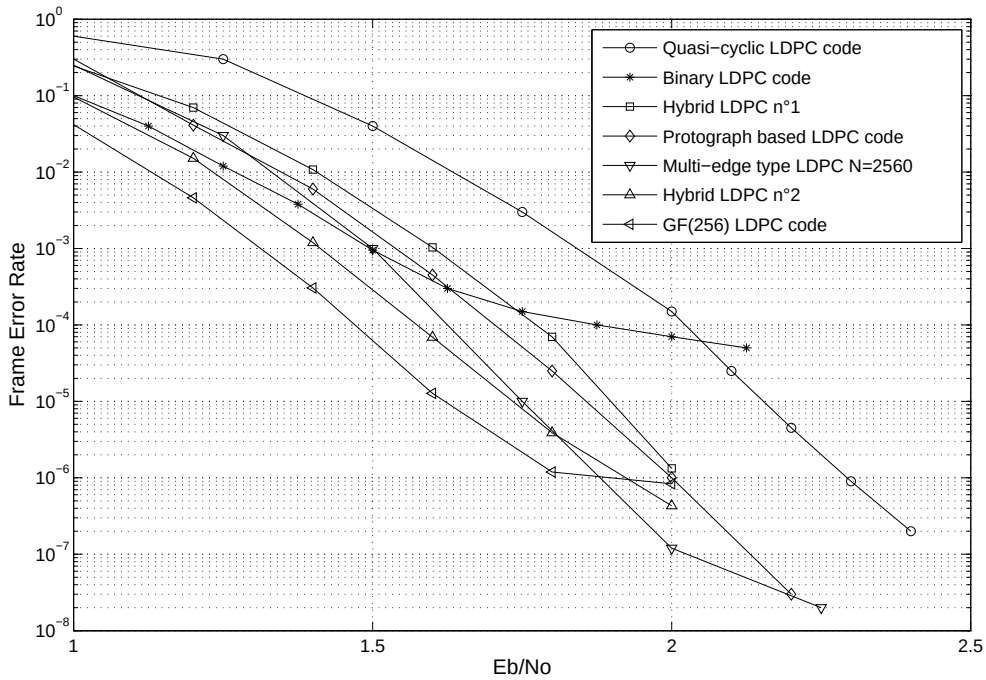


Figure 2.5 : FER versus $\frac{E_b}{N_o}$: code rate one-half. $K = 1024$ information bits except for the multi-edge type LDPC code for which $K = 1280$ information bits. No finite length optimization has been applied. $N_{iter} = 500$ except for quasi-cyclic LDPC code (from [1]) for which $N_{iter} = 50$.

Finite length optimized codes

The finite length optimization described in section 2.4 is applied to the hybrid LDPC code number 2, which has constant variable degree $d_v = 2$.

Figure 2.6 represents frame error rate (FER) curves for different codes with code rate one-half. The finite length optimization described in section 2.4 is applied to the hybrid LDPC code number 2, which has constant variable degree $d_v = 2$. The performance curves of hybrid LDPC codes 1 and 2 are compared with a protograph-based LDPC code from [26], and a multi-edge type (MET) LDPC code from [27]. This code has been specifically designed for low error-floor. All codes have $N_{bit} = 2048$ coded bits, except the MET LDPC code which has $N_{bit} = 2560$ coded bits. The graphs of hybrid LDPC codes have been built with the random PEG algorithm described in [51]. We see that the hybrid LDPC code 1 has performance very close to the protograph-based LDPC code. The hybrid LDPC code 2 has slightly better waterfall and slightly higher error-floor than the MET LDPC code, which is longer. Hybrid LDPC codes are therefore capable of exhibiting performance equivalent to MET LDPC codes, which are, to the best of our knowledge, among the most interesting structured codes. It is worthy to note that, unlike MET and protograph-based LDPC codes, the presented hybrid LDPC codes are non-structured codes.

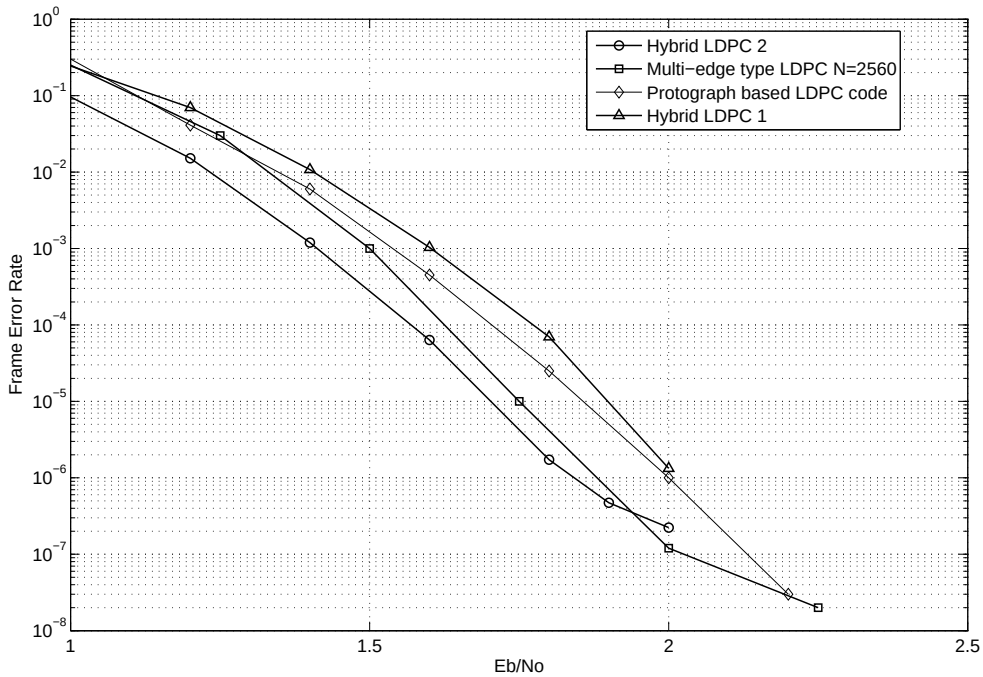


Figure 2.6 : FER versus $\frac{E_b}{N_o}$ (in dB): code rate one-half. $N_{bit} = 2048$ coded bits except for the multi-edge type LDPC code for which $N_{bit} = 2560$ coded bits. $N_{iter} = 500$ decoding iterations are performed.

Hence, hybrid LDPC codes can be a means to worsen the waterfall region of regular codes in the highest order field, in order to even lower the error-floor. They can be competitors for the best known codes for finite length performance.

2.5.2 Rate one-sixth codes

For communication systems operating in the low signal-to-noise ratio (SNR) regime (e.g., code-spread communication systems and power-limited sensor networks), low-rate coding schemes play a critical role. One important application of low-rate codes is in wide band data communications using code-division multiple-access (CDMA) systems [64], where they are used to replace the spreading code in traditional direct-sequence spread spectrum systems.

Although LDPC codes or Repeat-Accumulate (RA) codes can exhibit capacity-approaching performance for various code rates when the ensemble profiles are optimized [10], in the low-rate region, both RA and LDPC codes suffer from performance loss and extremely slow convergence using iterative decoding. To our knowledge, the most competitive codes at this time are Turbo-Hadamard (TH) [2] and various versions of Zigzag-Hadamard (ZH) codes [3]. All references of various low rate coding schemes can be found in [2][3][65]. We intend to illustrate the interest of hybrid LDPC codes for low-rate application requir-

ing short block length (from 200 to 1000 information bits).

The considered channel is still the BIAWGN channel. We compare the performance of our proposed hybrid LDPC codes with existing good codes related in [2][3]. K_{bit} is the number of information bits.

For a code rate $R = \frac{1}{6}$, a regular graph $(d_v = 2, d_c = 3)$ is considered, and the proportion of group orders has been optimized with EXIT charts techniques defined in section 2.3.3. With the order of the check nodes being set to $G(q_{max}) = G(256)$, the code resulting from the optimization has three different group orders $G(256) - G(16) - G(8)$ (table 2.3).

Table 2.3 : Nodewise distribution of the rate $\frac{1}{6}$ and $\frac{1}{12}$ hybrid LDPC codes

	Hybrid code $R = 1/6$	Hybrid code $R = 1/12$
$\tilde{\Pi}(i = 2, q_k = 2)$		0.184
$\tilde{\Pi}(i = 2, q_k = 4)$		0.150
$\tilde{\Pi}(i = 2, q_k = 8)$	0.227	
$\tilde{\Pi}(i = 2, q_k = 16)$	0.106	
$\tilde{\Pi}(i = 2, q_k = 256)$	0.667	0.667
$\tilde{\Pi}(j = 3, q_l = 256)$	1	1
$\left(\frac{E_b}{N_o}\right)^{\star}$ (dB)	-0.41	-0.59
Capacity (dB)	-1.08	-1.33

On figure 2.7, for $K_{bit} \simeq 200$, the hybrid LDPC code of code rate $1/6$ outperforms with 0.3 dB gain the ZH code of code rate $1/6$. Additionally, our hybrid code has no observed error floor up to a BER=10^{-7}. When comparing the computer simulation of the hybrid LDPC code with the union bound of ZH code, we observe that the BER of the hybrid LDPC code has gain of about one decade at $E_b/N_0 = 2$dB. Since union bounds are tight upper bounds on BER performances [2] for Turbo-Hadamard codes, we can predict from the figure that the error floors of our two simulated codes will be lower than the error floors of Turbo-Hadamard codes with random interleaver. Indeed, the minimum distance of our hybrid LDPC code has been estimated thanks to the impulse method [66] and is upper bounded by $d_{min} = 80$, which is by far superior to the minimum distance that can be achieved with TH or ZH codes.

The hybrid LDPC code of code rate $R = 1/12 = 0.083$ has poorer performance in the waterfall region than TH and ZH codes with comparable rates, but has much lower error floor when comparing the computer simulations to the union bound of the code rate 0.077 TH code. Indeed, its minimum distance is upper bounded by $d_{min} = 125$. Hence, although this $R = 0.083$ code suffers from 0.1 to 0.2 dB loss compared with the rate 0.077 TH code, the good error floor properties highlight the interest of hybrid LDPC codes for lower rates. As aforementioned, we can expect that introducing degree-1 variable nodes in hybrid LDPC code will allow to get thresholds closer to the capacity for very low code

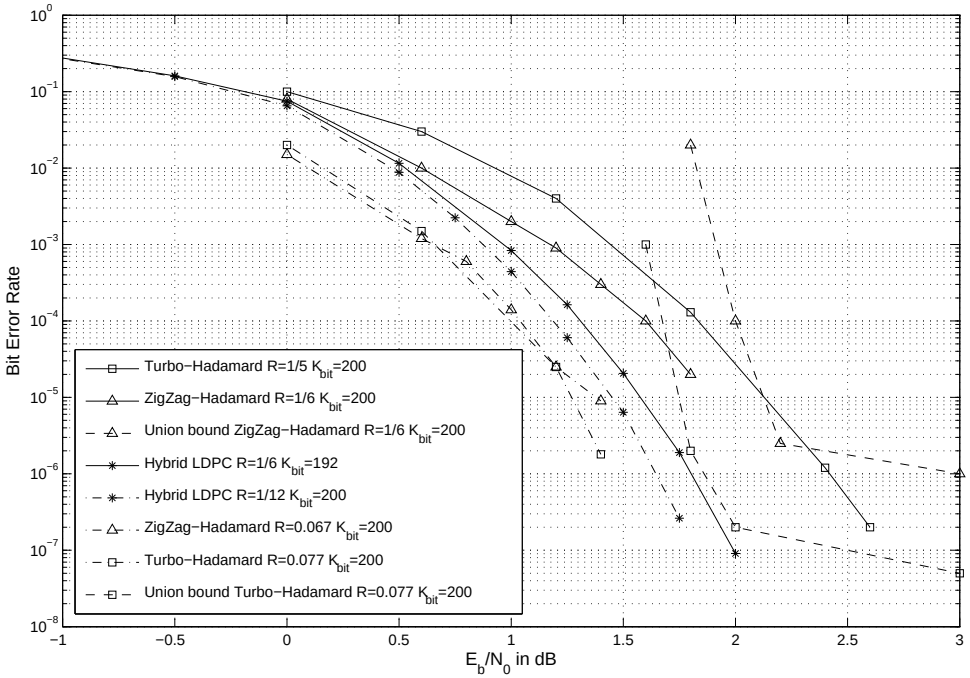


Figure 2.7 : Comparison of hybrid LDPC code with Turbo Hadamard codes (TH) taken from [2] and Zigzag Hadamard (ZH) codes taken from [3], for an information block length of $K_{bit} \simeq 200$. $N_{iter} = 30$ for Turbo Hadamard codes, and $N_{iter} = 200$ for the hybrid LDPC codes.

rate (less than $\frac{1}{10}$). Indeed, this seems to be important to have good thresholds with low rates [27].

In Figure 2.8, the FER comparison is drawn for code rate $1/6$ and $K_{bit} \simeq 1000$ information bits. The quasi-cyclic LDPC code is designed to have low error-floor [1]. The hybrid LDPC code is better that the quasi-cyclic LDPC and PTH codes, both in the waterfall and in the error-floor regions. The hybrid LDPC code has poorer waterfall region than the MET LDPC code [67], but better error-floor. Hence, for rate $1/6$ too, the performance of hybrid LDPC codes are equivalent to the one of MET LDPC codes, by allowing to reach comparable trade-off between waterfall and error-floor performance.

Remark: Let us mention that hybrid LDPC codes, with injective linear maps as non-zero elements, are well-fitted to low code rates thanks to their structure. Indeed, like all other kinds of codes with generalized constraint nodes (Turbo Hadamard code [2], LDPC Hadamard codes [68], GLDPC [28], D-GLDPC [29], or Tail-biting LDPC [30]), they are well-fitted to low code rates because the graph rate is higher than the code rate. This can help the iterative decoding: when the code rate is very low, decoding on a higher rate graph can lead to better performance.

It is worthy to note that the better performance of hybrid LDPC codes over codes based

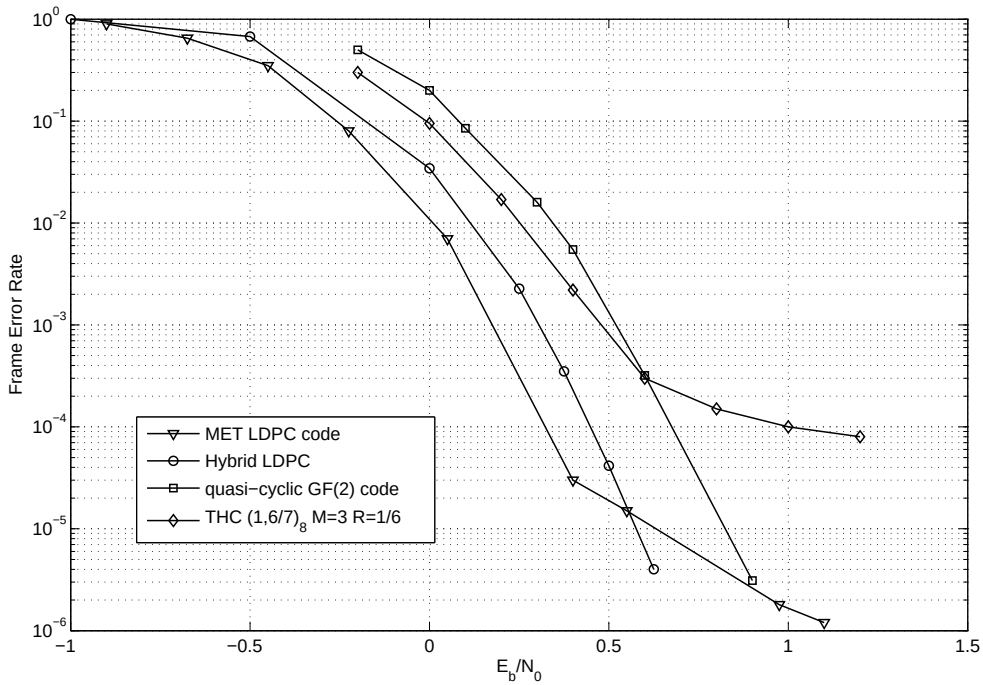


Figure 2.8 : Comparison of hybrid LDPC code with punctured Turbo Hadamard (PTH) taken from [4] and other powerful codes, for code rate $1/6$. The PTH code has $K_{bit} = 999$ information bits, and the other codes have $K_{bit} = 1024$ information bits. $N_{iter} = 50$ for the PTH code, and $N_{iter} = 200$ for the other codes.

on Hadamard codes are obtained with no complexity increase. Indeed, the complexity of these codes is dominated by the complexity of the fast Hadamard transform, which is $O(r \cdot 2^r)$ [2], where r is the order of the Hadamard code. The complexity of hybrid LDPC codes is dominated by the fast Fourier transform at check nodes $O(q \log(q))$, where q is the maximum group order. The complexity of Hadamard type codes and hybrid LDPC codes is therefore equivalent. However, contrary to TH codes, one should note that hybrid LDPC codes are suitable for decoding with reduced complexity and no loss, as described in [46].

2.6 Conclusions

In this work, asymptotic analysis of a new class of non-binary LDPC codes, named hybrid LDPC codes, has been carried out. Specific properties of considered hybrid LDPC code ensembles, like the Linear-Map invariance, have been studied to be able to derive both stability condition and EXIT charts. The stability condition of such hybrid LDPC ensembles shows interesting advantages over non-binary codes. Study of the condition allows to conclude that there exist many cases where any fixed point of density evolution for hybrid LDPC codes can be stable at lower SNR than for non-binary codes. The EXIT charts

analysis is performed on the BIAWGN channel, whereas studies of generalized LDPC codes usually consider the BEC [30, 29]. In order to optimize the distributions of hybrid LDPC ensembles, we have investigated how to project the message densities on only one scalar parameter using a Gaussian approximation. The accuracy of such an approximation has been studied, and used to lead to two kinds of EXIT charts of hybrid LDPC codes: multi-dimensional and mono-dimensional EXIT charts. Distribution optimization allows to get finite length codes with very low connection degrees and better waterfall region than protograph or multi-edge type LDPC codes. Moreover, hybrid LDPC codes are well fitted for the cycle cancellation presented in [34], thanks to the specific structure of the linear maps. The resulting codes appear to have, additionally to a better waterfall region, a very low error-floor for code rate one-half and codeword length lower than three thousands bits, thereby competing with multi-edge type LDPC. Thus, hybrid LDPC codes allow to achieve an interesting trade-off between good error-floor performance and good waterfall region with non-binary codes techniques.

We have also shown that hybrid LDPC codes can be very good candidates for efficient low rate coding schemes. For code rate one sixth, they compare very well to existing Turbo Hadamard or Zigzag Hadamard codes. In particular, hybrid LDPC codes exhibit very good minimum distances and error floor properties.

As future work, it would be of first interest to allow degree one variable nodes in the representation of hybrid LDPC codes, by, e.g., adopting a multi-edge type representation [27]. As shown in [30], this would allow to have better decoding thresholds, in particular for low rate codes.
This would give rise to the study and optimization, with the same tools, of non-binary protograph based or multi-edge type LDPC codes. However, the extension may be theoretically not completely straightforward as the non-zero values have to be carefully handled to define the code ensemble.

On the other hand, it would be interesting to study hybrid LDPC codes on other channels. Let us mention that we made some experiments on an AWGN channel with 16-QAM modulation. We restricted the connection profile to be regular, in order to not bias the results by the absence of special allocation on differently protected symbols. Only two group orders where allowed to avoid correlation between channel LLRs: $G(16)$ and $G(256)$. The optimization of fractions of variable nodes in these two different orders have been done. The results where slightly degraded compared to a $(2,4)$ $GF(256)$ LDPC codes. A study of these codes on the BEC would be also interesting, according to what has been done for D-GLDPC codes on the BEC [56].

2.7 Proofs of theorems in Chapter 2

Lemma 5 *Let $P_e^{(t)}(\mathbf{x})$ denote the conditional error probability after the tth BP decoding iteration of a $GF(q)$ LDPC code, assuming that codeword $\mathbf{x}$ was sent. If the channel is symmetric, then $P_e^{(t)}(\mathbf{x})$ is independent of $\mathbf{x}$.*

Proof: The proof has the same structure as the proof of Lemma 1 in [11]. The notations are the same as in [11] and Section 2.1.7.
Let $\Psi_v^{(t)}(\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_{d_v-1})$ denote the message map of any variable node at iteration t, according to equation (2.4). The size of argument messages is implicitly the one of the group of the variable node. Let $\Psi_c^{(t)}(\mathbf{m}_1, \ldots, \mathbf{m}_{d_c-1})$ be the message map of any check node. The sizes of argument messages are implicitly the one of the group of each variable node connected to the check node, according to equation (2.5).

- Check node symmetry: Let G be the Cartesian product group defined in Section 2.1.7. For any sequence $(b_1, \ldots, b_{d_c-1})$ in G such that $\bigoplus_{i=1}^{d_c-1} A_{v_i c} b_i \in Im(A_{vc})$, we have (see equation (2.5))

$$\Psi_c^{(t)}(\mathbf{m}_1^{+b_1}, \ldots, \mathbf{m}_{d_c-1}^{+b_{d_c-1}}) = \Psi_c^{(t)}(\mathbf{m}_1, \ldots, \mathbf{m}_{d_c-1})^{+A_{vc}^{-1}(\bigoplus_{i=1}^{d_c-1} A_{v_i c} b_i)}$$

- Variable node symmetry: We also have, for any $b \in GF(q_v)$:

$$\Psi_v^{(t)}(\mathbf{m}_0^{+b}, \mathbf{m}_1^{+b}, \ldots, \mathbf{m}_{d_v-1}^{+b}) = \Psi_v^{(t)}(\mathbf{m}_1, \ldots, \mathbf{m}_{d_c-1})^{+b}$$

Let $\mathbf{Z}_i$ denote the random variable being the channel output in probability form, conditionally to the transmission of the zero symbol. Each $\mathbf{Z}_i$ for any $i = 1 \ldots N$ has same size as the group of the corresponding codeword symbol. Any memoryless symmetric channel can be modeled as

$$\mathbf{Y}_i = \mathbf{Z}_i^{+x_i}$$

where x_i is the ith component of $\mathbf{x}$ which is a vector of size N, denoting an arbitrary codeword of the hybrid LDPC code. The channel output in probability form $\mathbf{Y}_i$ results from the transmission of $\mathbf{x}$.

Let v denote an arbitrary variable node and let c denote one of its neighboring check nodes. For any observation in probability form $\mathbf{w}$, let $\mathbf{m}_{vc}^{(t)}(\mathbf{w})$ denote the message sent from v to c in iteration t assuming $\mathbf{w}$ was received. The quantity $\mathbf{w}$ is hence a set of channel output vectors in probability form $\mathbf{w}_i$, for all $i = 1 \ldots N$. The same definition holds for $\mathbf{m}_{cv}^{(t)}(\mathbf{w})$ from c to v. From the variable node symmetry at $t = 0$ we have $\mathbf{m}_{vc}^{(0)}(\mathbf{y}) = \mathbf{m}_{vc}^{(0)}(\mathbf{z})^{+x_v}$. Assuming now that in iteration t we have $\mathbf{m}_{vc}^{(t)}(\mathbf{y}) = \mathbf{m}_{vc}^{(t)}(\mathbf{z})^{+x_v}$. Since $\mathbf{x}$ is a codeword, we have $\bigoplus_{i=1}^{d_c} A_{v_i c} x_i = 0$, and hence $\bigoplus_{i=1}^{d_c-1} A_{v_i c} x_i = A_{vc} x_v$. From the check node symmetry condition we conclude that

$$\mathbf{m}_{cv}^{(t+1)}(\mathbf{y}) = \mathbf{m}_{cv}^{(t+1)}(\mathbf{z})^{+x_v} \ .$$

Moreover, from the variable node symmetry condition, it follows that in iteration $t+1$ the message sent from v to c is

$$\mathbf{m}_{vc}^{(t+1)}(\mathbf{y}) = \mathbf{m}_{vc}^{(t+1)}(\mathbf{z})^{+x_v} \ .$$

Thus, all messages to and from variable node v when $\mathbf{y}$ is received are permutations by x_v of the corresponding message when $\mathbf{z}$ is received. Hence, both decoders commit exactly the same number of errors, which proves the lemma.

□

2.7.1 Symmetry

Lemma 6. *If* $\mathbf{W}$ *is a symmetric LDR random vector, then its extension* $\mathbf{W}^{\times A}$*, for any linear map* A *selected from* $E_{1,2}$*, is also symmetric. The truncation of* $\mathbf{W}$ *by the inverse of* A*, denoted by* $\mathbf{W}^{\times A^{-1}}$*, is also symmetric.*

Proof: We first prove that any q_2-sized extension of a q_1-sized symmetric random vector remains symmetric. We want to show that

$$\forall b \in [0, q_2 - 1], P(\mathbf{W}^{\times A} = \mathbf{w}) = e^{w_b} P(\mathbf{W}^{\times A} = \mathbf{w}^{+b}) \tag{2.26}$$

Case $b \notin Im(A)$:

- In the case when $w_b \neq -\infty$:
  We have to show that

  $$e^{-w_b} P(\mathbf{W}^{\times A} = \mathbf{w}) = P(\mathbf{W}^{\times A} = \mathbf{w}^{+b})$$

  If $w_b \neq \infty$, then $P(\mathbf{W}^{\times A} = \mathbf{w}) = 0$. If $w_b = \infty$, then $e^{-w_b} = 0$. Thus, we have to show that

  $$\forall b \notin Im(A), P(\mathbf{W}^{\times A} = \mathbf{w}^{+b}) = 0 \tag{2.27}$$

  This is equivalent to show that $\exists i \notin Im(A)$ such that $w_i^{+b} \neq \infty$. We have $w_i^{+b} = w_{b+i} - w_b$. It is sufficient to choose $i = b$, then $w_b^{+b} = -w_b$. Since $w_b^{+b} = -w_b \neq \infty$ by hypothesis, $P(\mathbf{W}^{\times A} = \mathbf{w}^{+b}) = 0$.

- In the case $w_b = -\infty$, to prove that equation (2.26) is fulfilled we have to prove that $P(\mathbf{W}^{\times A} = \mathbf{w}) = 0$, which is straight forward because $b \notin Im(A)$, and hence $P(\mathbf{W}^{\times A} = \mathbf{w}) \neq 0 \Rightarrow w_b = \infty$. By taking the contraposition, we end on the wanted result.

Hence we have proved equation (2.26) in the case where $b \notin Im(A)$.
Case $b \in Im(A)$:
We have

$$P(\mathbf{W}^{\times A} = \mathbf{w}) = P(\mathbf{W} = \mathbf{w}^{\times A^{-1}}) \Pi_{i \notin \mathrm{Im}(A)} \delta_{w_i, \infty}$$

Since b belongs to $Im(A)$, we denote by a the element in $[0, q_1 - 1]$ such that $b = Aa$. The input message $\mathbf{W}$ is symmetric, hence we have

$$P(\mathbf{W} = \mathbf{w}^{\times A^{-1}}) = e^{w_{Aa}} P(\mathbf{W} = (\mathbf{w}^{\times A^{-1}})^{+a})$$

$$\begin{aligned}
\forall i \in [0, q_1 - 1], \quad (\mathbf{w}^{\times A^{-1}})_i^{+a} &= w_{i+a}^{\times A^{-1}} - w_a^{\times A^{-1}} \\
&= w_{A(i+a)} - w_{Aa} \\
&= w_{Ai}^{+Aa} \\
&= (w^{+Aa})_i^{\times A^{-1}}
\end{aligned}$$

Thus

$$P(\mathbf{W}^{\times A} = \mathbf{w}) = e^{w_{Aa}} P(\mathbf{W} = (\mathbf{w}^{+Aa})^{\times A^{-1}}) \Pi_{i \notin \text{Im}(A)} \delta_{w_i,\infty} \tag{2.28}$$

But we note that:

$$P(\mathbf{W}^{\times A} = \mathbf{w}^{+Aa}) = P(\mathbf{W} = (w^{+Aa})^{\times A^{-1}}) \Pi_{j \notin \text{Im}(A)} \delta_{w_j^{+Aa},\infty} \tag{2.29}$$

- We first examine the case $w_{Aa} = \infty$:

  $$P(\mathbf{W}^{\times A} = \mathbf{w}^{+Aa}) \neq 0 \Rightarrow \forall i \notin Im(A), \quad w_i^{+Aa} = \infty$$

  But, if $\mathbf{y} = LDR^{-1}(\mathbf{w})$, $w_i^{+Aa} = \log\left(\frac{y_{Aa}}{y_{Aa+i}}\right)$, and since $y_{Aa} = 0$ because $w_{Aa} = \infty$, we cannot have $w_i^{+Aa} = \infty$, $\forall i \notin Im(A)$. Hence we have $w_{Aa} = \infty \Rightarrow P(\mathbf{W}^{\times A} = \mathbf{w}^{+Aa}) = 0$. This proof by contradiction ensures that equation (2.26) is fulfilled when $w_{Aa} = \infty$.

- Then we examine the case $w_{Aa} = -\infty$:

  $$P(\mathbf{W}^{\times A} = \mathbf{w}) \neq 0 \Rightarrow \forall i \notin Im(A), \quad w_i = \infty$$

  But $w_{Aa} = \log\left(\frac{y_0}{y_{Aa}}\right) = -\infty$ implies that $y_0 = 0$. Hence we cannot have $w_i = \log\left(\frac{y_0}{y_i}\right)$ for all $i \in [0, q_2 - 1]$. Hence we have $w_{Aa} = -\infty \Rightarrow P(\mathbf{W}^{\times A} = \mathbf{w}) = 0$. This proof by contradiction ensures that equation (2.26) is fulfilled when $w_{Aa} = -\infty$.

- Finally we examine the case $w_{Aa} \notin \{-\infty, \infty\}$:
  In this case, for all $j \in [0, q_2 - 1]$, $\delta_{w_j^{+Aa},\infty} = \delta_{w_{Aa+j} - w_{Aa},\infty} = \delta_{w_{Aa+j},\infty}$. For all $i \in [0, q_2 - 1]$, if $i \notin Im(A)$, then $\exists j \notin Im(A)$: $i = Aa + j$. Therefore $\{i \in [0, q_2 - 1] s.t. i \notin Im(A)\} = \{j \in [0, q_2 - 1] s.t. Aa + j \notin Im(A)\}$. We finally obtain:

  $$\Pi_{j \notin \text{Im}(A)} \delta_{w_j^{+Aa},\infty} = \Pi_{i \notin \text{Im}(A)} \delta_{w_i,\infty}$$

  The above equality allows to insert equation (2.29) into equation (2.28). We can now conclude that, when $w_{Aa} \notin \{-\infty, \infty\}$, equation (2.26) is satisfied.

This completes the proof of the first part of lemma 6.

We now prove that any truncation of a symmetric random vector remains symmetric. We have to prove that

$$\forall a \in [0, q_1 - 1], \quad P(\mathbf{W}^{\times A^{-1}} = \mathbf{w}) = e^{w_a} P(\mathbf{W}^{\times A^{-1}} = \mathbf{w}^{+a}) \tag{2.30}$$

Let call $b = Aa$.

$$\begin{aligned} P(\mathbf{W}^{\times A^{-1}} = \mathbf{w}) &= \sum_{\mathbf{x}:x_0=w_0,x_{A1}=w_1,\ldots,x_{A(q_1-1)}=w_{q_1-1}} P(\mathbf{W} = \mathbf{x}) \\ &= \sum_{\mathbf{x}:x_0=w_0,x_{A1}=w_1,\ldots,x_{A(q_1-1)}=w_{q_1-1}} e^{x_b} P(\mathbf{W} = \mathbf{x}^{+b}) \\ &= \sum_{\mathbf{x}:x_0=w_0,x_{A1}=w_1,\ldots,x_{A(q_1-1)}=w_{q_1-1}} e^{-w_a} P(\mathbf{W} = \mathbf{x}^{+b}) \\ &= e^{-w_a} \sum_{\mathbf{x}:x_0=w_0,x_{A1}=w_1,\ldots,x_{A(q_1-1)}=w_{q_1-1}} P(\mathbf{W} = \mathbf{x}^{+b}) \\ &= e^{-w_a} P(\mathbf{W}^{\times A^{-1}} = \mathbf{w}^{+a}) \end{aligned} \tag{2.31}$$

The last step is obtained by noting that:

$$\forall i \in Im(A), \quad x_i^{+Aa} = x_{Aa+i} - x_{Aa} = w_{a+A^{-1}i} - w_a = (w^{+b})_i^{\times A^{-1}}$$

We have obtained equation (2.30).

□

Please note that, in the sequel of this chapter, for all the proofs, we simplify the notations as follows: For all group $G(q)$, for all $i \in [0, q-1]$, the element α_i is now denoted by i. Also, since A is a linear map, the matrix of the application is also denoted by A. Hence, for all linear map A from $G(q_1)$ to $G(q_2)$, $A(\alpha_i) = \alpha_j$ with $\alpha_i \in G(q_1)$ and $\alpha_j \in G(q_2)$, is translated by $Ai = j$.

2.7.2 A useful lemma

Lemma 10 *$E_{k,l}$ denotes the set of extensions from $G(q_k)$ to $G(q_l)$. For given k and l,*

$$\forall (i,j) \in [1, q_k - 1] \times [1, q_l - 1], \quad \frac{\textit{Card}(A \in E_{k,l} : A^{-1}j = i)}{\textit{Card}(E_{k,l})} = \frac{1}{q_l - 1}$$

Proof: p_k and p_l denote $\log_2(q_k)$ and $\log_2(q_l)$, respectively.
Without any constraint to build a linear extension A from $G(q_k)$ to $G(q_l)$, except the one of full-rank, we have $2^{p_l} - 2^{n-1}$ choices for the nth row, $n = 1, \ldots, p_l$.
For given i and j, with the constraint that $Ai = j$, we have $2^{p_l - b_i} + 2^{\lfloor \frac{b_i}{2} \rfloor} - 2^{n-1}$ choices for the nth row, $n = 1, \ldots, p_l$, where b_i is the number of bits equal to 1 in the binary map of α_i. Thus, the number of A such that $Ai = j$ is dependent only on i. Let say

$$\text{Card}(A \in E_{k,l} : A^{-1}j = i) = \beta_i$$

we have

$$\sum_{j=1}^{q_l-1} \text{Card}(A \in E_{k,l} : Ai = j) = \text{Card}(E_{k,l})$$

Therefore

$$\forall (i,j) \in [1, q_k - 1] \times [1, q_l - 1], \qquad \frac{\text{Card}(A \in E_{k,l} : Ai = j)}{\text{Card}(E_{k,l})} = \frac{1}{q_l - 1}$$

2.7.3 LM-invariance

Lemma 7. *If a probability-vector random variable* $\mathbf{Y}$ *of size* q_2 *is LM-invariant, then for all* $(i,j) \in [0, q_2 - 1] \times [0, q_2 - 1]$*, the random variables* Y_i *and* Y_j *are identically distributed.*

Proof: For any (q_1, q_2), $q_1 < q_2$, $T_{1,2}$ denotes the set of all truncations from $G(q_2)$ to $G(q_1)$. We assume $\mathbf{Y}$ LM-invariant. A^{-1} and B^{-1} denote two truncations independently arbitrary chosen in $T_{1,2}$. For any l and k in $[0, q_2 - 1]$, we can choose extension A such that $l \in$ Im(A) and $A^{-1}l$ is denoted by i. Also, we choose B such that $Bi = k$. $\mathbf{Y}$ LM-invariant implies

$$\forall (i, A^{-1}, B^{-1}) \in [0, q_1 - 1] \times T_{1,2} \times T_{1,2}, P(Y_i^{\times A^{-1}} = x) = P(Y_i^{\times B^{-1}} = x)$$

This is equivalent to

$$P(\mathbf{Y}_{Ai} = x) = P(\mathbf{Y}_{Bi} = x)$$

and hence

$$P(\mathbf{Y}_l = x) = P(\mathbf{Y}_k = x), \qquad \forall (l,k) \in [0, q_2 - 1] \times [0, q_2 - 1]$$

□

Lemma 8. *A probability-vector random variable* $\mathbf{Y}$ *of size* q_2 *is LM-invariant if and only if there exist* q_1 *and a probability-vector random variable* $\mathbf{X}$ *of size* q_1 *such that* $\mathbf{Y} = \tilde{\mathbf{X}}$.

Proof: Let us first assume $\mathbf{Y} = \tilde{\mathbf{X}}$ and prove that $\mathbf{Y}$ is LM-invariant. This means that we want to prove that for any $(B, C) \in E_{1,2} \times E_{1,2}$, $Y^{\times B^{-1}}$ and $Y^{\times C^{-1}}$ are identically distributed.
By hypothesis $\mathbf{Y} = \mathbf{X}^{\times A}$, with A uniformly chosen in $E_{1,2}$. We define the matrix α_A of size $q_2 \times q_1$. This matrix is such that $\mathbf{Y} = \alpha_A \mathbf{X}$ and is defined by

$$\begin{aligned} \forall j = 0 \ldots q_1 - 1, \forall i = 0 \ldots q_2 - 1, \quad \alpha_A(i,j) &= 1 \quad \text{if i=Aj} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Thus, vector $\mathbf{Y}$ truncated by any linear map B is expressed by:

$$\mathbf{Y}^{\times B^{-1}} = \alpha_B^T \alpha_A \mathbf{X}$$

The same holds for linear map C:

$$\mathbf{Y}^{\times C^{-1}} = \alpha_C^T \alpha_A \mathbf{X}$$

$\alpha_B^T\alpha_A$ and $\alpha_C^T\alpha_A$ correspond to a selection of q_1 rows of α_A. Thus, showing that $\mathbf{Y}^{\times B^{-1}}$ and $\mathbf{Y}^{\times C^{-1}}$ are identically distributed is equivalent to show that both matrices $\alpha_B^T\alpha_A$ and $\alpha_C^T\alpha_A$ are identically distributed, for any B and C in $E_{1,2}$ and for A uniformly chosen in $E_{1,2}$. The number of elements of $\mathbf{X}$, whose indexes are in $Im(A)$ and which are selected by α_B^T, is equal to the cardinality of $Im(A) \cap Im(B)$. The same holds for C.
$\mathbb{E}_A(f(A,B))$ denotes the expectation of the function f applied to random variables A and B, over all the realizations of A.
Let us first show that

$$\mathbb{E}_A\left(Card(Im(B) \cap Im(A))\right) = \mathbb{E}_A\left(Card(Im(C) \cap Im(A))\right), \quad \forall (B,C) \in E_{1,2} \times E_{1,2}, A \sim \mathbb{U}_{E_{1,2}} \tag{2.32}$$

$$\begin{aligned}\mathbb{E}_A\left(Card(Im(B) \cap Im(A))\right) &= \frac{1}{Card(E_{1,2})} \sum_{A \in E_{1,2}} Card(Im(B) \cap Im(A)) \\ &= \frac{1}{Card(E_{1,2})} \sum_{r=1}^{q_1} r \cdot Card\left(A \in E_{1,2} : Card(Im(B) \cap Im(A)) = r\right)\end{aligned}$$

$$= \frac{1}{Card(E_{1,2})} \sum_{r=1}^{q_1} r \cdot \binom{q_1}{r} \sum_{\substack{i_1 \neq \cdots \neq i_r \\ \in G(q_1)}} Card\left(A \in E_{1,2} : Ai_1 = j_1, \ldots, Ai_r = j_r\right)$$

where $j_1 \ldots j_r$ are subsets of $Im(B)$.
In the same way as for lemma 10, we can show that $Card\left(A \in E_{1,2} : Ai_1 = j_1, \ldots, Ai_r = j_r\right)$ is independent of $j_1 \ldots j_r$. Hence we conclude on equality (2.32).
Let us now consider a given subset $j_1 \ldots j_r$ of size r, taken from the image of any linear map in $E_{1,2}$ (hence with $r \leq q_1$), and a given subset $i_1 \ldots i_r$ of $G(q_1)$ of size r. In the same way as lemma 10, we can prove that $Card\left(A \in E_{1,2} : Ai_1 = j_1, \ldots, Ai_r = j_r\right)$ is independent of $j_1 \ldots j_r$.
The first part of the proof ensures each row, of both matrices $\alpha_B^T\alpha_A$ and $\alpha_C^T\alpha_A$, to have the same probability to contain a 1 (they have at most one 1). The second part of the proof ensures that, given r rows of α_A of indexes $j_1, \ldots, j_r$, the combination of locations of ones in the matrix $\alpha_B^T\alpha_A$ is independent of which rows $j_1, \ldots, j_r$ of α_A have been selected by α_B^T. Hence, this combination is independent of α_B^T.

For any $(B,C) \in E_{1,2} \times E_{1,2}$, for A uniformly distributed in $E_{1,2}$, both matrices $\alpha_B^T\alpha_A$ and $\alpha_C^T\alpha_A$ are therefore identically distributed. Since $\mathbf{Y}^{\times B^{-1}} = \alpha_B^T\alpha_A\mathbf{X}$ and $\mathbf{Y}^{\times C^{-1}} = \alpha_C^T\alpha_A\mathbf{X}$, $\mathbf{Y}^{\times B^{-1}}$ and $\mathbf{Y}^{\times C^{-1}}$ are identically distributed for any $(B,C) \in E_{1,2} \times E_{1,2}$, that means that $\mathbf{Y}$ is LM-invariant.

Let us now assume $\mathbf{Y}$ LM-invariant, and define $\mathbf{X}$ by $\mathbf{X} = \mathbf{Y}^{\times A^{-1}}$ with A uniformly chosen in $E_{1,2}$ and independent of $\mathbf{Y}$. We have to show that $\mathbf{X}$ is independent of A.

$$P(\mathbf{X} = \mathbf{x}|A) = P(\mathbf{Y}^{\times A^{-1}} = \mathbf{x}|A)$$

We can write, thanks to definition 8, for all B arbitrary selected from $E_{1,2}$ independently on A,

$$P(\mathbf{Y}^{\times A^{-1}} = \mathbf{x}|A) = P(\mathbf{Y}^{\times B^{-1}} = \mathbf{x}|A) = P(\mathbf{Y}^{\times B^{-1}} = \mathbf{x})$$

We finally obtain

$$\begin{aligned} P(\mathbf{X}=\mathbf{x}|A) &= P(\mathbf{Y}^{\times B^{-1}}=\mathbf{x}) \\ &= P(\mathbf{Y}^{\times A^{-1}}=\mathbf{x}) \\ &= P(\mathbf{X}=\mathbf{x}) \end{aligned}$$

This completes the proof.

□

Lemma 11 *The product of two LM-invariant random probability-vectors is LM-invariant.*

Proof: Let $\mathbf{U}$ and $\mathbf{V}$ be two LM-invariant random LDR-vectors of size q_2. Let A and B be any two linear maps from $G(q_1)$ to $G(q_2)$. Since $\mathbf{U}$ is LM-invariant, $\mathbf{U}^{\times A^{-1}}$ and $\mathbf{U}^{\times B^{-1}}$ are identically distributed, by definition of LM-invariance. The same holds for $\mathbf{V}$. $\mathbf{U}^{\times A^{-1}}\mathbf{V}^{\times A^{-1}}$ and $\mathbf{U}^{\times B^{-1}}\mathbf{V}^{\times B^{-1}}$ are therefore identically distributed. Moreover, it is clear that $\mathbf{U}^{\times A^{-1}}\mathbf{V}^{\times A^{-1}} = \mathbf{U}\mathbf{V}^{\times A^{-1}}$, for any A. Hence, $\mathbf{U}\mathbf{V}^{\times A^{-1}}$ and $\mathbf{U}\mathbf{V}^{\times B^{-1}}$ is LM-invariant. This completes the proof.

□

2.7.4 Proof of Theorem 3

$\mathbf{X}^{(k)}$ denotes a probability-vector random variable of size q_k. The jth component of the random truncation of $\mathbf{X}^{(k)}$ is denoted by $\frac{\mathrm{rt}}{X_j^{(k)}}$. The jth component of the random extension of $\mathbf{X}^{(k)}$ is denoted by $\frac{\mathrm{re}}{X_j^{(k)}}$. The jth component of the random extension followed by a random truncation of $\mathbf{X}^{(k)}$ is denoted by $\frac{\mathrm{rt+re}}{X_j^{(k)}}$.

We define the operator D_a by:

$$D_a(\mathbf{X}^{(l)}) = \frac{1}{q_l-1}\sum_{j=1}^{q_l-1}\mathbb{E}\left(\sqrt{\frac{X_j^{(l)}}{X_0^{(l)}}}\right)$$

The following equalities are hence deduced from the previous definitions:

$$\begin{aligned} \mathbb{E}\left(\sqrt{\frac{\frac{\mathrm{re}}{X_j^{(k)}}}{\frac{\mathrm{re}}{X_0^{(k)}}}}\right) &= \sum_l \Pi(l|k)\frac{1}{q_l-1}\sum_{i=1}^{q_k-1}\mathbb{E}\left(\sqrt{\frac{X_i^{(k)}}{X_0^{(k)}}}\right) \\ \mathbb{E}\left(\sqrt{\frac{\frac{\mathrm{rt}}{X_i^{(l)}}}{\frac{\mathrm{rt}}{X_0^{(l)}}}}\right) &= \frac{1}{q_l-1}\sum_{j=1}^{q_l-1}\mathbb{E}\left(\sqrt{\frac{X_j^{(l)}}{X_0^{(l)}}}\right) \\ &= D_a(\mathbf{X}^{(l)}) \\ \mathbb{E}\left(\sqrt{\frac{\frac{\mathrm{rt+re}}{X_i^{(k)}}}{\frac{\mathrm{rt+re}}{X_0^{(k)}}}}\right) &= \sum_l \Pi(l|k)\frac{1}{q_l-1}\sum_{i=1}^{q_k-1}\mathbb{E}\left(\sqrt{\frac{X_i^{(k)}}{X_0^{(k)}}}\right) \end{aligned}$$

To shorten the notations we can omit the index of iteration t. Moreover, in the remainder of this proof, we choose to use simpler notations although not fully rigorous: $\mathbf{R}^{(j,l)}$ denotes a message going into a check node of degree j in $G(q_l)$ while $\mathbf{R}^{(i,k)}$ denotes a message going out of a variable of degree i in $G(q_k)$. However, there is not ambiguity in the following thanks to the unique use of indexes i, j, k, l and we always precise of which nature is a message.
The nth component of a message coming from a variable of degree i in $G(q_k)$ is denoted by $R_n^{(i,k)}$. The nth component of the initial message going into a variable in $G(q_k)$ is denoted by $R_n^{(0)^{(k)}}$. The nth component of a message going into a degree i variable in $G(q_k)$ is denoted by $L_n^{(i,k)}$. The data pass, through a variable node of degree n in $G(q_k)$, is translated by

$$R_n^{(i,k)} = R_n^{(0)^{(k)}} \prod_{p=1}^{i-1} L_n^{(i,k)}$$

Let $\mathbf{R}_t^{(k)}$ denote the average message going out of a variable node in $G(q_k)$. By noting that the messages $\mathbf{L}^{(i,k)}$ are i.i.d. when (i,k) is set, we have:

$$\begin{aligned}
D_a(\mathbf{R}_t^{(k)}) &= \sum_i \Pi(i|k) \frac{1}{q_k - 1} \sum_{n=1}^{q_k} \mathbb{E}\left(\sqrt{\frac{R_n^{(0)^{(k)}} \prod_{p=1}^{i-1} L_n^{(i,k)}}{R_0^{(0)^{(k)}} \prod_{p=1}^{i-1} L_0^{(i,k)}}}\right) \\
&= \sum_i \Pi(i|k) \frac{1}{q_k - 1} \sum_{n=1}^{q_k} \mathbb{E}\left(\sqrt{\frac{R_n^{(0)^{(k)}}}{R_0^{(0)^{(k)}}}}\right) \mathbb{E}\left(\sqrt{\frac{L_n^{(i,k)}}{L_0^{(i,k)}}}\right)^{i-1} \\
&= \sum_i \Pi(i|k) \frac{1}{q_k - 1} \sum_{n=1}^{q_k} \mathbb{E}\left(\sqrt{\frac{R_n^{(0)^{(k)}}}{R_0^{(0)^{(k)}}}}\right) D_a(\mathbf{L}^{(i,k)})
\end{aligned}$$

The last step is obtained thanks to the LM-invariance of $\mathbf{L}^{(i,k)}$. Finally we get:

$$D_a(\mathbf{R}_t^{(k)}) = D_a(\mathbf{R}^{(0)^{(k)}}) \sum_i \Pi(i|k) D_a(\mathbf{L}^{(i,k)}) \tag{2.33}$$

Moreover, if we consider two LM-invariant vectors $\mathbf{L}^{(k)}$ and $\mathbf{L}^{(l)}$, where $\mathbf{L}^{(k)}$ is the random truncation of $\mathbf{L}^{(l)}$, it is clear that $D_a(\mathbf{L}^{(k)}) = D_a(\mathbf{L}^{(l)})$. Hence:

$$D_a(\mathbf{L}^{(i,k)}) = \sum_{j,l} \Pi(j,l|i,k) D_a(\mathbf{L}^{(j,l)}) \tag{2.34}$$

where $\mathbf{L}^{(j,l)}$ is the message going out of a check node of degree j in $G(q_l)$.
Let us recall the result of equation (68) in [48]:

$$1 - D(\underline{\mathbf{L}}_t) \geq \sum_d \rho_d \left(1 - D(\underline{\mathbf{R}}_t)\right)^{d-1} + O\left(D(\underline{\mathbf{R}}_t)^2\right)$$

We can apply this result, since our definition of D_a corresponds to the definition the authors gave to D. We obtain

$$1 - D_a(\mathbf{L}^{(j,l)}) \geq (1 - D_a(\mathbf{R}^{(j,l)}))^{j-1} + O(D_a(\mathbf{R}^{(j,l)})^2) \tag{2.35}$$

where $\mathbf{R}^{(j,l)}$ is a message going into a check node of degree j in $G(q_l)$. It is straightforward from definition of $D_a(\cdot)$ to get:

$$D_a(\mathbf{R}^{(j,l)}) = \sum_{i',k'} \Pi(i',k'|j,l) \frac{q_{k'}-1}{q_l-1} D_a(\mathbf{R}^{(i',k')}) \tag{2.36}$$

By gathering equations (2.33), (2.34), (2.35) and (2.36), we obtain:

$$D_a(\mathbf{R}_t^{(k)}) \leq D_a(\mathbf{R}^{(0)^{(k)}}) \sum_i \Pi(i|k) \left[\sum_{j,l} \Pi(j,l|i,k) \left(1 - \sum_{i',k'} \Pi(i',k'|j,l) \left(\frac{q_{k'}-1}{q_l-1} D_a(\mathbf{R}^{(i',k')}) \right)^{j-1} + O(D_a(\mathbf{R}^{(i',k')})^2) \right) \right]^{i-1} \tag{2.37}$$

which is also:

$$D_a(\mathbf{R}_t^{(k)}) \leq D_a(\mathbf{R}^{(0)^{(k)}}) \sum_i \Pi(i|k) \left[\sum_{j,l} \Pi(j,l|i,k) \left(1 - \sum_{i',k'} \Pi(i',k'|j,l) \frac{q_{k'}-1}{q_l-1} D_a(\mathbf{R}^{(i',k')}) \right)^{j-1} \right]^{i-1} + O(D_a(\mathbf{R}_{t-1})^2) \tag{2.38}$$

where $D_a(\mathbf{R}_{t-1}) = \sum_k D_a(\mathbf{R}_{t-1}^{(k)})$. By power series in the neighborhood of zero, we finally get:

$$D_a(\mathbf{R}_t^{(k)}) \leq D_a(\mathbf{R}^{(0)^{(k)}}) \Pi(i=2|k) \sum_{j,l} \Pi(j,l|i,k)(j-1) \sum_{k'} \Pi(k'|j,l) \frac{q_{k'}-1}{q_l-1} D_a(\mathbf{R}_{t-1}^{(k')}) + O(D_a(\mathbf{R}_{t-1})^2) \tag{2.39}$$

Let $c^{(k)} = D_a(\mathbf{R}^{(0)^{(k)}})$ and $p(y|x)$ the transition probabilities of the memoryless output symmetric channel. We recall that we assume that the all-zeros codeword has been sent. Then

$$\begin{aligned} c^{(k)} &= D_a(\mathbf{R}^{(0)^{(k)}}) \\ &= \frac{1}{q_k-1} \sum_{i=1}^{q_k-1} \mathbb{E}\left(\sqrt{\frac{p(y|i)}{p(y|0)}} \right) \\ &= \frac{1}{q_k-1} \sum_{i=1}^{q_k-1} \int \sqrt{\frac{p(y|i)}{p(y|0)}} p(y|0) dy \\ &= \frac{1}{q_k-1} \sum_{i=1}^{q_k-1} \int \sqrt{p(y|i)p(y|0)} dy \end{aligned}$$

We introduce hereafter some notations, for ease of reading:
Let $\mathbf{x}$ be a positive real-valued vector of size the number of different group orders. Let us define the g function by:

$$g(k, c^{(k)}, \Pi, \mathbf{x}) = c^{(k)} \Pi(i=2|k) \sum_{j,l} \Pi(j,l|i,k)(j-1) \sum_{k'} \Pi(k'|j,l) \frac{q_{k'}-1}{q_l-1} x_{k'}$$

For more readable notations, we also define the vector output function $\mathbf{G}(\mathbf{x})$ by:

$$\mathbf{G}(\mathbf{x}) = \{g(k, c^{(k)}, \Pi, \mathbf{x})\}_k$$

which means that the pth component of $\mathbf{G}(\mathbf{x})$ is $G_p(\mathbf{x}) = g(p, c^{(p)}, \Pi, \mathbf{x})$. Let us denote the convolution by $\otimes$. Then $\mathbf{x}^{\otimes n}$ corresponds to the convolution of vector $\mathbf{x}$ by itself n times. With these notations, we can write, for all $n > 0$:

$$D_a(\mathbf{R}_{t+n}^{(k)}) \leq g(k, c^{(k)}, \Pi, \mathbf{G}^{\otimes(n-1)}(\{D_a(\mathbf{R}_t^{(k')})\}_{k'})) + O(D_a(\mathbf{R}_t)^2)$$

Let $P_e^{(k)^t} = P_e(\mathbf{R}_t^{(k)})$ be the error probability when deciding the value of a symbol in $G(q_k)$ at iteration t. The global error probability of decision is $P_e^t = \sum_k \Pi(k) P_e^{(k)^t}$. Let us recall lemma (34) in [48]:

$$\frac{1}{q_k^2} D_a(\mathbf{X}^{(k)})^2 \leq P_e(\mathbf{X}^{(k)}) \leq (q_k - 1) D_a(\mathbf{X}^{(k)}) \tag{2.40}$$

Let us consider a given k. If there exists a vector $\mathbf{x}$ such that $\lim_{n\to\infty} g(k, c^{(k)}, \Pi, \mathbf{G}^{\otimes(n-1)}(\mathbf{x})) = 0$, then there exist α and $n > 0$ such that if $\forall k, D_a(\mathbf{R}_{t_0}^{(k)}) < \alpha$, then

$$D_a(\mathbf{R}_{t_0+n}^{(k)}) < K_{k'} D_a(\mathbf{R}_{t_0}^{(k')}), \quad \forall k' \tag{2.41}$$

where, for all k', $K_{k'}$ is a positive constant smaller than 1. If we consider $P_e^{t_0} < \xi$ such that $\forall k, P_e^{(k)^{t_0}} < (q_k \alpha)^2$, then equation (2.40) ensures that $\forall k, D_a(\mathbf{R}_{t_0}^{(k)}) \leq \frac{\sqrt{P_e^{(k)^t}}}{q_k} < \alpha$. As previously explained, in this case, there exits $n > 0$ such that inequation (2.41) is fulfilled. By induction, for all $t > t_0$, there exists $n > 0$ such that

$$D_a(\mathbf{R}_{t+n}^{(k)}) < K_{k'} D_a(\mathbf{R}_t^{(k')}), \quad \forall k'$$

We have $\forall (k, t), D_a(\mathbf{R}_t^{(k)}) \geq 0$, therefore the sequence $\{D_a(\mathbf{R}_t^{(k)})\}_{t=t_0}^{\infty}$ converges to zero for all k. Finally, equation (2.40) ensures that, for all k, $P_e^{(k)^t}$ converges to zero as t tends to infinity. Thus, P_e^t, the global error probability, averaged on all symbol sizes, converges to zero as t tends to infinity.
This proves the sufficiency of the stability condition.

2.7.5 Information content Through Linear Maps

Lemma 12 *Let x_{in} denote the mutual information of a LDR-message $\mathbf{v}$ going out of a $G(q_1)$ variable node, and x_{out} the mutual information of a LDR-message $\mathbf{w}$ going into a $G(q_2)$ check node. x_{in} and x_{out} are the input and output of the extension. They are connected through the following expression, which is independent of the linear extension:*

$$(1 - x_{in}) \log_2(q_1) = (1 - x_{out}) \log_2(q_2) \tag{2.42}$$

Proof: By hypothesis $\mathbf{w} = \mathbf{v}^{\times A}$. We define the matrices α_A and β_A of size $(q_2 - 1) \times (q_1 - 1)$ and $(q_2 - 1) \times 1$, respectively. These matrices are such that $\mathbf{w} = \alpha_A \mathbf{v} + \beta_A$. There are defined by

$$\begin{aligned}
\forall j = 1 \dots q_1 - 1, \forall i = 1 \dots q_2 - 1, \quad \alpha_A(i,j) &= 1 \quad \text{if i=Aj} \\
&= 0 \quad \text{otherwise} \\
\forall i = 1 \dots q_2 - 1, \quad \beta_A(i) &= 0 \quad \text{if } i \in Im(A) \\
&= C \quad \text{otherwise}
\end{aligned}$$

where C is a strictly positive very big constant, representing infinity. The Jacobi matrix at point u of the linear map applied to LDR-vectors is hence $J_A(u) = \alpha_A$. We then have

$$\begin{aligned}
(1 - x_{out}) \log_2(q_2) &= \mathbb{E}_{\mathbf{W}} \left(\log_2 \left(1 + \sum_{i=1}^{q_2-1} e^{-w_i} \right) \right) \\
&= \int \dots \int \log_2 \left(1 + \sum_{i=1}^{q_2-1} e^{-w_i} \right) P(\mathbf{W} = \mathbf{w}) dw_1 \dots dw_{q_2-1}
\end{aligned}$$

But we know that

$$\begin{aligned}
\forall j = 1 \dots q_2 - 1, \quad W_j &= V_i \text{ if } \exists i : \quad j = Ai \\
&= 0 \text{ if } j \notin Im(A)
\end{aligned} \tag{2.43}$$

Hence

$$\begin{aligned}
&(1 - x_{out}) \log_2(q_2) \\
= &\int \dots \int \log_2 \left(1 + \sum_{i=1}^{q_1-1} e^{-w_i} \right) P(W_{A1} = w_1, \dots, W_{A(q_1-1)} = w_{q_1-1}) dw_1 \dots dw_{q_1-1} \\
= &\int \dots \int \log_2 \left(1 + \sum_{i=1}^{q_1-1} e^{-v_i} \right) P(V_1 = v_1, \dots, V_{q_1-1} = v_{q_1-1}) dv_1 \dots dv_{q_1-1} \\
= &(1 - x_{in}) \log_2(q_1)
\end{aligned}$$

□

2.7.6 Mutual information of a probability vector and its Fourier Transform

Let $\mathbf{p}$ be a probability vector of size q, associated to a symbol in $GF(q)$, and $\mathbf{f}$ its Discrete Fourier Transform of size q too. p_j and f_i are the k-th and the i-th components of $\mathbf{p}$ and $\mathbf{f}$, respectively. $\mathbf{f}$ is defined by:

$$f_i = \sum_{k=0}^{q-1} p_k (-1)^{i \cdot k}, \qquad \forall i \in GF(q)$$

$i \cdot k$ is the scalar product between the binary representations of both elements i and k. The mutual information I of a symmetric probability vector $\mathbf{p}$, under the all-zero codeword assumption, is defined by

$$x_{\mathbf{p}} = 1 - \mathbb{E}_{\mathbf{p}} \left(\log_q(1 + \sum_{i=1}^{q-1} \frac{f_i}{f_0}) \right)$$

As in the binary case, we want to prove that

$$x_{\mathbf{p}} = 1 - x_{\mathbf{f}}$$

where $x_{\mathbf{f}}$ is defined by $[x_{\mathbf{f}} = 1 - \mathbb{E}_{\mathbf{p}} \left(\log_q(1 + \sum_{i=1}^{q-1} \frac{f_i}{f_0}) \right)$ **Proof**:
We want to prove that

$$x_{\mathbf{p}} = 1 - x_{\mathbf{f}}$$

that says

$$\begin{aligned}
\mathbb{E}_{\mathbf{f}} \left(\log_q(1 + \sum_{i=1}^{q-1} \frac{f_i}{f_0}) \right) &= 1 - \mathbb{E}_{\mathbf{p}} \left(\log_q(1 + \sum_{i=1}^{q-1} \frac{p_i}{p_0}) \right) \\
\mathbb{E}_{\mathbf{f}} \left(\log_q(1 + \sum_{i=1}^{q-1} \frac{f_i}{f_0}) \right) &= \mathbb{E}_{\mathbf{p}} \left(1 - \log_q(\frac{1}{p_0}) \right) \\
\mathbb{E}_{\mathbf{f}} \left(\log_q(1 + \sum_{i=1}^{q-1} \frac{f_i}{f_0}) \right) &= \mathbb{E}_{\mathbf{p}} \left(\log_q(qp_0) \right) \\
f_0 = 1 \quad & \text{implies} \\
\mathbb{E}_{\mathbf{f}} \left(\log_q(\sum_{i=0}^{q-1} f_i) \right) &= \mathbb{E}_{\mathbf{p}} \left(\log_q(qp_0) \right) \qquad (2.44)
\end{aligned}$$

Since $\sum_{i=0}^{q-1} f_i = \sum_{i=0}^{q-1} \sum_{k=0}^{q-1} p_j (-1)^{i \cdot k}$, it finally remains to prove that

$$\begin{aligned}
\sum_{i=0}^{q-1} \sum_{k=1}^{q-1} p_j (-1)^{i \cdot k} &= 0 \\
\sum_{k=1}^{q-1} p_j \sum_{i=0}^{q-1} (-1)^{i \cdot k} &= 0 \qquad (2.45)
\end{aligned}$$

which is ensured by

$$\sum_{i=0}^{q-1} (-1)^{i \cdot k} = 0, \qquad \forall k = \{1 \ldots q-1\}$$

We are going to demonstrate this last expression.

Let say that k has m bits equal to 1 in its binary representation.

- m is even: $i \cdot k$ is

$$\text{even} \quad \frac{q}{2^m} \sum_{l=0}^{m/2} \binom{m}{2l} \quad \text{times} \tag{2.46}$$

$$\text{odd} \quad \frac{q}{2^m} \sum_{l=0}^{m/2-1} \binom{m}{2l+1} \quad \text{times} \tag{2.47}$$

- m is odd: $i \cdot k$ is

$$\text{even} \quad \frac{q}{2^m} \sum_{l=0}^{\frac{m-1}{2}} \binom{m}{2l} \quad \text{times} \tag{2.48}$$

$$\text{odd} \quad \frac{q}{2^m} \sum_{l=0}^{\frac{m-1}{2}} \binom{m}{2l+1} \quad \text{times} \tag{2.49}$$

We complete the proof by showing that equations (2.46) and (2.47) are equal, so are (2.48) and (2.49):

$$(1-1)^m = \sum_{k=0}^{m} \binom{m}{k} = \sum_{l=0}^{\lfloor m/2 \rfloor} \binom{m}{2l} - \sum_{l=0}^{\lfloor m/2-1 \rfloor} \binom{m}{2l+1} = 0$$

Hence

$$\sum_{l=0}^{\lfloor m/2 \rfloor} \binom{m}{2l} = \sum_{l=0}^{\lfloor m/2-1 \rfloor} \binom{m}{2l+1}$$

□

Chapter 3

Machine Learning Methods for Code and Decoder Design

The initial subject of the thesis was to investigate how machine learning methods might be used for optimizing finite-length codes, i.e., for lowering the sub-optimality of BP decoding by breaking cycles. The starting idea was to build the Tanner graph of a code, by means of a supervised learning process applied to the graph of a mother code, in order to decide which edges should be pruned away.

The first section presents works from the literature, focusing on the relations between machine learning and coding.

The second section details our studies done around this idea, among which the modeling of the BP decoding process by a neural network, and why such an approach has not been successful. The final goal was to consider hybrid LDPC codes as a tool to build codes with good finite-length properties. This was planned to be achieved by learning how to assemble hybrid nodes in order to lower the sub-optimality of the BP decoder on finite-length codes. We explain why we could not succeed in defining a valid framework for this purpose.

The third section investigates how to modify the BP decoder in order to lower its sensibility to graph cycles, by adapting it to the graph of a given code. For this purpose, the BP decoder has been considered as a classifier with room for improvement.

All the codes considered in this chapter are binary LDPC codes.

3.1 Previous works

3.1.1 Information-theoretic models of artificial neural networks

Early after Claude Shannon wrote the foundations of information theory, a paper by Attneave [69] introduced the idea that information theory may offer an explanation for perceptual processing. In Simon Haykin book [70], a thorough description of information-theoretic models that lead to self-organization is detailed. We can cite this book (chapter 10, page 506): *"A model that deserves special mention is the maximum mutual information principle due to Linsker [71]. This principle states that the synaptic connections of*

a multilayer neural network develop in such a way as to maximize the amount of information that is preserved when signals are transformed at each processing stage of the network, subject to certain constraints."
Based on this analysis, we are going to describe the decoding of LDPC codes as a process that can be code adaptive, and see where the mutual information should be maximized on the artificial neural network to model the decoding process.

3.1.2 Learning methods and error-correcting codes

Some articles have presented the link between neural network methods and error-correcting code approaches. In 1989, Bruck et al. [72] presented one of the most significant works in this field: "Neural Networks, Error-Correcting Codes, and Polynomials over the Binary n-Cube". The authors state that the Maximum-Likelihood (ML) decoding of a linear block error-correcting code, is equivalent to finding the min-cut of a specific graph. Hence, based on their work on the relation between the maximization of n-cubic polynomials and error-correcting codes, the author proposed to use decoding techniques to find the maximum of these polynomials.

In 1992, Tseng et al. [73] focused on decoding Hamming codes of type $(2^n - 1, 2^n - 1 - n)$ and extended Hamming codes of type $(2^n, 2^n - 1 - n)$ with a single-layer or a double-layer perceptron, of low complexity, whose discriminating functions were polynomials of high degrees.

3.2 Machine Learning Methods for Code Design

3.2.1 Problem

The aim is to modify the Tanner graph structure of a mother code in order to build a new code with a good minimum distance. What we consider as “good” will be detailed in the following. We have decided to remove some edges from the graph of the mother code to obtain the new code. However, we generally cannot increase the minimum distance of codes by lowering the density. Indeed, it has been shown in [17] that all the sequences of LDPC codes reaching the capacity of the erasure channel have a large fraction of degree two variable nodes which gives rise to low-weight codewords. Such codewords correspond to cycles, in the subgraph of the Tanner graph, which contain only degree two variable nodes. Thus, the problem we chose to address is: how to prune away edges in the Tanner graph of a mother code in order to obtain a less dense code, with a minimum distance higher than a code of same density, known for holding a good minimum distance.

For this purpose, we consider the impulse method presented in [66] to compute the minimum distance of LDPC codes. The basic principle of this method is to feed the BP decoder with impulses (an impulse being an all-zero vector except for one or very few components set to one), then the smallest weight codeword is decided by a list decoder.

In order to decide which edges of the mother code should be pruned away to lower the least the minimum distance, the idea is to formalize a certain analogy that may be found between the graph of a code and an artificial neural network (ANN). The ANN definition is presented further. In that case, the addressed problem of pruning edges appears to be not a common artificial learning problem. Indeed, applying a learning process to an ANN basically means that the structure, i.e., the connections between neurons, are already determined, and what is learnt is the weight of each connection. When the learning process is said "supervised", the desired output of each neuron, on the output layer, is known for each input prototype from a training set.

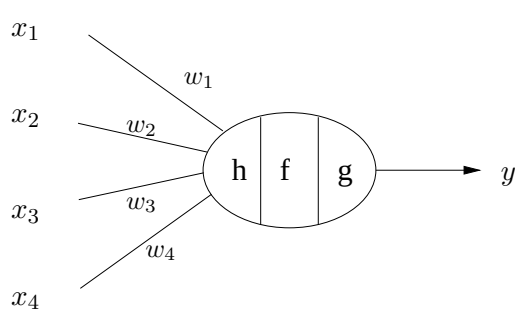
Our problem is rather different since it consists in finding the structure of the network: what should be the connections between neurons. However, the structure of the network is usually decided in an *ad hoc* way or with simple heuristic rules [74]. Indeed, except an exhaustive search, none method is known to determine the optimal architecture for a given problem. A suboptimal solution consists in using constructive algorithms starting from a minimal architecture then adding neurons and connections progressively during the learning process [74]. Another solution considers an inverse technique: starting from a fully interconnected structure, they remove neurons or connexions which seem non-essential. We are going to focus on the latter methods.

3.2.2 Neural networks and codes Tanner graphs

Definition

Definition 10 *A formal neuron is basically a processor which applies a simple operation to its inputs, and which can be connected to other identical processors in order to form a network.*

Such a neuron is depicted on figure 3.1, and defined in [75].



x_i: neuron inputs

A: neuron activation

y: neuron output

w_i: synaptic weights

h: input function

f: activation (or transfert) function

g: output function

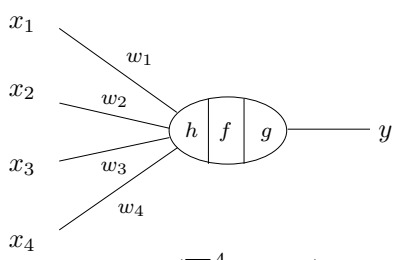
$$A = f(h(\{x_i\}_{i=1\ldots4}, \{w_i\}_{i=1\ldots4}))$$

$$y = g(A) \ (= A \text{ most often})$$

Figure 3.1 : General definition of a formal neuron

The (h, f, g) combination defines the type of the neuron.

Summator Neuron The most common definition of a formal neuron corresponds to the particular case where the input function h is a dot product between the input and the weights.



$y = f(\sum_{i=1}^{4} w_i x_i)$

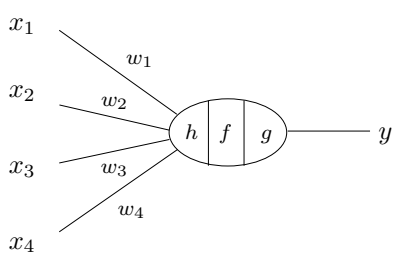
h: dot product

f: any kind of non-linear function (echelon, sigmoid, Gaussian, ...) defining the type of neuron

g: identity function

Figure 3.2 : An artificial neuron which computes the weighted sum of the inputs, and the apply the activation function f.

Polynomial Neuron Such a kind of neuron [75] is depicted on figure 3.3.



h: polynomial

f: any kind of non-linear function (echelon, sigmoid, Gaussian, ...)

g: identity function

E.g., for an order-2 neuron: $y = f(\sum_{i,k} w_i w_k x_i x_k)$

Figure 3.3 : A polynomial neuron.

Modelization of the decoder

Since the goal is to build a Tanner graph on which the BP decoder is as less suboptimal as possible, we translate the decoding on the Tanner graph of the code as the process of an Artificial Neural Network (ANN). Let a message from variable node v to check node c at iteration t be described by a 2-dimensional probability vector $\mathbf{x}_{vc}^{(t)} = (x_{vc}^{(t)}(0), x_{vc}^{(t)}(1))^T$, where $x_{vc}^{(t)}(0)$ and $x_{vc}^{(t)}(1)$ correspond to the conditional probabilities for the variable node v to be equal to 0 or 1, respectively. The Logarithmic Density Ratio (LDR) $m_{vc}^{(t)}$, associated with $\mathbf{x}_{vc}^{(t)}$, is defined as $m_{vc}^{(t)} = \log\left(\frac{x_{vc}^{(t)}(0)}{x_{vc}^{(t)}(1)}\right)$. The same holds for a message $p_{cv}^{(t)}$ from

check node c to variable node v. The Logarithmic Likelihood Ratio (LLR) corresponding to the observation of the nth bit is denoted by $LLR(n)$. $\mathcal{V}(v)$ denotes the first level neighborhood of variable node v, i.e., all the check nodes which are connected to v by a single edge. The same holds for $\mathcal{V}(c)$ of check node c. The BP equations corresponding to update these messages can then be expressed by

$$m_{vc}^{(t)} = LLR(v) + \sum_{d \in \mathcal{V}(v) \setminus c} p_{dv}^{(t-1)} \tag{3.1}$$

$$\tanh(\frac{p_{cv}^{(t)}}{2}) = \prod_{u \in \mathcal{V}(c) \setminus v} \tanh(\frac{m_{uc}^{(t)}}{2}) \tag{3.2}$$

As seen in the previous section, many inputs correspond to each neuron, but only one output does. Hence, the ANN, modeling the BP decoding, is made of summator and polynomial neurons and corresponds to unfold the decoding iterations. However, one pattern of the network, corresponding to one iteration, is not a copy of the factor graph. This is illustrated on figure 3.4.

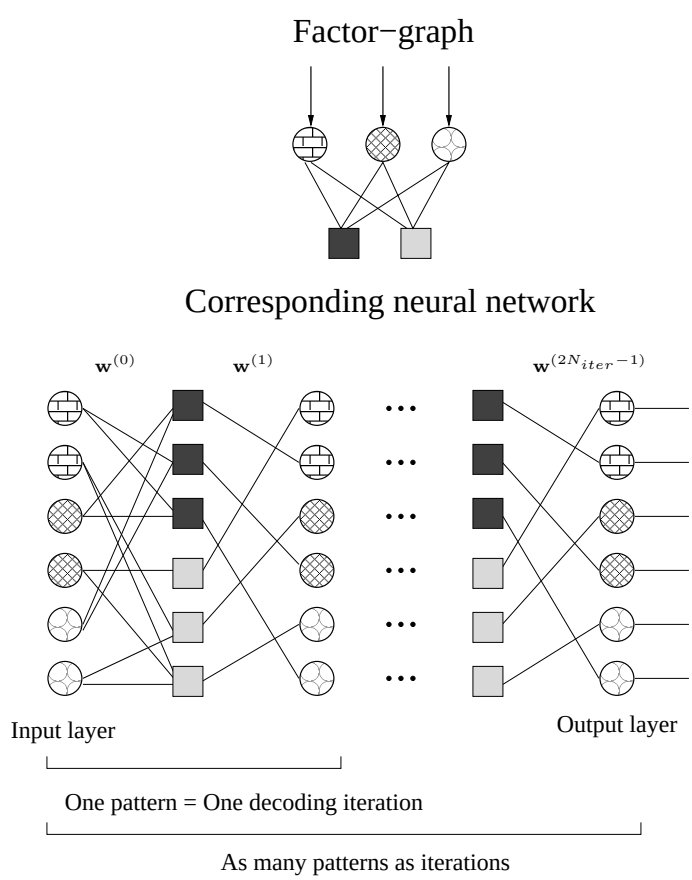


Figure 3.4 : A factor graph and its corresponding neural network. Each neuron corresponds to an edge of the factor graph, hence there are $2.N_{edge}.N_{iter}$ neurons in the network.

In this network, the number of layers is double the number of decoding iterations, and the number of nodes on each layer is equal to the number of edges in the Tanner

graph. Hence, removing an edge from the Tanner graph of the code will correspond to removing the corresponding neuron in all the layers. Following the convention of figure 3.4, a neuron which processes a message going out of a variable node is called circle neuron, while a neuron which processes a message going out of a check node is called square neuron. The message going out of a variable v towards a check c is computed by the corresponding circle summator neuron:

$$m_{vc}^{(t)} = LLR(v) + \sum_{d \in \mathcal{V}(v) \setminus c} w_{dv}^{(2t+1)} p_{dv}^{(t-1)} \tag{3.3}$$

where $w_{dv}^{(2t+1)}$ is the weight of the message $p_{dv}^{(t-1)}$ in the calculation of $m_{vc}^{(t)}$.

Analogously, the message going out of a check c towards a variable v is computed by the corresponding square polynomial neuron:

$$\tanh\left(\frac{p_{cv}^{(t)}}{2}\right) = \prod_{u \in \mathcal{V}(c) \setminus v} w_{uc}^{(2t)} \tanh\left(\frac{m_{uc}^{(t)}}{2}\right) \tag{3.4}$$

where $w_{uc}^{(2t)}$ is the weight of the message $m_{uc}^{(t)}$ in the calculation of $p_{cv}^{(t)}$.

In the particular case, where all the weights are equal to 1, the neural network is the BP decoder. We see how weights could be used to modify the BP decoder by adding degrees of freedom. This will be discussed in the section 3.3.

3.2.3 Cost function for pruning

In the following, we use identically the term cost function and the term error. If the global cost function for pruning edges is the minimum distance of the code with Tanner graph corresponding to the ANN, this criterion is a global criterion. However, we need to differentiate the Tanner graph edges, between each other, in order to choose which ones should be removed. Therefore, it is necessary to decide which are the desired outputs on each output layer neuron. In our case, this means that we need to decide what should be the value of each message going out of check nodes in the last iteration.

Making this choice has been the first critical issue for this approach. In a completely heuristic way, we chose to penalize edges connected to variable nodes whose value, in the smallest weight codewords, is zero. This means that at each input prototype, all the weights of the ANN corresponding to the same edge of the Tanner graph are updated in the same way: they are lowered when the variable value is zero in the smallest weight codewords, increased otherwise.

3.2.4 Pruning methods

In the aforementioned heuristic framework for pruning, two general pruning methods arise.

The first approach is to consider the sum of the weights of connections to each neuron, then prune away the neuron with the smallest sum. The methods based on such an

approach are known as *magnitude based methods* [76], because they eliminate weights that have the smallest magnitude. However, as mentioned in [77], this simple plausible idea unfortunately often leads to the elimination of wrong weights, as small weights may be necessary for low error. The second solution is to apply the Optimal Brain Surgeon (OBS) [77], which is far better than the magnitude based methods. OBS is based on the functional Taylor series of the network error E with respect to weights [77]:

$$\delta E = \frac{\partial E}{\partial \mathbf{w}}^T \cdot \delta \mathbf{w} + \frac{1}{2}\delta \mathbf{w}^T \cdot \mathbf{H} \cdot \delta \mathbf{w} + O(||\delta \mathbf{w}||^3) \tag{3.5}$$

Here is the OBS procedure:

1) Train a "reasonably large" network (i.e., adapt its weights stored in the vector $\mathbf{w}$) towards a minimal error E of the network.

2) Compute (iteratively) the inverse of the Hessian matrix $\mathbf{H}^{-1}$. $\mathbf{H}$ actually corresponds to the second order derivative of the network error related to the weights.

3) Find the index q of the weight w_q giving the smallest "saliency" L_q. The saliency is the increase of the network error from removing the corresponding edge. We get the following expression for L_q:

$$L_q = \frac{1}{2}\frac{w_q^2}{[H^{-1}]_{qq}}$$

If this candidate error increase is much smaller than E, then q^{th} weight should be deleted, and we proceed to step 4. Otherwise go to step 5.

4) Use the q of step 3 in order to update all the weights with the following formula :

$$\delta \mathbf{w} = -\frac{w_q}{[H^{-1}]_{qq}} c_q$$

with c_q the qth column of H^{-1}. Go back to step 2.

5) No more edge can be pruned without large increase in E. It may be necessary to retrain the network.

This algorithm is valid only when the first (linear) term of equation 3.5 vanishes, as well as the third and all higher order terms. OBS assumes that the third and all higher order terms can be neglected [77]. No more explanation of this assumption is given in [77]. For the first term to vanish, the network must have been trained to a local minimum in error.

In order to apply this algorithm, it is necessary to define the error of the network, thus, to determine the desired outputs of this network. We recall that an output neuron of the network corresponds to an edge of the Tanner graph. The impulse method allows to find low-weight codewords. Defining the desired outputs of the network is therefore equivalent to define the quality of an edge of the Tanner graph in terms of the output of

the impulse method. This means that we must decide what should be the value of the messages after a given number of iterations.

Our heuristic was to penalize edges connected to variable nodes whose value, in the smallest weight codewords, is zero in order to optimize the minimum distance of the code. However, this this tends to modify the Tanner graph and to turn it into a new code, whose minimum distance may have no relation with the minimum distance we were trying to optimize at first.

To see how finding a new Tanner graph by pruning a mother code is not fitted to be solved by pruning an artificial neural network, we can express in another way the above problem of the choice of the cost function: Modeling the belief propagation decoder by an artificial neural network, as done in figure 3.4, leads to consider the BP decoder as a classifier which, to a given noisy observation of a codeword, associates the most likely sent codeword. However, the above pruning approach aims at finding a Tanner graph. This does not consist in finding a good classifier for a given problem, as neural networks are meant to do, but in finding classes (the codewords) on which the classifier depends. Thus, due to the difficulty (impossibility?) to find the relation between minimum distances of the mother code and its pruned version, we could not find a relevant cost function in such a framework. Instead, we decided to focus on a better posed problem and to propose a relevant approach.

3.3 Machine Learning Methods for Decoder Design

In this section, we switch to another problem than code design. We consider a given code which sets the classes, and we are going to look for the best classifier to classify inputs in the right classes. The classifier is the decoder. The approach is detailed below.

3.3.1 Decoding is a classification problem

As aforementioned, the decoding problem can be seen as a classification problem, where, for each noisy observation received from the channel, one wants to find the corresponding sent codeword. If we assume a linear code of length N with K information bits and $M = N - K$ redundancy bits, decoding consists in finding to which class the observation belongs, among 2^K classes corresponding to all possible codewords, in the vector space of dimension K. Hence, a class corresponds to a codeword and is made of all the noisy variants of this codeword such that, for all $i \in 1, \ldots, N$, if the ith bit of the observation is different from the ith bit of the codeword, then the Hamming distance between the codeword and the observation must be lower than $\frac{d_{min}^{loc}(i)}{2}$, with $d_{min}^{loc}(i)$ being the local minimum distance of bit i in the code, as defined in [38]. In other words, the class of a given codeword $\mathbf{c}$ corresponds to the set of all points closer to $\mathbf{c}$ than to any other codeword. A class is therefore the interior of a convex polytope (in some cases unbounded) called the Dirichlet domain or Voronoi cell for $\mathbf{c}$. The set of such polytopes tessellates the whole space, and corresponds to the Voronoi tessellation of all codewords (i.e., to the

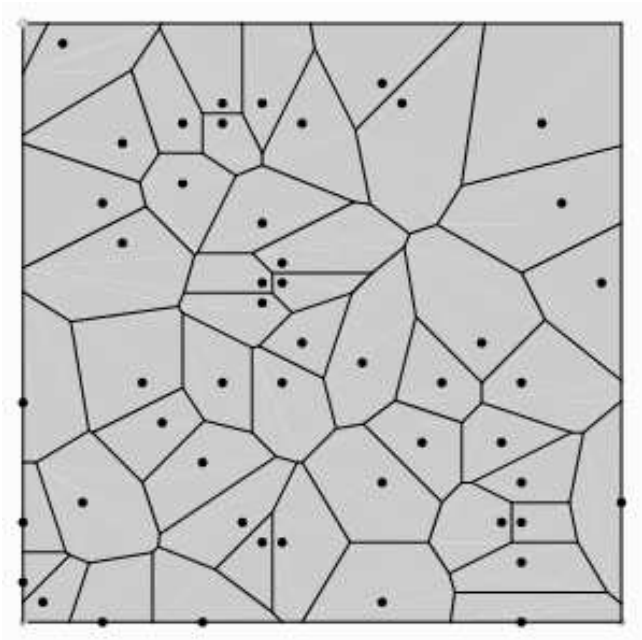


Figure 3.5 : Voronoi diagram (or Dirichlet tessellation): the partitioning of a plane with n points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other.

code). Hence, we know theoretically the optimal classifier, which corresponds to implement a K-dimensional Voronoi partition of the Euclidean space $GF(2)^N$ with codewords as cell centroids, as sketched on figure 3.5. However, implementing this partitioning is intractable in practice for long codes, and corresponds exactly to implement maximum-likelihood (ML) decoding. That is why this classification problem is usually solved with a BP decoder, which actually only implements an approximation of the Voronoi tessellation frontiers, i.e., of ML decoding. Many previous works [19, 20] have characterized the phenomenon which arises when BP decoder is used on loopy graphs, and which emphasizes the difference between ML decoding and BP decoding. ML decoding is always able to find the codeword closest to the observation (even though it makes errors because this closest codeword is not the one which has been sent), whereas BP decoding may converge to fixed points which are not codewords. These points are usually called pseudo-codewords, and it has been shown [19] that they are of first importance in the loss of performance of BP decoding compared to maximum-likelihood decoding.

To try to improve the BP decoding, we focus on pseudo-codewords, but indirectly. Indeed, we make the assumption that pseudo-codewords are the indicators that the frontiers of the classifier implemented by the BP decoder are not the frontiers of ML decoding. Hence, we are going to try to find a correction to BP decoding by considering it as a classifier.

3.3.2 Modelization of BP decoding

The classifier we decide to consider corresponds to a specific case of networks made of neurons defined in definition 10. BP decoding is modeled by an ANN in the same way as in section 3.2.2 (see figure 3.4). The operations processed by circle neurons and squared neurons are respectively:

$$m_{vc}^{(t)} = LLR(v) + \sum_{d \in \mathcal{V}(v) \setminus c} w_{dv}^{(2t+1)} p_{dv}^{(t-1)} \tag{3.6}$$

$$\tanh\left(\frac{p_{cv}^{(t)}}{2}\right) = \left(\prod_{u \in \mathcal{V}(c) \setminus v} sign(m_{uc}^{(t)})\right) \cdot \prod_{u \in \mathcal{V}(c) \setminus v} \left(\tanh\left(\frac{|m_{uc}^{(t)}|}{2}\right)\right)^{w_{uc}^{(2t)}} \tag{3.7}$$

In the particular case where all the weights are equal to 1, such a neural network is exactly the BP decoder. The weights are additional degrees of freedom that we intend to set in order to adapt the BP decoding rules to a given Tanner graph and thus lower its sub-optimality. We propose a modification of the BP decoder based on these weights. From now on, we call this BP decoder with added-weights: *weighted-BP*.

We have hence defined how the correction weights are going to modify the BP decoder: they are coefficients for the variable node update and exponent to the absolute value of the hyperbolic tangent for check node update. These weights are meant to turn the BP classifier into a classifier which matches the topologies of the graph of a given code, in order to better approach ML classifier. The problem now is to choose those correction weights. First of all, since our goal is to make decoding a success, we must define a cost function to measure the quality of the decoding. Determining those weights, which corresponds to solve a learning problem, will hence be addressed thanks to supervised learning.

3.3.3 Cost function

We now present the problem of the choice of the cost function that we have to minimize. Our problem is to make the weighted-BP decoder less sensitive to correlation of the messages on the factor graph of the code. We want thus to find the optimal weights for a given LDPC code $\mathcal{C}$, i.e. for a given parity-check matrix $\mathbf{H}$, which provide a weighted-BP decoder as close as possible to ML decoding. We hence have to look for a cost function that codes the loss of performance of the weighted-BP decoder applied to the given LDPC code $\mathcal{C}$, compared to the ML decoder. This means that we want to measure the loss of performance of weighted-BP applied to $\mathcal{C}$ compared to classical BP decoding applied on a cycle-free LDPC code with irregularity profile identical to $\mathcal{C}$. The key idea is to measure the mutual information between the input of the channel and messages of weighted-BP decoding of $\mathcal{C}$ at each iteration, and to compare it to the mutual information we would have if the graph was cycle-free.

In the sequel, we shorten the expression "mutual information between the input of the channel and messages" by "mutual information of messages".

The evolution of mutual information of BP decoder applied on a cycle-free graph has been extensively studied by TenBrink in [14], who calls this evolution EXtrinsic Information Transfer (EXIT) charts. Figure 3.6 depicts how the evolution of mutual information of messages, along the decoding iterations, can be seen as coding for the loss of optimality of BP decoder in the non free-cycle case. Indeed, we observe that the mutual information of messages is able to reach a higher value, when decoding the (3,6) MacKay LDPC code of length $N = 504$, than for length $N = 96$. For the latter, BP decoding gets stuck earlier. This is explained by the specific topologies of the factor graphs of the two codes: the length 504 code has girth 8 whereas the length 96 code has girth 6 (the girth is the size of the smallest cycle in the graph). The shortest graph is, hence, worst conditioned to run BP decoding than the longest one, since the messages will be more dependent and since bad topologies (like shorter stopping or trapping sets [35]) will be more likely to appear. Thus,

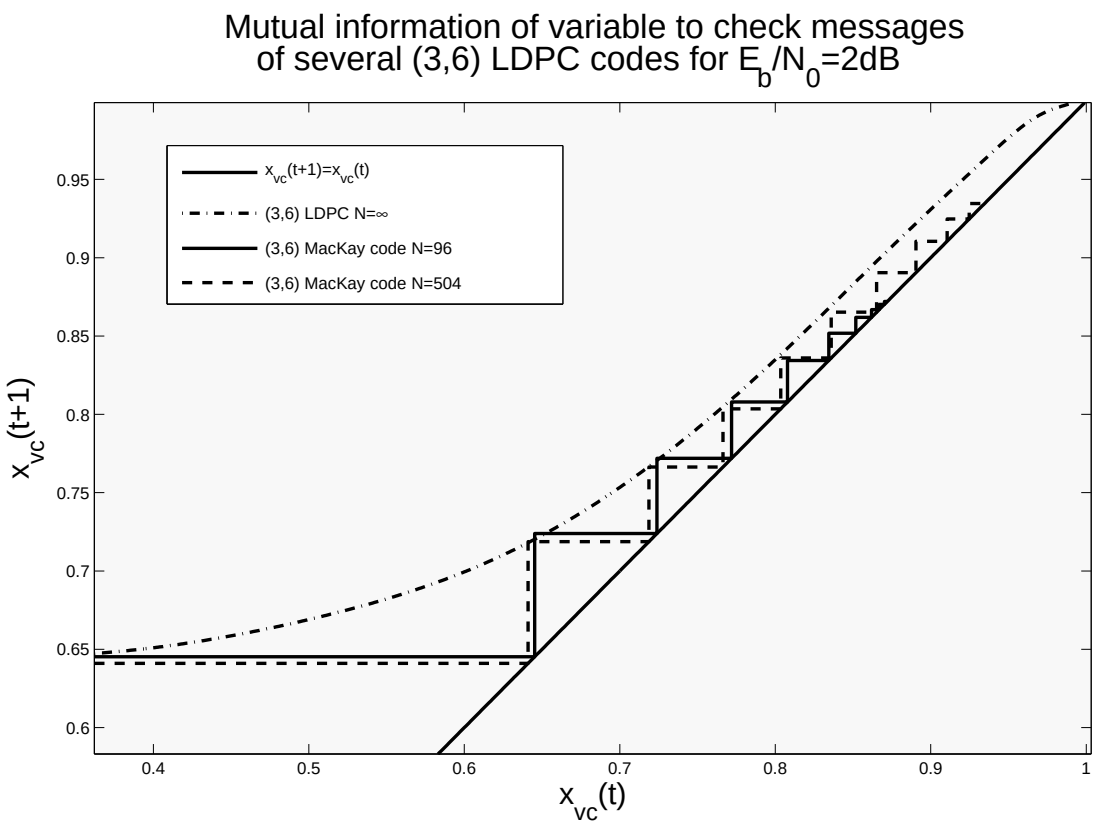


Figure 3.6 : Evolution of the mutual information of variable to check messages along iteration of BP decoding of various codes. Transmission on AWGN channel with $\frac{E_b}{N_o} = 2$dB. The upper hashed dotted curve corresponds to the EXIT function of a cycle-free (3,6) LDPC code. The steps correspond to BP decoding of various finite-length (3,6) LDPC codes.

when optimizing the weights of the weighted-BP decoder for a given code $\mathcal{C}$, the aim will be to minimize the difference between the EXIT function of the cycle-free case and the actual mutual information of the messages, when decoding a given finite-length code $\mathcal{C}$. We assume transmission on additive white Gaussian noise (AWGN) channel. At a given

iteration t, we denote by $x_{vc}^{(\mathcal{F})}(t)$ the mutual information of messages going out of variable nodes when decoding a code of the ensemble $\mathcal{F}$, made of all the possible cycle-free (infinitely long) codes, with same irregularity parameters as $\mathcal{C}$. The mutual information of messages going out of variable nodes, averaged over all the edges of the code $\mathcal{C}$ with parity-check matrix $\mathbf{H}$ at iteration t, is denoted by $y_{vc}^{(\mathcal{C})}(t)$. Hence, $y_{vc}^{(\mathcal{C})}(t)$ depends on $\mathcal{C}$ and on the weights of the weighted-BP decoder. The cost function at iteration t is:

$$f_{cost}^{(\mathcal{C},t)} = x_{vc}^{(\mathcal{F})}(t) - y_{vc}^{(\mathcal{C})}(t) \tag{3.8}$$

Thus, the optimization problem results in looking for the weights, stored in $\mathbf{w}_{opt}^{(\mathcal{C})}$, that minimize the cost function, for each iteration t:

$$\mathbf{w}_{opt}^{(\mathcal{C},t)} = \arg\min_{\mathbf{w}} \left(x_{vc}^{(\mathcal{F})}(t) - y_{vc}^{(\mathcal{C})}(t) \right) \tag{3.9}$$

Indeed, we will solve the optimization problem for each iteration, by assuming that the correction of stage t will depend only on previous iterations.

Let us point out is that the mutual information of a message, on a given edge, at a given iteration, quantifies the “quality” of this edge, i.e., how much this edge is involved in bad topologies (as cycles or combination of cycles). Experiments showed the difference between mutual information of messages on edges involved in very short cycles, and the mutual information of messages on other edges. This is consistent with the fact that errors are more likely to happen on variable nodes involved in such topologies.

The next section deals with the way to handle this optimization problem.

3.3.4 Solving the minimization problem

Backpropagation of the error gradient

To solve the minimization problem, one may think to consider the neural network which would process the mutual information. Indeed, we have seen in the section 1 of this chapter that, with binary LDPC codes, at both check or variable node sides, the mutual information of outgoing messages can be expressed as a sum of functions of the mutual information of incoming messages, using the $J(.)$ function, provided that the message independence assumption is fulfilled (see equation 1.20). This expression of mutual information, with sums, allows to consider the ANN of the type of figure 3.4, made of only summator neurons. This ANN would compute the mutual information of messages in the cycle-free case. Then, this ANN would be a multi-layer perceptron [70], and it would be possible to apply the well-known backpropagation of the error gradient algorithm [78] in order to find the weights leading to the minimization of the cost function. For this supervised learning method, the cost function would be the one of equation 3.8, and the expected value for each output neuron would be the mutual information given by the EXIT curve of the cycle-free code ensemble. Since each neuron corresponds to an edge of the Tanner graph, the output, compared to the expected value, would be the mutual information measured on this edge by empirical mean, when decoding the code $\mathcal{C}$.

The neural network equations would then be used to adapt the weights, thereby considering that the mutual information has been obtained by the sum equations with the $J(.)$ function.

We can see the paradox of this method: The error minimization by back propagating the error gradient is performed based on the neural network equations which assume the absence of cycle whereas the actual output is the mutual information of messages on the cycle graph of $\mathcal{C}$, and thus cannot respect the hypothesis. This is problematic since we want the weights to balance the message dependencies. This is the reason why we cannot use such a supervised learning approach for error minimization.

Genetic Algorithm to solve the optimization problem

The cost function defined in equation 3.8, we choose to minimize, has no analytical expression. Therefore, we are going to choose an optimization algorithm which does not require analytical expression of the cost function. We have decided to use a genetic algorithm [74]. The flow of the optimization procedure is depicted on figure 3.7. An allele of the population vectors is made of weights for the tth iteration: weights $\mathbf{w}^{(2t)}$ to balance messages going out of variable nodes and weights $\mathbf{w}^{(2t+1)}$ to balance messages going out of check nodes. The size of the vectors handled by the genetic algorithm is

$$D = \sum_i (d_v(i) - 1) * d_v(i) + \sum_j (d_c(j) - 1) * d_c(j)$$

where $d_v(i)$ and $d_c(j)$ are the connection degrees of the ith variable node and j^{th} check node, respectively.

In practice, we have implemented the genetic algorithm, thanks to the C library PGApack Parallel Genetic Algorithm Library provided at [79]. We have tried to find weights for the MacKay (3,6) code with code length $N = 96$ at various SNRs. For a population size of 200 vectors, $N_c = 10000$ and $N_{iter} = 10$, the algorithm takes about a week on a last generation CPU.

3.3.5 Estimating the mutual information

To implement the above approach, we have to evaluate the mutual information averaged over all the edges of the graph, at a given iteration. To do so, we use a mean estimator for the expectation of definition 5. We set the SNR, and then send a given number, say N_c, of noisy codewords. Then we evaluate the mutual information as:

$$1 - \frac{1}{N_c} \sum_{n=1}^{N_c} \log_2 \left(1 + e^{-w^{(t,n)}}\right) \tag{3.10}$$

where $w^{(t,n)}$ is any message of the chosen kind (from check-to-variable or variable-to-check) of the graph at the tth iteration when the nth observation is received. This has to be done to evaluate the cost function for each population vector. For good convergence

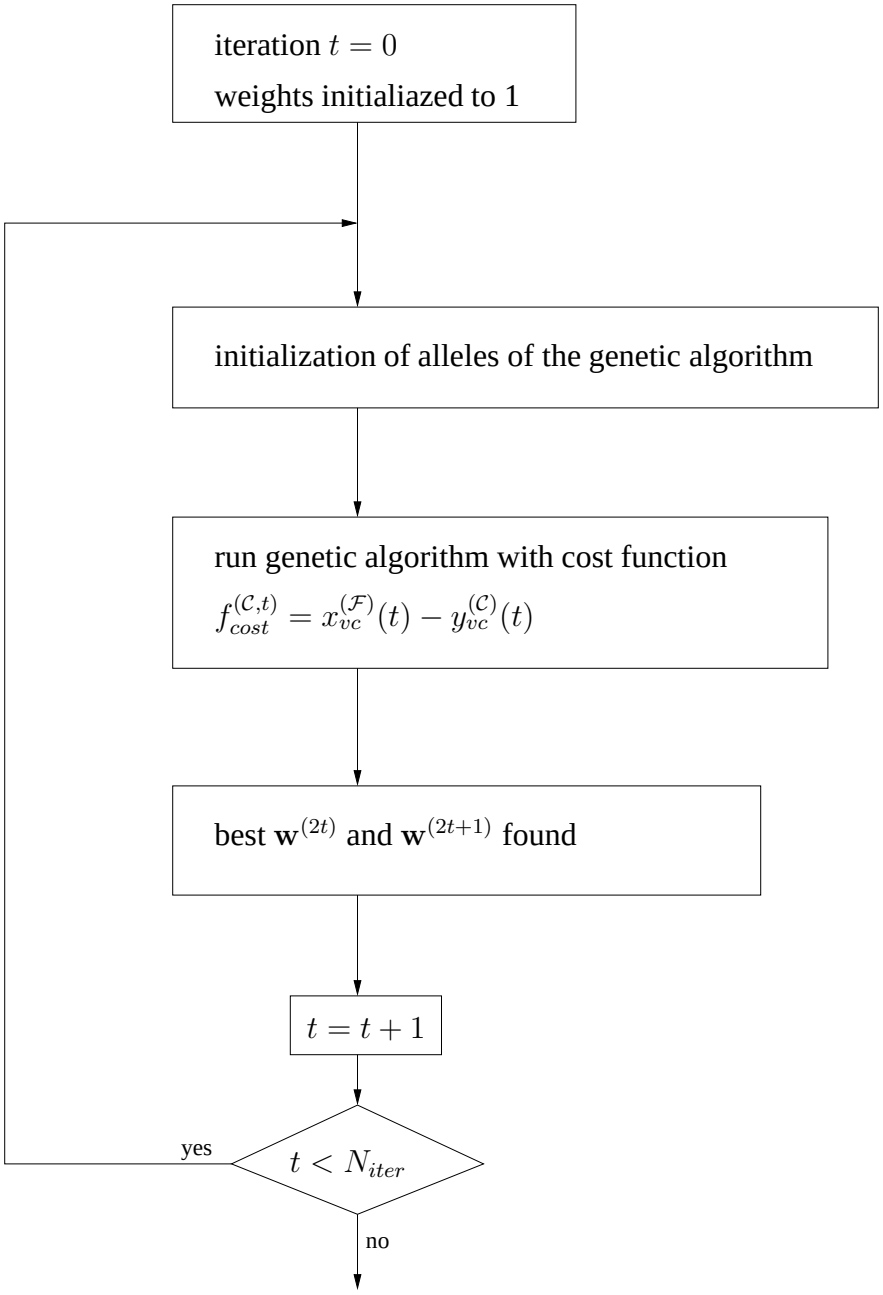


Figure 3.7 : Flow chart of the optimization procedure using a genetic algorithm to find the best weights minimizing the cost function, for each iteration. N_{iter} is the number of decoding iterations for which we look for the correcting weights.

of the genetic algorithm to the global minimum of the cost function, the size of the population must be as high as possible. In practice, to limit the computation time, it is widely accepted that the population size must be at least many hundreds [74]. When the mutual information is close to 1, it turns to be very difficult to get an accurate estimation of the actual mutual information of the messages of the code $\mathcal{C}$, thanks to equation 3.10. Indeed, the closer to 1 is the mutual information, the rarer are the observations which give rise to decoding errors. Since the number N_c of decodings, for one set of weights, has to be limited for computational time reasons, an accurate estimation of the mutual information becomes almost impossible. This problem is related to the error-floor estimation, about which works exist [35]. However, in our case, the method would require an error-floor estimation for each decoder, corresponding to each population vector. This is the prohibitive drawback of the method that made all our tries unsuccessful.Moreover, such a correction of the BP algorithm would be very interesting in the error-floor region, but the above mentioned prohibitive drawback is, more than ever, present in this region.

Finally, it is interesting to note that all these decoders inspired from neural network models do not preserve the symmetry of messages. Indeed, it is easy to check that if a random variable X (standing for a LDR message) is symmetric in the sense of definition 1 in [10] (which is just the binary instance of definition 1.13), then the the random variable $Y = \alpha X$, for any α in $\mathbb{R}$, is not symmetric anymore.

3.3.6 Some other methods

With the goal of investigating how artificial learning methods could contribute to the design of efficient coding systems, we have tried to see how other kinds of learning approaches could be applied to channel coding.

Min-cut max-flow analysis

Our purpose is to detect bad topologies in the Tanner graph of a code, bad topologies being sets of edges which make the decoding to get stuck. Still using the mutual information of messages on a given edge as a quality descriptor of this edge, one may think to consider the iteration when the mutual information on each edge remains stable or periodic but does not converge anymore to 1.

At this point, the idea would be to consider the mutual information as a quantity of liquid which has to increase until being maximum in a water pipe network. Let us consider a water pipe network. For each pipe, the theoretical maximum throughput of liquid inside is called the capacity. The current throughput is called the flow. If the capacity of each pipe is known, then the Ford-Fulkerson algorithm [80] allows to find the maximal flow, shorten as max-flow, between a source at the beginning of the network and a sink at the end. It also allows to detect the minimum cut, that says the set of pipes which limit the flow. The minimum cut is shorten as min-cut. For the pipes defining the min-cut, the flow in each pipe is equal its capacity.

Then the idea was to consider the mutual information of messages on each edge, when

the decoding stops converging, as the capacity of a corresponding pipe in a pipe network to be determined. By running the Ford-Fulkerson algorithm on this network, the goal was to associate the pipes of the min-cut to edges of the Tanner graph involved in limiting topologies (cycles or combination of cycles) for the decoding. We could not complete this investigation because we did not succeed in finding a relevant modeling of the pipe network matching the Tanner graph.

ICA

In this part, we try to see whether sub-optimality of BP decoding could be lowered by applying an independent component analysis to the graph messages.

The primary objective for a neural system with multiple inputs and outputs is to be self-organizing, designed for a specific task (e.g. modeling, extraction of statistically salient features, or signal separation). Haykin ([70], page 520) showed that these requirements can be satisfied by choosing the mutual information between certain variables of the system as the objective function to be optimized. This optimization is equivalent to adjust the free parameters (i.e., synaptic weights) of the system so as to optimize the mutual information. Depending on the application of interest, different scenarios can arise in practice. One of them consists in minimizing the statistical dependence between the components of the output vector $\mathbf{Y}$. This problem corresponds to the *blind source separation problem* and can be solved applying a learning algorithm for Independent Component Analysis (ICA) [81]. The objective of this learning algorithm is to minimize the Kullback-Leibler divergence between the probability density function of $\mathbf{Y}$ and the factorial distribution of Y_i, for $i = 1..m$, if m is the size of the output vector $\mathbf{Y}$. The goal of the algorithm is to find the weight matrix $\mathbf{W}$ which must be as close as possible to the inverse mixing matrix $\mathbf{A}$ with which the signals to be recovered are supposed to be mixed. Such a minimization may be implemented using the method of gradient descent.

We have tried to see whether sub-optimality of BP decoding could be lowered by applying this learning algorithm for ICA of the graph messages at each iteration. We did not succeed in decoding any noisy codeword with ICA output messages.

This way might be an interesting way to follow, but we would emit some reserve for this method: the graph messages are necessarily dependent because of the underlying structure of the finite-length code. Hence, trying to force the messages to be independent, instead of taking into account their statistical dependence, might bias the decoding.

Still with finding a better classifier than the BP algorithm as the objective, we present the two last methods we have investigated in the following section.

Classifiers from the learning literature for decoding LDPC codes ?

The last but one method we have focused on is the Support Vector Machine (SVM) [82]. SVM originally aims at separating two classes. SVM denotes the method which consists, for a given set of examples belonging to both classes, in finding the frontier such that the distance between the frontier and all learning vectors is maximized. Here is the reason why we have considered to use a SVM to find the codeword, i.e. the class, associated to

the input noisy observation of the codeword. Since SVM maximizes the distance between the frontier and the elements of both classes, presented during the learning process, the higher is the number of training patterns, the closer the frontier is to Voronoi, thereby getting closer to ML decoding. However, the generalization to more than two classes does not allow to handle the decoding of an LDPC code because of the complexity, as well as the fact that it is impossible to learn all frontiers between any two codewords.

Finally, we looked at methods for k-nearest neighbors research, or approximate k-nearest neighbors. These methods are spacial access methods, relying on a random partition of space. Among them, we can mention the k-dimensional trees (kd-trees), R-trees [83] or, a much more efficient and recent method, the local sensitive hashing (LSH) [84]. These method are very studied for the problem of multimedia classification, when a new entry has to be associated with the nearest element of a given database. This means that each element of the database can be considered as a class. When a new element is presented, the research consists in finding its nearest neighbor in the database, i.e., the class to which this new element belongs. Thus, we consider these methods as candidates to substitute to the BP decoder seen as a classifier.

The first reason why these methods cannot be applied to LDPC decoding is that they work well only when the distribution of the set of database points is far from uniform, i.e., when the set is "lumpy". Indeed, since these methods rely on a random partition of space, we can intuitively understand that they will be efficient when some parts of the space are almost empty, while other are almost full, thereby allowing favoured search directions.

In particular, in [85] and [86], authors introduced the concept of *fractal dimension* of a set of points to quantify the deviation from the uniformity distribution. Let the *embedding dimension* be defined as [87]: a set has embedding dimension n if n is the smallest integer for which it can be embedded into $\mathbb{R}^n$ without intersecting itself. Thus, the embedding dimension of a plane is 2, the embedding dimension of a sphere is 3.

Authors showed in [85, 86] that these spacial access methods for nearest neighbors search are efficient only when the fractal dimension is much lower that the embedding dimension. When both dimensions are equal, the methods do not work anymore as soon as the dimension is higher than 10 or 12. As previously said, by definition, the embedding dimension of a D-dimensional vector space is D. In [Observation 1, [85]], it is shown that euclidean volumes D-dimensional space have fractal dimension equal to D. The set of codewords of any linear (K, N) code is a vector space of dimension K. The fractal dimension is hence equal to the embedding dimension, both equal to K. In other words, the code space is dense, there is a codeword in each direction. Hence, these methods cannot be applied to LDPC decoding.

Another reason why these methods cannot be used in our case is that they face the problem of dimensionality in the case of LDPC decoding. This problem is also well-known as the "curse of dimensionality" in large scale databases classification domain. Indeed, these methods are non-parametric, which means that they do not take into account the structure of the data, i.e., any underlying model. They only rely on the non-uniform

distribution of the data in their space, as previously explained. Hence, the higher is the number of classes, the harder is the classification. In practice, the best known methods are able to handle databases with a number of classes less than 10^7, which has nothing to see with the channel decoding problem where the number of classes, i.e. of codewords, is at least 2^K with $K > 100$.

A channel code gives the model of the data, and decoding by BP on the Tanner graph of the code corresponds to take into account the underlying model of the data, which are hence completely structured. Thus, we were not able to see any contribution that these methods might bring to enhance the decoding of LDPC codes.

On the other way around we can cite the work of Broder et al [88] who improved the classification of webpages by modeling with a graph the underlying structure of these webpages given by the hyperlinks between each other. Applying a belief propagation on this graph improved the classification. Following these ideas, one could think to try to exploit the underlying structure of any multimedia database, by e.g., modeling it through a factor graph, and then use the BP algorithm for efficient classification. It is obvious that the main problem in that case is to extract a model from a multimedia database, before any try of using this model.

Other works which are representative of what can be done using factor graphs and belief propagation are [89, 90]. In these works, a factor graph framework is used to enforce some *a priori* spatio-temporal constraints for image or video classification. This means that data are assumed to follow a model: e.g., the sky is always in the top part of the scene. This kind of relation is translated by the check nodes of the factor graph, then belief propagation is used for the classification, the image or video query corresponding to the channel observation from a coding point of view.

Thus, at the end of this part of the thesis, it appeared that there may be much more ways to use iterative coding and decoding expertise to improve solving some classification problems currently solved by various machine learning algorithms, rather than paths on the other direction. This kind of investigation may be very interesting, but it is out of the scope of this thesis.

3.4 Conclusion

This work corresponds to the initial subject of the thesis. We have tried to determine which kind of machine learning methods would be useful to design better LDPC codes and decoders in the short code length case.

We have first investigated how to build the Tanner graph of a code by pruning away edges from the Tanner graph of a mother code, based on a machine learning algorithm. We showed that no relevant cost function can be found to be minimized by any learning algorithm. Hence, no pruning method could be applied. We have pointed out that this pruning problem was not a classification problem, and that is why this approach failed.

In the second part, we have investigated decoder design by machine learning methods in order to perform better than BP which is suboptimal as soon as there are cycles in

the graph. We have considered the decoding of a given code as a classification problem to which a better decoder than BP may be found, in order to handle message statistical dependencies. The chosen cost function based on the difference between an estimated mutual information and the EXIT chart appeared to be impossible to evaluate for value of mutual information close to one.

Finally, we have investigated several classification methods to see whether they might substitute the BP decoder. We gave the fundamental reason why this is not possible: those methods are non-parametric machine learning algorithms for databases where the elements must be highly non-uniformly distributed.

Hence, we were not able identify any contribution that machine learning methods might bring to LDPC code or decoder design.

However, this work gave some insights on how channel coding methods can help classification in high-dimensional massive databases, as soon as some structure or model can be assumed for the database.

Chapter 4

Two-Bit Message Passing Decoders for LDPC Codes Over the Binary Symmetric Channel

A class of two-bit message passing decoders for decoding column-weight-four LDPC codes over the binary symmetric channel is proposed. The thresholds for various decoders in this class are derived using density evolution. For guaranteed error correction capability, a decoder with provably relaxed requirements compared to Gallager type algorithms is found.

4.1 Introduction

The performance of various hard decision algorithms for decoding low-density parity-check (LDPC) codes on the binary symmetric channel (BSC), has been studied in great detail. The BSC is a simple yet useful channel model used extensively in areas where decoding speed is a major factor. For this channel model, Gallager [6] proposed two binary message passing algorithms, namely Gallager A and Gallager B algorithms. A code of length n is said to be (n, γ, ρ) regular if all the columns and all the rows of the parity-check matrix of the code have exactly γ and ρ non-zero values, respectively.

Gallager showed [6] that there exist (n, γ, ρ), $\rho > \gamma \geq 3$ regular LDPC codes, with column weight γ and row weight ρ, for which the bit error probability approaches zero when we operate below the threshold (precise definition will be given in Section 4.4). Richardson and Urbanke [11] analyzed ensembles of codes under various message passing algorithms. They also described *density evolution*, a deterministic algorithm to compute thresholds. Bazzi et al. [91] determined exact thresholds for the Gallager A algorithm and outlined methods to analytically determine thresholds of more complex decoders. Zyablov and Pinsker [92] were the first to analyze LDPC codes under parallel bit flipping algorithm, and showed that almost all codes in the regular ensemble with $\gamma \geq 5$ can correct a linear fraction of errors. Sipser and Spielman [93] established similar results using expander graph based arguments. Burshtein and Miller [94] considered expansion

arguments to show that message passing algorithms are also capable of correcting a linear fraction of errors.

We also consider hard decision decoding of a fixed LDPC code on the BSC. When the LDPC code is decoded by message passing algorithms, the frame error rate (FER) curve of has two regions: as the crossover probability α decreases, the slope of the FER curve first increases, and then sharply decreases. This region of low slope for small α is called the error floor region. The problem of correcting a fixed number of errors assumes significance in the error floor region, where the slope of the frame error rate (FER) curve is determined by the weight of the smallest error pattern uncorrectable by the decoder [95].

For iterative decoding over the binary erasure channel (BEC), it is known that avoiding stopping sets [96] up to size t in the Tanner graph [43] of the code guarantees recovery from t or less erasures. A similar result for decoding over the BSC is still unknown. The problem of guaranteed error correction capability is known to be difficult and in this work, we present a first step toward such result by finding three-error correction capability of column-weight-four codes.

Column-weight-four codes are of special importance because under a fixed rate constraint (which implies some fixed ratio of the left and right degrees), the performance of regular LDPC codes under iterative decoding typically improves when he right and left degrees decrease. Burshtein [97] showed that regular codes with $\gamma = 4$, like codes with $\gamma \geq 5$, are capable of correcting a fraction of errors under bit flipping algorithm. These results are perhaps the best (up to a constant factor) one can hope for in the asymptotic sense. The proofs are, however, not constructive and the arguments cannot be applied for codes of practical length. Chilappagari et al. [98] has shown that for a given column weight, the number of variable nodes having expansion required by the bit flipping algorithm grows exponentially with the girth of the Tanner graph of the code. However, since girth grows only logarithmically with the code length, construction of high rate codes, with lengths in the order of couple of thousands, even with girth eight is difficult.

Generally, increasing the number of correctable errors can be achieved by two methods: (a) by increasing the strength and complexity of a decoding algorithm or/and (b) by carefully designing the code, i.e., by avoiding certain harmful configurations in the Tanner graph. Powerful decoding algorithms such as belief propagation, can correct error patterns which are uncorrectable by simpler binary message passing algorithms like the Gallager A/B algorithm. However, the analysis of such decoders is complicated due to the statistical dependence of messages in finite graphs. It also depends on implementation issues such as the numerical precision of messages. For Gallager B decoder, avoiding certain structures (known as trapping sets [35]) in the Tanner graph has shown to guarantee the correction of three errors in column-weight-three codes [99], and this work is an extension of this result.

In this chapter, we apply a combination of the above methods to column-weight-four codes. Specifically we make the following contributions: (a) We propose a class of message-passing decoders whose messages are represented by two bits. We refer to these decoders as to two-bit decoders.(b) For a specific two-bit decoder, we derive sufficient conditions for a code with Tanner graph of girth six to correct three errors.

The idea of using message alphabets with more than two values for the BSC was first proposed by Richardson and Urbanke in [11]. They proposed a decoder with erasures in the message alphabet. The messages in such a decoder have hence three possible values. They showed that such decoders exhibit thresholds close to the belief propagation algorithm. The class of two-bit decoders that we propose is a generalization of their idea, since we consider four possible values for the decoder messages.

Since the main focus of the chapter is to establish sufficient conditions for correction of three errors, we do not optimize the decoders, but instead choose a specific decoder. Also, for the sake of simplicity we only consider universal decoders, i.e., decoders which do not depend on the channel parameter α.

The rest of the chapter is organized as follows. In Section II, we establish the notation and define a general class of two-bit decoders. For a specific two-bit decoder, the sufficient conditions for correction of three errors are derived in Section III. In Section IV, we derive thresholds for various decoders. Simulation results in Section V illustrate that, on a given code, lower frame error rates (FER) can be achieved by a two-bit decoder compared to FER achieved by Gallager B algorithm.

4.2 The class of two-bit decoders

The Tanner graph of a code, whose parity-check matrix $\mathbf{H}$ has size $m \times n$, is a bipartite graph with a set of n variable nodes and a set of m check nodes. Each variable node corresponds to a column of the parity-check matrix, and each check node corresponds to a row. An edge connects a variable node to a check node if the corresponding element in the parity-check matrix is non-zero. A Tanner graph is said to be γ-left regular if all variable nodes have degree γ, ρ-right regular if all check nodes have degree ρ, and (n, γ, ρ) regular if there are n variable nodes, all variable nodes have degree γ and all check nodes have degree ρ.

Gallager type algorithms for decoding over the BSC run iteratively. Let $\mathbf{r}$ be a binary n-tuple input to the decoder. In the first half of each iteration, each variable node sends a message to its neighboring check nodes. The outgoing message along an edge depends on all the incoming messages except the one coming on that edge and possibly the received value. At the end of each iteration, a decision on the value of each bit is made in terms of all the messages going into the corresponding variable node.

Let $w_j(v, c)$ be the message that a variable node v sends to its neighboring check node c in the first half of the jth iteration. Analogously, $\overline{w}_j(c, v)$ denotes the message that a check node c sends to its neighboring variable node v in the second half of the jth iteration. Additionally, we define $w_j(v, :)$ as the set of all messages from a variable v to all its neighboring checks at the beginning of the jth iteration. We define $w_j(v, : \backslash c)$ as the set of all messages that a variable node v sends at the beginning of the jth iteration to all its neighboring checks except c. The sets $\overline{w}_j(c, :)$ and $\overline{w}_j(c, : \backslash v)$ are similarly defined.

Remark: In the case of general two-bit decoders, a number of rules are possible for update. However, we consider only rules which are symmetric Boolean functions that have a simple algebraic expression. We consider symmetric Boolean functions whose

value depends only on the weight in the argument vector, not on positions of zeros and ones. Symmetric Boolean functions are natural choice for regular codes. For irregular codes, asymmetric Boolean functions may lead to improved decoders, but this problem is out of the scope of this work.

These symmetric rules can be seen as follows. The messages are of two kinds: strong and weak. One of the two bits of a message going into a variable node corresponds to the value of this variable node this message votes for, from a majority decoding point of view. The other bit determines the kind of the message. A strong message has a higher number of votes than a weak message. At the variable node, the votes of incoming messages, except the one being computed, are summed up. The value of the variable the outgoing message will carry is determined by the value getting the highest number of votes, while the strength of the outgoing message is determined by this number of votes.

In order to algebraically define the decoder, the message alphabet is denoted by $M = \{-S, -W, W, S\}$ with $S > W$, where the symbols S and W correspond to "strong" and "weak", respectively. Although other equivalent descriptions of the two-bit decoders are possible, we choose to describe them by introducing different quantization levels to the messages. The decoder is then defined by the specific set of quantization levels.

The channel received value alphabet is denoted by $\{-C, C\}$. For any variable node v, R_v is defined as $R_v = (-1)^{r_v} C$. It is important to note that, in this work, the channel amplitude C is not a quantized likelihood [10], since the BSC output is still $\{0, 1\}$, mapped to $\{-C, C\}$. All symbols S, W and C are assumed to be integers. It should be also noted that this representation is as general as representing message alphabet by $\{11, 01, 00, 10\}$ and channel output alphabet by $\{0, 1\}$.

For the sake of clarity, we also define the quantities $t_j(v, :)$ and $t_j(v, c)$, $j > 0$:

$$t_j(v, c) = \sum \overline{w}_{j-1}(: \backslash c, v) + R_v$$

and

$$t_j(v, :) = \sum \overline{w}_{j-1}(:, v) + R_v \tag{4.1}$$

Decoder: The message update and decision rules are expressed as follows.
The messages $\overline{w}_j(c, v)$ are defined as:

$$\overline{w}_j(c, v) = \begin{cases} S \cdot \prod sign(w_j(: \backslash v, c)), & \text{if } \forall v_i \neq v, |w_j(v_i, c)| = S \\ W \cdot \prod sign(w_j(: \backslash v, c)), & \text{otherwise} \end{cases}$$

The messages $w_j(v, c)$ are defined as:

- If $j = 0$, $w_j(v, c) = W \cdot sign(R_v)$.
- If $j > 0$,

$$w_j(v, c) = \begin{cases} W \cdot sign(t_j(v, c)), & \text{if } 0 < |t_j(v, c)| < S \\ S \cdot sign(t_j(v, c)), & \text{if } |t_j(v, c)| \geq S \\ W \cdot sign(R_v), & \text{if } t_j(v, c) = 0 \end{cases}$$

Table 4.1 : Examples of message update for a column-weight-four code, when $C = 2$, $S = 2$ and $W = 1$.

# incoming $-S$ messages	2	1	0	1
# incoming $-W$ messages	0	1	2	0
# incoming W messages	1	0	0	1
# incoming S messages	0	1	1	1
R_v	$-C$	C	C	$-C$
$w_j(v,c)$	$-S$	W	S	$-W$

Decision: After the jth iteration, the decision rule consists in setting the value of the variable v to the sign of $t_j(v,:)$.

Table 4.1 gives an example of message update for a column-weight-four code, when $C = 2$, $S = 2$ and $W = 1$. The message $w_j(v,c)$ goes out of variable node v, and is computed in terms of the three messages going into v from the neighboring check nodes different of c.

The above update and decision rules define the considered class of two-bit decoders. A particular decoder in this class is determined by the set (C, S, W). In the next section, we focus on the two-bit decoder with $(C, S, W) = (2, 2, 1)$, and provide the conditions on the Tanner graph of the code to correct three errors. As shown in Section IV, this decoder has better thresholds than one-bit decoders for various code rates.

4.3 Guaranteed weight-three error correction

In this section, we first find sufficient conditions on the graph of a code to ensure that the code can correct up to three errors in the codeword, when the decoding is performed with the two-bit decoder with $(C, S, W) = (2, 2, 1)$. As justified in the Introduction, we consider only left-regular codes with column weight four.

4.3.1 Sufficient condition for correction of three errors

As mentioned in the Introduction, the higher the code rate, the more difficult the problem of correcting a fixed number of errors. This is the reason why we are interested in finding only sufficient conditions that are as weak as possible in order to be satisfied for high rate codes. That is why we have selected the two-bit decoder defined by $(C, S, W) = (2, 2, 1)$. This decoder has better thresholds than one-bit decoders. The thresholds for various code rates are discussed in Section IV.

For this two-bit decoder, we show that the conditions to guarantee weight-three error correction, are weaker than when Gallager B decoder is used. This means that two-bit decoders permit codes of higher rates than those permitted by one-bit decoders. We note that the problem of establishing correspondence between code rate and absence of a given

topological structure in the Tanner graph is generally difficult and is beyond scope of this work.

Let us first give some additional definitions and notations. We define a path of length d as a set of d connexe edges.

Definition 11 *The neighborhood of order one of a node n is denoted by $\mathcal{N}_1(n)$ and is composed of all the nodes such that there exists an edge between these nodes and n. By extension, $\mathcal{N}_d(n)$ denotes the neighborhood of order d of node n, which is composed of all the nodes such that there exists a path of length d between these nodes and n.*

When T is a set of nodes, say $T = \cup_i n_i$, then the order d neighborhood of T is $\mathcal{N}_d(T) = \cup_i \mathcal{N}_d(n_i)$. Let v_1^1, v_2^1 and v_3^1 the variable nodes on which the errors occur. Let $V^1 = \{v_1^1, v_2^1, v_3^1\}$ and $C^1 = \mathcal{N}_1(V^1)$. For more easily readable notations, we denote $\mathcal{N}_2(V^1)\backslash V^1$ by V^2 and $\mathcal{N}_1(V^2)\backslash C^1$ by C^2. Also we say that a variable is of type $(p|q)$ when it has p connections to V^1 and q connection to V^2. The union of order d neighborhoods of all the $(p|q)$ type variable nodes is denoted by $\mathcal{N}_d(p|q)$.

Now we state the main theorem.

Theorem 5 *[Irregular expansion theorem] Let $\mathcal{G}$ be the Tanner graph of a column-weight-four LDPC code with no 4-cycles, satisfying the following expansion conditions: each variable subset of size 4 has at least 11 neighbors, each one of size 5 at least 13 neighbors and each one of size 8 at least 16 neighbors. Then the code can correct up to three errors in the codeword, provided the two-bit decoder, with $C = 2$, $S = 2$ and $W = 1$, is used.*

For lighter notations, each expansion condition according to which each variable subset of size i has at least j neighbors, will be denoted by "$i \rightarrow j$ expansion condition".

Proof:
Remark: The proof can be followed more easily by looking at Tables 4.2 and 4.3. Table 4.2 draws the decision rule in terms of the numbers of messages $-S$, $-W$, W and S going into a variable, when this variable node is decoded as 0 (resp. 1) and when the channel observation is 1 (resp. 0). Table 4.3 draws update rule in terms of the numbers of messages $-S$, $-W$, W and S going into the variable node v leading to different values of the message $w_j(v, c)$ going out of v, when the received value is r_v. We consider all the subgraphs subtended by three erroneous variable nodes in a graph and prove that, in each case, the errors are corrected. The possible subgraphs are shown in Figure 4.1. As shown, five cases arise. In the reminder, we assume that the all-zero codeword has been sent.

Case 1: Consider the error configuration shown in Figure 4.1(a). In this case, variables 1, 2 and 3 send incorrect $-W$ messages to their neighbors. They receive W messages from all their neighboring check nodes, they are therefore decoded correctly. Error occurs only if there exists a variable node with correct received value that receives four $-W$ messages from its neighboring check nodes (see Table 4.2). However, since variables 1, 2 and 3 are the only variables that send incorrect messages in the first iteration, it is impossible to encounter such a variable node without introducing a 4-cycle. Hence,

Table 4.2 : Decision rule for the two-bit decoder defined by $(C, S, W) = (2, 2, 1)$.

	$\# -S$ mess.	$\# -W$ mess.	$\# W$ mess.	$\# S$ mess.
Received value 1 Decoded as 0	0	0	0	4
	0	0	1	3
	0	0	2	2
	0	0	3	1
	0	0	4	0
	0	1	0	3
	0	1	1	2
	0	1	2	1
	1	0	0	3
	1	0	1	2
Received value 0 Decoded as 1	0	4	0	0
	1	2	1	0
	1	3	0	0
	2	1	0	1
	2	1	1	0
	2	2	0	0
	3	0	0	1
	3	0	1	0
	3	1	0	0
	4	0	0	0

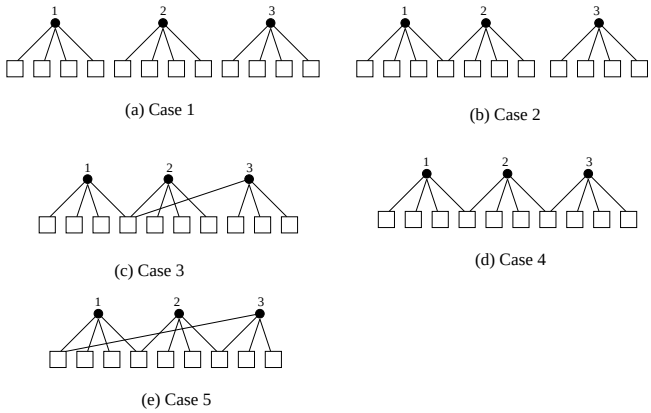


Figure 4.1 : All possible subgraphs subtended by three erroneous variable nodes.

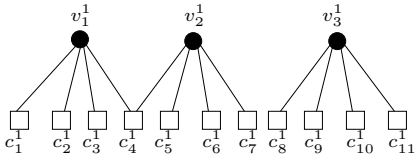


Figure 4.2 : Errors configuration for Case 2.

Table 4.3 : Update rule for the two-bit decoder defined by $(C, S, W) = (2, 2, 1)$.

	$\# -S$ mess.	$\# -W$ mess.	$\# W$ mess.	$\# S$ mess.
$r_v = 1$	0	0	2	1
$w_j(v,c) = W$	0	0	3	0
	0	1	0	2
$r_v = 1$	0	0	0	3
$w_j(v,c) = S$	0	0	1	2
$r_v = 0$	2	1	0	0
$w_j(v,c) = -S$	3	0	0	0
	0	3	0	0
$r_v = 0$	1	2	0	0
$w_j(v,c) = -W$	2	0	1	0
	0	2	0	1
	0	2	1	0
	0	3	0	0
	1	0	2	0
	1	1	0	1
$r_v = 1$	1	1	1	0
$w_j(v,c) = -S$	1	2	0	0
	2	0	0	1
	2	0	1	0
	2	1	0	0
	3	0	0	0
	0	1	1	1
$r_v = 1$	0	1	2	0
$w_j(v,c) = -W$	1	0	0	2
	1	0	1	1
	0	2	1	0
$r_v = 0$	1	1	0	1
$w_j(v,c) = W$	1	1	1	0
	2	0	0	1
	0	0	0	3
	0	0	1	2
	0	0	2	1
	0	0	3	0
$r_v = 0$	0	1	0	2
$w_j(v,c) = S$	0	1	1	1
	0	1	2	0
	0	2	0	1
	1	0	0	2
	1	0	1	1
	1	0	2	0

this configuration converges to the correct codeword at the end of the first iteration.

Case 2: Consider the error configuration shown in Figure 4.1(b) and Figure 4.2.

At the end of the first iteration, we have:

$$\begin{aligned}
{}_1(c_4^1, v) &= -W, \quad v \in \{v_1^1, v_2^1\} \\
\overline{\mathbf{w}}_1(c, v) &= -W, \quad v \in V^2, \quad c \in C^1 \backslash c_4^1 \\
\overline{\mathbf{w}}_1(c, v) &= W, \quad \text{otherwise}
\end{aligned}$$

In the first half of the second iteration, according to Table 4.3 no $-S$ messages can be sent by variables neither in $V \backslash V^1$ because no $-S$ message propagate in the first iteration, nor variables in V^1 because they all receive at least three W messages:

$$\begin{aligned}
w_2(v, c) &= -W, \quad v \in \{v_1^1, v_2^1\}, \quad c \in C^1 \backslash c_4^1 \\
w_2(v, c_4^1) &= W, \quad v \in \{v_1^1, v_2^1\} \\
w_2(v_3^1, c) &= W, \quad c \in C^1 \\
w_2(v, c) &= -W, \quad v \in \mathcal{N}_0(3|1), \quad c \in C^2 \\
w_2(v, c) &= W, \quad v \in \mathcal{N}_0(2|2), \quad c \in C^2 \\
w_2(v, c) &= W, \quad v \in \mathcal{N}_0(3|1), \quad c \in C^1 \\
w_2(v, c) &= S, \quad \text{otherwise}
\end{aligned}$$

In the second half of the second iteration, the messages going out of certain check nodes depend on the connection degree of these check nodes. However, we do not want that the proof be dependent on the degree of connection of check nodes. Hence, we consider in the following the "worst" case, that is the configuration where each message has the smallest possible value. In that case, the messages along the edges in the second half of the second iteration are such that:

$$\begin{aligned}
\overline{\mathbf{w}}_2(c, v) &= -W, \quad v \in V^2 \cap \mathcal{N}_2(\{v_1^1, v_2^1\}), \quad c \in C^1 \backslash c_4^1 \\
\overline{\mathbf{w}}_2(c_4^1, :) &= W \\
\overline{\mathbf{w}}_2(c, : \backslash v) &= -W, \quad v \in \mathcal{N}_0(3|1), \quad c \in C^2 \cap \mathcal{N}_1(3|1) \\
\overline{\mathbf{w}}_2(c, v) &= W, \quad v \in V^2, \quad c \in \{c_8^1, c_9^1, c_S^1, c_{-S}^1\} \\
\overline{\mathbf{w}}_2(c, :) &= W, \quad c \in C^1 \cap \mathcal{N}_1(3|1) \\
\overline{\mathbf{w}}_2(c, :) &= W, \quad c \in C^2 \cap \mathcal{N}_1(2|2) \\
\overline{\mathbf{w}}_2(c, v) &= S, \quad \text{otherwise}
\end{aligned}$$

At the end of the second iteration, all $v \in V^1$ receive all correct messages W or S. According to Table 4.2, all variables in V^1 are hence corrected at the end of the second iteration. For variables in V^2, since no $-S$ messages propagate in the second half of the second iteration, we see on Table 4.2 that variables in V^2, which are not received in error, are decoded as 1 if and only if they receive four $-W$ messages. The following lemma prove that this is not possible.

Lemma 13 *No variable node receives four incorrect $-W$ messages at the end of second iteration.*

Proof: Let v be such a variable. Then the four neighboring checks of v must belong to $\{c_1^1, c_2^1, c_3^1, c_5^1, c_6^1, c_7^1\} \cup (C^2 \cap \mathcal{N}_1(3|1))$. Note that only two neighbors of v can belong to $\{c_1^1, c_2^1, c_3^1, c_5^1, c_6^1, c_7^1\}$ without introducing a 4-cycle. This implies that there are only three cases:

- v has two neighboring checks, say c_1^2 and c_2^2, in $C^2 \cap \mathcal{N}_1(3|1)$. Let v_1^2 and v_2^2 be the $(3|1)$ type variables connected to c_1^2 and c_2^2. It results that the set of variables $\{v_1^1, v_2^1, v_1^2, v_2^2, v\}$ is connected to only 11 checks, which contradicts the 5→13 expansion condition. This case is hence not possible.

- v has one neighbor in $\{c_1^1, c_2^1, c_3^1, c_5^1, c_6^1, c_7^1\}$ and three neighbors in $C^2 \cap \mathcal{N}_1(3|1)$, say c_1^2, c_2^2 and c_3^2. Let v_1^2, v_2^2 and v_3^2 be the $(3|1)$ type variables connected to c_1^2, c_2^2 and c_3^1. It results that the set of variables $\{v_1^1, v_2^1, v_1^2, v_2^2, v\}$ is connected to only 12 checks, which contradicts the 5→13 expansion condition. This case is hence not possible.

- v has four neighbors in $C^2 \cap \mathcal{N}_1(3|1)$, say c_1^2, c_2^2, c_3^2 and c_4^1. Let v_1^2, v_2^2, v_3^2 and v_4^2 be the $(3|1)$ type variables connected to c_1^2, c_2^2, c_3^1 and c_4^1. It results that the set of variables $\{v_1^1, v_2^1, v_3^1, v_1^2, v_2^2, v_3^2, v_4^2, v\}$ is connected to only 15 checks, which contradicts the 8→16 expansion condition. This case is hence not possible.

■

Hence, the decoder converges at the end of the second iteration.

Case 3: Consider the error configuration shown in Figure 4.1(c). In the first iteration, the variables 1, 2 and 3 send incorrect $-W$ messages to their neighboring checks. At the end of the first iteration, they receive correct messages from all their neighboring checks. There is no variable that receives four incorrect messages (as it will cause a four-cycle). Hence, the decoder successfully corrects the three errors.

Case 4: Consider the error configuration shown in Figure 4.1(d) and Figure 4.3. In

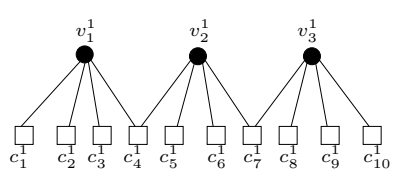


Figure 4.3 : Errors configuration for Case 4.

the second half of the first iteration we have:

$$\begin{aligned}
\overline{\mathbf{w}}_1(c,:\backslash V^1) &= -W \quad , \quad c \in C^1 \backslash \{c_4^1, c_7^1\} \\
\overline{\mathbf{w}}_1(c,v) &= -W \quad , \quad v \in V^1, c \in \{c_4^1, c_7^1\} \\
\overline{\mathbf{w}}_1(c,v) &= W \quad , \quad \text{otherwise}
\end{aligned}$$

Let us analyse the second iteration. For any $v \in V \backslash V^1$ and $c \in C^1$, $\mathbf{w}_2(v,c)$ can never be $-S$ because no $-S$ messages propagate in the first iteration. So, for any $v \in V \backslash V^1$ and $c \in C^1$, $\mathbf{w}_2(v,c) = -W$ if and only if $\overline{\mathbf{w}}_1(:\backslash c, v) = -W$, which implies that v must

have four connections to C^1. This is not possible as it would cause a 4-cycle. Hence:

$$\begin{aligned}
w_2(v_2^1, c) &= -S \quad , \quad c \in \{c_5^1, c_6^1\} \\
w_2(v_2^1, c_4^1) &= -W \\
w_2(v_2^1, c_7^1) &= -W \\
w_2(v_1^1, : \backslash c_4^1) &= -W \\
w_2(v_3^1, : \backslash c_7^1) &= -W \\
w_2(v, c) &= -W \quad v \in \mathcal{N}_0(3|1), \quad c \in C^2 \cap \mathcal{N}_1(3|1) \\
w_2(v_1^1, c_4^1) &= W \\
w_2(v_3^1, c_7^1) &= W \\
w_2(v, c) &= W \quad v \in \mathcal{N}_0(2|2), \quad c \in C^2 \cap \mathcal{N}_1(2|2) \\
w_2(v, c) &= W \quad v \in \mathcal{N}_0(3|1), \quad c \in C^1 \cap \mathcal{N}_1(3|1) \\
w_2(c, v) &= S \quad , \quad \text{otherwise}
\end{aligned}$$

In the first half of the third iteration, we have

$$\begin{aligned}
w_3(v_2^1, :) &= W \\
w_3(v_1^1, : \backslash c_4^1) &= -W \quad , \quad w_3(v_1^1, c_4^1) = W \\
w_3(v_3^1, : \backslash c_7^1) &= -W \quad , \quad w_3(v_3^1, c_7^1) = W
\end{aligned}$$

Lemma 14 *All variables in V^1 are corrected at the end of the third iteration because, for any $v \in V^1$, $\overline{w}_3(:, v) = W$ or S.*

Proof: The proof is by contradiction. Let assume that there exists a variable in $V \backslash V^1$, say v, such that there exists $c \in C^1$ and $w_3(v, c) = -W$ or $w_3(v, c) = -S$. Since it is impossible that two $-S$ messages go into v, as it would cause a 4-cycle, $w_3(v, c) = -W$ or $w_3(v, c) = -S$ implies that v receives from its neighbors different of c three $-W$ messages, or one $-S$ and two $-W$ (see Table 4.3).

- If v receives three $-W$: As proved previously, v cannot have four neighbors in C^1. Hence, v must be connected to $c_1^2 \in C^2$ such that $\overline{\mathbf{w}}_2(c_1^2, v) = -W$. With the above described values of the messages in the second half of the second iteration, we see that c_1^2 must be connected to a $(3|1)$ type variable in V^2, let say x_1^2. Let notice that there cannot be more than one $(3|1)$ type variable in V^2, otherwise five variables would be connected to only twelve checks. Two cases arise:
  - If v has at least two neighbors in $C^2 \cap \mathcal{N}_1(3|1)$, there are at least two $(3|1)$ type variables in V^2, which has been proved to be impossible.
  - If v has exactly one neighbor in $C^2 \cap \mathcal{N}_1(3|1)$, there would exist two $(3|1)$ type variables in V^2: v and x_1^2. This case is not possible for the same reason as above.
- If v receives two $-W$ messages and one $-S$ message:

- If v is of type $(3|1)$, the neighboring check of v in C^2 must be connected to another $(3|1)$ type variable, let say x_1^2. It results that the set $\{v_1^1, v_2^1, v_3^1, v, x_1^2\}$ has only eleven neighboring checks, which contradicts the 5→13 expansion condition. This case is hence not possible.
- If v is of type $(2|2)$, both neighboring checks of v in C^2 must be connected each to another $(3|2)$ type variables, let say x_1^2 and x_2^2. It results that the set $\{v_1^1, v_2^1, v_3^1, v, x_1^2, x_2^2\}$ has only twelve neighboring checks, which contradicts the 5→13 expansion condition. This case is hence not possible.

Hence, since $\overline{w}_3(c_4^1, v_1^1)$, $\overline{w}_3(c_4^1, v_2^1)$, $\overline{w}_3(c_7^1, v_2^1)$ and $\overline{w}_3(c_7^1, v_3^1)$ are equal to W or S, v_1^1, v_2^1 and v_3^1 are corrected at the end of the third iteration.

■

Lemma 15 *No variable in $V\backslash V^1$ can propagate $-W$ at the beginning of the third iteration, except variables of type $(3|1)$.*

Proof: Consider a variable v which has at most two connections to C^1. For this variable v to propagate $-W$ at the beginning of the third iteration, two cases arise:

- If v is of type $(2|2)$, v must have at least one connection to $C^2 \cap \mathcal{N}_1(3|1)$. Let the $(3|1)$ type variable be v_1^2, then the set $\{v_1^1, v_2^1, v_3^1, v_1^2, v\}$ is connected to only twelve checks. This case is hence not possible.
- If v has q connections outside C^1, with $q > 2$, there must exist $q-1$ variables of type $(3|1)$ connected to those $q-1$ checks of v. It results that it would be necessary that at least 2 variables of type $(3|1)$ exist, which is not possible as previously proved.

■

Lemma 16 *Any variable in $V\backslash V^1$ is correctly decoded at the end of the third iteration.*

Remark: That is to say that any variable in $V\backslash V^1$ is decoded to its received value since it is not received in error by hypothesis. *Proof*: According to Table 4.3, no message $-S$ propagates in the third iteration since all variables in V^1 receive at least three W messages at the end of the second iteration, and variables in $V\backslash V^1$ cannot receive more than one $-S$ message. In that case, to be decoded as a one, a bit whose received value is zero has to receive only $-W$ messages according to the decision rule (see Table 4.2). That is for any $v \in V\backslash V^1$, v is wrongly decoded if and only if $\overline{w_3}(:, v) = -W$. No $v \in V\backslash V^1$ can have more than two neighboring checks in $\{c_1^1, c_2^1, c_3^1, c_8^1, c_9^1, c_{10}^1\}$, otherwise it would introduce a 4-cycle. Lemma 15 implies that a variable in $V\backslash V^1$ is wrongly decoded if it has at least two connections to $C^2 \cap \mathcal{N}_1(3|1)$, which implies that there exist two $(3|1)$ type variables. This is not possible as previously proved. This completes the proof of the Lemma.

■

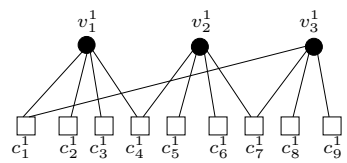


Figure 4.4 : Errors configuration for Case 5.

Thus, the decoder converges to the valid codeword at the end of the third iteration.

Case 5: Consider the error configuration shown in Figure 4.1(a) and Figure 4.4. Neither $(3|1)$ nor $(4|0)$ type variable can exist in V^2 because it would contradict the $4 \rightarrow 11$ expansion condition. Any type $(2|2)$ variables cannot share a check in C^2 as it would result in a set of five variables connected to only twelve checks. At the end of the first iteration, we have:

$$\begin{aligned}
\overline{w}_1(c,v) &= -W, \quad c \in C^1 \backslash \{c_1^1, c_4^1, c_7^1\}, v \in V^2 \\
\overline{w}_1(c,v) &= -W, \quad c \in \{c_1^1, c_4^1, c_7^1\}, v \in V^1 \\
\overline{w}_1(c,v) &= W, \quad \text{otherwise}
\end{aligned}$$

At the end of the second iteration, we have in the worst case, that is in the case where each message has the smallest possible value:

$$\begin{aligned}
\overline{w}_2(c,v) &= -S, \quad c \in C^1 \backslash \{c_1^1, c_4^1, c_7^1\}, v \in V^2 \\
\overline{w}_2(c,v) &= -W, \quad c \in \{c_1^1, c_4^1, c_7^1\}, v \in V^1 \\
\overline{w}_2(c,:) &= W, \quad c \in C^2 \cap \mathcal{N}_1(2|2) \\
\overline{w}_2(c,v) &= S, \quad \text{otherwise}
\end{aligned}$$

Also, at the end of the third iteration:

$$\begin{aligned}
\overline{w}_3(c,v) &= -S, \quad c \in C^1 \backslash \{c_1^1, c_4^1, c_7^1\}, v \in V^2 \\
\overline{w}_3(c,v) &= W, \quad c \in \{c_1^1, c_4^1, c_7^1\}, v \in V^1 \\
\overline{w}_3(c,v) &= S, \quad \text{otherwise}
\end{aligned}$$

At the end of the third iteration, all variables in V^1 are corrected because they receive two S and two W messages, and all variables in $V \backslash V^1$ are well decoded to the received value since they receive at most two $-S$ messages from checks in C^1, and neither $-S$ nor $-W$ messages from checks in C^2 (see Table 4.2). Hence, the decoder converges to the valid codeword at most at the end of the third iteration. This completes the Proof.

■

Note that similar conditions for a column-weight-four LDPC code of girth six to correct any weight-three error pattern, when it is decoded with Gallager B algorithm, has been found by Krishnan [100]. The conditions are that each variable subset of size 4 has at least 11 neighbors, each one of size 5 at least 13 neighbors, each one of size 6 at least

15 neighbors and each one of size 7 at least 17 neighbors. These conditions are stronger than the ones of Theorem 5. Besides, the higher is the rate of the code, the more difficult it is for the Tanner graph of the code to satisfy the expansion conditions, since the variable nodes tend to be less and less connected when the code rate increases. Hence, the weaker expansion conditions obtained for the two-bit decoder make possible the construction of higher rate codes, with weight-three error correction capability, than with the one-bit Gallager B decoder.

4.4 Asymptotic analysis

This section intends to illustrate the interest of two-bit decoders over one-bit decoders, in terms of decoding thresholds. In particular, we show that the two-bit decoder, for which expansion conditions for weight three error correction has been derived, has better thresholds than one-bit decoders, for various code rates.

4.4.1 Density evolution

$$P\{W_j = X\} = \sum_{\substack{r\in\{-C,C\},n(W),n(S),n(-W):\\ f(T,r)=X}} K_\gamma P\{R=r\} \prod_{Y\in M\setminus\{-S\}} P\{\overline{W}_{j-1}=Y\}^{n(Y)} P\{\overline{W}_{j-1}=-S\}^{n(-S)} \tag{4.2}$$

$$P\{\overline{W}_j = X\} = \sum_{\substack{n(W),n(S),n(-W):\\ g(n(-S),n(-W),n(W))=X}} K_\rho \prod_{Y\in M\setminus\{-S\}} P\{W_j=Y\}^{n(Y)} P\{W_j=-S\}^{n(-S)} \tag{4.3}$$

Asymptotically in the codeword length, LDPC codes exhibit a threshold phenomenon [10]. In other words, for α smaller than a certain threshold, it is possible to achieve an arbitrarily small bit error probability under iterative decoding, as the codeword length tends to infinity. On the contrary, for noise level larger than the threshold, the bit error probability is always larger than a positive constant, for any codeword length [10, 11].

In [11] and [10], Richardson and Urbanke presented a general method for predicting asymptotic performance of binary LDPC codes. They proved a so-called concentration theorem [11] according to which decoding performance over any random graph converges, as the code length tends to infinity, to the performance when the graph is cycle-free. Thus, relevant evaluation of performance of binary LDPC codes is possible in the limit case of infinite codeword lengths. The proposed density-evolution method consists in following the evolution of probability densities of messages along the decoding iterations. The messages in each direction are assumed to be independent and identically distributed.

For the class of two-bit decoders, we derive thresholds for different values of C and S. The code is assumed to be regular with column weight γ and row degree ρ. The numbers of W, S and $-W$ messages are denoted by $n(W)$, $n(S)$ and $n(-W)$, respectively. In the sets of equations (4.2) and (4.3), $n(W) \in [0,\ldots,d]$, $n(S) \in [0,\ldots,d-n(W)]$,

$n(-W) \in [0, \ldots, d - n(W) - n(S)]$, where d is either γ or ρ, depending on the context. The number of $-S$ messages $n(-S)$ is hence $d - 1 - n(W) - n(S) - n(-W)$, with $d = \gamma$ or ρ depending on the context. Since the messages of the graph, in each direction, are assumed to be independent and identically distributed, W_j (resp. $\overline{W}_j$) denote the random variables distributed as $w_j(v, c)$ (resp. $\overline{w}_j(c, v)$) for any pair (v, c) of connected variable and check nodes. X denotes an element of M. Also, $R \in \{-C, C\}$ denotes the random variable which corresponds to the initial value of the bit. The density evolution equations are given by the sets of equations (4.2) and (4.3), where:

$$\begin{aligned}
T &= \sum_{Y \in M} n(Y) \cdot Y \\
K_\gamma &= \binom{\gamma - 1}{n(W)} \binom{\gamma - 1 - n(W)}{n(S)} \binom{\gamma - 1 - n(W) - n(S)}{n(-W)} \\
K_\rho &= \binom{\rho - 1}{n(W)} \binom{\rho - 1 - n(W)}{n(S)} \binom{\rho - 1 - n(W) - n(S)}{n(-W)}
\end{aligned}$$

The two functions f and g are defined as follows:

$$\begin{aligned}
f : \mathbb{Z}^2 &\rightarrow M \\
f(T, r) &= \begin{cases} W \cdot sign(T), & \text{if } 0 < |T| < S \\ S \cdot sign(T), & \text{if } |T| \geq S \\ W \cdot sign(r), & \text{if } T = 0 \end{cases}
\end{aligned}$$

$$\begin{aligned}
g : \mathbb{N}^3 &\rightarrow M \\
g(n_1, n_2, n_3) &= \\
&\begin{cases} W, & \text{if } n_3 + n_2 > 0, n_2 + n_1 = 0 \bmod (2) \\ S, & \text{if } n_3 + n_2 = 0, n_2 + n_1 = 0 \bmod (2) \\ -W, & \text{if } n_3 + n_2 > 0, n_2 + n_1 = 1 \bmod (2) \\ -S, & \text{if } n_3 + n_2 = 0, n_2 + n_1 = 1 \bmod (2) \end{cases}
\end{aligned}$$

4.4.2 Thresholds of quantized decoders

Table 4.4 encompasses thresholds for various code parameters and decoding rules. Thresholds are given in probability of crossover on the BSC. Algorithm E is presented in [11]. For the two-bit decoders, the set (C,S,W) is given. When the threshold is below 0.001, $\times$ is put in the box. The code rate is defined by $1 - \frac{\gamma}{\rho}$.

We have computed thresholds for various two-bit decoders. Table 4.4 shows that the specific two-bit decoder with parameters $(C, S, W) = (2, 2, 1)$, has better thresholds than one-bit decoders Gallager A and B algorithms.

Table 4.4 : Thresholds of different decoders for column-weight-four codes with row degree ρ.

ρ	Rate	Gallager A	Gallager B	Algorithm E	
8	0.5	0.0474	0.0516	0.0583	
16	0.75	0.0175	0.0175	0.0240	
32	0.875	0.00585	0.00585	0.00935	
ρ	Rate	(1,1,1)	(1,2,1)	(1,3,1)	(1,4,1)
8	0.5	0.0467	0.0509	0.0552	0.0552
16	0.75	0.0175	0.0165	0.0175	0.0175
32	0.875	0.00585	0.00562	0.00486	0.00486
ρ	Rate	(2,1,1)	(2,2,1)	(2,3,1)	(2,4,1)
8	0.5	0.0467	0.0567	0.0532	0.0552
16	0.75	0.0175	0.0177	0.0168	0.0175
32	0.875	0.00585	0.00587	0.00568	0.00486
ρ	Rate	(3,1,1)	(3,2,1)	(3,3,1)	(3,4,1)
8	0.5	$\times$	0.0467	0.0657	0.0620
16	0.75	$\times$	0.0218	0.0222	0.0203
32	0.875	$\times$	0.00921	0.00755	0.00691
ρ	Rate	(4,1,1)	(4,2,1)	(4,3,1)	(4,4,1)
8	0.5	$\times$	$\times$	0.0486	0.0657
16	0.75	$\times$	$\times$	0.0227	0.0222
32	0.875	$\times$	$\times$	0.00871	0.00755
ρ	Rate	Dynamic two-bit decoder with $S = 2$ and $W = 1$			
8	0.5	0.0638			
16	0.75	0.0249			
32	0.875	0.00953			

However, this decoder has not the best threshold among the two-bit decoders. Indeed, we tried to achieve a trade-off between good thresholds and not too strong conditions for three error correction. Nevertheless, the method of analysis applied in the proof of the previous section is general, and can be applied to a variety of decoders to obtain similar results.

Remark: Algorithm E and the presented dynamic two-bit decoder outperform the other ones, especially for code rates $\frac{3}{4}$ (i.e., $\rho = 16$) and $\frac{7}{8}$ (i.e., $\rho = 32$). Algorithm E, described in [11], is the aforementioned decoder with erasures in the message alphabet. At each iteration, the weight affected to the channel observation (equivalent to C in the two-bit decoder) is optimized [11]. The dynamic two-bit decoder is based on the same idea: for $S = 2$ and $W = 1$, C is chosen at each iteration. The better thresholds of the presented dynamic two-bit decoder over Algorithm E indicates that it is interesting to consider decoding on a higher number of bits, even if the channel observation is still one bit, to get better thresholds.

4.5 Numerical results

We have formally proved the capability of weight-three error correction of an LDPC code satisfying conditions of Theorem 5 and decoded with the two-bit decoder with $(C, S, W) = (2, 2, 1)$. To compare this two-bit decoder with another one-bit decoder, namely Gallager B, we have plotted FER in Figure 4.5. We consider a MacKay code, with column weight four, 1998 variable nodes and 999 check nodes. The code rate is 0.89. This code has been decoded with Gallager B and the above two-bit decoder. Figure 4.5 shows that the two-bit decoder has lower FER than Gallager B decoder. In particular, we observe better waterfall performance using the two-bit decoder, and about 1dB gain in the error-floor.

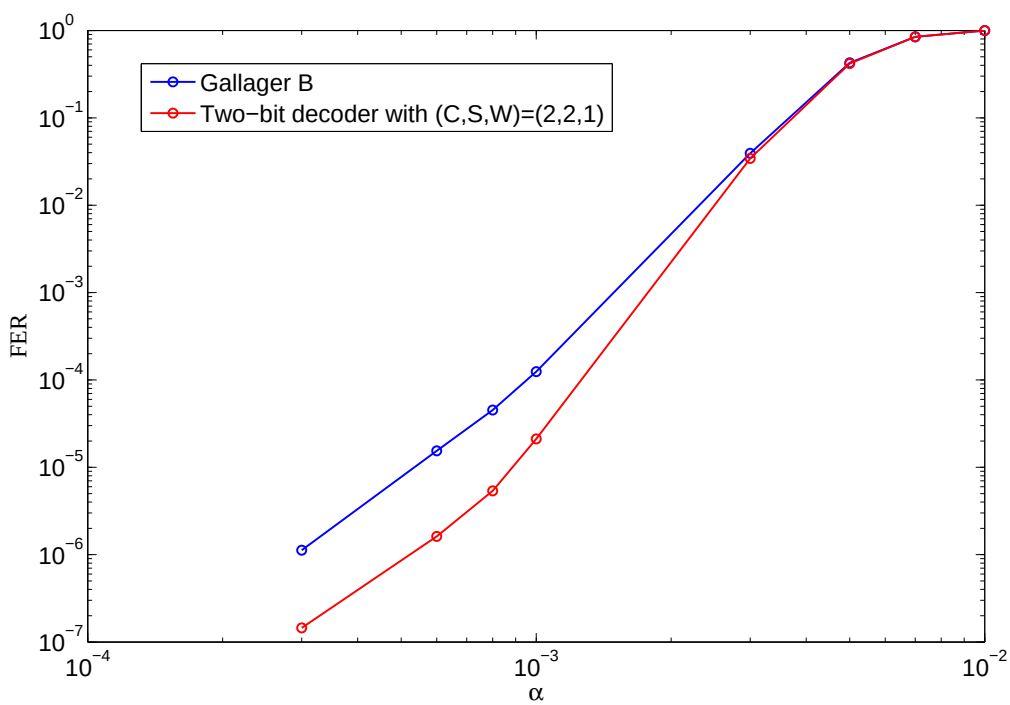


Figure 4.5 : FER versus the crossover probability α for regular column-weight-four MacKay code. The code rate is 0.89 and the code length is $n = 1998$.

4.6 Conclusion

In this chapter, we proposed a class of two-bit decoders. We have focused on a specific two-bit decoder for which we have derived sufficient conditions for a code with Tanner graph of girth six to correct any three errors. These conditions are weaker than the conditions for a code to correct three errors when it is decoded with Gallager B algorithm, which uses only one bit. Hence, two-bit decoder may allow to ensure weight-three error

correction capability for higher rate codes than one-bit Gallager type decoding. We have computed thresholds for various two-bit decoders, and shown that the decoder for which the previous conditions has been derived has better thresholds than one-bit decoders, like Gallager A and B. Finally, we have compared the frame error rate performance of the two-bit decoder and Gallager B algorithm for decoding a column-weight-four code with high rate. The two-bit decoder performs better than Gallager B both in the waterfall and in the error-floor region. This illustrates that it is interesting to use two bits rather than one bit for decoding.

Conclusions and Perspectives

Conclusions

In this thesis, we have first proposed a new class of non-binary LDPC codes, named hybrid LDPC codes. The asymptotic analysis of this new class has been carried out. Specific properties of considered hybrid LDPC code ensembles, like the Linear-Map invariance, have been expressed to be able to derive both stability condition and EXIT charts. The stability condition of such hybrid LDPC ensembles shows interesting advantages over non-binary codes. The EXIT charts analysis is performed on the BIAWGN channel. In order to optimize the distributions of hybrid LDPC ensembles, we have investigated how to project the message densities on only one scalar parameter using a Gaussian approximation. The accuracy of such an approximation has been studied, and has led to two kinds of EXIT charts for hybrid LDPC codes: multi-dimensional and mono-dimensional EXIT charts. The distribution optimization allows to get finite length codes with very low connection degrees and better waterfall region than protograph or multi-edge type LDPC codes. Moreover, hybrid LDPC codes are well fitted for the cycle cancellation presented in [34], thanks to their specific structure. Additionally to a better waterfall region, the resulting codes have a very low error-floor for code rate one-half and codeword length lower than three thousands bits, thereby competing with multi-edge type LDPC. Thus, hybrid LDPC codes allow to achieve an interesting trade-off between good error-floor performance and good waterfall region with non-binary codes techniques.

We have also shown that hybrid LDPC codes can be very good candidates for efficient low rate coding schemes. For code rate one sixth, they compare very well to existing Turbo Hadamard or Zigzag Hadamard codes. More particularly, hybrid LDPC codes exhibit very good minimum distances and error floor properties.

In the second part of the thesis, we have tried to determine which kind of machine learning methods would be useful to design better LDPC codes and better decoders in the short code length case.

We have first investigated how to build the Tanner graph of a code by pruning away edges from the Tanner graph of a mother code, using a machine learning algorithm, in order to optimize the minimum distance. We showed that no relevant cost function can be found for this problem. Hence, no pruning method could be applied. We have pointed out that this pruning problem was not a classification problem, and that is why this approach failed.

We have then investigated decoder design by machine learning methods in order to perform better than BP which is suboptimal as soon as there are cycles in the graph. We have considered the decoding of a given code as a classification problem to which a better decoder than BP may be found, in order to handle message statistical dependencies. The chosen cost function based, on the difference between an estimated mutual information and the EXIT chart, appeared to be impossible to evaluate for value of mutual information close to one.

Finally, we have investigated several classification methods to see whether they might substitute the BP decoder. We gave the fundamental reason why this is not possible: those methods are non-parametric machine learning algorithms based on the assumption that the elements to be classified, must be highly non-uniformly distributed, which is the opposite case of the channel coding problem.

Hence, we were not able to identify any contribution that machine learning methods might bring to LDPC code or decoder design.

The third part still aims at finding good decoders for finite length LDPC codes, but with also good asymptotic behavior. We have switched from continuous BP decoding to quantized decoding. The idea is still to find a decoding rule adapted to hard-to-decode topologies. We have first proposed a class of two-bit decoders and computed thresholds for various decoders in this class. Based on those thresholds, we have focused on a specific two-bit rule. We have derived sufficient conditions for a code with Tanner graph of girth six to correct any three errors. These conditions are less stringent than the conditions for a code to correct three errors when it is decoded with Gallager B algorithm, which relies on only one bit. Hence, decoding with the two-bit rule allows to ensure weight-three error correction capability for higher rate codes than the decoding with one bit, like Gallager B decoding. Finally, we have compared the frame error rate performance of the two-bits rule and Gallager B algorithm to decode a given code satisfying the conditions for weight-three error correction with both decoders. The two-bits rule decoding performs up to three decades better than Gallager B on the same code, thereby indicating that the highest weight error corrigible by the two-bits rule is higher than that of Gallager B. This illustrates how it is interesting to use two bits rather than one bit for decoding.

Perspectives

As perspectives, it would be of first interest to allow degree one variable nodes in the representation of hybrid LDPC codes, by, e.g., adopting a multi-edge type representation [27]. As shown in [30], this would allow to have better decoding thresholds, particularly for low rate codes.

This would give rise to the study and the optimization, with the same tools, of non-binary protograph-based LDPC codes or multi-edge type LDPC codes. However, the extension may be theoretically not fully straightforward as the non-zero values have to be carefully handled to define the code ensemble.

On the other hand, it would be interesting to study hybrid LDPC codes on other chan-

nels. Let us mention that we made some experiments on an AWGN channel with 16-QAM modulation. We restricted the connection profile to be regular, in order to not bias the results by the absence of specific allocation on unequally protected symbols. Only two group orders where allowed to avoid correlation between channel LLRs: $G(16)$ and $G(256)$. The optimization of fractions of variable nodes in these two different orders have been done. The results where slightly degraded compared to a $(2,4)$ $GF(256)$ LDPC codes. A study of these codes on the BEC would be also interesting, according to what has been done for D-GLDPC codes on the BEC [56], as well as for code rates higher than one-half.

The investigations on connections between machine learning algorithms and BP decoding of LDPC codes, viewed as a classification problem, gave some insights on how channel coding methods can help classification in high-dimensional massive databases, as soon as some structure or model can be assumed for elements to be classified [89, 90, 88].

In terms of quantized decoding rules as defined in the last part of the thesis, many directions are possible. First, still for column-weight four codes, it would be interesting to see what is the minimum weight of an incorrigible error pattern. The following extension would be to lead the same study to determine which two-bit rule could have the best properties in terms of decoding threshold as well as correction capability, for column-weight three codes. Finally, an aim could be to extend the set of two-bit decoding rules to similarly defined sets of rules with any given number of bits, and finding a general condition for correction capability in terms of the number of quantization bits.

Bibliography

[1] G. Liva, W. Ryan, and M. Chiani, "Design of quasi-cyclic Tanner codes with low error floors," in *Proceedings of IEEE International Symposium on Turbo Codes*, Munich, Germany, April 2006.

[2] L. Ping, W. Leung, and K. Wu, "Low-rate Turbo-Hadamard codes," *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3213–3224, December 2003.

[3] G. Yue, W. Leung, L. Ping, and X. Wang, "Low rate concatenated Zigzag-Hadamard codes," in *Proceedings of International Conference on Communications*, Istanbul, Turkey, June 2006.

[4] N. Shimanuki, B. Kurkoski, K. Yamagichi, and K. Kobayashi, "Improvements and extensions of low-rate Turbo-Hadamard codes," in *Proceedings of ISITA*, Seoul, Korea, October 2006.

[5] C. E. Shannon, "A mathematical theory of communication," *Bell Systems Technical Journal*, vol. 27, pp. 379–423,623–656, July 1948.

[6] R. Gallager, "Low-density parity-check codes," PhD dissertation, MIT press, Cambridge, Massachusetts, 1963.

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

[8] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, November 2001.

[9] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Transactions on Information Theory*, vol. 47, pp. 585–598, February 2001.

[10] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular LDPC codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, February 2001.

[11] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, February 2001.

[12] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261–1271, October 1996.

[13] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

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

[15] S. Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications Letters*, vol. 5, pp. 58–60, February 2001.

[16] K. Price and R. Storn, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal on Global Optimization*, vol. 11, pp. 341–359, 1997, code available at: http://www.icsi.berkeley.edu/ storn/code.html.

[17] P. Oswald and M. A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Transactions on Information Theory*, vol. 48, pp. 364–373, December 2002.

[18] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free tanner graphs?" *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2173–2181, 1999.

[19] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *accepted for IEEE Tansactions on Information Theory*, 2007, available at: http://arxiv.org/abs/cs.IT/0512078/.

[20] V. Chernyak, M. Chertkov, M. Stepanov, and B. Vasic, "Error correction on a tree: An instanton approach," *Physical Review Letters*, vol. 93, no. 19, p. 198, November 2004.

[21] M. Chiani and A. Ventura, "Design and performance evaluation of some high-rate irregular low-density parity-check codes," in *Proceedings of IEEE Global Telecommunications Conference*, San Antonio, USA, November 2001.

[22] C. Di, R. Urbanke, and T. Richardson, "Weight distribution of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4839–4855, November 2006.

[23] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Transactions on Information Theory*, vol. 51, pp. 386–398, January 2005.

[24] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Transactions on Information Theory*, vol. 50, pp. 1788–1793, August 2004.

[25] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proceedings of Int. Symp. on Turbo codes and Related Topics*, Brest, France, September 2000.

[26] D. Divsalar, C. Jones, S. Dolinar, and J. Thorpe, "Protograph based LDPC codes with minimum distance linearly growing with block size," in *Proceedings of IEEE Global Telecommunications Conference*, St. Louis, USA, November 2005.

[27] T. Richardson and R. Urbanke, "Multi-edge type LDPC codes," *available online*, April 2004.

[28] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes," in *Proceedings of IEEE Int. Conf. on Communications*, Vancouver, Canada, June 1999.

[29] E. Paolini, M. Fossorier, and M. Chiani, "Analysis of doubly-generalized LDPC codes with random component codes for the binary erasure channel," in *Proceedings of Allerton Conference on Communications, Control and Computing*, Monticello, USA, Sept 2006.

[30] I. Andriyanova, "Analysis and design of a certain family of graph-based codes: TLDPC," PhD dissertation, Ecole Nationale Supérieure des Télécommunications, Paris, France, 2006.

[31] M. Davey and D. MacKay, "Low density parity check codes over GF(q)," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, June 1998.

[32] M. Davey, "Error-correction using low density parity check codes," PhD dissertation, University of Cambridge, Cambridge, UK, December 1999.

[33] X.-Y. Hu and E. Eleftheriou, "Binary representation of cycle Tanner-graph GF(2^q) codes," in *Proceedings of IEEE International Conference on Communications*, Paris, France, June 2004, pp. 528–532.

[34] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2,dc)-LDPC codes over GF(q) using their binary images," *accepted in IEEE Transactions on Communications*, 2008.

[35] T. J. Richardson, "Error floors of LDPC codes," in *Proceedings of 41st Annual Allerton Conf. on Communications, Control and Computing*, 2003, pp. 1426–1435.

[36] J. G. Proakis, *Digital communications. Fourth edition.* MacGraw-Hill, 2001.

[37] F. MacWilliams and N. Sloane, *The theory of error-correcting codes*. North Holland, 1978.

[38] B. Masnick and J. Wolf, "On Linear Unequal Error Protection Codes," *IEEE Trans. on Inform. Theory*, vol. 3, no.4, pp. 600–607, Oct. 1967.

[39] S. lin and D. J. Costello, *Error-control coding*. Prentice Hall, 1983.

[40] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM Journal of Applied Mathematics*, vol. 8, pp. 300–304, 1960.

[41] A. J. Viterbi, "Error bounds for convolutional codes and asymtotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, April 1967.

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

[43] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, September 1981.

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

[45] A. Goupil, M. Colas, G. Gelle, and D. Declercq, "FFT-based BP decoding of general LDPC codes over abelian groups," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 644–649, April 2007.

[46] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low complexity, low memory EMS algorithm for non-binary LDPC codes," in *Proceedings of IEEE International Conference on Communications*, Glasgow, UK, June 2007.

[47] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, August 2005.

[48] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 549–583, February 2006.

[49] G. Li, I. Fair, and W. Krzymien, "Analysis of nonbinary LDPC codes using Gaussian approximation," in *Proceedings of IEEE International Symposium on Information Theory*, Yokohama, Japan, July 2003.

[50] S. Chung, T. Richardson, and R. Urbanke, "Analysis of sum-product decoding LDPC codes using a Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, February 2001.

[51] A. Venkiah, D. Declercq, and C. Poulliat, "Design of cages with a randomized progressive edge growth algorithm," *IEEE Communications Letters*, vol. 12, no. 4, pp. 301–303, February 2008.

[52] K. Kasai, T. Shibuya, and K.Sakaniwa, "Detailedly represented irregular LDPC codes," *IEICE Transactions on Fundamentals*, vol. E86-A, no. 10, pp. 2435–2443, October 2003.

[53] G. Liva, S. Song, L. Lan, Y. Zhang, S. Lin, and W. E. Ryan, "Design of LDPC codes: a survey and new results," *to appear in Journal on Communication Software and Systems*, 2006, available online.

[54] D. Sridhara and T. Fuja, "Low density parity check codes over groups and rings," in *Proceedings of IEEE Information Theory Workshop*, Bangladore, India, October 2002.

[55] J. Boutros, A. Ghaith, and Y. Yuan-Wu, "Non-binary adaptive LDPC codes for frequency selective channels: code construction and iterative decoding," in *Proceedings of IEEE Information Theory Workshop*, Chengdu, China, October 2006.

[56] E. Paolini, "Iterative decoding methods based on low-density graphs," PhD dissertation, Universita degli studi di Bologna, Bologna, Italia, 2007.

[57] C. Poulliat, M. Fossorier, and D. Declercq, "Design of non binary LDPC codes using their binary image: algebraic properties," in *Proceedings of IEEE International Symposium on Information Theory*, Seattle, USA, July 2006.

[58] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, pp. 670–678, April 2004.

[59] A. G. Kolpakov, "The solution of the convex combination problem," *Journal on Computational mathematics and mathematical physics*, vol. 32, no. 8, pp. 1183–1188, 1992.

[60] W. Press, S. Teukolsky, , W. Vetterling, and B. Flannery, *Numerical recipes in C. Second edition*. Cambridge University Press, 1992.

[61] S. ten Brink, "Code doping for triggering iterative decoding convergence," in *Proceedings of IEEE International Symposium on Information Theory*, Washington DC, USA, 2001.

[62] A. Brouwer and T. Verhoeff, "An updated table of minimum distance for binary linear codes," *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 662–677, March 1993.

[63] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, April 2007.

[64] A. Viterbi, "Very low rate convolutional codes for maximum theoretical performance of spread-spectrum multiple-access channels," *IEEE Journal on Selected Areas on Communications*, vol. 8, pp. 641–649, May 1990.

[65] M. González-López, F. J. Vázquez-Araújo, L. Castedo, and J. Garcia-Frias, "Layered LDGM codes: a capacity-approaching structure for arbitrary rates," in *Proc. ISWCS*, Trondheim, Norway, September 2007.

[66] X.-Y. Hu and M. Fossorier, "On the computation of the minimum distance of low-density patity-check codes," in *Proceedings of IEEE International Conference on Communications*, Paris, June 2004.

[67] T. Richardson, "in review of this paper," 2008.

[68] G. Yue, L. Ping, and X. Wang, "Low-rate generalized LDPC codes with Hadamard constraints," in *Proceedings of IEEE International Symposium on Information Theory*, Adelaide, Australia, September 2005.

[69] F. Attneave, "Informational aspects of visual perception," *Psychological Review*, vol. 61, pp. 183–193, 1954.

[70] S. Haykin, *Neural Networks. A Comprehensive Foundation*. Prentice Hall, 2005.

[71] R. Linsker, "An application of the principle of maximum information preservation to linear systems," in *Advances in Neural Information Processing Systems Conference*, Denver, USA, 1988.

[72] J. Bruck and M. Blaum, "Neural networks, error-correcting codes, and polynomials over the binary n-cube," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 976–987, September 1989.

[73] Y.-H. Tseng and J.-L.Wu, "High-order perceptrons for decoding error-correcting codes," in *IEEE International Joint Conference on Neural Networks*, vol. 3, Baltimore, USA, June 1992, pp. 24–29.

[74] A. Cornuéjols and L. Miclet, *Apprentissage artificiel. Concepts et algorithmes*. Eyrolles, 2002.

[75] L. Personnaz and I. Rivals, *Réseaux de neurones formels pour la modélisation, la commande et la classification*, ser. Sciences et Techniques de l'Ingénieur. CNRS, 2003.

[76] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the theory of neural computation*. Westview Press, 1991.

[77] B. Hassibi, D. Stork, and G. Wolff, "Optimal brain surgeon and general network pruningoptimal brain surgeon and general network pruning," in *IEEE International Conference on Neural Networks*, vol. 1, San Francisco, USA, March 1993, pp. 293–299.

[78] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error backpropagation," *Parallel distributed processing, MIT Press*, vol. 1, pp. 318–362, 1986.

[79] D. Levine, "PGApack: Parallel genetic algorithm library," in *Argonne National Laboratory*, UChicago, USA, 2000, http://www-fp.mcs.anl.gov/CCST/research/reports_pre1998/comp_bio/stalk/pgapack.html.

[80] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[81] P. Comon, "Independent component analysis: a new concept ?" *Elsevier Signal Processing. Special Issue on Higher Order Statistics*, vol. 36, no. 3, pp. 287–314, April 1994.

[82] N. Cristianini and J. Shawe-Taylor, *Support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[83] A. Guttman, "A dynamic index structure for spatial searching," in *ACM SIGMOD International Conference on Management of Data*, 1984, pp. 47–57.

[84] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," *Proceedings of the 13th annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998.

[85] C. Faloutsos and I. Kamel, "Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension," in *ACM SIGACT-SIGMOD-SIGART PODS*, Minneapolis, USA, May 1994, pp. 4–13.

[86] A. Belussi and C. Faloutsos, "Estimating the selectivity of spatial queries using the 'correlation' fractal dimension," in *International Conf. on Very Large Data Base*, Zurich, Switzerland, September 1995, pp. 299–310.

[87] S. R. Simanca and S. Sutherland, "Mathematical problem solving with computers," in *The University at Stony Brook*, ser. Lecture Notes for MAT 331, 2002, available at: http://www.math.sunysb.edu/ scott/Book331/Fractal_Dimension.html.

[88] A. Broder, R. Krauthgamer, and M. Mitzenmacher, "Improved classification via connectivity information," in *Symposium on Discrete Algorithms*, 2000.

[89] M. Naphade, I. Kozintsev, and T. Huang, "A factor graph framework for semantic video indexing," *IEEE Transactions on circuits and systems for video technology*, vol. 12, no. 13, pp. 40–52, 2002.

[90] M. Boutell, J. Luo, and C. Brown, "Factor-graphs for region-based whole-scene classification," in *International Workshop on Semantic Learning Applications in Multimedia (in conjunction with CVPR2006)*, New York, USA, June 2006.

[91] L. Bazzi, T. Richardson, and R. Urbanke, "Exact thresholds nd optimal codes for the binary-symmetric channel and gallager's decoding algorithm a," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2010–2021, 2004.

[92] V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity for gallager low-density codes," *Problems of Information Transmission*, vol. 11, no. 1, pp. 18–28, 1976.

[93] M. Sipser and D. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, November 1996.

[94] D. Burshtein and G. Miller, "Expander graph arguments for message passing algorithms," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 782–790, February 2001.

[95] M.Ivkovic, S. K. Chilappagari, and B. Vasic, "Eliminating trapping sets in low-density parity check codes using tanner graph lifting," in *Proceedings of IEEE International Symposium on Information Theory*, Nice, France, June 2007, pp. 2266–2270.

[96] C. Di, D. Proietti, T. Richardson, E. Teletar, and R. Urbanke, "Finite length analysis of low-density parity-check codes," *IEEE Tansactions on Information Theory*, vol. 48, pp. 1570–1579, June 2002.

[97] D. Burshtein, "On the error correction of regular LDPC codes using the flipping algorithm," in *Proceedings of IEEE International Symposium on Information Theory*, Nice, France, June 2007, pp. 226–230.

[98] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, "On guaranteed error correction capability of LDPC codes," in *Proceedings of IEEE International Symposium on Information Theory*, Toronto, Canada, July 2008.

[99] S. K. Chilappagari, A. R. Krishnan, and B. Vasic, "LDPC codes which can correct three errors under iterative decoding," in *Proceedings of IEEE Information Theory Workshop*, May, 2008.

[100] A. R. Krishnan, S. K. Chilappagari, and B. Vasic, "On error correction capability of column-weight-four ldpc codes," *to be submitted to IEEE Transactions on Information Theory*, September 2008.

[101] E. Sharon, A. Ashikhmin, and S. Litsyn, "EXIT functions for the Gaussian channel," in *Proceedings of 40th Annu. Allerton Conf. Communication, Control, Computers*, Allerton, IL, October 2003, pp. 972–981.

[102] C. Poulliat, M. Fossorier, and D. Declercq, "Using binary image of nonbinary LDPC codes to improve overall performance," in *Proceedings of IEEE International Symposium on Turbo Codes*, Munich, Germany, April 2006.

[103] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low complexity decoding for non-binary LDPC codes in high order fields," *accepted for publication in IEEE Transactions on Communications*, 2008.

[104] L. Sassatelli and D. Declercq, "Non-binary hybrid LDPC codes: Structure, decoding and optimization," in *Proceedings of IEEE Information Theory Workshop*, Chengdu, China, October 2006.

[105] ——, "Analysis of non-binary hybrid LDPC codes," in *Proceedings of IEEE International Symposium on Information Theory*, Nice, France, June 2007.

[106] L. Sassatelli, W. Henkel, and D. Declercq, "Check irregular LDPC codes for unequal error protection under iterative decoding," in *Proceedings of IEEE International Symposium on Turbo Codes*, Munich, Germany, April 2006.

[107] G. Byers and F. Takawira, "EXIT charts for non-binary LDPC codes," in *Proceedings of IEEE International Conference on Communications*, Seoul, Corea, May 2005, pp. 652–657.

[108] D. Declercq, M. Colas, and G. Gelle, "Regular GF(2^q)-LDPC coded modulations for higher order QAM-AWGN channels," in *Proceedings of ISITA*, Parma, Italy, October 2004.

[109] R. Gallager, "Low-density parity check codes," *IEEE Transactions on Information Theory*, vol. 39, no. 1, pp. 37–45, January 1962.

[110] O. Wintzell, M. Lentmaier, and K. Zigangirov, "Asymptotic analysis of super-orthogonal turbo codes," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 253–258, January 2003.

[111] K. Li, X. Wang, and A. Ashikhmin, "Exit functions of Hadamard components in Repeat-Zigzag-Hadamard codes," in *Proceedings of IEEE International Symposium on Information Theory*, Nice, France, June 2007.

[112] J. Hamkins and D. Divsalar, "Coupled receiver-decoders for low rate turbo codes," in *Proceedings of IEEE International Symposium on Information Theory*, Yokohama, Japan, June 2003.

[113] H. Jin and R. McEliece, "RA codes achieve awgn channel capacity," in *Proceedings of IEEE International Symposium on Applied Algebra and Error-Correcting Codes*, Honolulu, HI, November 1999, pp. 14–19.

Résumé : Codes LDPC multi-binaires hybrides et méthodes de décodage itératif

Cette thèse porte sur l'analyse et le design de codes de canal définis par des graphes creux. Le but est de construire des codes ayant de très bonnes performances sur de larges plages de rapports signal à bruit lorsqu'ils sont décodés itérativement.

Dans la première partie est introduite une nouvelle classe de codes LDPC, nommés code LDPC hybrides. L'analyse de cette classe pour des canaux symétriques sans mémoire est réalisée, conduisant à l'optimisation des paramètres, pour le canal gaussien à entrée binaire. Les codes LDPC hybrides résultants ont non seulement de bonnes propriétés de convergence, mais également un plancher d'erreur très bas pour des longueurs de mot de code inférieures à trois mille bits, concurrençant ainsi les codes LDPC multi-edge. Les codes LDPC hybrides permettent donc de réaliser un compromis intéressant entre région de convergence et plancher d'erreur avec des techniques de codage non-binaires.

La seconde partie de la thèse a été consacrée à étudier quel pourrait être l'apport de méthodes d'apprentissage artificiel pour le design de bons codes et de bons décodeurs itératifs, pour de petites tailles de mot de code.

Dans la troisième partie de la thèse, nous avons proposé une classe de décodeurs utilisant deux bits de quantification pour les messages du décodeur. Nous avons prouvé des conditions suffisantes pour qu'un code LDPC, avec un poids de colonnes égal à quatre, et dont le plus petit cycle du graphe est de taille au moins six, corrige n'importe quel triplet d'erreurs. Ces conditions montrent que décoder avec cette règle à deux bits permet d'assurer une capacité de correction de trois erreurs pour des codes de rendements plus élevés qu'avec une règle de décodage à un bit.

Mots clefs : théorie de l'information - codage correcteur d'erreur - codes LDPC - évolution de densité - apprentissage artificiel - décodage quantifié

Abstract : Multi-binary hybrid LDPC codes and iterative decoding methods

This thesis is dedicated to the analysis and the design of sparse-graph codes for channel coding. The aim is to construct coding schemes having high performance both in the waterfall and in the error-floor regions under iterative decoding.

In the first part, a new class of LDPC codes, named hybrid LDPC codes, is introduced. Their asymptotic analysis for memoryless symmetric channel is performed, and leads to code parameter optimization for the binary input Gaussian channel. Additionally to a better waterfall region, the resulting codes have a very low error-floor for code rate one-half and codeword length lower than three thousands bits, thereby competing with multi-edge type LDPC. Thus, hybrid LDPC codes allow to achieve an interesting trade-off between good error-floor performance and good waterfall region with non-binary coding techniques.

In the second part of the thesis, we have tried to determine which kind of machine learning methods would be useful to design LDPC codes and decoders well performing in the short code length case.

In the third part of the thesis, we have proposed a class of two-bit decoders. We have derived sufficient conditions for a column-weight four code with Tanner graph of girth six to correct any three errors. These conditions show that decoding with the two-bit rule allows to ensure weight-three error correction capability for higher rate codes than the decoding with one bit.

Keywords : information theory - error correcting codes - LDPC codes - density evolution - machine learning - quantized decoding