

Etude d'une certaine construction des codes definis par les graphes: codes TLDPC

Iryna Andriyanova

▶ **To cite this version:**

Iryna Andriyanova. Etude d'une certaine construction des codes definis par les graphes: codes TLDPC. domain_other. Télécom ParisTech, 2006. English. NNT : . pastel-00002465

HAL Id: pastel-00002465
https://pastel.hal.science/pastel-00002465

Submitted on 25 Jun 2007

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 Nationale Supérieure des Télécommunications
Département Communications et Electronique

Etude d'une certaine construction des codes définis par les graphes: codes TLDPC

Iryna Andriyanova

Jury:

Gérard Cohen	Président
Rudiger Urbanke	Rapporteurs
Gilles Zémor	
Jean-Claude Carlach	Examinateurs
Jean-Pierre Tillich	
Claude Berrou	Invité
Joseph Boutros	Directeur de thèse

Paris - 2006

National Superior School of Telecommunications
Communications and Electronics Department

Analysis and design of a certain family of graph-based codes: TLDPC codes

Iryna Andriyanova

Technical committee:

Gérard Cohen	President
Rudiger Urbanke	Reviewers
Gilles Zémor	
Jean-Claude Carlach	Examinators
Jean-Pierre Tillich	
Claude Berrou	Invited member
Joseph Boutros	Ph.D. supervisor

Paris - 2006

Résumé

Ce travail est consacré à l'analyse et la construction de codes définis par des graphes dans le but d'obtenir des familles de codes ayant, pour une complexité de décodage faible, de très bonnes performances pour une large plage de rapports signal-à-bruit.

Nous nous intéressons à une famille de codes que nous appelons TLDPC (pour Tail-biting Trellis Low-Density Parity Check) qui contient, comme sous-familles, à la fois les Turbo codes de Berrou et Glavieux et les codes de Gallager appelés aussi codes LDPC.

La première partie de cette thèse est consacrée à l'étude des codes TLDPC binaires. Nous nous sommes intéressés au caractère asymptotiquement bon de ces codes et avons obtenus des conditions nécessaires ou suffisantes en étudiant les polynômes énumérateurs moyens des poids d'ensembles de ces codes, ou en introduisant un certain graphe associé à ces codes dont les cycles sont reliés à des mots de code de poids potentiellement faibles.

Cela nous a donné des bornes sur la proportion des noeuds de degré 2 pour respecter le caractère asymptotiquement bon. Nous avons ensuite optimisé (avec cette contrainte sur les noeuds de degré 2) la distribution des noeuds des autres variables au moyen des courbes d'entropie (analogues des courbes EXIT ou Extrinsic Information Transfer charts) pour maximiser les performances de l'algorithme standard de décodage itératif (Belief Propagation) et nous obtenons par ce moyen de très bonnes performances.

Dans la deuxième partie de la thèse, nous étudions certains codes LDPC et TLDPC non-binaires. Nous y présentons une famille de codes TLDPC non-binaires, ayant à la fois une structure simple et de très bonnes performances de décodage itératif, dont l'un des avantages est la pente de leurs courbes de taux d'erreur dans la région de faible rapport signal-à-bruit plus forte que pour les codes binaires correspondants. Il est à noter qu'un code LDPC ayant au moins deux symboles de degré 2 par équation de parité peut être représenté comme un code TLDPC ayant des symboles de degré 1 dans sa structure. Ces codes LDPC peuvent donc être vus et décodés comme des TLDPC. Dans le cas particulier des codes de cycles d'un graphe, cette façon de faire nécessite beaucoup moins d'itérations de décodage grâce aux symboles de degré 1 sans pour autant changer le seuil de correction. En introduisant dans la structure de ces codes de cycles d'un graphe des noeuds de degré 1, nous obtenons pour le canal à effacements en autorisant une petite fraction de symboles effacés après décodage, une famille de codes dont les performances se rapprochent encore des limites théoriques de Shannon.

Abstract

This study is dedicated to the analysis and the design of sparse-graph codes in order to construct codes having high performances both in waterfall and error-floor regions under an iterative decoding algorithm of low complexity.

In particular, we explore a class of Tail-biting trellis LDPC (TLDPC) codes involving the class of turbo codes of Berrou and Glavieux as well as the class of codes of Gallager known as LDPC codes.

In the first part of the thesis, binary TLDPC codes are investigated. We found sufficient and necessary conditions to ensure that they are asymptotically good by calculating their average weight enumerator and studying a certain graph in which the cycles correspond to potentially low weight codewords. These conditions give us an upper bound on the fraction of degree-2 nodes in the Tanner graph. By keeping the fraction of degree-2 nodes below the upper bound, we optimised the degree distribution of other variable nodes by EXIT chart techniques and thus we obtained good performances under standard iterative decoding algorithm (belief propagation).

In the second part of the thesis, some non-binary TLDPC and LDPC codes are investigated. We propose a family of non-binary TLDPC codes with a very simple structure and a steep waterfall region. We also noticed that any LDPC code with at least two degree-2 symbols per parity-check equation can be represented as a TLDPC code with symbols in degree 1 in its structure. Thus, it can be decoded like a TLDPC code. In the case of cycle codes, such a decoding decreases significantly the number of iterations while the iterative decoding threshold does not seem to change. Moreover, by allowing a constant fraction of degree 1 symbols for this class of codes and a small fraction of erased bits after decoding over binary erasure channel, we obtained codes with improved iterative decoding performances.

Acknowledgements

I owe my deep gratitude to Jean-Pierre Tillich, for his guidance through the work on this thesis, and give special thanks to my supervisors from France Telecom and Telecom Paris, Jean-Claude Carlach and Joseph Boutros, for the opportunity of working on this subject.

I would also like to thank the members of the technical committee for their appreciation of the thesis and especially my reviewers for their advice and help.

My warm thanks to former and current members of the IRI group of France Telecom R&D, Projet CODES at INRIA Rocquencourt and COMELEC department at Telecom Paris as well as to my friends for their support and encouragement.

The research was supported by France Telecom R&D.

Contents

1 Introduction **11**
- 1.1 Context and historical background . . . 11
- 1.2 Motivation . . . 12
- 1.3 Outline . . . 12
- 1.4 Contributions . . . 13

2 Background **15**
- 2.1 Sparse-graph codes . . . 15
  - 2.1.1 Unstructured code ensembles . . . 16
  - 2.1.2 Structured code ensembles . . . 17
  - 2.1.3 Belief propagation decoding algorithm . . . 20
  - 2.1.4 Asymptotic analysis tools . . . 22
  - 2.1.5 Average weight distribution . . . 25

3 Family of binary TLDPC codes **27**
- 3.1 Introduction . . . 27
- 3.2 General framework for irregular code constructions and their average spectrum . . . 28
- 3.3 Tanner codes. Sufficient conditions for being asymptotically good. . . . 30
  - 3.3.1 Average spectrum of Tanner codes . . . 30
  - 3.3.2 Sufficient conditions for Tanner codes for being asymptotically good 32
- 3.4 Tail-biting trellis LDPC codes . . . 34
- 3.5 Sufficient conditions for TLDPC codes for being asymptotically good . . . 35
  - 3.5.1 Average spectrum of the TLDPC code ensemble . . . 35
  - 3.5.2 Sufficient conditions for being asymptotically good . . . 37
- 3.6 Matching condition for entropy curves of TLDPC codes on the BEC . . . 38
  - 3.6.1 TLDPC code families without bits of degree 1 . . . 38
  - 3.6.2 TLDPC code families with bits of degree 1 . . . 42
- 3.7 Necessary conditions for being asymptotically good . . . 44
  - 3.7.1 Definition of the graph of codewords of (partial) weight 2 . . . 44
  - 3.7.2 Cycles in the graph of codewords of (partial) weight 2 . . . 46
  - 3.7.3 Upper bound on the average degree of the graph of codewords of (partial) weight 2 . . . 50
  - 3.7.4 Conditions on permutation of bits of degree > 2 . . . 50
- 3.8 Particular TLDPC code families . . . 51

3.8.1 Family A . . . 52
3.8.2 Family B . . . 61
3.8.3 Introduction to TLDPC code families with bits of degree 1 . . . 67
3.8.4 Family C . . . 69
3.8.5 Family D . . . 71
3.8.6 Family E . . . 73
3.8.7 Family F . . . 75

4 Trapping sets of TLDPC codes **79**
4.1 Introduction: trapping sets of LDPC codes . . . 79
4.2 Experimental results for a family-E code . . . 81
4.2.1 Properties of decoding failures . . . 81
4.3 Generalized definition of trapping sets . . . 84
4.4 Error-floor estimation . . . 86
4.4.1 Evaluation of $\mathbf{P}\{\xi_{\mathcal{T}}\}$ in AWGN case . . . 86
4.5 Error-floor estimation of the family-E code over AWGN channel . . . 87
4.5.1 Trapping sets configurations . . . 87
4.5.2 Detection of dominant configurations . . . 88
4.5.3 Error-floor estimation results . . . 89

5 Codes over $\mathbb{F}_q$ **91**
5.1 Motivation . . . 91
5.2 State of art . . . 92
5.2.1 General definitions . . . 92
5.2.2 Definition of some useful operations over $\mathbb{R}^q$ and $\mathbb{R}^{m+1}$ . . . 93
5.2.3 LDPC codes over $\mathbb{F}_q$ . . . 95
5.3 TLDPC codes over $\mathbb{F}_q$ . . . 96
5.3.1 Definition . . . 96
5.3.2 Decoding operations for $((a,b))$ TLDPC code family . . . 98
5.3.3 Density evolution equations . . . 99
5.3.4 Results on thresholds . . . 105
5.3.5 EXIT chart for the (2,3) LDPC code ensemble and the $((1,1))_1$-TLDPC code ensemble . . . 107
5.3.6 Convergence of the (2,3) LDPC code ensemble and of the $((1,1))_1$-TLDPC code ensemble . . . 108
5.3.7 $((1,b))_1$ TLDPC codes as irregular LDPC codes with a large fraction of symbols of degree 2 . . . 110
5.3.8 Performance of some families of non-binary TLDPC codes . . . 112
5.4 Analysis of LDPC codes over $\mathbb{F}_q$ having a small constant fraction of symbols of degree 1 . . . 116
5.4.1 Code rate as a function of λ_1 . . . 117
5.4.2 Density evolution equations . . . 117
5.4.3 Obtained thresholds . . . 118

6 Conclusions and perspectives **121**
6.1 Conclusions . . . 121
6.2 Future work . . . 122

A **125**
A.1 APP modified decoding algorithm . . . 125

B **127**
B.1 Optimisation of the degree distribution $\Lambda(x)$ to maximise the iterative decoding threshold . . . 127
B.2 Optimisation of degree distributions for the binary erasure channel . . . 128
B.2.1 Entropy curves of base codes for families A and B . . . 128
B.3 Comparison of different types of channel . . . 129

C **131**
C.1 Trapping set configurations observed during simulations . . . 131
C.2 Low-weight codewords . . . 131

D **133**
D.1 Properties of $\boxtimes$ and $\boxdot$ operations . . . 133
D.2 Computation of (5.8) in Lemma 19 . . . 133
D.3 Entropy curve of the $((1,1))_1$-TLDPC base code . . . 134

E **137**
E.1 Noisy channel coding theorem for binary erasure channel . . . 137

List of abbreviations

3GPP	:	3rd Generation Partnership Project
AWGN	:	Additive White Gaussian Noise
BCJR	:	Bahl-Cocke-Jelinek-Raviv algorithm
BEC	:	Binary Erasure Channel
BER	:	Binary Error Rate
BSC	:	Binary Symmetric Channel
EXIT	:	EXtrinsic Information Transfer chart
GEXIT	:	Generalized EXtrinsic Information Transfer chart
LDPC	:	Low-Density Parity-Check codes
MAP	:	Maximum A Posteriori
MBIOS	:	Memoryless Binary Input-Output Symmetric channel
ML	:	Maximum Likelihood
MSE	:	Mean Square Error
PEG	:	Progressive Edge Growth
RA	:	Repeat-Accumulate codes
SNR	:	Signal-to-Noise Ratio
SP	:	Sum-Product
TLDPC	:	Tail-biting Low-Density Parity-Check codes
WER	:	Word Error Rate

Chapter 1

Introduction

1.1 Context and historical background

In his paper [64] Shannon stated the result that reliable communication is only possible at rates up to channel capacity. This means that there exists a channel code and a decoding algorithm that can achieve any designed word error probability as long as the rate of the channel code is smaller than the channel capacity. On the other hand, if the rate is larger than the capacity, the word error probability goes to one exponentially.

The proof of the above result only shows the existence of such a channel code and such a decoding algorithm but says nothing about the code construction or, more importantly, how to decode the code efficiently.

All the approaches proposed since the appearance of Shannon's theorem can be classified in two big groups: "classical" approach or "modern" approach. For a "classical" approach it is typical to find a code with some desirable properties, e.g. large minimum distance, and then to look for a decoding algorithm for the given code. In the "modern approach" the situation is reversed: given an iterative decoding algorithm one seeks codes which work well with this algorithm.

For "modern" codes the underlying objects are so called graph-based codes.

We give a brief description of the historical development of "modern" channel coding techniques. At the beginning of the 1960s Gallager presented the so called low-density parity-check (LDPC) codes [36] and proposed an iterative decoding algorithm for them. However, at that time LDPC codes received little attention. In the early 1980s Tanner introduced bipartite graphs to describe a class of codes[1] including LDPC codes and the sum-product algorithm based on these graphs [67]. The publishing of turbo codes by Berrou *et al.* in 1993 [11] had a big impact on the channel coding community.

Since then there has been a lot of research activity and many improvements in the area of codes defined on graphs. Undoubtedly, research on LDPC codes has played a central role in this field, as many of the new classes of codes which are defined on graphs are influenced by the structure of LDPC codes. Some of the key improvements in the field of graphical codes and their importance are highlighted below

- Irregular LDPC codes: Luby *et al.* [49] introduced irregular LDPC codes and showed that they can attain the channel capacity on the erasure channel. The discovery of

[1] Tanner codes in further

irregular LDPC codes influenced considerations of irregular structures for other codes defined on graphs.

- Density evolution: Richardson, Shokrollahi and Urbanke [57] proposed a method called density evolution to analyze arbitrarily long LDPC codes.

- Extrinsic Information Transfer (EXIT) chart analysis: EXIT chart analysis [68] is similar to density evolution, except that it follows the evolution of a single parameter that represents the density of messages. This evolution can be visualized in a graph called an EXIT chart. EXIT charts have become very popular, as they provide deep insight into the behaviour of iterative decoders.

The research in the area of graphical codes is still very active and there are many open problems under study, for example how to attain capacity on other channels or to find a code performing well on several different channels.

1.2 Motivation

Several different coding schemes such as turbo-codes, low-density parity check codes, repeat-accumulate codes or product codes are able to operate with reasonably low error probability after iterative decoding at rates extremely close to the Shannon limit. However, it seems that in all cases, this comes at the cost of having a poor minimum distance, which itself gives a rather high error floor. This turns out to be quite apparent in the schemes which are the closest to the Shannon limit (either irregular LDPC codes [22] or irregular turbo-codes [15]).

On the other hand, there are code families which are asymptotically good (in the sense of having minimum distance linear in the length of the code) which can be successfully iteratively decoded, for instance LDPC codes with a small enough fraction of edges of left degree 2 [29, 30] or Tanner codes built from local codes of distance greater than 2 [16], but they are only able to operate with vanishing error probability after iterative decoding at rates that are much further from the Shannon limit than the aforementioned constructions.

This raises the issue of whether or not asymptotically good codes can approach capacity with low complexity iterative decoding.

The aim of this thesis is to propose an alternative code family (different from turbo codes or LDPC codes) which, with a low complexity iterative decoding algorithm, could simultaneously approach the Shannon capacity and exhibit a very low error-floor.

1.3 Outline

We explore a particular family of codes which we call **Tail-biting Trellis Low-Density Parity Check** codes (TLDPC).

Since we want to obtain a good code behaviour in the error-floor region, we are interested in the members of the family which are asymptotically good and we look for conditions which ensure such a property.

Our codes are defined by using two constituents: a base code and a bipartite graph. Therefore, we find to find what conditions on the constituents will ensure that the overall

code is asymptotically good. Sufficient conditions for the base code side were presented in [71], in the next Section we give more details about it. As for the bipartite graph, we develop these constraints in Section 3.7.

Our codes being irregular, we optimize degree distributions of code ensembles to minimize the threshold of the code family and to maximize the design code rate by using EXIT chart techniques [69].

We present six different TLDPC code families and their performances in Section 3.8.

Two of these code families experience error-floors at word error-rate $10^{-5} - 10^{-6}$ which are caused by so called trapping set configurations [56]. In Chapter 4 we investigate the structure of trapping sets of two code families and their failure probabilities that permits us to estimate their error-floor without performing Monte-Carlo decoding simulations.

Further investigations show that all six proposed TLDPC families perform very well at rates from 1/10 till 1/2 because of their base code structures which seem to be very well adapted to the above rates. To construct a high-rate TLDPC code family (from the rate 2/3 and higher), we need to find an appropriate base code (whose rate must be higher than the given rate of the code family we want to construct) which could be decoded with a low complexity iterative decoding algorithm and would be asymptotically good. There are different ways to obtain such a high-rate base code, for example to make its structure more complex by increasing the number of states of its trellis.

To enlarge the number of possible constructions of the base code and to make an optimum choice between the decoding complexity and the performance, we do not restrict ourselves to the binary field anymore and we work over the class of non-binary codes. The question arising with the use of non-binary base codes is their asymptotic analysis (i.e. optimisation of degree distribution of TLDPC code families, computation of thresholds). We perform the asymptotic analysis of non-binary TLDPC codes and we describe density evolution operations for them in Section 5.3.

A family of non-binary TLDPC codes with symbols of degree 1 happens to have the same iterative decoding thresholds for different alphabets as the non-binary cycle code ensemble. We study whether such TLDPC and LDPC constructions represent the same code.

We finish by investigation of non-binary LDPC codes with a constant fraction of symbols of degree 1 to study their influence on iterative decoding threshold of the code ensemble.

1.4 Contributions

In the present thesis the following contributions are made:

- The necessary condition to avoid small weight codewords in graph-based codes is to choose the bipartite structure in such a way that a certain graph contains only cycles of linear size. This graph is generalized over all graph-based codes and it is called the **graph of codewords of weight 2**. With the help of the graph of codewords of weight 2 it can be easily shown that the minimal distance of turbo codes is logarithmically bounded. We compute the upper bound on the average degree of the graph of codewords of weight 2 above of which the associated graph-based code will contain codewords of logarithmic size.

- The family of binary TLDPC codes is proposed. It combines several interesting features when the parameters of the members are well chosen, namely:
  - a very simple structure which yields low-complexity iterative decoding,
  - a minimum distance linear in the length of the code,
  - good performances after iterative decoding which beat standard turbo-codes for code lengths of several thousand for instance. We present examples of members of this class with rates 1/2, 1/3, 3/10, 1/4 and 1/10 where this fact is striking.
- One of the presented TLDPC code families is proven to have typical minimum distance linear in the codelength if its graph of codewords of weight 2 does not contain cycles.
- We construct families of TLDPC codes having bits of degree 1 in their structure. We investigate the influence of degree-1 bits on EXIT charts of code families and we make a structured choice of the permutation to avoid codewords of logarithmic length.
- An algorithm for estimating the error-floor of TLDPC codes is presented. As such an error-floor phenomenon is due to the presence of trapping sets in the Tanner graph, the algorithm is based on detecting potential trapping sets of small size and then on evaluating their failure probability.
- The density evolution equations for non-binary TLDPC codes, when the transmission takes place over the binary erasure channel, are defined.
- Any LDPC code, binary or not, with a large enough fraction of degree-2 symbols can be represented as a TLDPC code with symbols of degree 1 and vice versa. Such TLDPC representation gives a fast convergence decoding algorithm of the LDPC codes.

  It was observed that thresholds for non-binary LDPC codes $(2, 3)$ and $(2, 4)$ under both standard and TLDPC-like decoding algorithms coincide.
- Performances of TLDPC codes over larger alphabets are presented.
- Iterative decoding thresholds of some non-binary LDPC code ensembles with a small constant fraction of symbols of degree 1 are computed.

Chapter 2

Background

The goal of this chapter is to review the necessary background on sparse-graph code families, structured and unstructured, iterative decoding algorithms and the existing analysis methods in the infinite-length case.

2.1 Sparse-graph codes

Very often a bipartite graph representation is used to describe graph-based codes. Behind this representation there is an efficient decoding algorithm.

A code may be characterized as the set of configurations that satisfy a certain set of constraints. For example, a linear code may be characterized as the set of configurations that satisfy a certain set of parity checks. Such a representation was proposed by Gallager [36]. Generally, such a representation is specified by local constraints where each constraint involves a subset of code bits and defines a set of valid local configurations [67].

It also has a graphical model called a Tanner graph. A Tanner graph is a bipartite graph in which a first set of vertices represents the bit variables , a second set of vertices represents the local constraints, and a variable node is connected to a constraint node by an edge if the corresponding bit is involved in the corresponding local constraint.

A code is a sparse-graph if the number of edges in its corresponding Tanner graph is linear in the number of nodes. In the following sections only sparse-graph codes are considered.

For a Tanner graph of a code, we define a degree of a variable (constraint) node to be the number of constraint (variable) nodes connected to it. To each variable (constraint) node of degree l we assign l sockets. The total number m of sockets of variable nodes is equal to the total number of sockets of constraint nodes. The sockets of variable and constraint nodes are matched by a random permutation chosen with uniform probability among a set of permutations of size m. Sets of variable nodes V and of constraint nodes C of a Tanner graph together with the set of allowed edge permutations define a code ensemble.

An *unstructured* code ensemble is a code ensemble defined over all the possible socket matchings.

2.1.1 Unstructured code ensembles

In this section we review some well-known unstructured code ensembles. We begin with the LDPC code family.

LDPC codes

An LDPC code is a linear block code with a sparse parity-check matrix. In the Tanner graph, constraint nodes represent parity-check equations from the parity-check matrix of the code and are usually called check nodes.

An LDPC code ensemble is (d_v, d_c)-regular if all the variable and check nodes have corresponding degrees d_v and d_c. For a regular LDPC code the minimum code rate (also called design code rate) is

$$R = \frac{n-r}{n} = 1 - \frac{d_v}{d_c}.$$

An LDPC code is irregular if not all of the variable (check) nodes have equal degree. By carefully designing the irregularity in the graph, one can construct codes which perform very close to the capacity [49].

Let $\lambda = (\lambda_2, \ldots, \lambda_s)$ be a probability distribution over the set of integers $\{2, \ldots, s\}$ and let $\rho = (\rho_3, \ldots, \rho_l)$ be a probability distribution over the set of integers $\{3, \ldots, l\}$. Let $\bar{\lambda} \stackrel{\text{def}}{=} \frac{1}{\sum_i \lambda_i / i}$ and let $n\bar{\lambda} = m$ for some positive integer n. Then an ensemble of irregular LDPC codes of length n is defined by its variable (left) degree distribution λ and its check (right) degree distribution ρ if $\lambda_i m/i$ and $\rho_j m/j$ are integers for any $i \in \{2, \ldots, s\}$ and $j \in \{2, \ldots, l\}$. Note that λ_i denotes the fraction of edges incident on variable nodes of degree i and ρ_j denotes the fraction of edges incident on check nodes of degree j, $\lambda_i m/i$ ($\rho_j m/j$) is the fraction of variable (check) nodes of degree i (j), $\bar{\lambda}$ ($\bar{\rho} \stackrel{\text{def}}{=} \frac{1}{\sum_j \rho_j / j}$) represents the average left (right) degree of the corresponding Tanner graph.

It is also possible to describe the same code ensemble by representing the sequences λ and ρ by the following polynomials $\lambda(x) = \sum_{i>2} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{j>3} \rho_j x^{j-1}$.

Given the degree distribution for an irregular LDPC code, it is easy to find its design code rate:

$$R = 1 - \frac{\sum_j \frac{\rho_j}{j}}{\sum_i \frac{\lambda_i}{i}} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}.$$

Finding a good family of irregular LDPC codes of infinite length is equivalent to finding a good degree distribution.

For n large enough, the average behaviour of almost all instances of the regular or irregular LDPC ensemble concentrates around the expected behaviour [57]. Additionally, the expected behaviour converges to the cycle-free case [57].

Tanner codes

This class of codes was first presented by Tanner in [67].

A Tanner code of length n is defined with the help of $\rho_1 r$ local codes C_1 of length n_1, $\rho_2 r$ local codes C_2 of length n_2,..., $\rho_t r$ local codes C_t of length n_t[1], and of the bipartite graph

[1] $\sum_1^t \rho_i = 1$

$G = (V, W)$ with V to be a set of variable nodes and W to be a set of constraint nodes, each constraint node representing a local code, $|V| = n$ and $|W| = r\sum_{i=1}^{t} \rho_i n_i = m$. A left degree distribution $\lambda(x) = \sum_{i=1}^{s} \lambda_i x^{i-1}$ and a right degree distribution $\rho(x) = \sum_{i=1}^{t} \rho_j x^{n_j - 1}$ are respectively associated with variable and local code nodes.

If a Tanner code is constructed by using r identical local codes, its right degree distribution only contains one degree, and the code is called simple.

Remark that LDPC codes are a particular case of Tanner codes when it has a parity code as the local code.

2.1.2 Structured code ensembles

It was already pointed out that for unstructured code ensembles no restriction is put on the permutation choice.

In order to have iterative decoding performances close to channel capacity, we need to have a large fraction of degree-2 variable nodes in the code structure. In particular, all the sequences of LDPC codes in [65] attaining the capacity on the erasure channel, satisfy

$$\lambda_2 \to \frac{1}{p(\bar{\rho} - 1)}.$$

On the other hand, a large fraction λ_2 gives rise to a rather high error floor as the error-floor phenomenon can be caused by low-weight codewords corresponding to cycles in the subgraph of the Tanner graph which contain only degree-2 variable nodes. To avoid such cycles, the following condition should be satisfied:

$$\lambda_2(\bar{\rho} - 1) < 1.$$

Then the typical minimum distance of the code ensemble is linear in the codelength and the error-floor for a code from the code ensemble is low. In the opposite case, when $\lambda_2(\bar{\rho} - 1) > 1$, the typical minimum distance of the code ensemble is logarithmic in the codelength.

But it does not mean that code ensembles with linear minimum distance for which $\lambda_2 > \frac{1}{\bar{\rho}-1}$ do not exist, however, these code ensembles are rare and cannot be found by random choice of permutation. So, to construct code ensembles with iterative decoding performances close to channel capacity and having a low error-floor, we need to choose permutations in a structured way in order to avoid low-weight codewords of sublinear size.

The code ensembles with a structured choice of permutation are called structured. We give some examples of structured code ensembles below.

Parallel turbo codes

The first example of structured ensembles are the parallel turbo-codes. Their factor graph contains two types of variable nodes, information and redundancy nodes, and two constraint nodes, corresponding to two convolutional codes C^1 and C^2 of the same length m and rate R_C. For the constraint node i, $i = 1, 2$, we distinguish two subsets of information and redundancy positions C_I^i and C_R^i which correspond to information and redundancy

bits of the convolutional code C^i. Similarly, we denote the subset of information variable nodes by V_I and the subset of redundancy variable bits by V_R. Then we define a permutation $\Pi = \{\Pi_I, \Pi_R\}$ as follows:

$$\begin{aligned} \Pi_I : V_I &\mapsto C_I^1 \\ V_I &\mapsto C_I^2 \\ \Pi_R : V_R &\mapsto C_I^1 \cup C_I^2 \end{aligned}$$

In other words, two edge subsets are defined. The first subset connects information variable nodes to information positions of constraint nodes and the second one connects redundancy variable nodes to redundancy positions of constraint nodes.

The structured code ensemble of parallel turbo codes with two convolutional codes C^1 and C^2 is an ensemble of codes obtained by all possible random permutations within every edge subset.

Note that it is not possible to avoid cycles of logarithmic size for parallel turbo codes, so their minimum distance is logarithmic in the code length [17]. However, they have a very large fraction of degree-2 variable nodes which determines their good performances in the waterfall region.

Repeat-accumulate codes

An important example of structured ensembles is the repeat-accumulate (RA) code ensemble [31]. Its Tanner graph contains two subsets of variable nodes, information V_I and V_P parity nodes, as well as check nodes C. The common degree of the information variable nodes is d_v, the degree of the parity variable nodes is 2 and the check node degree is d_c. The number of check nodes for RA codes is equal to the number of parity variable nodes. Let $|C| = |V_P| = r$. We distinguish two edge subsets, E_I and E_R, the first one connecting information variable nodes to check nodes, and the second one connecting parity variable nodes to check nodes.

We make a structured choice of permutation for E_R by connecting each of parity variable nodes to exactly two check nodes in such a way that parity variable nodes and check nodes form a cycle of length $2r$. The permutation for E_I is chosen uniformly at random.

Note that RA codes do not have cycles of logarithmic size with degree-2 variable nodes only in their structure. As there is a large fraction of degree-2 variable nodes, they have very good performances in the waterfall region.

LDPC codes constructed from protographs

The next example of structured code ensembles are codes constructed from protographs [70].

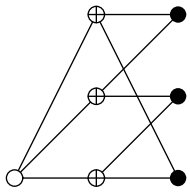
A protograph is a bipartite graph G with a relatively small number of variable and check nodes.

Definition 1 [Edge type] *We say that an edge in a Tanner graph is of type $t_{i,j}$ if it connects a variable node of degree i and a check node of degree j. If there are n variable nodes and m check nodes, the number of all possible edge types is mn.*

A LDPC code of length Nn constructed from the protograph G is defined by its Tanner graph, when the Tanner graph is obtained by the following way:

- N copies of G are made,
- edges of the same edge type are randomly permuted.

Necessary conditions for being asymptotically good and examples of asymptotically good code ensembles constructed from protographs are presented in [25].
Note that RA codes can also be viewed as codes constructed from protographs. For example, the protograph of a rate-1/3 RA code with $d_v = 3$ is presented in the figure on the right.



Circles represent variable nodes (filled circles represent parity variable nodes) and $\oplus$ represent check nodes. During the transmission, only values of parity variable nodes are sent to the channel.

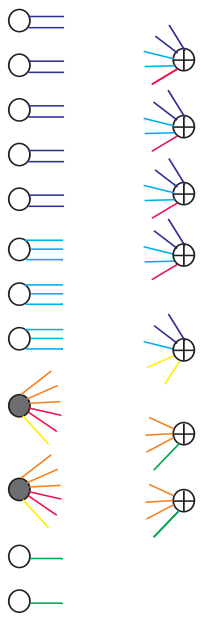
Multi-edge LDPC codes

Irregular LDPC codes and LDPC codes based on protographs are two particular cases of multi-edge type codes introduced in [59]. In general, the structure of multi-edge codes is a modification of the structure of irregular LDPC codes for which

- different types of sockets for variable and check nodes are defined and only sockets of the same type are matched,
- hidden variable nodes are allowed.

Let G be a Tanner graph of a multi-edge LDPC code ensemble. As it was already said, constraint nodes of G are parity check nodes. Variable nodes of G are divided in two subsets: the nodes corresponding to symbols transmitted over the channel and the nodes, called hidden, corresponding to non-transmitted symbols.

To each variable (check) node of degree l we assign l sockets. To each socket a type i is also assigned. Let the total number of socket types be t. Then to each variable (check) node we associate a vector D of length t, the i-th element of which is the number of sockets of type i assigned to it. The type of a variable (check) node is determined by its vector D.
We match sockets of variable and check nodes by random permutation if they are of the same type. Thus, the Tanner graph of the code contains edges of t types each of them corresponding to a socket type.
An example of a multi-edge structure taken from Table 8 in [59] is presented in the figure at right. Unfilled circles represent transmitted variable nodes, filled circles represent hidden ones, $\oplus$ represent check nodes. There are 6 socket types, each of them represented by different color. Let the vector D be { blue, cyan, orange, red, yellow, green}. Then variable nodes have four types, namely $\{2,0,0,0,0,0\}$, $\{0,3,0,0,0,0\}$, $\{0,0,3,2,1,0\}$ and $\{0,0,0,0,0,1\}$.



Check nodes have three types: $\{2,2,0,1,0,0\}$, $\{2,1,0,0,1,0\}$ and $\{0,0,3,0,0,\ 1\}$. The number of sockets of given type for variable nodes and for check nodes is the same and they are matched by random permutation.

Different socket types impose substructures in the Tanner graph which can have a positive effect on the typical minimum distance of the code ensemble. In the multi-edge structure presented as an example, each check node will be connected to at most 2 variable nodes of degree 2. Therefore, by matching sockets of type 'blue' carefully, the subgraph of the Tanner code induced by variable nodes of degree 2 will not contain cycles and the code will not have low-weight codewords corresponding to such cycles. Another advantage is the possibility to include bits of degree 1 (which have a positive effect on the convergence and performances of the iterative decoding in the waterfall region) without harming much the typical minimum distance of the code ensemble. For instance, from the figure of the example it is seen that two bits of degree 1 can not be connected to the same parity check node and this avoids having codewords of weight 2 in the code.

Note that the standard irregular LDPC code ensemble is a particular case of the multi-edge structure when all variable nodes are transmitted over the channel and there is only one edge type, i.e. any variable node can be connected with any check node when the random permutation of edges is performed.

The protograph ensemble corresponds to a particular case of multi-edge LDPC ensembles when the node and edge types are determined by nodes and edges in its protograph.

2.1.3 Belief propagation decoding algorithm

Consider a transmission over a noisy channel. Let $\mathbf{X}$ be a random vector on its input and let $\mathbf{Y}$ be a random vector on its output. 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 $\mathbf{P}_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y})$. 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

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

The ML decoder is optimum in terms of word error probability. To minimize the bit error probability for a bit x_i, we maximize $\mathbf{P}(x_i = x|\mathbf{y})$ over all x. The decoder maximizing this quantity is called maximum a posteriori (MAP) decoder and the i-th position of its output is written as follows

$$\hat{x}_i = \arg\max_{x\in\mathcal{A}} \mathbf{P}((x_i = x|\mathbf{y}),$$

the maximisation is done over the input alphabet of the channel. In what follows we consider the binary alphabet.

ML decoding becomes exponentially difficult as the codelength becomes large. To decode graph-based codes, we use hard or soft iterative decoding algorithms which operate on their graphs. The decoding complexity of such algorithms per iteration is linear in the codelength. Soft iterative decoding algorithms can be viewed as applying Bayes' rule locally and iteratively to calculate approximate marginal a posteriori probabilities for

graph-based codes. If the graph has no cycles then these algorithms compute marginal posterior probabilities exactly.

Let us describe in general decoding operations for LDPC codes. Under an iterative decoding algorithm, variable and check nodes exchange messages iteratively. The messages in this algorithm are probabilities. The message passed from a variable node v to a check node c is the probability that v has a certain value given the value received from the channel and all the values communicated to v on the previous semi-iteration from check nodes incident to v other than c. The message passed from c to v is the probability that v has a certain value given all the messages passed to c on the previous semi-iteration from variable nodes other than v. This two-step procedure is repeated many times. After n such iterations, each variable node decodes its associated bit based on all information obtained from its depth-n subgraph of neighbours. Under local tree assumption, i.e. if the girth of the subgraph is large enough, the subgraph has no repeated nodes and thus forms a tree. Therefore, all the incoming messages to any node are statistically independent. The notion of independence is very important while doing the analysis of the average behaviour of a code ensemble. Even it is not the case, the local subgraphs contain cycles and incoming messages are dependent, we assume their independence when analysing the code ensemble.

The decoding procedure described above also holds when we have a code ensemble for which variable nodes are not connected to separated parity-check nodes but to one or several codes with more complicated relations between the bits (for example, convolutional codes in the turbo code structure). During the code ensemble analysis we also assume the independence between incoming messages [76]. Let us describe a decoding algorithm called the sum-product (SP) [67, 76, 33], also known as belief propagation [43] in more general case than LDPC codes when there is only one constraint node representing a code of length n bigger that the length of the global code n'. We denote by m_i the degree of the i-th variable node, $i = 0, \ldots, n'-1$. Let $q_i^0 = P(x_i = 1|y_i)$ be a priori probabilities received from the channel, $i = 0, \ldots, n'-1$, and let t_{max} be the maximum number of iterations. We denote by p_j^t and q_j^t extrinsic and intrinsic probabilities of the i-th bit of the constraint code at the t-th iteration, $j = 0, \ldots, n-1$. q_i^{final} denotes the a posteriori probability of the i-th bit at the decoder output, $i = 0, \ldots, n'-1$. The stages of the SP algorithm are represented as follows:

1.	Put the number of iterations $t = 1$.
2.	Compute extrinsic probabilities $p_j^t = P(x_j = 1 \vert y_0, y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_{n-1})$ with the help of the modified APP algorithm (Section A.1).
3.	If $t = t_{max}$ go to stage 6.
4.	Update intrinsic probabilities: $q_j^t = \frac{q_i^0 \cdot \prod_{k=1\ldots m_i, k\neq j} p_k^t}{q_i^0 \cdot \prod_{k=1\ldots m_i, k\neq j} p_k^t + (1-q_i^0) \cdot \prod_{k=1\ldots m_i, k\neq j} (1-p_k^t)}$
5.	$t = t + 1$. Go to stage 2.
6.	Compute q_i^{final}: $q_i^{final} = \frac{q_i^0 \cdot \prod_{j=1\ldots m_i} p_j^t}{q_i^0 \cdot \prod_{j=1\ldots m_i, j\neq i} p_j^t + (1-q_i^0) \cdot \prod_{j=1\ldots m_i} (1-p_j^t)}$

2.1.4 Asymptotic analysis tools

An iterative decoder at each iteration uses two sources of knowledge about the transmitted codeword: the information from the channel (the intrinsic information) and the information from the previous iteration (the extrinsic information). During decoding iterations only the extrinsic information changes while the intrinsic information is fixed. In all methods of analysis of iterative decoders, the statistics of the extrinsic messages at each iteration are studied. Studying the evolution of the probability distribution of extrinsic messages iteration by iteration is the most complete analysis (known as density evolution). However, as an approximate analysis, one may study the evolution of a representative of this density.

Density evolution

In his initial work, Gallager provided an analysis of the decoder in the case of binary regular LDPC codes. The main idea of his analysis is to characterize the error rate of the messages in each iteration in terms of the channel situation and the error rate of messages in the previous iteration. This analysis is based on the tree assumption, i.e. for a fixed iteration number, any message, computed at some node on this iteration, is based on the neighbouring graph[2] which is a tree.

In 2001, Richardson and Urbanke extended the main idea of LDPC code analysis used for Algorithm A and B and also BEC-decoding to other decoding algorithms [60]. Considering the general case, where the message alphabet is the set of real numbers, they proposed a technique called density evolution, which tracks the evolution of the probability distribution of the messages, iteration by iteration.

To be able to define a density for the messages, they needed a property for channel and decoding, called the symmetry conditions. The symmetry conditions require the channel and the decoding update rules to satisfy symmetry properties. As for channel symmetry, the channel is said to be output-symmetric if $\mathbf{P}_{Y|X}(y|x = 1) = \mathbf{P}_{Y|X}(-y|x = -1)$. Concerning decoding update rules symmetry, a variable node symmetry and a function node symmetry must hold. Details can be found, for instance, in [58].

Under the symmetry conditions, the convergence behaviour of the decoder is independent of the transmitted codeword, and we can assume that the all-zero codeword is transmitted.

Furthermore, by the general concentration theorem [60], we are assured that, for almost all randomly constructed codes and for almost all inputs, the decoder performance will be close to the decoder performance under the local tree assumption with high probability, if the block length of the code is long enough.

Given the probability distribution of received messages (since we know the channel) and update rules for function and variable nodes, we are able to compute the probability distributions of the outcoming messages first for function nodes and then for variable nodes. The process can be continued for n iterations.

The probability distribution of a message will also give us the corresponding error probability. It was proved in [60] that there exists a worst case channel condition for

[2] this neighbouring graph is also called the computation graph

which, the message error rate approaches zero as the number of iterations approaches infinity. This channel condition is called the iterative decoding threshold of the code.

On the BEC, the density evolution does not involve updating probability distributions, but simply a single erasure probability.

EXIT chart

In extrinsic information transfer (EXIT) chart analysis, instead of tracking the density of messages, we track the evolution of a single parameter, iteration by iteration. This term is usually used in literature when mutual information is the parameter the evolution of which is tracked. The tool has been introduced by ten Brink (see for instance [68]). Alternatively, one can also track the SNR of the extrinsic messages [37] or their error probability [6] as well as the mean square error (MSE) of messages [14].

When we use EXIT charts in further work, we track down during the iterations entropy curves, i.e. the evolution of the expectation of the average entropy of the extrinsic probability of a bit versus the average entropy of the intrinsic probability of a bit. As usual, we define the binary entropy function by $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$. Definitions of entropy curves used in this thesis are given in Section 3.6.

Note that for the erasure channel, the average (extrinsic or intrinsic) entropy is given by a single value, and entropy curves can be defined rigorously without making any assumption. On the Gaussian channel however, the average entropy is given by a probability density function. It is generally assumed that the log-likelihood ratios of the extrinsic and intrinsic probabilities conditioned on the value of the bit under consideration have Gaussian distributions and that, in the case the all zero codeword was sent, the log-likelihood ratios of the extrinsic (or intrinsic) probability are distributed as Gaussian variables with mean μ and variance $\nu^2 = 2\mu$. Under these assumptions, the probability density function defining the average entropy is characterised by the single parameter μ and the expected value of the average entropy is given by

$$f(\mu) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{4\pi\mu}} h(x) e^{\frac{(x-\mu)^2}{4\mu}} dx.$$

We define two entropy curves. The first one summarizes how the extrinsic probabilities behave in terms of the intrinsic probabilities and is defined as a function $g(f, f_0)$, f_0 being the initial parameter of the transmission channel. The second entropy curve summarizes how the intrinsic probabilities behave in terms of the extrinsic probabilities and is given by $h(f, f_0)$. These two entropy curves predict that at the t-th iteration the expectations of the average extrinsic entropies E_t and the average intrinsic entropies I_t are given by

$$\begin{array}{rcl} I_0 & = & f_0 \\ E_1 & = & g(f^{-1}(I_0), f_0) \\ I_1 & = & h(f^{-1}(E_1), f_0) \\ \cdots & \cdots & \cdots \\ E_t & = & g(f^{-1}(I_{t-1}), f_0) \\ I_t & = & h(f^{-1}(E_t), f_0) \end{array}$$

We illustrate this with an example in the figure below. In other words the entropy curves

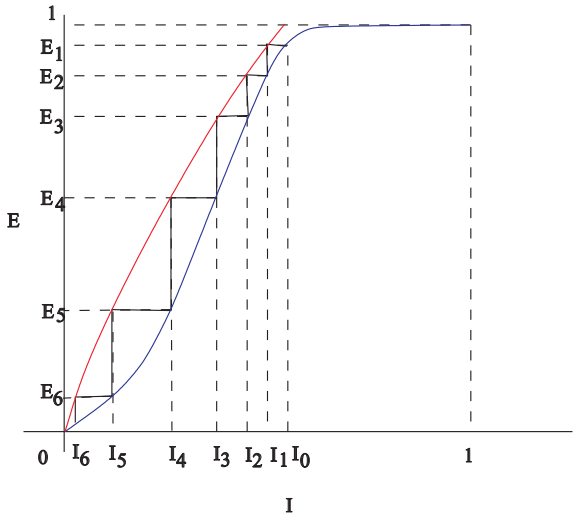


Figure 2.1: Entropy curves and average decoding trajectory.

predict that the E_t's and the I_t's are given by taking vertical and horizontal steps between both curves and that the decoding converges to the right codeword whenever the entropy curve $g(f, f_0)$ is below the entropy curve $h(f, f_0)$.

Entropy curves have many interesting properties. On the erasure channel, it has been proven in [7] that for a code ensemble with left degree distribution $\lambda(x)$ and for any value of the erasure channel probability p_0 which is such that the entropy curves do not cross, the gap between the capacity C and the design rate R is proportional to the area $\mathcal{A}$ between entropy curves,

$$C - R = \bar{\lambda}(1 - \lambda_1)\mathcal{A},$$

where λ_1 is the fraction of bits of degree 1 and $\bar{\lambda} = \frac{\sum_{i=2}^{s} \lambda_i}{\sum_{i=2}^{s} \lambda_i / i}$. Such a relation holds also approximately over the Gaussian channel. If we want to minimise the gap to capacity we should strive for entropy curves $g(f, f_0)$ which can be well approximated by an entropy curve $h(f, f_0)$.

GEXIT chart

Generalized extrinsic information transfer (GEXIT) charts form a generalization of the concept of EXIT charts and were introduced by Measson *et al.* [20]. The important point is that they satisfy the area theorem for an arbitrary MBIOS channel by definition (see [58]).

This area conservation theorem also enables to get upper bounds on the thresholds of sparse-graph code ensembles under bit-MAP decoding. The bound was shown to be tight for the BEC [19], and is conjectured to be tight in general for MBIOS channels [20].

2.1.5 Average weight distribution

Since all the code ensembles are sequences of sets of codes of given rate and length, we are interested in the typical behavior of codes of the same length. In order to study it, define $\bar{a}_i^{(n)}$ to be the average number of codewords of weight i, where the average is taken over all codes of length n in the given code ensemble.

Definition 2 [Asymptotically good codes] *An ensemble of codes is asymptotically good if there exists $\delta > 0$ so that $\sum_{i=1}^{\delta n} \bar{a}_i^{(n)} = o(1)$ for $n \to \infty$.* By Markov's inequality this implies that all but perhaps a small fraction $o(1)$ of codes of length n in such an ensemble have minimum distance at least δn.

A weaker statement is formulated as follows:

Definition 3 [Weakly asymptotically good codes] *An ensemble of codes is weakly asymptotically good if there exist constants $\delta > 0$ and $p < 1$ such that $\sum_{i=1}^{\delta n} \bar{a}_i^{(n)} \leq p$ for n sufficiently large.* In this case all but possibly a fraction p of codes of length n in this family have minimum distance at least δn, for n sufficiently large.

Note that the average distance spectrum of regular LDPC codes was computed in [36], The average distance spectrum of irregular LDPC codes was computed in [46, 18]. Moreover, it has been realized in [29] that from the behavior of the average weight distribution for low weights, a very simple criterion ensuring that most codes in an LDPC code ensemble are asymptotically good can be derived. It was also observed that is possible to generalize the generating function approach of [36] to obtain the distance spectrum of regular Tanner codes, see [54, 16]. For the average distance spectrum of ensembles of repeat-accumulate codes and variations see [31, 38], of turbo codes - [10, 61, 41].

As a_w has an exponential behaviour, then w/n tends to a constant as $n \to \infty$. We define $\alpha(\delta)$ to be the growth rate of a code ensemble:

$$\alpha(\delta) = \lim_{n \to \infty} \frac{1}{n} \ln \bar{a}_{\delta n}^{(n)}. \tag{2.1}$$

Chapter 3

Family of binary TLDPC codes

3.1 Introduction

The goal of this part is to construct a family of binary codes with low error-floor having an iterative decoding algorithm of low complexity. To provide the low error-floor behaviour, we focus ourselves on the design of asymptotically good codes, i.e. codes with linear minimum distance in the codelength.

The basic idea of our design is to modify a (not necessarily asymptotically good) Tanner code so as to obtain a code with an improved weight distribution. The hope is, that if the initial Tanner code has a low complexity iterative decoding, then the obtained code can be also decoded with a low complexity.

We optimize parameters of the obtained code family by using entropy curves (which are essentially similar to EXIT charts). We also make a structured choice of the permutation in order to avoid codewords of sublinear size and thus to preserve the property for being asymptotically good; this adds up to avoiding cycles of sublinear size in the special graph called graph of codewords of (partial) weight 2.

Codes, constructed in such a way, are called Tail-biting Trellis LDPC codes (TLDPC) and are defined further.

An important point is that an asymptotically good TLDPC code ensemble can still have a large number of codewords of the base code of weight 2 and 3 which is of benefit for the entropy curve of its base code (see Section 3.6 for details) as it improves the threshold of the ensemble and decreases the gap to channel capacity.

Moreover, TLDPC codes allow to have bits of degree 1 in their structure. The inclusion of bits of degree 1 significantly changes the behaviour of the entropy curve of the base code at the origin which leads to a lowering of the threshold and to improving the performance of the code ensemble under iterative decoding (see Section 3.8.3).

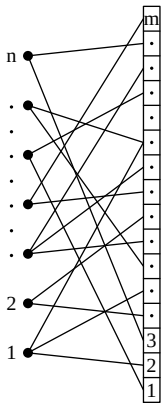
A good threshold (for example, 0.15 dB from the channel capacity for a code of rate 1/3) together with a linear minimum distance in the codelength, enables us to obtain code families with a good behaviour in the waterfall region and with no error-floor up to the word error rate in the range $10^{-5} - 10^{-6}$.

In Section 3.5 we present sufficient conditions for being asymptotically good for the TLDPC code family. The result is taken from [71]. Then in Section 3.6 we show why it is important to have a large number of codewords of weight 2 and 3 in the base code.

We present necessary conditions for being asymptotically good in Section 3.7. Further we consider particular TLDPC code families, we construct codes of rates 1/10, 1/4, 3/10, 1/3 and 1/2 and we present their performances.

3.2 General framework for irregular code constructions and their average spectrum

In this section we present a new representation of graph-based codes which embraces all the code ensembles presented as examples in the previous chapter. This general construction was first described in [71], then mentioned in [4] and [5]. The representation permits to apply directly the simple mono-dimensional optimisation of the degree distribution of a given code ensemble if the ensemble is unstructured.



Definition 4 [Construction, base code] *The construction starts with a binary code C of length m and yields a code of smaller length n with the help of a degree distribution $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_s)$ over the set of integers $\{1, 2, \ldots, s\}$ so that $\lambda_i n$ is an integer for any $i \in \{1, 2, \ldots, s\}$ and such that $\sum_{i=1}^{s} i\lambda_i = 1$ and $n\sum_{i=1}^{s} i\lambda_i = m$. The construction uses a bipartite graph between two sets V and W of vertices of size n and m respectively, where the degree of any vertex in W is exactly one and the number of vertices in V of degree i is $\lambda_i n$. We match vertices in W with m positions of the base code C.*
The code C is called the base code of the construction.

The bipartite graph together with the base code specifies a code of length n as the set of binary assignments of V such that the induced assignments[1] of vertices of W belong to C.

We present several known code constructions viewed as particular cases of the general construction.

Example

- Tanner codes: The base code of a Tanner code defined in Section 2.1.1 is the juxtaposition of $\rho_1 r$ codes C_1, $\rho_2 r$ local C_2,..., $\rho_t r$ local C_t. The set W of positions of the base code is divided into r subsets, each subset being associated to the positions of one local code.

- LDPC codes (Section 2.1.1): The base code of an LDPC code is a particular case of a Tanner base code when local codes are parity check codes.

- Parallel turbo codes (Section 2.1.2): The base code of a parallel turbo code is a juxtaposition of one or several convolutional codes. The set of variable nodes V is divided into subsets of information and redundancy variable nodes. Information variable nodes have degrees which correspond to the number of times they are used in the encoding process, redundancy variable nodes are of degree 1.

[1] a vertex in W receives the same assignment as the vertex in V it is connected to.

$\diamond$

The point of the construction that even if C has a poor minimum distance, the obtained random code of length n has a better minimum distance, and, if C has an efficient maximum likelihood decoding algorithm, there is an iterative decoding algorithm with the comparable decoding complexity to decode the obtained code of length n.

Theorem 1 *Let R_b be the rate of the base code and $\bar{\lambda}$ the average degree of the bipartite graph. Then the design rate of the constructed code R can be calculated as*

$$R = 1 - (1 - R_b)\bar{\lambda}. \tag{3.1}$$

Proof : Let n_i be the set of variable nodes of degree i, $i = 1, \ldots, s$. We obtain that

$$n = \sum_{i=1}^{s} n_i,$$
$$m = \sum_{i=1}^{s} i n_i.$$

The number k of information bits of the constructed code is computed as

$$k = R_b m - \sum_{i=1}^{s} (i-1) n_i = R_b(1-m) + n$$

where $\sum_{i=1}^{s}(i-1)n_i$ is the number of equality conditions. Thus, the design rate of the code is

$$R = 1 + R_b \frac{m}{n} = 1 - (1 - R_b)\bar{\lambda}.$$

$\blacksquare$

There is a simple formula for the average number $\bar{a}_w$ of codewords of weight w for an irregular code ensemble defined using the general construction. Let m be the number of edges in the bipartite graph and let s be the maximum degree of variable nodes. We define a polynomial $q(x, y)$ as follows

$$q(x, y) \stackrel{\text{def}}{=} \prod_{j=1}^{s} (1 + xy^j)^{\lambda_j n}.$$

Let $b(x)$ the weight enumerator of the base code.
Notation $\lfloor q(x, y) \rfloor_{w,e}$ denotes the coefficient of $x^w y^e$ of the polynomial $q(x, y)$. $\lfloor b(x) \rfloor_e$ denotes the coefficient of x^e in $b(x)$.

The expression for the average number $\bar{a}_w$ of codewords of weight w is written as

$$\bar{a}_w = \sum_{e=1}^{m} \frac{\lfloor b(x) \rfloor_e \ \lfloor q(x, y) \rfloor_{w,e}}{\binom{m}{e}}. \tag{3.2}$$

3.3 Tanner codes. Sufficient conditions for being asymptotically good.

In this section the asymptotic average weight distribution of Tanner code ensembles is presented. This generalizes formulas known for LDPC code ensembles. A sufficient condition for being asymptotically good for such a family is also derived.

The results described in this section are taken from [71].

3.3.1 Average spectrum of Tanner codes

In the case of Tanner codes, when the base code is a juxtaposition of local codes, the weight enumerator of the base code $b(x)$ has a very simple expression in terms of weight enumerator polynomials of local codes. Denote by $b_i(x)$ the weight enumerator of a local code C_i,$1 \leq i \leq t$, where t is the maximum degree of constraint nodes in the bipartite graph (i.e. the maximum length of local codes). Then, if r denotes the number of local codes, we obtain

$$b(x) = \prod_{i=1}^{t} b_i(x)^{\rho_i r}.$$

We are interested in estimating the behaviour of $\bar{a}_{\delta n}$ [2], $\delta \in [0,1]$. According to (3.2), this boils down to estimating $\lfloor q(x,y) \rfloor_{\delta n, \varepsilon n}$ and $\lfloor b(x) \rfloor_{\varepsilon n}$ for some $\varepsilon \in (0,1)$. These polynomials have nonnegative coefficients and we can use the following upper bound on the coefficient of the monomial $x^i y^j$ of a polynomial $p(x,y)$ with negative coefficients:

$$\lfloor p(x,y) \rfloor_{i,j} \leq \inf_{u>0, v>0} \frac{p(u,v)}{u^i v^j}. \tag{3.3}$$

Using the upper bound for polynomials $q(x,y)$ and $b(x)$ yields the correct exponent of the exponential behaviour of their coefficients; the reason for this is that $q(x,y)$ and $b(x)$ are powers of some fixed polynomials with positive coefficients $Q(x,y)$ and $B(x)$. We quote the following theorem which has been proved in [18]:

Theorem 2 *Let γ be some rational number and let $p(x,y)$ be a function such that $p(x,y)^\gamma$ is a polynomial with nonnegative coefficients. Let δ and ε be some positive reals, and let n_i be the set of indexes such that n_i/γ is an integer and $\lfloor p(x,y)^{n_i} \rfloor_{\delta n, \varepsilon n} \neq 0$, then*

$$\lfloor p(x,y)^{n_i} \rfloor_{\delta n, \varepsilon n} \leq \inf_{u>0, v>0} \frac{p(u,v)^{n_i}}{(u^\delta v^\varepsilon)^{n_i}} \tag{3.4}$$

and

$$\lim_{i \to \infty} \ln \lfloor p(x,y)^{n_i} \rfloor_{\delta n, \varepsilon n} = \ln \inf_{u>0, v>0} \frac{p(u,v)}{u^\delta v^\varepsilon}. \tag{3.5}$$

Now we can present the proof of the following theorem:

[2] by slight abuse of notation we denote $\bar{a}_{\lfloor \delta n \rfloor}$ by $\bar{a}_{\delta n}$

Theorem 3

$$\alpha(\delta) = \sup_{\Delta_{min}\delta \le \varepsilon \le \Delta_{max}\delta} \beta(\delta, \varepsilon)$$

where

$$\beta(\delta,\varepsilon) = \frac{r}{n} \inf_{x>0} \left[\sum_{j=1}^{t} \rho_j \ln b_j(x) - \frac{\varepsilon n}{r} \ln x \right] + \inf_{x>0, y>0} \left[\sum_{i=1}^{s} \lambda_i \ln(1+xy^i) - \delta \ln x - \varepsilon \ln y \right] - \bar{\lambda} h(\frac{\epsilon}{\bar{\lambda}}).$$

Moreover,

$$\bar{a}_w^{(n)} \le e^{n\alpha(w/n)}. \tag{3.6}$$

Proof : From Theorem 2, $\lfloor q(x,y) \rfloor_{w,e}$ and $\lfloor b(x) \rfloor_e$ have an exponential behaviour when e/n and w/n tend to a constant. This is also the case for $\binom{m}{n}$. Therefore, a_w has an exponential behaviour when w/n tends to a constant as $n \to \infty$. Let us compute the growth rate $\alpha(\delta)$ of the Tanner ensemble.

From (3.2),

$$\alpha(\delta) = \lim_{n\to\infty} \frac{1}{n} \ln \bar{a}_{\delta n} = \lim_{n\to\infty} \frac{1}{n} \max_{1 \le e \le m} \ln \frac{\lfloor b(x) \rfloor_e \lfloor q(x,y) \rfloor_{\delta n, e}}{\binom{m}{e}},$$

This leads to estimate $\lfloor q(x,y) \rfloor_{\delta n, \theta m}$ and $\lfloor b(x) \rfloor_{\theta m}$, $\theta \in [0,1]$. Note that $\varepsilon = \theta\bar{\lambda}$, then

$$\frac{1}{n} \ln \bar{a}_{\delta n} = \max_{\delta\Delta_{min} \le \varepsilon \le \delta\Delta_{max}} \left(\frac{1}{n} \ln \lfloor b(x) \rfloor_{\varepsilon n} + \frac{1}{n} \ln \lfloor q(x,y) \rfloor_{\delta n, \varepsilon n} - \bar{\lambda} h(\frac{\epsilon}{\bar{\lambda}}) + o(1) \right), \tag{3.7}$$

where Δ_{min} and Δ_{max} are the smallest and the highest left degrees of the bipartite graph and where last terms are obtained using the following approximation.

$$\binom{m}{\theta m} = \binom{\bar{\lambda} n}{\epsilon n} = \bar{\lambda} n [h(\frac{\epsilon}{\bar{\lambda}}) + o(1)].$$

By using the upper bound (3.5), we obtain that

$$\frac{1}{n} \ln \lfloor b(x) \rfloor_{\varepsilon n} = \frac{1}{n} \sum_{j=1}^{t} \ln \lfloor b_i^{\rho_j r}(x) \rfloor_{\varepsilon n}.$$

Thus, by Theorem 2,

$$\lim_{n\to\infty} \frac{1}{n} \ln \lfloor b(x) \rfloor_{\varepsilon n} = \frac{r}{n} \sum_{j=1}^{t} \rho_j \ln \inf_{x>0} \frac{b_j(x)}{x^{\varepsilon n/(\rho_j r)}} = \frac{r}{n} \inf_{x>0} \left[\sum_{j=1}^{t} \rho_i \ln b_j(x) - \frac{\varepsilon n}{r} \ln x \right]. \tag{3.8}$$

We also use Theorem 2 to get the limit for $\lfloor q(x,y) \rfloor_{\delta n, \varepsilon n}$:

$$\begin{aligned} \lim_{n\to\infty} \frac{1}{n} \ln \lfloor q(x,y) \rfloor_{\delta n, \varepsilon n} &= \ln \inf_{x>0, y>0} \sum_{i=1}^{s} \frac{(1+xy^i)^{\lambda_i}}{x^\delta y^\varepsilon} \\ &= \inf_{x>0, y>0} \left[\sum_{i=1}^{s} \lambda_i \ln(1+xy^i) - \delta \ln x - \varepsilon \ln y \right]. \end{aligned}$$

By using (3.7), (3.8) and (3.9) we obtain the given expression for $\bar{a}_w^{(n)}$.

Now we prove Inequality (3.6). We begin from (3.2) and write:

$$\begin{aligned}\bar{a}_w &= \exp\left[\ln \sum_{e=1}^{m} \frac{\lfloor b(x)\rfloor_e \ \lfloor q(x,y)\rfloor_{w,e}}{\binom{m}{e}}\right] \\ &\approx \exp\left[\max_{1\le e\le m} \ln\lfloor b(x)\rfloor_e + \ln\lfloor q(x,y)\rfloor_{w,e} - \ln\binom{m}{e}\right].\end{aligned}$$

Using Theorem 2 and keeping w/n fixed, we obtain that

$$\begin{aligned}\bar{a}_w &\le \exp\left[\sup_{\Delta_{min}\delta\le\varepsilon\le\Delta_{max}\delta} n\beta(\delta,\varepsilon)\right] \\ &= e^{n\alpha(w/n)}.\end{aligned}$$

∎

3.3.2 Sufficient conditions for Tanner codes for being asymptotically good

Intuitively, if $\alpha(\delta)$ is small around the origin, then the code ensemble should be asymptotically good. For a Tanner code, the following lemma settles the issue of how small α has to be:

Lemma 1 *Consider an ensemble of Tanner codes. If* $-\infty < \lim_{\delta\to 0^+} \frac{\alpha(\delta)}{\delta} < -\ln 2$*, then the code ensemble is weakly asymptotically good. If* $\lim_{\delta\to 0^+} \frac{\alpha(\delta)}{\delta} \to -\infty$*, the code ensemble is asymptotically good.*

Proof : Assume first that $\lim_{\delta\to 0^+} \frac{\alpha(\delta)}{\delta}$ exists, is finite and is smaller than $-\ln 2$. This implies that there exists $C < -\ln 2$ and $\delta_0 > 0$ such that

$$\alpha(\delta) = C\delta$$

for any $0 < \delta < \delta - 0$. By inequality (3.6), $\bar{a}_i^{(n)} \le e^{n\alpha(i/n)} \le K^i$ with $K < 1/2$ for any $0 < i < \delta_0 n$. Therefore, provided that i/n stays smaller than some well chosen constant $\delta_1 > 0$, we have

$$\sum_{i=1}^{\delta_1 n} \bar{a}_i^{(n)} \le \sum_{i=1}^{\delta_0 n} K^i \le \frac{K}{1-K} < 1.$$

Consider now that $\lim_{\delta\to 0^+} \frac{\alpha(\delta)}{\delta} = -\infty$. Let $f(n) = \sup_{0<u<\frac{1}{\sqrt{n}}} \frac{\alpha(u)}{u}$. Note that $f(n)$ goes to $-\infty$ as n tends to infinity. From the hypothesis on $\alpha(\delta)$ we know that there exists $\delta_0 > 0$ such that for any positive integer $i \le \delta_0 n$ we have $\alpha(i/n) \le -\ln 2$. Observe that

$$\begin{aligned}\sum_{i=1}^{\delta_0 n} \bar{a}_i^{(n)} &= \sum_{i=1}^{\lfloor\sqrt{n}\rfloor} \bar{a}_i^{(n)} + \sum_{\lfloor\sqrt{n}\rfloor+1}^{\delta_0 n} \bar{a}_i^{(n)} \\ &\le \sum_{i=1}^{\lfloor\sqrt{n}\rfloor} e^{if(n)} + \sum_{\lfloor\sqrt{n}\rfloor+1}^{\delta_0 n} e^{-i\ln 2} \\ &\le \frac{e^{f(n)}}{1-e^{f(n)}} + \frac{1}{2^{\lfloor\sqrt{n}\rfloor}}\end{aligned}$$

The last term tends to 0 as n tends to infinity. ∎

Remark that it does not hold for other irregular code ensembles, even if $\alpha(\delta)$ exists.

To calculate $\lim_{\delta\to 0+}\frac{\alpha(\delta)}{\delta}$, we use two following lemmas. The first of them will permit to settle the behaviour of the first term in the expression $\beta(\delta,\varepsilon)$ as $\delta\to 0$ in Theorem 3:

Lemma 2 *Let f be a C^∞ function defined over $[0,\infty)$. Assume that $f(0)=1$ and $f'(0)\geq 0$, and that the behaviour of f' is given by $f'(x)=\alpha x^{\beta-1}+O(x^\beta)$, with $\alpha>0$ and $\beta>1$. Assume also that $\frac{xf'(x)}{f(x)}$ is increasing. Let $g(t)=\inf_{u>0}\ln f(x)-t\ln u$. Then*

$$\frac{g(t)}{t}=\frac{1+\ln\alpha-\ln t+o(1)}{\beta}$$

as $t\to 0^+$.

The second lemma is used to determine the behaviour of the second term in the expression $\beta(\delta,\varepsilon)$, namely

$$\pi(\delta,\varepsilon)=\inf_{x>0,y>0}\sum_{i=1}^{s}\lambda_i\ln(1+xy^i)-\delta\ln x-\varepsilon\ln y,$$

as $\delta\to 0^+$ and the ratio $\tau=\varepsilon/\delta$ is kept fixed.

Lemma 3 *As $\delta\to 0^+$ and $\tau=\varepsilon/\delta$ is kept fixed and and equal to some constant τ,*

$$\pi(\delta,\varepsilon)=h(\delta)+\delta K(\tau)+o(\delta),$$

where $K(\tau)=\inf_{x>0}\ln q(x)-\tau\ln x$ and $q(y)=\sum_{t=1}^{s}\lambda_i y^i$. $K(\tau)$ is an increasing and continuous function in τ which satisfies:
$K(\tau)=-\infty$ for $\tau<\Delta_{min}$,
$K(\Delta_{min})=\ln(\lambda_{\Delta_{min}})$.

We formulate the following proposition:

Proposition 1 *Let d_{min} be the smallest minimum distance of local codes C_i and let n_i be the number of codewords of weight d_{min} in the local code C_i. If $\Delta_{min}>2$ or $d_{min}>2$, then a Tanner code ensemble is asymptotically good. If $\Delta_{min}=d_{min}=2$, then the Tanner code ensemble is weakly asymptotically good if*

$$\frac{8\lambda_{\Delta_{min}}\sum_{i=1}^{t}\rho_i n_i}{\bar\lambda\bar\rho}<1.$$

Proof : By using Lemmas 2 et 3, we write the expression for $\alpha(\delta)$ in the form

$$\alpha(\delta)=A\delta\log\delta+B\delta+o(\delta),$$

where

$$A=\Delta_{min}-\frac{\Delta_{min}}{d_{min}}-1;$$

$$B=\frac{\Delta_{min}}{d_{min}}\left[1+\log(\frac{\bar\lambda}{\bar\rho\sum_{j=1}^{t}\rho_j n_j})\right]+\log\lambda_{\Delta_{min}}+\Delta_{min}\log\frac{\Delta_{min}}{\bar\lambda}.$$

So we have that

- if $\Delta_{min} > 2$ or $d_{min} > 2$ as $\delta \to 0^+$, we approximate $\alpha(\delta)$ by its second term

$$\alpha(\delta) = (\Delta_{min} - \frac{\Delta_{min}}{d_{min}} - 1)\delta \ln \delta + O(\delta)$$

and

$$\lim_{\delta \to 0^+} \frac{\alpha(\delta)}{\delta} = -\infty.$$

- if $\Delta_{min} = d_{min} = 2$ as $\delta \to 0^+$, then $A = 0$ and

$$\alpha(\delta) = \delta(1 + \log \frac{4\lambda_{\Delta_{min}} \sum_{i=1}^{t} \rho_i n_i}{\bar{\lambda}\bar{\rho}}) + o(\delta)$$

and

$$\lim_{\delta \to 0^+} \frac{\alpha(\delta)}{\delta} = \log \frac{8\lambda_{\Delta_{min}} \sum_{i=1}^{t} \rho_i n_i}{\bar{\lambda}\bar{\rho}}.$$

From these calculations and lemma 1 we can deduce the expression above. ∎

3.4 Tail-biting trellis LDPC codes

In this section we define a TLDPC code ensemble by defining what is the TLDPC base code. The TLDPC code ensemble is obtained by a simple modification of a Tanner code ensemble where the base code of a simple Tanner code is replaced by a tail-biting convolutional code.

Let C_0 be the local code from which the simple Tanner code is constructed. Say that this code has length n_0 and dimension k_0. Let r be the number of local codes in the Tanner code we consider. Assume that for some integer c we have a couple of linear maps

$$\begin{array}{rcccl} R & : & \{0,1\}^c \times \{0,1\}^{k_0} & \to & \{0,1\}^{n_0-k_0} \\ S & : & \{0,1\}^c \times \{0,1\}^{k_0} & \to & \{0,1\}^c \end{array}$$

such that C_0 is formed by all the couples of the form $(x, R(e,x))$, where $x \in \{0,1\}^{k_0}$ and $e \in \{0,1\}^c$ satisfy $S(e,x) = e$.

In other words, we have defined a one-sectional linear tail-biting trellis T for C_0 with state complexity 2^c using two linear maps R and S, where R maps trellis states and information bits to redundancy bits and S maps input state bits and information bits to output state bits.

Assumption 1 *We assume that*

1. *for $\forall e \in \{0,1\}^c$, the set $\{S(e,x), x \in \{0,1\}^{k_0}\}$ is equal to $\{0,1\}^c$ (i.e. the state complexity is constant and equal to 2^c).*

2. *there is only one state $e \in \{0,1\}^c$ such that $R(e,0) = 0$, namely $e = 0$ (i.e. this amounts to the uniqueness of the all-zero codeword).*

Definition 5 [TLDPC base code] *The base code of the TLDPC code is a tail-biting convolutional code obtained by the serial concatenation of r copies of T, where the first and the last states of serial concatenation are identified. In other words, the base code of the TLDPC code is the set of vectors of the form $(x_1, y_1, x_2, y_2, \ldots, x_r, y_r)$ (with $x_i \in \{0,1\}^{k_0}$, $y_i \in \{0,1\}^{n_0-k_0}$ for $i \in \{1,2,\ldots,r\}$) for which there exists $(e_1, e_2, \ldots, e_r)$ (with $e_i \in \{0,1\}^c$ for $i \in \{1,2,\ldots,r\}$) such that:*

1. *$e_0 = e_r$ (condition of trellis termination),*
2. *$\forall i \in \{1,r\}$, $S(e_{i-1}, x_i) = e_i$ and $R(e_{i-1}, x_i) = y_i$.*

Remark that the definition can be generalised to TLDPC codes constructed from more complicated Tanner codes. In this case it is essential that every couple (R_i, S_i) defining a one-sectional linear tail-biting trellis for a local code C_i be defined over a common state space (i.e. all the $\{0,1\}^c$ have to be the same).

3.5 Sufficient conditions for TLDPC codes for being asymptotically good

3.5.1 Average spectrum of the TLDPC code ensemble

The average spectrum of TLDPC codes is obtained using a similar approach as for Tanner codes in Section 3.3.1. We compute the average spectrum of TLDPC code family using (3.2) with $b(x)$ being the weight enumerator polynomial of the TLDPC base code. The average spectrum $\bar{a}_{\lfloor \delta n \rfloor}$ having an exponential behaviour when w/n tends to a constant and n to infinity, we compute the growth rate of the TLDPC code family.

Note that the weight enumerator $b(x)$ of the base code has a simple expression in terms of the following matrix $M(x) = (m_{i,j})$, $i \in \{0,1\}^c$ and $j \in \{0,1\}^c$: the entry $\{i,j\}$ of M is equal to the weight enumerator (assumed to be a polynomial in x) of the set

$$\{(u, R(i,u));\ u \in \{0,1\}^{k_0} \text{ such that } S(i,u) = j\},$$

namely

$$b(x) = \mathbf{tr} M(x)^r.$$

To bound coefficients $\lfloor b(x) \rfloor_{\varepsilon r}$[3], we adapt Theorem 2 to our case and we obtain the following expression:

$$\lfloor b(x) \rfloor_{\varepsilon r} \leq \inf_{x>0} \frac{\mathbf{tr} M(x)^r}{x^{\epsilon r}}. \tag{3.9}$$

Let $\mu(x)$ be the largest positive eigenvalue of $M(x)$. By the Perron-Frobenius theorem [35], since $M(x)$ is a positive matrix for any $x > 0$, $\mu(x)$ is simple and larger than or equal to the absolute values of the other eigenvalues. From this we deduce that

$$\frac{1}{r} \ln \lfloor b(x) \rfloor_{\varepsilon r} \leq \frac{c \ln 2}{r} + \inf_{x>0} (\ln \mu(x) - \varepsilon \ln x).$$

By doing calculations similar to those in [18], one can show that that $\lim_{r\to\infty} \frac{1}{r} \ln \lfloor b(y) \rfloor_{\epsilon r}$ exists and is equal to $\inf_{x>0}(\ln \mu(x) - \varepsilon \ln x)$. So, the growth rate of the TLDPC code ensemble $\alpha(\delta)$ can be calculated as follows

[3]Recall that $\lfloor b(x) \rfloor_{\varepsilon r}$ is the coefficient of $x^{\varepsilon r}$ of the polynomial $b(x)$

Theorem 4

$$\alpha(\delta) = \sup_{\Delta_{min}\delta \le \varepsilon \le \Delta_{max}\delta} \beta(\delta, \varepsilon)$$

where

$$\beta(\delta,\varepsilon) = \frac{r}{n} \inf_{x>0} \left[\ln \mu(x) - \frac{n\varepsilon}{r} \ln x\right] + \inf_{x>0, y>0} \left[\sum_{i=1}^{s} \lambda_i \ln(1 + xy^i) - \delta \ln x - \varepsilon \bar{\lambda} \ln y\right] + \bar{\lambda} h(\epsilon).$$

For the average number of codewords of weight w we can write

$$\begin{aligned} \bar{a}_w &= \exp\left[\ln \sum_{e=1}^{m} \frac{\lfloor b(x) \rfloor_e \ \lfloor q(x,y) \rfloor_{w,e}}{\binom{m}{e}}\right] \\ &\approx \exp\left[\max_{1 \le e \le m} \ln \lfloor b(x) \rfloor_e + \ln \lfloor q(x,y) \rfloor_{w,e} - \ln \binom{m}{e}\right]. \end{aligned}$$

By using Theorem 2 and keeping w/n fixed, we obtain that

$$\begin{aligned} \ln \bar{a}_w \le \sup_{\Delta_{min}\delta \le \varepsilon \le \Delta_{max}\delta} & nc \ln 2 + \frac{r}{n} \inf_{x>0} \left[\ln \mu(x) - \frac{n\varepsilon}{r} \ln x\right] + \\ & + \inf_{x>0, y>0} \left[\sum_{i=1}^{s} \lambda_i \ln(1 + xy^i) - \delta \ln x - \varepsilon \bar{\lambda} \ln y\right] + \bar{\lambda} h(\epsilon), \end{aligned}$$

and thus, for TLDPC codes we have the weaker statement than for Tanner codes

$$\bar{a}_i \le 2^c e^{n\alpha(i/n)}. \tag{3.10}$$

This is a consequence of $b(x) = \mathbf{tr} M(x)^r \le 2^c \mu(x)^r$. However, for small values of i/n the following inequality holds.

Proposition 2 *For every $\varepsilon' > 0$ there exists $\beta > 0$ such that for any positive integers i and n, $i/n > \beta$, we have*

$$\bar{a}_i \le (1 + \varepsilon') e^{n\alpha(i/n)}. \tag{3.11}$$

Proof : We start from Inequality (3.9). Note that for every γ there exists some constant $K(\gamma)$ such that for every $0 < \varepsilon < K(\gamma)$ the infimum is attained for some $x \in (0, \gamma)$.

$M(0)$ is a matrix with only one non-zero coefficient $m_{0,0}$, equal to 1. As a consequence, $M(0)$ has a single positive eigenvalue which is 1, all the other eigenvalues equal to 0. From the fact that the eigenvalues of $M(x)$ are continuous functions of x [66] and the trace of $M(x)^r$ is equal to the sum of the r-th powers of the eigenvalues of $M(x)$, we deduce that there is a $\gamma > 0$ such that $\mathbf{tr} M(x)^r \le (1 + \varepsilon')\mu(x)^r$ for every $0 < x < \gamma$. We choose such a γ and then for $\varepsilon < K(\gamma)$ we obtain that

$$\lfloor b(x) \rfloor_{\varepsilon r} \le (1 + \varepsilon') \inf_{x>0} \frac{\mu(x)^r}{x^{\epsilon r}}.$$

By using calculations similar to those leading up to (3.10), we obtain the inequality (3.11). ∎

Note that Inequality (3.11) is stronger than Inequality (3.6). This is due to the fact that $\mathbf{tr} M(x)^r \approx \mu(x)^r$ for values of x around 0.

It is highly instructive to estimate what can be gained in the average distance spectrum if we take a TLDPC code instead of a simple Tanner code. Recall that for a right-regular Tanner code

$$\alpha(\delta) = \sup_{\Delta_{min}\delta \leq \varepsilon \leq \Delta_{max}\delta} \beta(\delta, \varepsilon)$$

where

$$\beta(\delta,\varepsilon) = \frac{r}{n} \inf_{x>0} \left[\sum_{j=1}^{t} \rho_j \ln b_j(x) - \frac{\varepsilon n}{r} \ln x \right] + \inf_{x>0, y>0} \left[\sum_{i=1}^{s} \lambda_i \ln(1+xy^i) - \delta \ln x - \varepsilon \ln y \right] - \bar{\lambda} h(\frac{\epsilon}{\bar{\lambda}}).$$

For a simple Tanner code the first term in the expression $\beta(\delta, \varepsilon)$ reduces to $\ln b_1(x)$, $b_1(x)$ being the weight enumerator polynomial of the local code chosen to construct the simple Tanner code. If we take the associated TLDPC code (constructed with the help of the same local code), then in the corresponding expression for $\beta(\delta, \varepsilon)$ $b_1(x)$ will be replaced by $\mu(x)$ (since $b_1(x) = \mathbf{tr}M(x)$ and we replace the trace of $M(x)$ by the largest eigenvalue of $M(x)$). Note that the tail-biting trellis of the local code can be chosen in such a way that the largest eigenvalue of $M(x)$ will be significantly smaller than the trace of $M(x)$.

3.5.2 Sufficient conditions for being asymptotically good

The calculations for TLDPC codes are similar to those done for Tanner codes with the only difference that before to apply Lemma 2 we need to study the behaviour of $\mu(x)$ around 0. This is done by matrix perturbation argument. To obtain the behaviour of $\alpha(\delta)$ around 0 by using only the first order approximation of $\mu(x)$ and thus to simplify the results, we make the following assumption:

Assumption 2 *All entries of $M(x)$ except $m_{0,0}$ are divisible by x^2 and $(m_{0,0}-1)$ is also divisible by x^2. This amounts to the fact that all vectors $(x, R(e,x))$ with $x \in \{0,1\}^{k_0}$, $e \in \{0,1\}^c$ are of weight at least 2, except $(0, R(0,0))$ which has Hamming weight 0.*

Let $E(x) = M(x) - M(0)$. Note that the largest eigenvalue of $M(0)$ is simple and equal to 1, and the associated eigenvector can be chosen to be the vector u with the 1 in the first position equal and 0 elsewhere. By applying Theorem 2.3 of [66] we obtain that

$$\mu(x) = 1 + \frac{(u, E(x)x)}{(u,u)} + O(\|E(x)\|^2) = m_{0,0}(x) + O(\|E(x)\|^2),$$

where (u,v) denotes the standard inner product in $\mathcal{R}^{2^c}$. By using the assumption we obtain $\mu(x) = m_{0,0}(x) + O(x^4)$. This leads us to consider the subcode $C^{(0)}$ of the local code given by $(x, R(0,x))$ with $x \in \{0,1\}^{k_0}$ such that $S(0,x) = 0$. Denote by $d^{(0)}_{min}$ the minimum distance of the subcode and by $n^{(0)}_{min}$ the number of codewords in this subcode of weight $d^{(0)}_{min}$. With this notation, we finally obtain

$$\mu(x) = n^{(0)}_{min} x^{d^{(0)}_{min}} + O(x^4).$$

By using this expression in Lemma 1 we obtain that

- if either $\Delta_{min} > 2$ or $4 \geq d^{(0)}_{min} > 2$ as $\delta \to 0^+$, then

$$\alpha(\delta) = \left(\Delta_{min} - \frac{\Delta_{min}}{d^{(0)}_{min}} - 1 \right) \delta \ln \delta + O(\delta);$$

- if $\Delta_{min} = d^{(0)}_{min} = 2$, as $\delta \to 0^+$, then

$$\alpha(\delta) = \ln\left(\frac{4\lambda_{\Delta_{min}} \sum_{i=1}^{t} \rho_i n_i}{\bar{\lambda}\bar{\rho}}\right)\delta + o(\delta).$$

As for Tanner codes, For TLDPC ones we have the following lemma:

Lemma 4 *Consider an ensemble of TLDPC codes. If* $-\infty < \lim_{\delta\to 0^+} \frac{\alpha(\delta)}{\delta} < -\ln 2$*, then the code ensemble is weakly asymptotically good. If* $\lim_{\delta\to 0^+} \frac{\alpha(\delta)}{\delta} \to -\infty$*, the code ensemble is asymptotically good.*

The proof of the lemma is similar to the proof of Lemma 1, and we use Proposition 2 with an ε' such that $(1+\varepsilon')C \leq -\ln 2$ to show that for i/n smaller than some constant $\delta_1 > 0$ we have $\bar{a}_i^{(n)} \leq K^i$.

From previous calculations and Lemma 4 we deduce the following proposition

Proposition 3 *Let C be a TLDPC code ensemble which satisfies the Assumption 2. If either $\Delta_{min} > 2$ or $d^{(0)}_{min} > 2$ then C is asymptotically good. If $\Delta_{min} = d(0)_{min} = 2$ and if $\frac{4\lambda_{\Delta_{min}} n^{(0)}_{min}}{\bar{\lambda}\bar{\rho}} < 1$, then C is weakly asymptotically good.*

Note that for TLDPC codes we can allow twice more codewords of weight 2 in the base code than for Tanner codes and still be asymptotically good.

3.6 Matching condition for entropy curves of TLDPC codes on the BEC

In this section, with the aim to define what base codes are more appropriate to obtain a code family with good decoding performances, we study the first and the second derivatives of entropy curves of the base code at the origin and show that they are functions of the number of codewords of the base code of weight 2 and 3.

Though in general entropy curves (see Section 2.1.4 for definition) are only an approximation to describe the behaviour of the decoding algorithm, on the BEC they give exactly the average behaviour of a code ensemble when the codelength tends to infinity. For the BEC there also exist relations connecting the threshold of iterative decoding and some parameters of the code ensemble. These relations can be used to optimise the parameters of the code in order to improve the performances under iterative decoding.

We study two different cases of TLDPC code families: without and with bits of degree 1 in their structure.

3.6.1 TLDPC code families without bits of degree 1

Properties of entropy curves of variable nodes and of the base code

Let us define the entropy curves of variable nodes and of the base code on the BEC.

Definition 6 *We define extrinsic average entropy and intrinsic average entropy at the t^{th} iteration as follows*

$$\begin{aligned} E_t &\stackrel{\text{def}}{=} \frac{1}{n_b}\sum_{i=1}^{n_b} E_t(i) \\ I_t &\stackrel{\text{def}}{=} \frac{1}{n_b}\sum_{i=1}^{n_b} I_t(i) \end{aligned}$$

with $E_t(i)$ ($I_t(i)$) being the entropy of the extrinsic (intrinsic) probability associated to the i-th position of the base code at the t-th iteration.

We begin with the entropy curve of the base code:

Lemma 5 *The entropy curve of the base code on the BEC with erasure probability p is given by the set of points (I_0, E_1) with:*

$$\begin{aligned} I_0 &= p \\ E_1 &= \frac{1}{n_b}\sum_{i=1}^{n_b} h_i(p) \end{aligned}$$

where

$$h_i(p) = \sum_{e\in\mathcal{E}_i} p^{|e|-1}(1-p)^{n_b-|e|} \tag{3.12}$$

with $\mathcal{E}_i$ to be the erasure configurations of bits of the base code for which the value of bit i cannot be found; for every such configuration e, $|e|$ denotes the number of erasures in it.

Lemma 5 is a slight modification of the result given in [7].

Lemma 6 *An erasure configuration e is included in $\mathcal{E}_i$ if and only if there exists a codeword of the base code with 0 on all the unerased positions of e and 1 at position i.*

Proof : An erasure at position i cannot be compensated by the decoding of the base code if and only if there exist two codewords being respectively equal to 1 and to 0 at position i which are compatible with the received word. By the code linearity, we may assume that the transmitted codeword was the all-zero one. Thus one of the compatible codewords will be the all-zero codeword, and there exists a codeword with 1's at position i compatible with the received codeword if and only if all the unerased positions of this codeword are equal to 0. ∎

We compute the behaviour of entropy curves of variable nodes and of the base code at the origin. This computation is fundamental to explain our approach to construct base codes with a given trellis complexity.

For the entropy curve of variable nodes we have the following theorem:

Theorem 5 *The derivative of the entropy curve of variable nodes at the origin for the BEC with erasure probability p is equal to*

$$\frac{1}{p\lambda_2}.$$

Proof : The inverse function of the entropy curve of variables nodes is expressed as $p\Lambda(x)$. Its derivative at the origin is $p\Lambda'(0) = p\lambda_2$, so the derivative of the entropy curve of variable nodes at the origin is equal to $1/p\lambda_2$. ∎

Note that the entropy curve of variable nodes is a function of the erasure channel probability p. When p decreases, the entropy curve of variable nodes moves away from the entropy curve of the base with the rate inversely proportional to the average degree $\bar{\lambda}$. Note also that greater the gap between two entropy curves for some p, the faster is the average convergence of the iterative decoding in this case. This fact is another reason to have $\bar{\lambda}$ small besides the low complexity of iterative decoding.

For the entropy curve of the base code we have

Theorem 6 *Let m_2 be the number of codewords of weight 2 in the base code of length n_b. The derivative at the origin of the entropy curve of the base code is equal to*

$$\frac{2m_2}{n_b}.$$

Proof : Let $m_2(i)$ the number of codewords of the base code of weight 2 equal to 1 at the position i. By using Lemmas 5 and 6 we obtain for $p \to 0^+$:

$$h_i(p) = m_2(i)p + O(p^2).$$

Therefore, the derivative of the entropy curve of the base code at the origin is equal to

$$\frac{\sum_{i=1}^{n_b} m_2(i)}{n_b} = \frac{2m_2}{n_b}. \tag{3.13}$$

∎

We also calculate the second derivative of the entropy curve of the base code at origin in the next theorem:

Theorem 7 *Let m_t be the number of codewords of the base code of weight t, and $m_t(i)$ be the number of codewords of the base code of weight t being equal to 1 at the position i. The second derivative of the entropy curve of the base code at the origin is equal to*

$$\frac{6m_3 + m_2 - \sum_{i=1}^{n} m_2(i)^2}{n_b}.$$

Proof : By using Lemmas 5 et 6 and by letting $p \to 0^+$ we obtain:

$$h_i(p) = m_2(i)p + (A + B)p^2 + O(p^3),$$

where the coefficient $A + B$ of p^2 is given by

- $A = m_2(i)(-n_b + 2)$, which comes from the contribution of $p(1-p)^{n_b-2}$ erasure configurations of weight 2 for which the position i cannot be found,
- B is the number of configurations with 3 erasures for which the position i cannot be found by the base code decoding. This set is the union of two following sets:

– the set of erasure configurations of weight 3 which coincide with the support of a codeword of weight 3 having 1 at the position i. There are $m_3(i)$ such configurations.

– the set of erasure configurations of weight 3 which coincide with the support of a codeword of weight 3 which covers the support of a codeword of weight 2 having 1 at the position i. Every codeword of weight 2 having 1 at the position i can be completed in $n_b - 2$ different ways to have such a configuration. The number of such configurations is thus equal to $m_2(i)(n_b - 2)$ from which we subtract the number of doubly counted configurations $\binom{m_2(i)}{2}$.

We deduce that the second derivative of the entropy curve of the base code at the origin is equal to

$$2\frac{\sum_{i=1}^{n} m_3(i) - \frac{m_2(i)(m_2(i)-1)}{2}}{n_b} = \frac{6m_3 + m_2 - \sum_i m_2(i)^2}{n_b}.$$

∎

Approach to construct base codes without bits of degree 1

We seek a code having a low decoding complexity, the iterative threshold of which is close to capacity. To have a low decoding complexity we fix the average degree $\bar{\lambda}$ to be small. In this case the form of the entropy curve of variable nodes is close to the straight line. To optimize the iterative decoding threshold, we would like to have a base code whose entropy curve form is similar to the form of the entropy curve of variable nodes.

Denote by p_c the threshold erasure probability, i.e. the minimum erasure probability of the channel for which the probability of recovering of the codeword after iterative decoding is equal to $1/2$. Consider an ideal case when the entropy curve of the base code is of form $y = \frac{1}{p_c x}$ for $x \in (0, p_c)$ and is equal 1 for $x \in [p_c, 1)$. In this case, one could choose all the bits to be of degree 2 and to attain the capacity of the erasure channel.

For TLDPC codes, the entropy curve of the base code is very close to 1 for $x = p_c$, and, as the entropy curve of variable nodes begins at the point $(p_c, 1)$, the area difference between two entropy curves in this region will be very small.

A different thing happens for entropy curves at the origin. Note that if we want the entropy curves of variable nodes and of the base code to be close at the origin, their derivatives at the origin have to be approximately close:

$$\frac{2m_2}{n_b} \approx \frac{1}{\lambda_2 p_c}, \tag{3.14}$$

i.e.

$$\frac{2\lambda_2 m_2}{n_b} \approx \frac{1}{p_c}. \tag{3.15}$$

In the case of asymptotically good TLDPC codes the number of codewords of the base code of weight 2 is bounded by Proposition 3 and so is the first derivative of the entropy curve of the base code. The condition (3.15) can not be satisfied, except in the case when the code rate is close to 0 or $p_c \approx 1$. Thus the entropy curve of the base code is convex at the origin and its slope at the origin is different from the slope at the origin of the entropy curve

of variable nodes (which is concave). Owing to this difference, there is always some area between entropy curves at the origin which influences the gap to capacity. The solution to minimise the area at origin between the two entropy curves is to choose the second derivative at origin of the entropy curve of the base code to be the largest possible. This implies that the number of codewords of weight 3 in the base code should be maximised.

Finally, with the aim to construct an asymptotically good TLDPC code family having a small gap to capacity and a low complexity of decoding, we do the following: while designing the base code, we seek to minimise $\sum_i m_2(i)^2$ and maximise m_3 for a fixed ratio $\frac{m_2}{n_b}$.

3.6.2 TLDPC code families with bits of degree 1

So far TLDPC code families without bits of degree 1 have been considered.

Now, let us have a sparse-graph graph code family with a degree distribution $\Lambda(x) = \sum_{i=1}^{s} \lambda_i x^{i-1}$ ($\lambda_1 > 0$) and a base code C_b and let us define a new degree distribution $\tilde{\Lambda}(x) = \sum_{i=2}^{s} \tilde{\lambda}_i x^{i-1} = \sum_{i=2}^{s} \lambda_i x^{i-1} / \sum_{i=2}^{s} \lambda_i$. The average degree $\bar{\lambda}$ is defined over $\tilde{\lambda}_i$:

$$\bar{\lambda} = \frac{1}{\sum_{i=2}^{s} \tilde{\lambda}_i / i}$$

We focus on TLDPC code families with a constant fraction of bits of degree 1 and we study properties of entropy curves of their base codes.

Let us define the entropy curves of variable nodes and of the base code for a code family containing bits of degree 1 on the BEC.

Definition 7 *Let the base code of a TLDPC code family with bits of degree 1 and of length n_b have n_1 positions of degree > 1. Then we define extrinsic average entropy and intrinsic average entropy at the t^{th} iteration as follows*

$$E_t \stackrel{\text{def}}{=} \frac{1}{n_b - n_1} \sum_{i:deg(i)>1} E_t(i)$$

$$I_t \stackrel{\text{def}}{=} \frac{1}{n_b - n_1} \sum_{i:deg(i)>1} I_t(i)$$

with $E_t(i)$ ($I_t(i)$) being the entropy of the extrinsic (intrinsic) probability associated to the i-th position of the base code of degree > 1 at t-th iteration, $0 \leq i \leq n_b - 1$.

We define the entropy curve of variable nodes:

Lemma 7 *The entropy curve of variable nodes on the BEC with erasure probability x is given by the set of points $(p\tilde{\Lambda}(x), x)$, with $\tilde{\Lambda}(x) = \sum_{i=2}^{s} \tilde{\lambda}_i x^i$ where $\tilde{\lambda}_i = \lambda_i / \sum_{i=2}^{s} \lambda_i$.*

We define the entropy curve of the base code:

Lemma 8 *The entropy curve of the base code on the BEC with erasure probability p is given by the set of points (x, E_1) with:*

$$E_1 = \frac{1}{n_b - n_1} \sum_{i:deg(i)>1} h_i(x, p)$$

where $0 \leq x \leq 1$, $0 \leq i \leq n_b - 1$, *and*

$$h_i(x,p) = \sum_{(e,e_1)\in\mathcal{E}_i} p^{|e_1|}x^{|e|-1}(1-p)^{n_1-|e_1|}(1-x)^{n_b-n_1-|e|} \tag{3.16}$$

with $\mathcal{E}_i$ *to be the erasure configurations of bits of the base code for which the value of bit* i *cannot be found; a configuration consists of a set* e *of bits of the base code of degree* > 1 *and of a set* e_1 *of bits of the base code of degree* 1. $|e|$ *denotes the number of erasures in the set* e.

Similarly to the case without bits of degree 1,

Lemma 9 *An erasure configuration* (e, e_1) *is included in* $\mathcal{E}_i$ *if and only if there exists a codeword of the base code with* 0 *on all the unerased positions of* (e, e_1) *and* 1 *at position* i.

Note that in contrast to the case $\lambda_1 = 0$, the entropy curve of the base code h is not only function of the average intrinsic probability x but also of the channel erasure probability p, $h = h(x, p)$. This implies that, when p decreases, the entropy curves of the code family with $\lambda_1 > 0$ will move away from each other more quickly compared to the case $\lambda_1 = 0$ due to the dependence on p of the entropy curve of the base code. In its turn, such a divergence of entropy curves gives a faster convergence under iterative decoding and might lead to improved performances in the waterfall region. An example of influence of bits of degree 1 on the entropy curve of the base code is presented in Sections 3.8.3 and 5.3.5.

We compute the behaviour of entropy curves of variable nodes and of the base code at the origin for a TLDPC code family with bits of degree 1.

Theorem 8 *The derivative of the entropy curve of variable nodes at the origin for the BEC with erasure probability* p *is equal to*

$$\frac{1}{p\tilde{\lambda}_2}.$$

For the entropy curve of the base code we have

Theorem 9 *Consider transmission over the BEC with channel erasure probability* p. *Consider a base code of length* n_b *containing* n_1 *bits of degree 1. Let* m_2 *be the number of codewords of the base code of weight 2 containing only positions of degree* > 1 *and* $m_{2,j}$ *be the number of codewords of weight* $2 + j$ *containing 2 positions of degree* > 1 *and* j *positions of degree 1. Then the derivative at the origin of the entropy curve of the base code is equal to*

$$\frac{2}{n_b - n_1}(m_2 + \sum_{j=1}^{n_1} m_{2,j}p^j).$$

Proof : Let $m_2(i)$ be the number of codewords of the base code of weight 2 only containing positions of degree > 1, $m_{2,j}(i)$ be the number of codewords of weight $2 + j$ containing 2

positions of degree > 1 and j positions of degree 1 and all these codewords are equal to 1 at the position i. By using Lemmas 5 and 6 we obtain for $x \to 0^+$:

$$h_i(p,x) = m_2(i)x + x\sum_{j=1}^{n_1} m_{2,j}(i)p^j + O(x^2).$$

Therefore, the derivative of the entropy curve of the base code at the origin is equal to

$$\sum_{i=1}^{n_b-n_1} \frac{m_2(i) + \sum_{j=1}^{n_1} m_{2,j}(i)p^j}{n_b - n_1} = \frac{2}{n_b - n_1}(m_2 + \sum_{j=1}^{n_1} m_{2,j}p^j). \tag{3.17}$$

∎

There is a big advantage to allow bits of degree 1 in the code structure. It is observed in [72] that the derivative of area between the entropy curve of the base code for a graph-based code ensemble and the entropy curve of variable nodes is a function with two terms, the first one inversely depending of the average degree $\bar{\lambda}$ and the other one depending on λ_1. This implies that the convergence of the iterative decoding for a code with bits of degree 1 is faster and its performance curve in the waterfall region is steeper.

Examples of entropy curves for TLDPC code families with and without bits of degree 1 are presented in sections below. Before presenting particular examples, we study conditions to be put on the bipartite graph in order to have an asymptotically good code family.

3.7 Necessary conditions for being asymptotically good

An analysis of the number of codewords of a given weight w over the ensemble of n length codes obtained from the previous construction shows that their average a_w is of order $\Omega(1)$. This behaviour comes from the existence of cycles of a small size in a special graph which is derived from the bits of degree 2 and codewords of weight 2 in the base code. If one insists upon having a code family where almost all members are asymptotically good, it is crucial to have the average number of codewords of constant size be of order $o(1)$.

In this section we define this special graph for two cases: when the code family has no bits of degree 1 in its structure and when it has. The graph is called the graph of codewords of weight 2 in the first case and the graph of codewords of partial weight 2 in the second case[4]. We determine some restrictions on the average degree of the graphs which have to be satisfied to construct a family of asymptotically good codes.

Notice that the graph of codewords of (partial) weight 2 is nothing but a slight generalization of a graph which has been considered in [30] in the case of LDPC code ensembles.

3.7.1 Definition of the graph of codewords of (partial) weight 2

For code families without bits of degree 1, we define the graph of codewords of weight 2 with a help of the following definition:

[4]We write *graph of codewords of (partial) weight 2* if we refer to both graphs

Definition 8 [Clusters] *Two bits x_i and x_j of the base code are equivalent if and only if the base code contains a codeword of weight 2 with these bits equal to 1 and the others equal to 0. The ensembles of equivalent bits of the base code are called* **clusters**.

Now we are ready to define the graph of codewords of weight 2:

Definition 9 [Graph of codewords of weight 2, edge labelling] *For code families without bits of degree 1, the graph of codewords of weight 2 is the graph $G = (V, E)$ where V is the ensemble of clusters of equivalent bits from Definition 8 and E is the ensemble of edges e_{ij} between the clusters v_i and v_j if and only if there exist two bits x_k and x_l of the base code belonging to the clusters v_i and v_j respectively which join the same degree-2 variable node. The edge e_{ij} is labelled by the number t of the position of the corresponding variable node in the Tanner graph of the code.*

Examples

1. LDPC codes: recall that the base code of an LDPC code is a juxtaposition of single parity-check codes. Clusters in the graph of codewords of weight 2 will correspond to parity checks as any two bits involved in the same parity check form a support for a codeword of weight 2 in the base code. Two clusters are connected only if their corresponding parity checks are joined to the same variable node of degree 2 in the Tanner graph.

2. TLDPC code without bits of degree 1: in Fig.3.1a, we present a toy example of a code of length 6 defined by the base code having a six-section tail-biting trellis with two bits per section and by a bipartite graph between variable nodes representing bits of the code and positions of the base code. All the variable nodes are of degree 2.

   In this case we have six clusters, each of them corresponding to a single codeword of weight 2. The graph is shown in Fig.3.1b.

◇

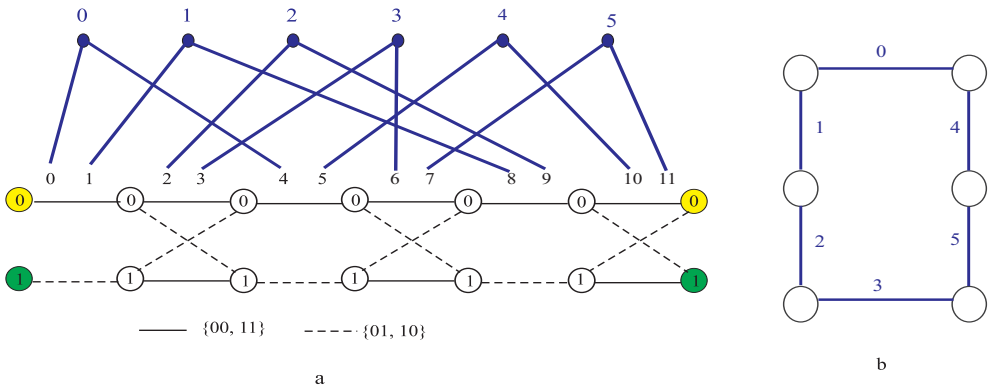


Figure 3.1: a) base code and bipartite graph for a code of length 6; b) its graph of codewords of weight 2.

For code families with degree 1 bits we consider codewords of the base code of partial weight 2:

Definition 10 [Codewords of the base code of partial weight 2] *For a base code with positions of degree 1, codewords of the base code of partial weight 2 are codewords which contain exactly two non-zero positions of degree 2 and all other non-zero positions of which are of degree 1.*

Example

For the TLDPC code with bits of degree 1, presented in Fig.3.2a, the base code contains two sections of length 1 which carry bits of degree 1; all the other trellis sections are of length 2 and carry bits of degree 2. There are 24 codewords of the base code: four of them, namely 1100000000, 0011000000, 0000011000 and 0000000110, have supports on positions of degree 2 only; 12 of codewords of partial weight 2 have one bit of degree 1 in their supports (as, for instance, 0010101000); last 8 of them contain 2 bits of degree 1 in their supports. ◇

The definition of equivalency between bits is based on codewords of partial weight 2 and is formulated as follows:

Definition 11 [Clusters] *Two bits x_i and x_j of the base code containing bits of degree 1 are equivalent if and only if the base code contains a codeword of partial weight 2 for which the non-zero positions of degree 2 are i and j. The ensembles of equivalent bits of the base code are called* **clusters**.

We can see that the definition of clusters is different from the previous one, when there were no bits of degree 1 in the code structure. Now the definition for the graph of codewords of partial weight 2 is the following one:

Definition 12 [Graph of codewords of partial weight 2, edge labelling] *For code families with bits of degree 1, the graph of codewords of partial weight 2 is the graph $G' = (V', E')$ where V' is the ensemble of clusters of equivalent bits from Definition 11 and E' is the ensemble of edges e_{ij} between the clusters v_i and v_j if and only if there exist two bits x_k and x_l of the base code belonging to the clusters v_i and v_j respectively which join the same degree-2 variable node. The edge e_{ij} is labelled by the number t of the position of the corresponding variable node in the Tanner graph of the code.*

Example

We continue the example of the code with bits of degree 1 in Fig.3.2a. Remark that any two positions of degree 2 of the base code form a support of a codeword of a partial weight 2. So, there is only one cluster in the corresponding graph of codewords of partial weight 2, and it contains all the positions of degree-2 of the base code. Such the graph of codewords of partial weight 2 with one cluster is presented in Fig.3.2b. ◇

3.7.2 Cycles in the graph of codewords of (partial) weight 2

In this section we show the correspondence between cycles in the graph of codewords of (partial) weight 2and codewords of the code. First we define weights of nodes and of cycles in the graph of codewords of (partial) weight 2:

Definition 13 [Node weights] *For a node v in the graph of codewords of (partial) weight 2 and for two edges i and j connected to it we define a node weight $w^v_{i,j}$ to be the number of positions of degree 1 in the base code in the corresponding codeword of partial weight 2 containing two non-zero positions of degree 2 in v which are connected by edges i and j.*

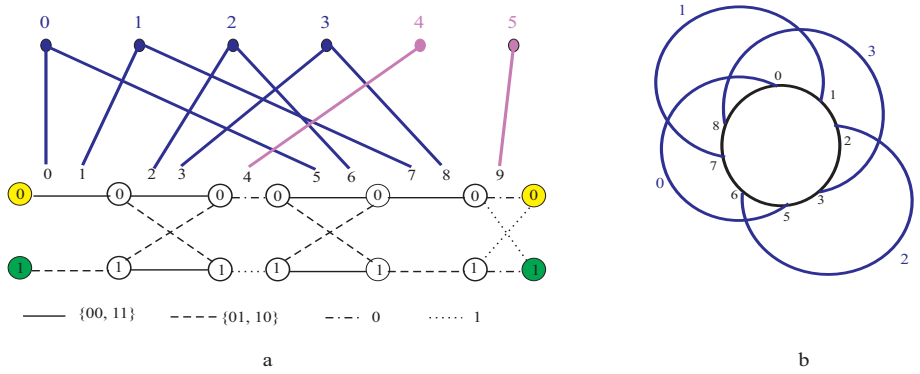


Figure 3.2: a) base code and bipartite graph for a code of length 6; b) its graph of codewords of partial weight 2 containing only one cluster.

Definition 14 [Cycle weight] *The weight l of a cycle $C = (V_C, E_C)$ in the graph of codewords of (partial) weight 2 is equal to*

$$l = |E_C| + \sum_{v \in V_C} w^v,$$

where w^v is the node weight associated with v and two edges in E_C connected to v.

The following proposition shows the influence of cycles in the graph of codewords of (partial) weight 2 on the weight distribution of the code:

Proposition 4 *A cycle of weight l in the graph of codewords of (partial) weight 2 induces a codeword of weight l in the code.*
Proof : If $C = (V_C, E_C)$ is a cycle in G (G'), we associate to it a configuration $\mathbf{x} = (x_1, x_2, \ldots, x_{n_b})$[5] of positions of the base code in which

- positions of the base code of degree 2 are put to 1 if in the Tanner graph they are connected to the variable nodes of degree 2 which are associated with edges in E_C;
- a set B of positions of degree 1 is put to 1 if they form a codeword of the base code of partial weight 2 with two positions of degree 2;
- all other positions in $\mathbf{x}$ are put to 0.

Denote by w^v the size of the set B for a node $v \in V_C$. The configuration $\mathbf{x}$ gives a codeword of weight $2|E_C| + \sum_{v \in V_C} w^v$ in the base code. $2|E_C|$ non-zero bits of $\mathbf{x}$ are connected to degree-2 variable nodes and the rest of them is connected to degree-1 variable nodes. Thus, there are $|E_C| + \sum_{v \in V_C} w^v$ of variable nodes participating in the configuration $\mathbf{x}$ and they correspond to a codeword of weight $|E_C| + \sum_{v \in V_C} w^v$. ∎

Note that the weight of the smallest cycle in the graph of codewords of (partial) weight 2 is an upper bound on the code minimum distance. We formulate two following corollaries:

[5] n_b is the length of the base code

Corollary 1 *For a code ensemble without bits of degree 1 to which the graph of codewords of weight 2 G is associated to, all the node weights are 0 and the minimum distance is upper bounded by the value $|E_C|$ of the smallest cycle in G.*

Corollary 2 *For a code ensemble with bits of degree 1 if the node weights $w_{i,j}^v$ of its corresponding graph of codewords of partial weight 2 are smaller than some small positive integer a, the minimum distance is upper bounded by $(a+1)|E_C|$.*

Thus, if the graph of codewords of (partial) weight 2 in these two cases contains a cycle of logarithmic length, the minimum distance will be logarithmic in the codelength. It can be also noticed that it is possible to obtain a lower average of the number of codewords of given weight a_w of order $o(1)$ by avoiding cycles of weight l smaller than a fraction δ of the code length n ($l \geq \delta n$) in the graph of codewords of (partial) weight 2.

All the TLDPC code families with bits of degree 1 presented in the thesis have bounded node weights.

It may happen for a given code family with bits of degree 1 (for instance, for turbo codes) that node weights in the corresponding graph of codewords of partial weight 2 are unbounded. Then the cycle weights are not bounded either, and thus small cycles do not necessarily correspond to low-weight codewords. We treat this particular case below.

Particular case of unbounded node weights

We proceed in a slightly more complicated way by considering a new graph G_k called graph of codewords of partial weight 2 of order k:

Definition 15 [Graph of codewords of partial weight 2 of order k] *A graph of codewords of partial weight 2 G_k of order k is defined by*

- *the set of vertices V_k corresponding to the set of positions of degree 2 of the base code,*
- *the set of edges E_k partitioned into two subsets E_k^R and E_k^B of red and blue edges,*
- *two vertices in V_k are connected by a red edge of weight t if they form a codeword of the base code of partial weight 2 and of total weight $t+2$, for some $t \leq k$,*
- *two vertices in V_k are connected by a blue edge of weight 1 if their corresponding positions of degree 2 are associated to the same degree-2 variable node in the Tanner graph of the code.*

The graph of codewords of partial weight 2 can be viewed as a graph G_k of the largest possible order k if edges in G' are represented by blue ones and the vertices in G' are represented by red-connected components of sets of vertices connected by red edges.

Example

We show an example of the node v of degree 8 in the graph of codewords of partial weight 2 in Fig.3.2b viewed as red-connected component in Fig.3.4a. Nodes of the component are associated to positions of degree-2 in the base code, the tail-biting trellis of which is recalled in Fig.3.3. Lines of different styles represent edges of different weight and the weight of an edge between positions i and j corresponds to the node weight w_{ij}^v.

As the nodes of G' are equivalent classes, i.e. any position of degree 2 in a node form a codeword of base code of partial weight 2 with any other position in the same node, its corresponding red-connected component is a complete graph. In Fig.3.4b-d corresponding red-connected components of G_0 and G_1 are represented. $\diamond$

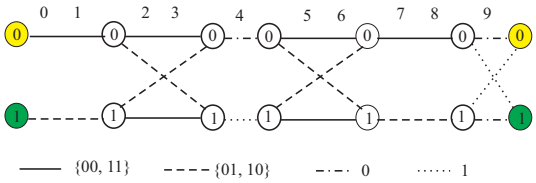


Figure 3.3: Tail-biting trellis of the base code from the previous example.

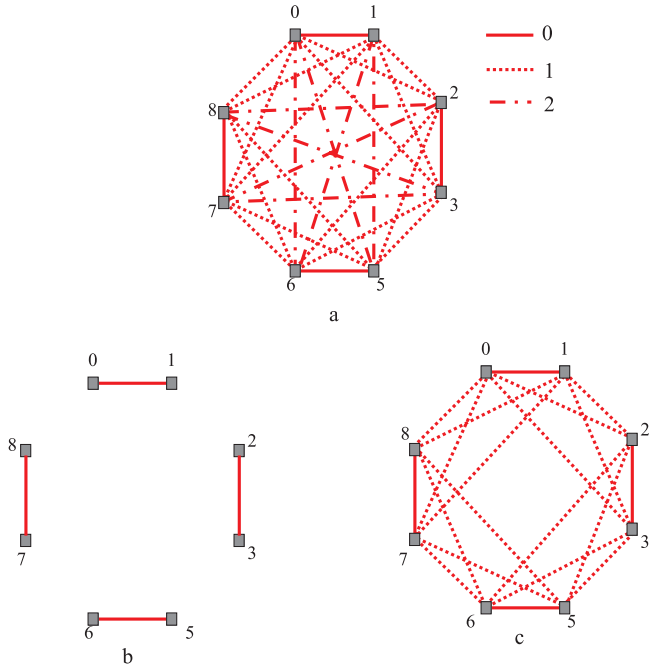


Figure 3.4: a) complete red-connected component of the node of degree 8 in G' in Fig.3.2b; b) component in G_0; c) component in G_1.

Now consider cycles in G_k. We define the weight of a cycle for a graph of codewords of partial weight 2 G_k of order k

Definition 16 [Cycle weight for G_k] *The weight of a cycle in G_k is the sum of edge weights participating in it.*

We do the following proposition:

Proposition 5 *A cycle in G_k of weight l containing alternating blue and red edges induces a codeword of weight l in the code.*

Considering graphs of codewords of partial weight 2 of order k, we can give an upper bound on the minimum distance of the corresponding code ensemble by using Proposition 5:

Corollary 3 *If the graph G_k of order k contains cycles with alternating red and blue edges, then the minimum distance is bounded by $(k+1)E_C$, where E_C is the number of edges of the smallest cycle.*

3.7.3 Upper bound on the average degree of the graph of codewords of (partial) weight 2

It is known that the girth of a d-regular graph with $d > 2$ is logarithmic in the size of the graph. Recently, Alon et al. have proved ([3]) that the same result holds for an irregular graph of average degree d, $d > 2$. We present it in the following lemma:

Lemma 10 *Let (V',E') be an irregular graph of average degree $\bar{d}$. If N_j^l denotes the depth-l neighbour set of a node j such that $N_j^l = V'$, then, when $d > 2$, l is upper bounded by*

$$l \leq \lceil t \rceil$$

where t satisfies

$$t = \begin{cases} \frac{\log(m-2m/\bar{d}+1)}{\log(\bar{d}-1)} & \text{for odd g} \\ \frac{\log(m(\bar{d}-2)+\bar{d})-\log 2}{\log(\bar{d}-1)} & \text{for even g} \end{cases}, \tag{3.18}$$

$\lceil \cdot \rceil$ indicates the ceiling of a real number and m denotes the cardinality of the set V'.

The proof follows directly from the result of Alon et al. in [3].

In the previous section it was shown the connection between codewords of a TLDPC codes family with or without bits of degree 1 and cycles in its corresponding graph of codewords of (partial) weight 2. When constructing a family of asymptotically good TLDPC codes, we must avoid codewords of sublinear weight and thus avoid cycles of sublinear weight in its corresponding graph of codewords of (partial) weight 2. This consideration naturally gives us the upper bound on the average degree of the graph of codewords of (partial) weight 2:

Theorem 10 *A TLDPC code ensemble without bits of degree 1 or a TLDPC code ensemble with bits of degree 1 having node weights bounded by a small positive integer a is not asymptotically good when the average degree of the graph of codewords of weight 2 of its members is greater than $2 + \epsilon$ for some $\epsilon > 0$.*

The proof follows directly from Lemma 10 and Proposition 4.

In the unbounded case, we have the following theorem

Theorem 11 *A TLDPC code ensemble with bits of degree 1 and with unbounded node weights is not asymptotically good when the average degree of its graph G'_k of order k is greater than $2 + \epsilon$ for some $\epsilon > 0$.*

The fact that the parallel turbo-codes have at most logarithmic minimum distance in the codelength is a consequence of Theorem 11: one can find a graph G_k of order with the average degree greater than two.

3.7.4 Conditions on permutation of bits of degree > 2

By choosing the structured permutation for degree-2 variable nodes in the Tanner graph of of an TLDPC code we avoid configurations corresponding to codewords of logarithmic

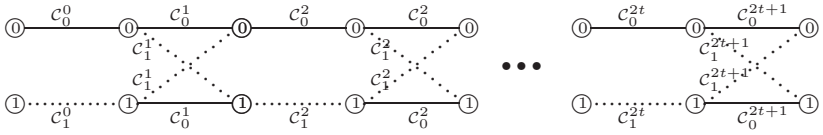


Figure 3.5: Tail-biting trellis of the base code for different TLDPC code families.

size which appear with constant probability when the permutation is chosen at random. However, in order to construct an asymptotically good code, permutation for variable nodes of degrees > 2 must be taken into account.

It was recently shown in [73] that for LDPC codes of length n having the same number of variable nodes of degree-2 and of check nodes, for which all other variable nodes are of degree $c > 2$ and the corresponding graph of codewords of weight 2 contains only a single cycle, the minimum distance is bounded by $\Theta(n^{1-1/c})$. The similar result holds in more general case when the graph of codewords of weight 2 is arbitrary (but always has m edges). The upper bound is due to the presence of codewords of polynomial weight which correspond to configurations in the Tanner graph having a small number (say, j) of variable nodes of degree c and the rest of variable nodes being of degree 2, for which the graph of codewords of weight 2 has $cj/2$ connected components.

3.8 Particular TLDPC code families

Our approach to obtain codes with good performances under a low-complexity iterative decoding algorithm is to seek for base codes for which the EXIT chart [7] comes close to *a straight line*. If this were the case, one could choose all the bits to be of degree 2 and attain the capacity of the erasure channel. Even if this is not exactly the case, the optimisation of the degree distribution $\Lambda(x)$ yields a *large* fraction of degree 2 nodes which gives an iterative decoding algorithm of low complexity.

For this purpose, we have noticed that base codes defined from convolutional structures help us to obtain such a straight line. We have even found out that there is no need to choose large complexity trellises.

More precisely we have focused on base codes obtained from the (multi-edge) tail-biting trellis presented in Fig.3.5.

The C_0^i's are chosen as single parity-check codes and $\mathcal{C}_1^i$ is the complementary coset of $\mathcal{C}_0^i$. Note that the tail-biting trellis can have bits of degree 1 as its labels.

In what follows, we focus on the 6 families derived from the following choices:

Code families without bits of degree 1

(A) $\mathcal{C}_0^i = \{00, 11\}$ and $\mathcal{C}_1^i = \{10, 01\}$

(B) $\mathcal{C}_0^i = \{000, 011, 101, 110\}$ and $\mathcal{C}_1^i = \{001, 010, 100, 111\}$

Code families with bits of degree 1

(C) $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$ for $i \equiv 0 \pmod 4$, $\mathcal{C}_0^i = \{00, 11\}$ and $\mathcal{C}_1^i = \{01, 10\}$ otherwise

(D) $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$ for $i \equiv 0 \pmod 4$, $\mathcal{C}_0^i = \{000, 011, 101, 110\}$, and $\mathcal{C}_1^i = \{001, 010, 100, 111\}$. for $i \equiv 6 \pmod 8$, $\mathcal{C}_0^i = \{00, 11\}$ and $\mathcal{C}_1^i = \{01, 10\}$ otherwise

(E) $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$ for $i \equiv 0 \pmod 5$, $\mathcal{C}_0^i = \{00, 11\}$ and $\mathcal{C}_1^i = \{01, 10\}$ otherwise

(F) $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$

Base codes of all the presented TLDPC code ensembles are defined by concatenation of trellis sections as shown in Fig.3.5.

Families A and B were first presented in [4], families C, D and E - in [5]. We describe properties and performances of each of code families in the sections below.

3.8.1 Family A

For the family A $C_0 = \{00, 11\}$ and $C_1\{01, 10\}$.

We can easily calculate two ratios - number of codewords of weight 2 or 3 divided by the length of the base code

$$\begin{aligned} \frac{m_2}{n_b} &= 1/2 \\ \frac{m_3}{n_b} &= 2. \end{aligned}$$

The first ratio assures that the corresponding graph of codewords of weight 2 has average degree at most 2 (under condition that all the variable nodes have degree 2). In this case we can choose the permutation such as to avoid cycles in the graph of codewords of weight 2. The second ratio (which is quite high for a low-complexity trellis) assures that the slope of the entropy curve of the base code at the origin quickly increases when the intrinsic entropy increases.

EXIT chart analysis

Note that the entropy curve of the family-A base code on the BEC can be calculated analytically and it is done in Appendix B.2.1.

If we trace the entropy curve of the family-A base code and the entropy curve of the degree distribution $\Lambda(x) = x$ for different erasure channel probabilities p, we can see that this curve is very close to the entropy curve of the base code when p is around 0.45301 (see Fig.3.6). This indicates that the threshold for the iterative decoding of this code ensemble is around this value.

For the Gaussian channel we have similar curves (see Fig.3.7), and the threshold of the Gaussian noise for which the entropy curve of variable nodes is above the entropy curve of the base code is of about 0.77 dB. Note that for the code rate 1/2 the entropy of the Gaussian noise of 0.77 dB corresponds approximatively to the entropy of the BEC with erasure probability $p = 0.45$ (see Appendix B.3).

When the fraction of bits of degree 2 satisfies $\lambda_2 < 1$ and the permutation is appropriately chosen to avoid cycles in the graph of codewords of weight 2 as it will be explained further, then such a constructed code family is asymptotically good. We show this in the next section.

The introduction of bits of higher degrees in the code structure represents several advantages:

- to obtain asymptotically good codes in a simple way,

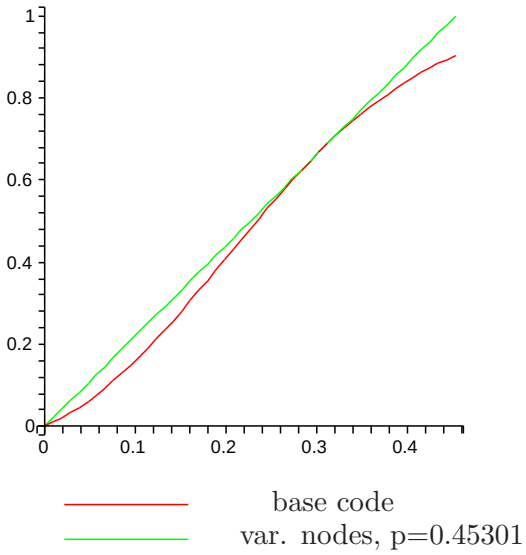


Figure 3.6: Entropy curve of the family-A base code and entropy curve for $\Lambda(x) = x$ and $p = 0.45301$.

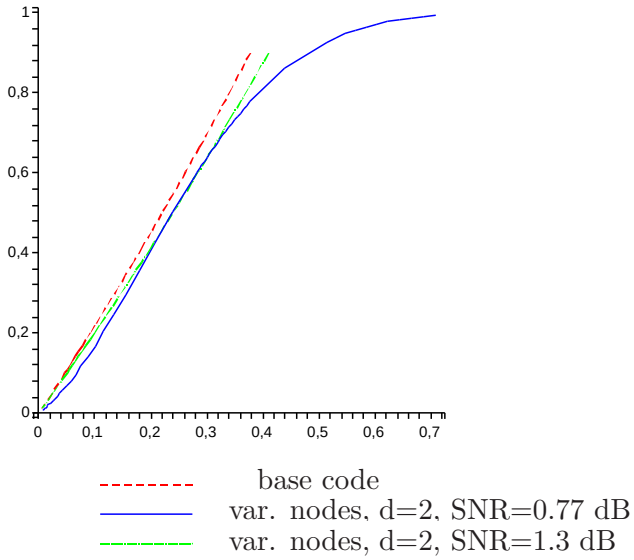


Figure 3.7: Entropy curve of the family-A base code and entropy curve for $\Lambda(x) = x$ and for the channel of SNR's 0.77 and 1.3dB respectively.

- to increase the number of parameters and thus to increase the degree of freedom for the code optimisation.

For example, we construct an irregular family-A code family of rate 1/3 and after the degree distribution optimisation, we obtain $\Lambda(x) = 0.7114317527x + 0.08251239426x^{13} +$

$0.06786376632x^{14} + 0.1381920867x^{15}$. The threshold erasure probability for this code family is 0.635 as can be seen from Fig.3.8. We did the same kind of optimisation for

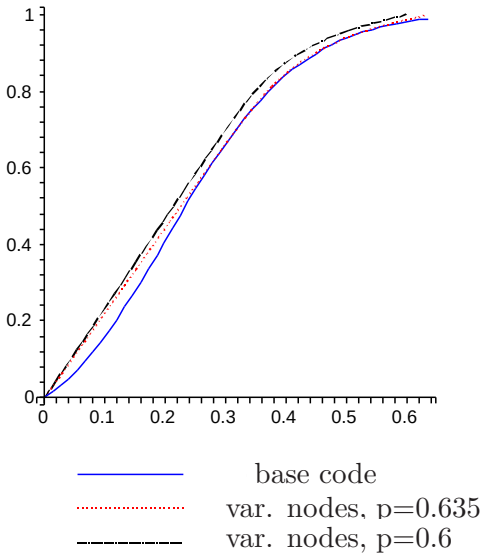


Figure 3.8: Entropy curve of the family-A base code and entropy curve associated to $\Lambda(x) = 0.7114317527x + 0.08251239426x^{13} + 0.06786376632x^{14} + 0.1381920867x^{15}$ for erasure probabilities 0.635 and 0.6 respectively.

the Gaussian channel and found the degree distribution $\Lambda(x) = 0.7x + 0.3x^{11}$ for the code rate 1/3. If we plot the entropy curve associated to $\Lambda(x)$ for different SNR's (see Fig.3.9), we can see that the entropy curve obtained for 0.0186 dB is slightly above of the entropy curve of the base code. This gives an estimate of the noise threshold which is sustained by this code family.

Permutation choice

For the family-A base code all bits associated with the same trellis section belong to the same cluster. So, clusters of the graph of codewords of weight 2 always contain 2 bits. Thus, by Theorem 10, we can always avoid cycles of sublinear size in the graph of codewords of weight 2 for any value of λ_2.

Let n be the code length, n_s be the number of sections in the base code and λ_i be the fraction of degree-i variable nodes. To construct a permutation we use the following procedure:

- generation of a random cyclic permutation of length n_s with the vertices being associated to sections of the base code;
- uniform choice of $\lambda_2 n$ edges of this cycle and their association to degree-2 variable nodes: for the edge connecting a section i to a section j there is a corresponding degree-2 variable node in the bipartite graph of the code which connects these two sections of the base code. We notice that as the codewords of weight 2 are only

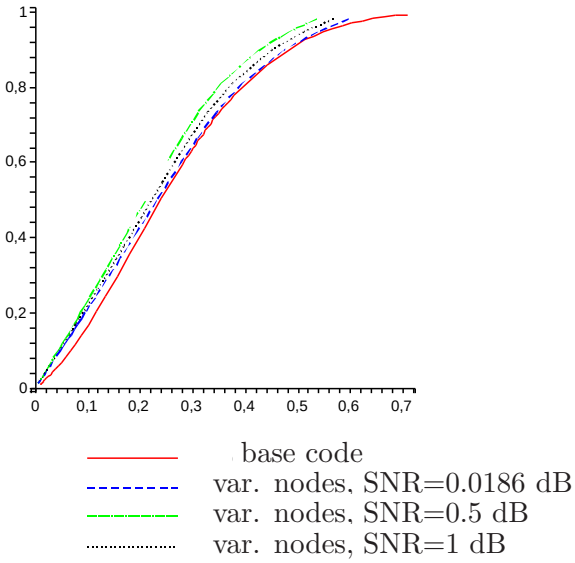
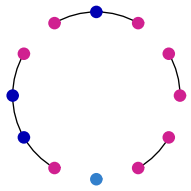


Figure 3.9: Entropy curve of the family-A base code and entropy curve associated to $\Lambda(x) = 0.7x + 0.3x^{11}$ for Gaussian channel with SNR 0.0186dB, 0.5dB, 1dB.

located within the sections, they completely define the graph of codewords of weight 2 associated to our code.

- given the structure of the bipartite graph for the degree-2 variable nodes, the choice of connections for variable nodes of higher degrees is made at random among the remaining positions in the sections of the base code.

In the case of $\lambda_2 = 1$, $n = n_s$ and the graph of codewords of weight 2 is a length-n cycle; it is a forest when $\lambda_2 < 1$ as it is shown in the figure on the right. Light-blue points represent clusters of degree 0, violet red points - clusters of degree 1 and blue - clusters of degree 2.



Proof for being asymptotically good when $\lambda_2 < 1$

In this section we prove that most codes of family A are asymptotically good when the random bipartite graph G is chosen as explained in Section 3.8.1 and $\lambda_2 < 1$. The proof can be decomposed in four steps.

1. Calculation of the average number $\bar{a}_w$ of codewords of given weight w.

   We compute the average number $\bar{a}_w$ of codewords of given weight w taking the average over all the bipartite graphs which are defined in Section 3.8.1; we use a generating function approach by conditioning on well chosen events.

   We say that a binary word **c**overs t sections of the base code C_b if its non-zero elements are located on exactly t sections of C_b. As before, let $b(x, y)$ be the weight

enumerator polynomial of the base code,

$$b(x,y) = \sum b_{i,j} x^i y^j,$$

where $b_{i,j}$ is the number of codewords of the base code of weight i covering j sections.

Let I be a matching function from $\{0,1\}^n$ to $\{0,1\}^m$ [6] which gives the assignments of the bits of the base code C_b induced by the assignments of the bits of the code C through the Tanner graph G of the code C. Then we use the following notation:

- $\mathbf{x}_w \in \{0,1\}^n$ be the binary word of weight w whose first w coordinates are equal to 1;
- S be the number of sections of the base code C_b covered by $I(\mathbf{x}_w)$;
- T be the weight of $I(\mathbf{x}_w)$.

By linearity of expectation we have

$$\begin{aligned} \bar{a}_w &= \binom{n}{w} \mathbf{P}(I(\mathbf{x}_w) \in C_b) \\ &= \binom{n}{w} \sum_{s,t} \mathbf{P}(I(\mathbf{x}_w) \in C_b \mid S = s, T = t) \mathbf{P}(S = s, T = t) \end{aligned}$$

The last conditioning is valid due to the fact that the distribution of the matching function $I(\mathbf{x}_w)$ is *uniform* over all words of weight t covering s sections of C_b. We denote by $a_{s,t}$ the number of weight-t codewords of C_b covering s sections, by $b_{s,t}$ the number of words of length m and of weight t with the 1's located in s subvectors obtained by partitioning the initial word into subvectors of length 2. Thus, the probability $\mathbf{P}(I(\mathbf{x}_w) \in C_b \mid S = s, T = t)$ is simply the quotient of $a_{s,t}$ by $b_{s,t}$,

$$\mathbf{P}(I(\mathbf{x}_w) \in C_b \mid S = s, T = t) = \frac{a_{s,t}}{b_{s,t}}. \tag{3.19}$$

Let us consider the denominator $b_{s,t}$. It is easy to see that $b_{s,t}$ is simply

$$b_{s,t} = 2^{2s-t} \binom{\frac{\bar{\lambda}n}{2}}{2s-t} \binom{\frac{\bar{\lambda}n}{2} - 2s + t}{t-s},$$

which is equal to choose $2s-t$ subvectors of weight 1 and $t-s$ subvectors of weight 2. This gives us $2s - t + t - s = s$ covered subvectors and $2s - t + 2(t-s) = t$ 1's in every word.

As for the numerator $a_{s,t}$, we introduce the following lemma using a generating function approach:

Lemma 11 *For two consecutive trellis sections of the family-A base code let*

$$A \stackrel{\text{def}}{=} \begin{bmatrix} 1 + 2x^2y + x^4y^2 & 4x^2y^2 \\ 4x^2y^2 & 2xy + 2x^3y^2 \end{bmatrix}$$

[6] Note that the length of the base code is $m = \bar{\lambda}n$, $\bar{\lambda}$ being the the average degree of the Tanner graph of the code

be a matrix where the $A_{i,j}$-th element is equal to the extended path enumerator of the set of words with corresponding trellis paths going from the state i to the state j. We note the path weight variable as x and covered sections variable as y. Then we calculate $a_{s,t}$ as follows

$$a_{s,t} = \lfloor \mathbf{tr} A^{\frac{\bar{\lambda} n}{4}} \rfloor_{s,t}.$$

Now let us obtain a formula for $\mathbf{P}(S = s, T = t)$. We define the number of bits of the base code equal to 1 which are connected to the degree-2 variable nodes and to variable nodes of higher degrees as T_1 and T_2 correspondingly. More precisely, $T_1 = \#\{i \mid I(\mathbf{x}_w)_i = 1, \deg(i) = 2\}$ and $T_2 = \#\{i \mid I(\mathbf{x}_w)_i = 1, \deg(i) > 2\}$. We rewrite $\mathbf{P}(S = s, T = t)$ in the form:

$$\mathbf{P}(S = s, T = t) = \tag{3.20}$$

$$= \sum_{t_1 + t_2 = t} \mathbf{P}(T_1 = t_1)\mathbf{P}(T_2 = t_2 \mid T_1 = t_1)\mathbf{P}(S = s \mid T_1 = t_1, T_2 = t_2) \tag{3.21}$$

Let $n_1 \stackrel{\text{def}}{=} \frac{\bar{\lambda}\lambda_2 n}{2}$ be the total number of bits of the global code of degree 2. As all the bits of the global code are assembled along a unique cycle, the number of "touched" bits of the global code of degree 2 is simply equal to $w_1 \stackrel{\text{def}}{=} t_1/2$. So, the first term in (3.20) is equal to

$$\mathbf{P}(T_1 = t_1) = \frac{\binom{n_1}{w_1}\binom{n-n_1}{w-w_1}}{\binom{n}{w}}, \tag{3.22}$$

and the second term is equal to

$$\mathbf{P}(T_2 = t_2 \mid T_1 = t_1) = \frac{q_{w-w1,t_2}}{\binom{n-n_1}{w-w_1}}, \tag{3.23}$$

where q_{w-w_1,t_2} is given by the generating function $q(x, y) \stackrel{\text{def}}{=} \prod_{j>2}(1 + xy^j)^{\tilde{\lambda}_j n}$. In other words, $q_{k,l}$ is the number of possibilities to choose k variable nodes of degree $j > 2$ with exactly l outgoing edges.

Let us estimate the last term in (3.20). We denote S_1 to be the number of sections covered by degree-2 bits in $I(\mathbf{x}_w)$. Notice that $w_1+1 \leq S_1 \leq 2w_1$. By conditioning on the event $S_1 = s_1$ we obtain the following expression for $\mathbf{P}(S = s \mid T_1 = t_1, T_2 = t_2)$;

$$\mathbf{P}(S = s \mid T_1 = t_1, T_2 = t_2) = \tag{3.24}$$

$$= \sum_{s_1 = w_1 + 1}^{2w_1} \mathbf{P}(S_1 = s_1 \mid T_1 = t_1)\mathbf{P}(S = s \mid T_1 = t_1, T_2 = t_2, S_1 = s_1). \tag{3.25}$$

The first term can be calculated by a generating function approach.

Lemma 12 *For a trellis section of the family-A base code we make a correspondence to a trellis section with the transition matrix B described as follows*

$$B \stackrel{\text{def}}{=} \begin{bmatrix} 1 & xy \\ xy & x^2 y \end{bmatrix},$$

where x is the weight variable and y is the covered sections variable. Let $\sum_{t,s} d_{t,s} x^t y^s \stackrel{\text{def}}{=} \mathbf{tr} B^{\frac{\bar{\lambda}n}{2}}$, then

$$\mathbf{P}(S_1 = s_1 \mid T_1 = t_1) = \frac{d_{t_1,s_1}}{\sum_s d_{t_1,s}} = \frac{d_{t_1,s_1}}{\binom{\frac{\bar{\lambda}n}{2}}{\frac{t_1}{2}}}.$$

The last term in (3.24) can be expressed as

$$\mathbf{P}(S = s \mid T_1 = t_1, T_2 = t_2, S_1 = s_1) = \tag{3.26}$$

$$= \sum_u \frac{\binom{2s_1 - t_1}{u} 2^{2(s-s_1)-(t_2-u)} \binom{\frac{\bar{\lambda}n}{2} - s_1}{2(s-s_1)-(t_2-u)} \binom{\frac{\bar{\lambda}n}{2} - 2s + s_1 + t_2 - u}{(t_2-u)-(s-s_1)}}{\binom{\bar{\lambda}n - t_1}{t_2}}, \tag{3.27}$$

where a term of the sum is the number of possibilities to choose $s - s_1$ sections of the base code which contain t_2 bits equal to 1 and connected to variable nodes of degrees greater than 2 partitioned in the following way: u sections contain already one bit connected to a degree-2 variable node, $2(s - s_1) - (t_2 - u)$ sections contain exactly one bit from t_2 and $(t_2 - u) - (s - s_1)$ sections contain two bits from t_2.

Putting all these formulas together, we obtain an explicit expression of $\bar{a}_w$:

$$\bar{a}_w = \binom{n}{w} \sum_{s,t} \frac{a_{s,t}}{2^{2s-t} \binom{\frac{\bar{\lambda}n}{2}}{2s-t} \binom{\frac{\bar{\lambda}n}{2} - 2s + t}{t-s}} \cdot \sum_{t_1+t_2=t} \frac{\binom{n_1}{w_1} q_{w-w_1,t_2}}{\binom{n}{w}} \cdot \sum_{s_1=w_1+1}^{2w_1} \frac{d_{t_1,s_1}}{\binom{\frac{\bar{\lambda}n}{2}}{\frac{t_1}{2}}} \cdot$$

$$\sum_u \frac{\binom{2s_1 - t_1}{u} 2^{2(s-s_1)-(t_2-u)} \binom{\frac{\bar{\lambda}n}{2} - s_1}{2(s-s_1)-(t_2-u)} \binom{\frac{\bar{\lambda}n}{2} - 2s + s_1 + t_2 - u}{(t_2-u)-(s-s_1)}}{\binom{\bar{\lambda}n - t_1}{t_2}}$$

It involves sums of terms each of which represents a product of binomial coefficients and terms $a_{s,t}$, q_{w-w_1,t_2}, d_{t_1,s_1}.

2. Calculation of $\alpha(\delta)$.

   Let us compute the average growth rate $\alpha(\delta)$ of family-A codes. To do this, we have to estimate $\bar{a}_w$. As we have seen in the previous step, the formula of $\bar{a}_w$ is a sum of a polynomial number of terms in n, each of them displaying an exponential behaviour in n. Obviously, $\alpha(\delta)$ is simply the largest exponent in the aforementioned sum.

   Given that the binomial coefficients $\binom{n}{i}$ are readily estimated by the well-known formula

$$\binom{n}{i} = 2^{h(\frac{i}{n})n(1+o(1))},$$

   we only have to estimate the coefficients $a_{s,t}$, q_{w-w_1,t_2}, d_{t_1,s_1}. All these terms are coefficients of some polynomial $p(x,y) = \sum_{i,j} p_{i,j} x^i y^j$ with only nonnegative coefficients. We use the upper bound 2 of coefficients $p_{i,j}$. By using large deviation results (local forms of the Gaerther-Ellis theorem, [27],Section 2.3) it can be proved that this upper bound captures the correct exponent of the considered terms $a_{s,t}$, q_{w-w_1,t_2}, d_{t_1,s_1}.

Lemma 13 *Let $\mu_A(x,y)$ and $\mu_B(x,y)$ be the largest eigenvalues of A and B respectively. For $\sigma > 0$, $\tau > 0$, $\omega > 0$ we have*

$$\lim_{n\to\infty} \frac{\log(a_{\sigma n,\tau n})}{n} = \inf_{x>0,y>0} \frac{\bar{\lambda}}{4} \ln \mu_A(x,y) - \sigma \ln x - \tau \ln y, \tag{3.28}$$

$$\lim_{n\to\infty} \frac{\log(q_{\sigma\omega,\tau n})}{n} = \inf_{x>0,y>0} \sum_{i>2} \tilde{\lambda}_i \log(1+xy^i) - \omega \ln x - \tau \ln y, \tag{3.29}$$

$$\lim_{n\to\infty} \frac{\log(d_{\sigma n,\omega n})}{n} = \inf_{x>0,y>0} \frac{\bar{\lambda}}{2} \ln \mu_B(x,y) - \sigma \ln x - \omega \ln y. \tag{3.30}$$

The result is obtained by using the techniques described above as we have done this in (3.9).

3. Behaviour of $\alpha(\delta)$ around 0.

To prove that the family-A code family is asymptotically good, we explore the behaviour of $\alpha(\delta)$ around 0. For this purpose we capture the behaviour for small σ, τ, ω of the expressions in Lemma 13, which turns out to be rather simple. We use it to prove that $\alpha(\delta)$ can be expressed in the form $-K\delta\log\delta + O(\delta)$ for small δ and some positive $K > 0$.

We obtain the following results:

Lemma 14 *For $\omega \to 0^+$ and $\sigma \le K\omega$, $\tau \le K'\omega$ for some constants K, $K' \ge 0$ we have*

$$\lim_{n\to\infty} \frac{\log(a_{\sigma n,\tau n})}{n} = -(\sigma-\tau)\log(\sigma-\tau) - (2\sigma-\tau)\log(2\sigma-\tau) + O(\omega).$$

Lemma 15 *As $\omega \to 0^+$ and $\frac{\tau}{\omega}$ is kept fixed and equal to some constant α, we can estimate the behaviour of $c_{s,t}$ as given:*

$$\lim_{n\to\infty} \frac{\log(q_{\sigma\omega,\tau n})}{n} = h(\omega) + \delta K(\alpha) + o(\omega),$$

where $K(\alpha) \stackrel{\text{def}}{=} \inf_{x>0} \ln q(x) - \alpha \ln x$, $q(x)$ being equal to $q(x) = \sum_{i>2} \tilde{\lambda}_i x^i$. $K(\alpha)$ is an increasing and continuous function for $\alpha \in \left[d, \bar{\lambda}\right]$, $d \stackrel{\text{def}}{=} \min\{i > 2 \mid \lambda_i \neq 0\}$, which satisfies:

- *$K(\tau) = -\infty$ for $\tau < d$,*
- *$K(d) = \ln(\lambda_d)$.*

Lemma 16 *For $\omega < \sigma \le 2\omega$, let k be an integer such that $\frac{k+1}{k} \le \frac{\sigma}{\omega} < \frac{k}{k-1}$ and $\varepsilon = \sigma - \omega$, then*

$$\lim_{n\to\infty} \frac{\log(d_{\sigma n,\omega n})}{n} = -\varepsilon\log\varepsilon - (\tau - k\varepsilon)\log(\tau - k\varepsilon) + O(\tau).$$

4. Proof that the code family A is asymptotically good when $\lambda_2 < 1$.
   Gathering together Lemmas 14, 15 and 16, for $\lambda_2 < 1$ we obtain that $\alpha(\delta) = -C\delta\log(\delta) + O(\delta)$ for some positive constant C. We choose $\delta_0 > 0$ so that for all $0 < \delta < \delta_0$ the following inequality holds: $\alpha(\delta) \leq -\frac{C}{2}\delta\log(\delta)$. We use the inequality

$$\mathbf{P}(\text{minimum distance of the global code} \leq \delta_0 n) \leq \sum_{i=1}^{\delta_0 n} \bar{a}_i$$

   to show that the probability for being asymptotically good tends to 1 as the n tends to infinity.

Remark that in order to study the behaviour of the family-A code ensemble with $\lambda_2 = 1$ we need to take into account terms of second order in our computations. It seems to us that the proof technique used in [73] also applies here, and that the expected minimum distance of such codes would be of order $\Theta(n^{1/3})$.

Simulation results

In this part we present simulation results for family-A codes of different lengths and rates.

In Fig.3.10 and 3.11 we present simulations results on the Gaussian channel for family-A codes of rate 1/2 and of length 2000, 4000 and 16000 respectively. The maximum number of iterations is fixed to 50. The degree distribution is $\Lambda(x) = x$.

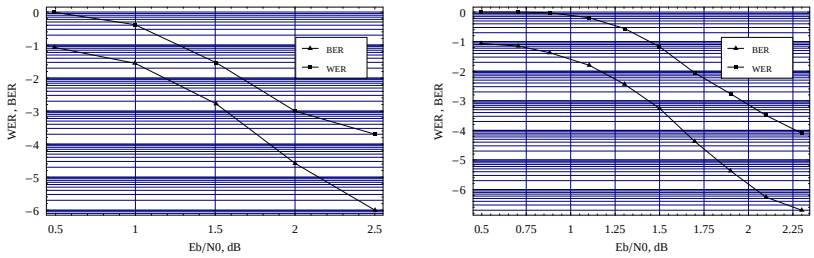


Figure 3.10: Performance of family-A codes of length 2000 (left) and 4000 (right) of rate 1/2 on the Gaussian channel, $\Lambda(x) = x$.

We compare the family-A code $[16000, 8000]$ with a duo-binary turbo code of the same length and dimension decoded with 8 iterations. In all three cases we see the phenomenon of error-floor. Its study did not reveal the presence of low-weight codewords but error configurations for which either the decoding process is extremely slow (we need 60-120 iterations to make such decodings successful) or it does not converge. Nevertheless, we note that the family-A code of length 16000, though having a very simple structure, already competes with the WER of the duo-binary turbo code. The BER of the turbo code, however, is better that of the family-A code. This is explained by the fact that when the decoding process for the family-A code fails, there is a quite large number of erroneous bits (of about 10%).

Family-A codes seem to be quite effective for rates around 1/3. The degree distribution was optimised to improve iterative decoding performances (i.e. to lower the asymptotic

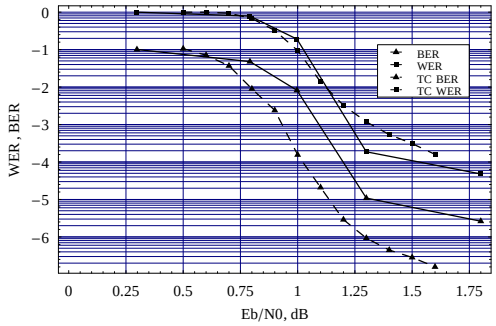


Figure 3.11: Performance of a family-A code of length 16000 of rate 1/2 on Gaussian channel, $\Lambda(x) = x$, vs. duo-binary turbo code of the same length and rate.

noise threshold) for a given rate. By curve fitting techniques (see for instance [69]) we found the degree distribution $\Lambda(x) = 0.7x + 0.3x^{11}$ which gave codes of rate 1/3 with the asymptotic noise threshold being slightly below 0 dB.

In Fig. 3.12 we compare a family-A code of rate 1/3 and with the degree distribution above decoded with 100 iterations with an 8-state Turbo code of length 15342 and of rate 1/3 decoded with 8 iterations from the 3GPP norm [21].

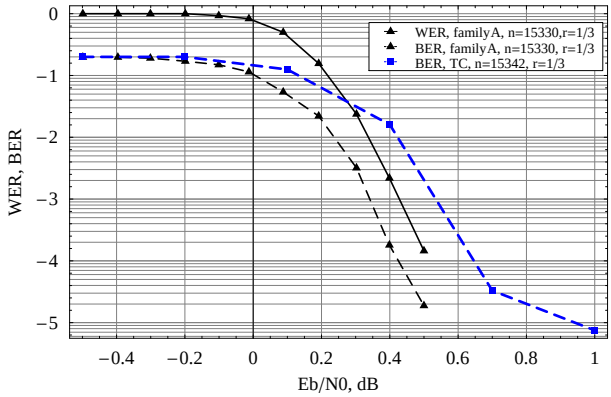


Figure 3.12: Family-A code of length 15330 and of rate 1/3 with degree distribution $\Lambda(x) = 0.7x + 0.3x^{11}$ compared with an 8-state Turbo code of length 15342 and of rate 1/3.

Notice that there is a 0.5 dB improvement in the error-floor region of the turbo-code at a bit error rate of 10^{-5} and that even the packet error rate of our code is below the bit-error rate of the turbo-code.

3.8.2 Family B

Trying to optimise the degree distribution of the previous family to obtain codes of rate 1/2 is not a good idea as there will always be a significant gap between the EXIT chart of the base code and the EXIT chart of variable nodes for large values of the average

intrinsic entropy (say in the range 0.4-0.5). As a consequence, the resulting codes will have thresholds rather far away from capacity. However, this behaviour can be significantly improved by changing the base code. For instance, we reduce the aforementioned gap by taking the base code of family (B).

For the family-B code $C_0 = \{000, 011, 101, 110\}$ and $C_1 = \{001, 010, 100, 111\}$. So, the base code is of rate 5/6.

The numbers of codewords of weight 2 or 3 divided by the length of the base code are given by

$$\begin{aligned} \frac{m_2}{n_b} &= 1 \\ \frac{m_3}{n_b} &= 4.5. \end{aligned}$$

EXIT chart analysis

The analytical expression for the entropy curve of the family-B base code on the BEC is similar to the one for family A and also can be found in Appendix B.2.1. The entropy curves of the base codes for both families, A and B, are shown in Fig.3.13. Comparing them, we can see that the curve for family B is less convex at the origin that the curve for family A, which is due to the fact that the number of codewords of weight 2 in the family-B base code is two times larger that for family A. This implies a smaller gap to capacity for family B than for family A. As it will be shown in the next part about the permutation

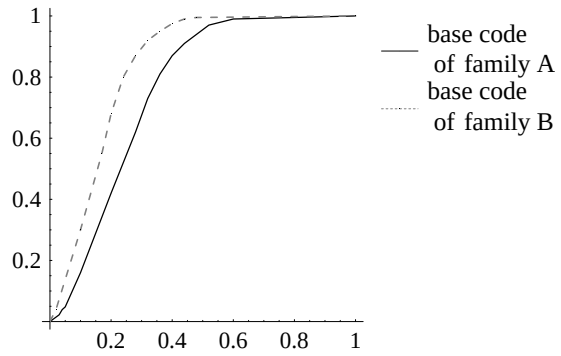


Figure 3.13: Comparison between entropy curves of the base code for families A and B.

choice, for family B we cannot choose $\lambda_2 = 1$ as the typical minimum distance is at most logarithmic in this case. However, we still can have a large fraction of degree-2 variable nodes, up to 2/3, still being asymptotically good. For example, by optimising the degree distribution of the code family B for the BEC we obtain the following degree distribution

$$\Lambda(x) = 0.6141700231x + 0.02989470758x^7 + 0.07007531202x^8 + 0.2858599574x^{19}.$$

The obtained code ensemble has rate 1/2 and corrects 48.3% erasures with probability $1 - o(1)$.

For the Gaussian channel, we choose $\lambda_2 = 1/2$ by following reasons:

- the ratio of 1/2 is still close to the limit of 2/3,

- in this case it is easy to find structured permutations so that the graph of codewords of weight 2 does not have cycles.

By optimising the degree distribution for the Gaussian channel, we obtain

$$\Lambda(x) = 0.5x + 0.182x^2 + 0.069x^{12} + 0.249x^{13}.$$

The threshold predicted by entropy curves is about 0.45 dB.

Choice of permutation

For the family-B base code, similarly with the previous code family, all bits associated with the same trellis section belong to the same cluster. So, clusters of the graph of codewords of weight 2 contain 3 bits in this case. Thus, the average degree of the graph of codewords of weight 2 is 3, and by Theorem 10, it is impossible to avoid sublinear-size cycles when $\lambda_2 > 2/3$; we construct family-B TLDPC codes with the fraction of degree-2 variable nodes $\lambda_2 \leq 1/2$.

In this case, there are several different ways to put some structure on the bipartite graph in order to have a graph of codewords of weight 2 without cycles of sublinear size. Probably the simplest way would be to choose the graph of codewords of weight 2 to be a union of disjoint paths. In this case, the same proof as in the previous case applies to show that the new code family contains almost only asymptotically good codes. However, it turns out that the prediction of the threshold given by the EXIT charts is underestimated by a bit more than 0.1dB. This might be due to the fact that that the EXIT chart implicitly assumes that we choose positions to be of degree 2 independently of each other with probability λ_2. This would imply that the expected number of vertices of degree 3 in the graph of codewords of weight 2 would be λ_2^3. However with the previous choice there are no vertices of degree 3.

This fact motivated us to choose the positions of bits of degree 2 in the base code in a more sophisticated way: the distribution of sections of the base code having t bits of degree 2, $t = 0, \ldots, 3$, is chosen to be the same as if the positions of degree 2 were chosen independently at random with probability λ_2, i.e. the 1/8-th part of all the sections does not contain bits of degree 2, the 3/8-th part contains one bits of degree 2, the 3/8-th part contains two bits of degree 2 and finally the 1/8-th part contains three bits of degree 2.

Let n_s the number of sections in the base code and let n_s be divisible by 8; the procedure of the choice of an permutation is the following one:

- random choice of $3n_s/8$ sections and generation of their cyclic permutation;

- uniform choice of $3n_s/8$ degree-2 variable nodes and their association to cycle edges: for the edge connecting a section i to a section j there is a corresponding degree-2 variable node in the bipartite graph of the code which connects these two sections of the base code.

- choice of $n_s/8$ remaining sections and matching each of them to three sections from the rest: if we match the section i with sections j, k, l, then three bits i_1, i_2 and i_3 of the section i will be connected with one bit in each of three given sections j, k, l.

Associating a degree-2 variable node with every edge, we define all the edges of left degree 2 in G.

- given the structure of the bipartite graph for the degree-2 variable nodes, the choice of connections for variable nodes of higher degrees is made at random among the left positions in the sections of the base code.

We note that the structure of the graph of codewords of weight 2 is a cycle of length $3n_s/8$ and the union of $n_s/8$ 3-stars. In this case the proportion of vertices of degree $0, 1, 2$ and 3 in the graph of codewords of weight 2 is $1/8$, $3/8$, $3/8$ and $1/8$ respectively. These figures are exactly the expected proportions we would have obtained if the positions of the base code were chosen to be of degree 2 with probability $1/2$. The predictions of the threshold given by the EXIT chart seem to be accurate in this case.

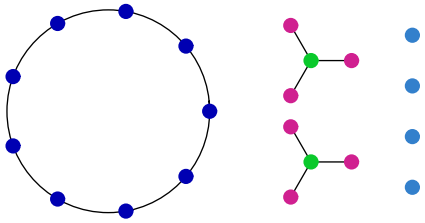


Figure 3.14: graph of codewords of weight 2 of the family-B code.

It is easy to check that the average degree of the graph of codewords of weight 2 is equal to $3/2$.

Simulation results

Fig.3.15 shows a family-B code of length 16000 and of rate $1/2$ performing 0.6 dB better at WER of $2 \cdot 10^{-4}$ than an 8-state duo-binary Turbo code of the same length and rate [1]. Note that for our family-B code, the maximum number of iterations is 100, but depending on the signal to noise ratio the average number of decoding iterations necessary to decode successfully is in the range $20 - 40$ only.

In Fig.3.16 performances of family-B codes of lengths 1000, 2000, 4000, 8000 and 12000 and of rate $1/2$ are presented.

In Fig.3.17 we present WER performances of family B codes of length 1000 and 2000 and of rate $1/2$ having the degree distribution $\Lambda(x) = 0.5x + 0.182x^2 + 0.069x^{12} + 0.249x^{13}$ and WER (floating-point) performances of the multi-edge LDPC code of length 1280 and of rate $1/2$ taken from Fig.1 in [59]. The interpolation of performances for the family B code of length 1280 is represented by dotted line.

It can be seen from the figure that if we simulate the family-B code of length 1280, the performance loss in comparison to the multi-edge code would be about 0.15 dB. We also observe that family-B codes do not reveal the error-floor region till $WER = 3 \cdot 10^{-5}$, in contrast to the multi-edge code. This is due to the fact that the multi-edge code is not asymptotically good. It can be easily shown that its corresponding graph of codewords of partial weight 2 is of degree 2 and the first degree being greater than 2 is 3, thus, by in [73], its expected minimum distance is at most of order $\Theta(n^{1/3})$.

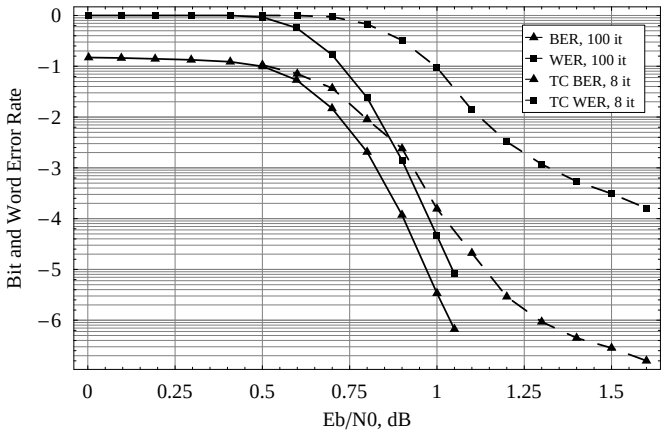


Figure 3.15: Family-B code of length 16000 and of rate 1/2 with degree distribution $\Lambda(x) = 0.5x + 0.182x^2 + 0.069x^{12} + 0.249x^{13}$ compared with a duo-binary Turbo code of the same length and rate.

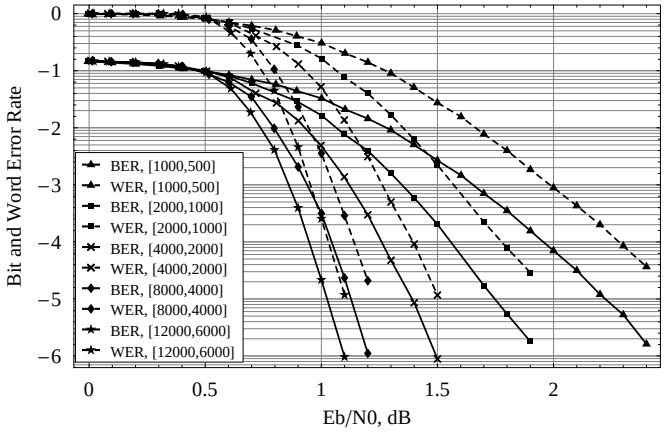


Figure 3.16: Family-B codes of of rate 1/2 with degree distribution $\Lambda(x) = 0.5x + 0.182x^2 + 0.069x^{12} + 0.249x^{13}$ of length 1000, 2000, 4000, 8000 and 16000 correspondingly.

In Fig.3.18 we compare performances of family-B codes of rate 1/2 and of lengths 8000 and 12000 performances (hardware simulations) of two multi-edge LDPC codes of length 10240 and of rate 1/2 taken from Fig.1 and 2 in [59]. The interpolation of performances for the family B code of length 10000 is represented by dotted line. The first multi-edge code is of the same family as the previous one from Fig.3.17, and it is not asymptotically good. It can be easily verified that necessary conditions for being asymptotically good hold for the second multi-edge code. By doing interpolation of the performances for the family-B code of length 10000, we obtain that its WER is about 0.1 dB worse than the WER of the first multi-edge code and is 0.1 dB better than the WER of the second one. Family-B codes are conjectured to be asymptotically good.

In our simulations we performed a maximum of 100 decoding iterations. To give an

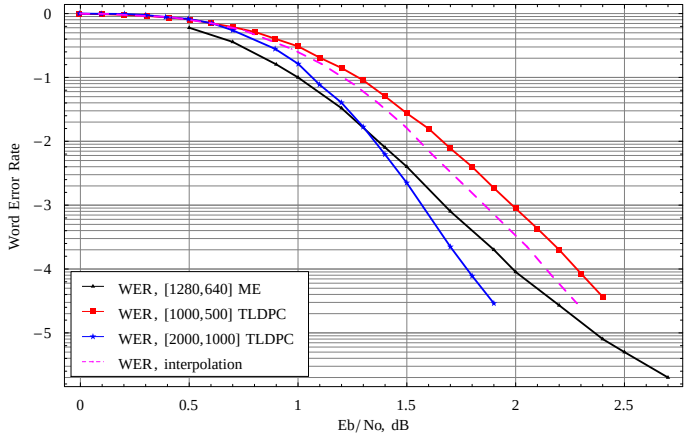


Figure 3.17: WER vs. SNR for family-B codes of of rate 1/2 with degree distribution $\Lambda(x) = 0.5x + 0.182x^2 + 0.069x^{12} + 0.249x^{13}$ of lengths 1000 and 2000 compared with the WER of the multi-edge LDPC code of length 1280 and of rate 1/2 from Fig.1 in [59].

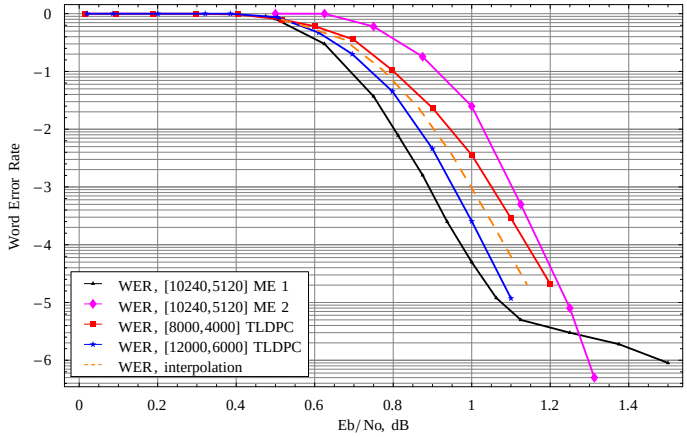


Figure 3.18: WER vs. SNR for family-B codes of of rate 1/2 with degree distribution $\Lambda(x) = 0.5x + 0.182x^2 + 0.069x^{12} + 0.249x^{13}$ of lengths 8000 and 12000 compared with the WER (hardware simulations) of multi-edge LDPC codes of length 10240 and of rate 1/2 from Fig.1 and 2 in [59].

idea of the number of iterations needed for decoding of TLDPC codes, for the family-B code of length 1000 we plot the entropy of extrinsic probabilities as a function of the number of iterations (see Fig.3.19) for 1.3 dB corresponding to $WER = 10^{-2}$ and for 1.8 dB corresponding to $WER = 5 \cdot 10^{-3}$. Note that for the beginning of the waterfall region (1.3 dB) 96 % of the test frames needed only 22 iterations to be decoded and the number of required iterations decreases with the SNR increase (13 iterations for 1.8 dB).

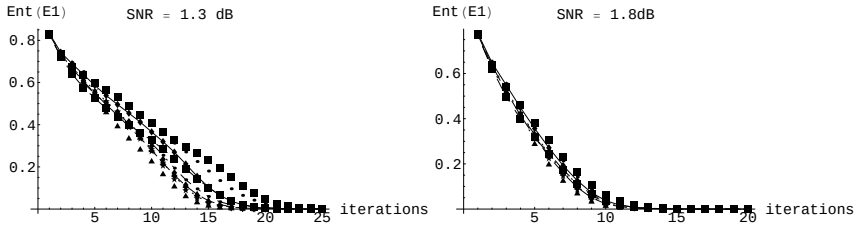


Figure 3.19: Convergence of a 2-regular family-B code of length 1000 and of rate 1/4 on Gaussian channels with SNRs 1.3 dB and 1.8 dB, 10 test examples.

3.8.3 Introduction to TLDPC code families with bits of degree 1

It was underlined in Section 3.6.2 that inclusion of bits of degree 1 in the code structure gives a faster convergence under iterative decoding and might lead to improved performances in the waterfall region. Let us present an example of the influence of bits of degree 1 on the waterfall region.

Example

We compare entropy curves on the BEC for the family of (2,3) LDPC codes and of a TLDPC code family with bits of degree 1 where the TLDPC code family is defined as follows:

- the base code C_b is presented by a tail-biting trellis in Fig.3.5 for which $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$ [7],
- $\tilde{\Lambda}(x) = x$ and $\lambda_1 = 1/2$,
- bits of degree 1 are put on the parallel sections of the trellis of the base code.

It is easy to verify that the rate of the TLDPC base code is 2/3 and the rate of the defined TLDPC code family is 1/3. It can be also checked that thresholds of (2,3) LDPC code family (which is of the same rate) and the defined TLDPC code family coincide and are equal to 1/2. This comes from the fact that, by Theorems 6 and 9, the first derivatives at origin for their entropy curves of the base code are equal.

In Fig.3.20 we present entropy curves of these two code families. For details of computation of the entropy curves see Section 5.3.5. We see that for $p = 1/2$ the entropy curve of the TLDPC base code containing bits of degree 1 is below of the entropy curve of the base code of the LDPC code ensemble which implies that the TLDPC code family will have a faster convergence under iterative decoding algorithm. Moreover, the entropy curve of the TLDPC base code depends on the channel erasure probability p and moves away from the entropy curve of the variable nodes when p decreases (see Section 5.3.5, Fig.5.5). This fact determines a better performances of the TLDPC code family in the waterfall region. $\diamond$

[7] see also Section 5.3 on $((1,1))_1$-TLDPC codes

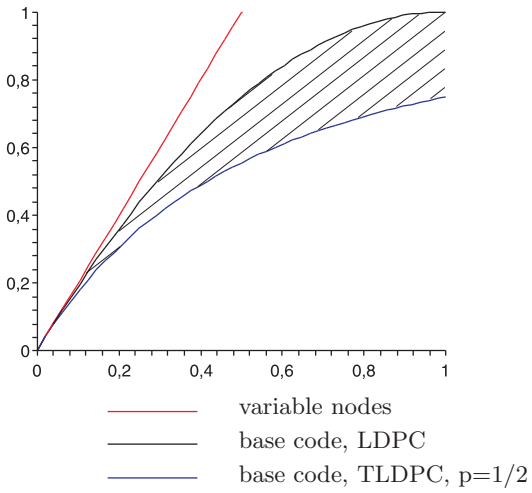


Figure 3.20: Entropy curves for base codes of the (2,3) LDPC code ensemble and of the TLDPC code ensemble with bits of degree 1 and their entropy curve of variable nodes for $p = 1/2$.

Thus, we further consider TLDPC code families by allowing bits of degree 1 in their architecture, as it has a positive impact both on the EXIT chart of the base code and on the performances under iterative decoding. At present we do not have a proof that the families with bits of degree 1 are asymptotically good [6], but we conjecture that this is still the case and simulations results did not show error floors behaviors for block error rates up to 10^{-6}.

The previous choices of the base code do not allow to have a constant fraction of degree 1 nodes as in this case the corresponding graph of codewords of partial weight 2 contains necessarily a cycle of fixed size and thus by Lemma 4 there will be a codeword of constant weight. However, if we allow certain sections of the tail-biting trellis to be of length one (i.e the C_0^i's associated to these sections are equal to $\{0\}$) then positions associated to these sections can be chosen to be of degree one without harming the minimum distance of the code.

This approach turns out to be particularly useful to lower the threshold of family A for which the area difference at origin between two entropy curves is quite significant. We replace some of the trellis sections by sections carrying a single bit (i.e. the associated code C_0^i is chosen to be $\{0\}$) and we put the positions of degree 1 precisely at these sections. This drastically changes the behavior of the entropy curve of the base code, the slope at the origin becomes steeper without changing too much the behavior of the curve elsewhere. It can even be proved that for the family A on the erasure channel, choosing one section out of 3 to be of unit length yields a concave entropy curve of the base code. Such a base code can give an efficient low-rate code as it is shown in Section 3.8.7.

In this part we present four code families, C, D, E and F, having degree-1 bits in their

[6] In principle it is possible to extend the results of [4] to this case, but the calculations are much more involved.

structure.

3.8.4 Family C

For the family C we have $\mathcal{C}_0^i = \{00, 11\}$ and $\mathcal{C}_1^i = \{01, 10\}$ but every forth section is of unit length, i.e. $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$ for $i \equiv 0 \pmod 4$. Thus, we have

$$\lambda_1 = \frac{m_1}{n_b} = \frac{1}{7}.$$

EXIT chart

For the BEC channel let us denote p the erasure probability of bits of degree 1 and x the erasure probability of bits of higher degrees. For fixed p, we plot the entropy curve as a function of x. Such an entropy curve of the family-C base code is presented in Fig.3.21 for $p = 0, 0.65, 1$. We also plot the entropy curve of the family-A code. By comparing curves of families A and C we can see that the slope at the origin of the family-C curve is much steeper and it changes more slowly in the range $x = 0.4 \ldots 0.6$ which can lower the maximum degree during the degree distribution optimisation. For

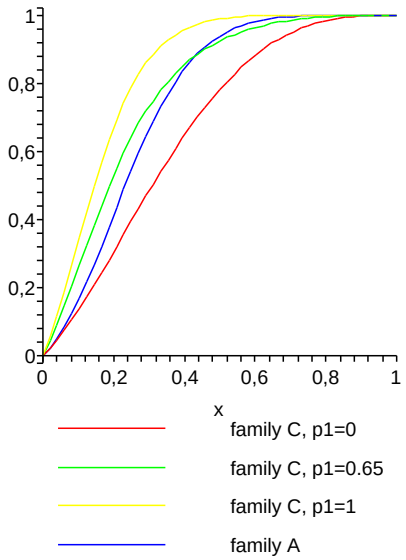


Figure 3.21: Entropy curve of the family-C base code plotted for different erasure probabilities p of degree-1 bits and compared with the entropy curve of the family-A code.

the Gaussian channel, we optimise the rest of the degree distribution so as to maximise the sustainable noise threshold for several rates $\frac{3}{10}, \frac{1}{3}, \frac{1}{4}$. For rate $\frac{1}{3}$ we obtained $\tilde{\Lambda}(x) = 0.5x + 0.0027x^2 + 0.082x^5 + 0.219x^6 + 0.0012x^{12} + 0.1865x^{13} + 0.0086x^{14}$. For rate $\frac{3}{10}$ we have the following degree distribution $\tilde{\Lambda}(x) = 0.5x + 0.0027x^2 + 0.082x^5 + 0.219x^6 + 0.0012x^{12} + 0.1865x^{13} + 0.0086x^{14}$ the threshold given by entropy curves is situated at -0.44 dB, whereas the channel capacity at rate 0.3 is around -0.617 dB. The degree distribution for rate $\frac{1}{4}$ is $\tilde{\Lambda}(x) = 0.4522x + 0.2967x^6 + 0.0103x^7 + 0.0219x^{21} + 0.2189x^{22}$.

For the rate 0.3 entropy curves seem to indicate that the gap to capacity in this case is about 0.18dB, which is quite good. However, for smaller rates the degree optimization yields even lower gaps to capacity at the cost of increasing the maximum degree. This has an impact on not only the complexity of iterative decoding (which increases) but also the performances of iterative decoding: now, when the decoding algorithm fails, this is generally due to a failure of reducing the extrinsic entropy in the first steps of iterative decoding and a large number of errors usually remains. It has a negative influence on the bit error probability and the slope in the waterfall error region is also decreased. It is possible to give a heuristic explanation of these negative phenomena by entropy curves considerations along the lines of [44]. Assume we seek a code of rate R with iterative threshold close to capacity. Assume also that the entropy curve of the base code is almost tangent to the horizontal line of equation $y = 1$ at the abscissa $1 - R$ (this is exactly what happens for the base code when we seek for rates smaller than 1/4). Then the degree optimization yields large degrees to allow the entropy curve of variables nodes to be almost horizontal at the abscissa $1-R$. A closer examination of both entropy curves for values of the signal to noise ratio of interest reveals that the bottleneck between them lies precisely in this region at abscissa around $1 - R$ where both curves are almost horizontal. Following [44] such a bottleneck is in general the point where iterative decoding fails (in our case it corresponds to the first steps of decoding). Roughly speaking a narrow bottleneck is related to a large probability that iterative decoding fails and this has a negative impact on the slope of the waterfall region.

Permutation choice

For family C there are two kind of clusters defining the graph of codewords of partial weight 2: the first one is composed by 4 bits in 2 sections around a section of length 1, and the other one - by 2 bits in sections not adjacent to any section of length 1. The number of clusters of maximum degree 2 and 4 is equal. Similarly to the case of family B, it can be checked that the average degree of the graph of codewords of partial weight 2 is of degree strictly larger than 2 when $\tilde{\lambda_2} > \frac{2}{3}$, so that there is necessarily a cycle of logarithmic size in the graph of codewords of partial weight 2 and thus the associated code is not asymptotically good. For the same considerations as for family B, we choose $\tilde{\lambda_2} = \frac{1}{2}$ and we select positions of bits of degree 2 so that for any degree $i \in \{0, 1, 2, 3, 4\}$, the number of vertices of degree i in the graph of codewords of partial weight 2 is equal to the expected number of degree i vertices in this graph when all the positions of degree 2 are chosen independently at random with probability $\frac{1}{2}$. For fixed $\tilde{\lambda}_2$ the average a_i of clusters of degree can be simply calculated as follows:

$$\begin{aligned} a_i &= \frac{1}{2}\binom{4}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{4-i} + \frac{1}{2}\binom{2}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{2-i}, \quad i = 0, 1, 2, \\ a_i &= \frac{1}{2}\binom{4}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{4-i}, \qquad\quad i = 3, 4. \end{aligned}$$

After performing such a random choice we match them together two by two in order to avoid any cycle in the graph of codewords of partial weight 2. Clearly, the way to perform the matching depends on fractions a_i. We choose the structure of the graph of codewords of partial weight 2 to be as it is shown in Fig.3.22.

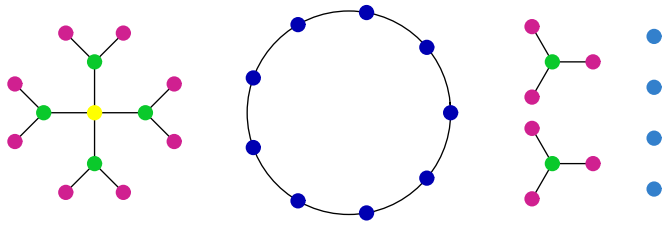


Figure 3.22: graph of codewords of partial weight 2 for the family C

Simulation results

In order to compare codes of family C with the ones obtained for the family A, we choose code lengths to be around 15000, see Fig.3.12. For rate $\frac{1}{3}$ we obtain slightly better performances than for family A (see Fig.3.23), but we will see in the next section that family-D and family-E codes give better results.

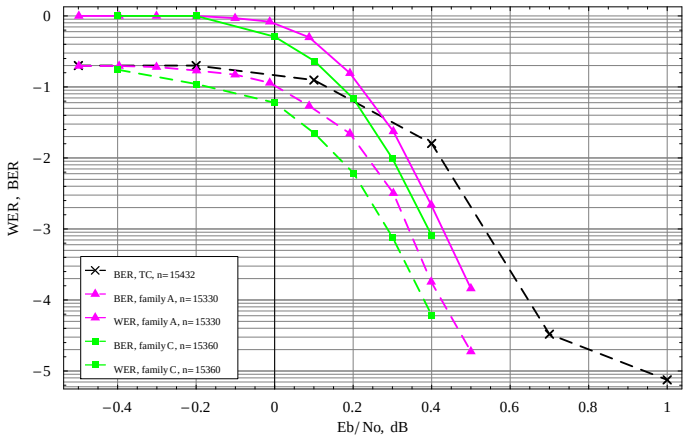


Figure 3.23: Family-C code of length 15000 and of rate 1/3 with $\tilde{\Lambda}(x) = 0.5x + 0.0027x^2 + 0.082x^5 + 0.219x^6 + 0.0012x^{12} + 0.1865x^{13} + 0.0086x^{14}$ compared with the family-A code of the same length and rate with $\Lambda(x) = 0.7x + 0.3x^{11}$ and the 8-state Turbo code of length 15432 and of rate 1/3.

Comparing the performances of family-C codes for rates 1/4, 0.3 and 1/3 (Fig.3.24) we can conclude that the family C displays quite good iterative decoding performances for rates around 3/10. Note that the code of rate 3/10 has a steeper waterfall region than the code of rate 1/4.

3.8.5 Family D

To circumvent the problem of bottleneck discussed for the code family C, we slightly change the base code to match the behavior of the entropy curve of the base code we are

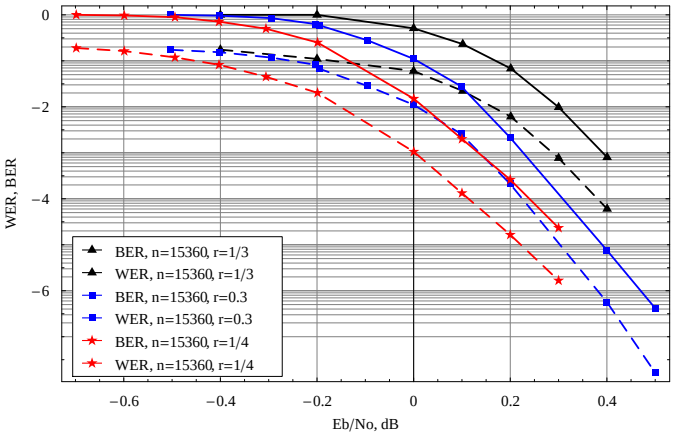


Figure 3.24: Performance of family-C codes of length 15360 and of rates 1/4,0.3 and 1/3 with following degree distributions: $\tilde{\Lambda}(x) = 0.4522x + 0.2967x^6 + 0.0103x^7 + 0.0219x^{21} + 0.2189x^{22}$ for $r = 1/4$, $\tilde{\Lambda}(x) = 0.5x + 0.0027x^2 + 0.082x^5 + 0.219x^6 + 0.0012x^{12} + 0.1865x^{13} + 0.0086x^{14}$ for $r = 0.3$ and $\tilde{\Lambda}(x) = 0.5x + 0.0027x^2 + 0.082x^5 + 0.219x^6 + 0.0012x^{12} + 0.1865x^{13} + 0.0086x^{14}$ for rate $r = 1/3$.

after. In other words, we look for an entropy curve of the base code which is reasonably close to the line $y = 1$ at the abscissa $1 - R$ with a large enough slope at that point. The two following code families, D and E, yield exactly such a behavior.

For the family D we have $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$ for $i \equiv 0 \pmod 4$, $\mathcal{C}_0^i = \{000, 011, 101, 110\}$, and $\mathcal{C}_1^i = \{001, 010, 100, 111\}$. for $i \equiv 6 \pmod 8$, $\mathcal{C}_0^i = \{00, 11\}$ and $\mathcal{C}_1^i = \{01, 10\}$ otherwise. The fraction of bits of degree-1 in this case is slightly lower than for family C:

$$\lambda_1 = \frac{2}{15}$$

EXIT chart optimisation

We choose $\tilde{\lambda}_2$ as the maximum possible value for $\tilde{\lambda}_2$ below the threshold $\tilde{\lambda}_2 \leq 8/13$ (which corresponds to the largest possible value when being asymptotically good) such that the gap to capacity is no more that 0.2 dB. Thus for family D for Gaussian channel we obtain $\tilde{\Lambda}(x) = 0.44x + 0.136x^2 + 0.424x^9$.

Permutation choice

Every local code of family D corresponds to a cluster of maximum degree 4 followed by a degree-3 cluster which in its turn is followed by one cluster of degree 4 and then by one cluster of degree 2. Thus, the graph of codewords of partial weight 2 consists of clusters of maximum degree 2,3 and 4 with fractions 1/4, 1/4 and 1/2 correspondingly. For family D, cycles of logarithmic size in their associated graph of codewords of partial weight 2 are unavoidable as soon as $\tilde{\lambda_2} > \frac{8}{13}$. Applying the approach that the number of vertices of degree i in the graph of codewords of partial weight 2 must be equal to the expected number of degree i vertices in this graph when all the positions of degree 2 are chosen

independently at random with probability $\frac{1}{2}$, it is simple to calculate necessary fractions of clusters of degree $0, 1, 2, 3, 4$:

$$\begin{aligned} a_i &= \frac{1}{2}\binom{4}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{4-i} + \frac{1}{4}\binom{3}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{3-i} + \frac{1}{4}\binom{2}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{2-i}, \quad i = 0, 1, 2, \\ a_3 &= \frac{1}{2}\binom{4}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{4-i} + \frac{1}{4}\binom{3}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{3-i}, \\ a_4 &= \frac{1}{2}\binom{4}{i}(\tilde{\lambda}_2)^i(1-\tilde{\lambda}_2)^{4-i}. \end{aligned}$$

Given a_i, a simple structure for the graph of codewords of partial weight 2 seems to be as it is presented in Fig.3.25.

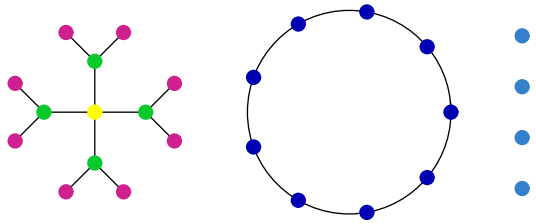


Figure 3.25: graph of codewords of partial weight 2 for the family D

Simulation results

In Fig.3.26 we compare performance of the family-D code having the degree distribution $\tilde{\Lambda}(x) = 0.44x + 0.136x^2 + 0.424x^9$ with the family-A code with $\Lambda(x) = 0.7x + 0.3x^{11}$. The family-D code performs 0.1 dB better and it does not have an error floor till $WER = 3 \cdot 10^{-6}$.

3.8.6 Family E

The code family E is another alternative to family C, their entropy curves have a quite large slope at the abscissa $1 - R$ which prevents decoding failures due to the proximity of entropy curves of the base code and of variable nodes at this point.

The trellis of the base code of family E can be described by C_0 and C_1 as follows: $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$ for $i \equiv 0 \pmod 5$, $\mathcal{C}_0^i = \{00, 11\}$ and $\mathcal{C}_1^i = \{01, 10\}$ otherwise. So, we have that

$$\lambda_1 = \frac{2}{18} = \frac{1}{9}.$$

EXIT chart analysis

To have a large fraction of degree-2 nodes, we choose $\tilde{\lambda_2}$ to be closely below the maximum value above which the graph of codewords of partial weight 2 will contain cycles of logarithmic size and so that the degree distribution optimization by using EXIT

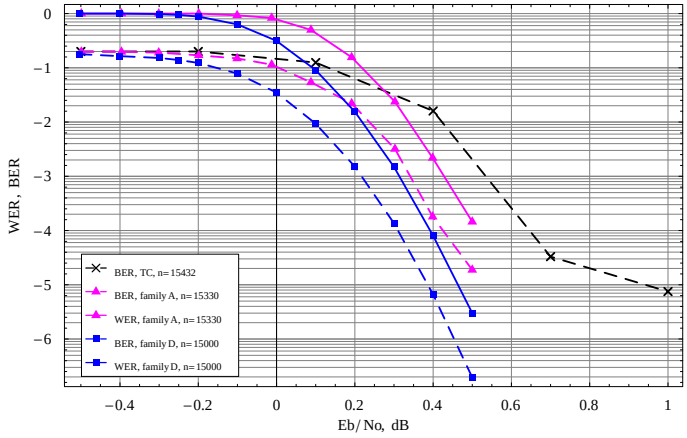


Figure 3.26: Family-D code of length 15000 and of rate 1/3 with degree distribution $\tilde{\Lambda}(x) = 0.44x + 0.136x^2 + 0.424x^9$ compared with an 8-state Turbo code of the length 15432 and of same rate and with the family-A code with $\Lambda(x) = 0.7x + 0.3x^{11}$ of the same length and rate.

charts yields gaps to capacity of about 0.2dB. For the family E such $\tilde{\lambda}_2$ is chosen to be 0.5. After the degree distribution optimisation for the Gaussian channel we obtain $\tilde{\Lambda}(x) = 0.5x + 0.181x^2 + 0.198x^8 + 0.121x^9$.

Entropy curves of the E base code and of variables nodes for -0.25 dB and 0.4 dB are shown in Fig.3.27. Once more, note that in the presence of bits of degree 1 the entropy curve of the base code depends on the channel noise and, when the channel noise decreases, the curve moves away from the entropy curve of variable nodes. This has a positive effect on performance under iterative decoding and the convergence.

Permutation choice

It is easy to check that the graph of codewords of partial weight 2 for family E contains 3/5 clusters of maximum degree 4 and 2/5 clusters of maximum degree 2. For family E cycles of logarithmic size are unavoidable if $\tilde{\lambda}_2 > \frac{5}{8}$. If the degree of clusters in the graph of codewords of partial weight 2 is chosen at random given $\tilde{\lambda}_2 = 1/2$, the average a_i of the proportion of degree i is the following:

$$a_0 = \frac{11}{80}; \quad a_1 = \frac{28}{80}; \quad a_2 = \frac{26}{80}; \quad a_3 = \frac{12}{80}; \quad a_4 = \frac{3}{80}.$$

Based on the proportions, we choose the structure of the graph of codewords of partial weight 2 as it is shown in Fig.3.28.

Simulation results

As it is shown in Fig.3.29, the family-E code of length 15000 and of rate 1/3 performs 0.15 dB better than the family-A code of the same length and rate. However, the point

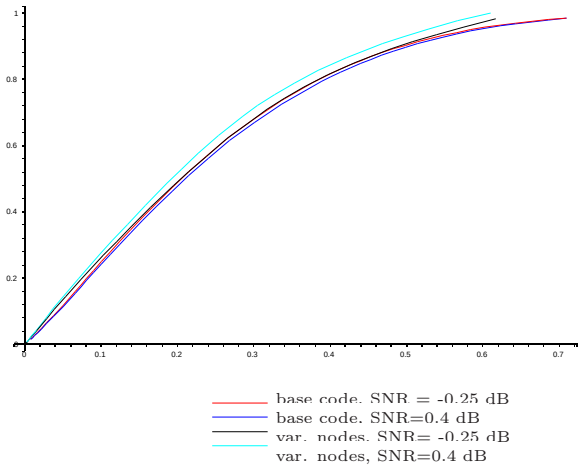


Figure 3.27: Entropy curve of the family-E base code and entropy curves of variables nodes with $\tilde{\Lambda}(x) = 0.5x + 0.181x^2 + 0.198x^8 + 0.121x^9$ for Gaussian channel with SNR -0.25 dB and 0.4 dB.

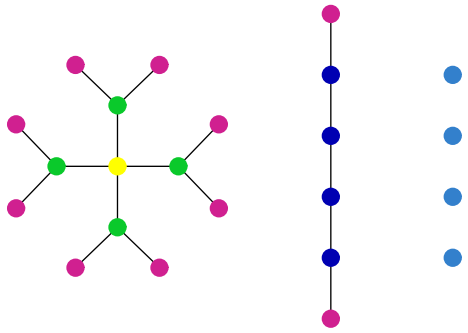


Figure 3.28: graph of codewords of partial weight 2 for the family E

at 0.5 dB reveals the beginning of the error floor region. The experiments show that similarly to the case of LDPC codes the error floor of the family-E code is due to special configurations of bits called trapping configurations which are not codewords. The study of trapping configurations is a problem isolated from the construction of codes by asymptotic tools, and a separated section is devoted to it.

In the following Fig.3.30 we present all the constructed codes of families A, C, D and E of rate 1/3.

3.8.7 Family F

It seems that to construct very low-rate codes with good performance based on graphs is a difficult task. To do it we found very helpful to have a large fraction of degree-1 bits in the code structure. Graph-based codes which are very efficient in middle and high rates such as RA and LDPC codes both suffer from the performance loss and the extremely slow convergence using iterative decoding at the low rate region. Standard turbo codes

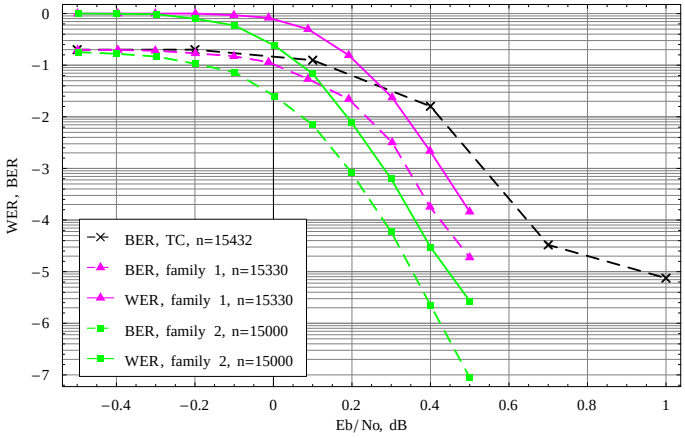


Figure 3.29: Family-E code of length 15000 and of rate 1/3 with degree distribution $\tilde{\Lambda}(x) = 0.5x + 0.181x^2 + 0.198x^8 + 0.121x^9$ compared with an 8-state Turbo code of the length 15432 and of same rate and with the family-A code with $\Lambda(x) = 0.7x + 0.3x^{11}$ of the same length and rate.

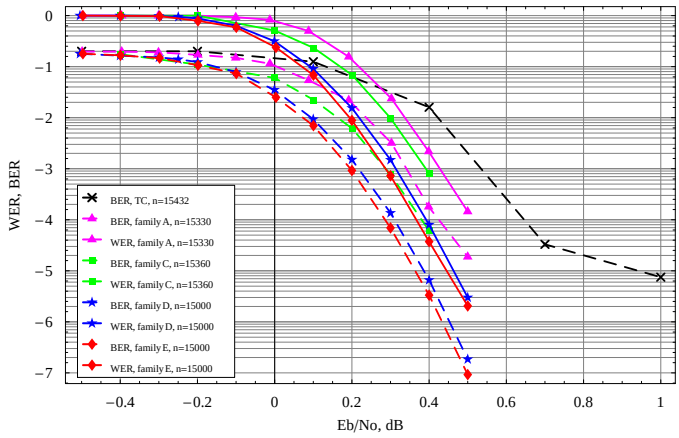


Figure 3.30: Performance of codes of families A, C, D and E of rate 1/3 and of length close to 15000.

are efficient around the rate 1/3 and they have a large fraction of degree-1 $\lambda_1 = 1/2$. One of the best low-rate code families proposed by now is Hadamard zigzag concatenated codes [45] whose construction is based on Hadamard arrays and which have a zigzag graph structure.

With the intention to construct codes of rates $\leq 1/10$, we study a TLDPC code family F the base code of which contains the maximum number of section containing bits of degree 1 under the condition for being asymptotically good. It has $\mathcal{C}_0^i = \{0\}$ and $\mathcal{C}_1^i = \{1\}$, and each third section of the family-F base code contains a bit of degree 1 while all the others contain bits of higher degrees. Thus, we have

$$\lambda_1 = \frac{1}{3}.$$

EXIT chart optimisation

By performing the optimisation for Gaussian channel we obtain the following degree distribution $\tilde{\Lambda}(x) = 0.4x + 0.264209x^2 + 0.090866x^4 + 0.236716x^8 + 0.008209x^9$.

Permutation choice

The family-F base code contains the maximum possible number of trellis sections containing bits of degree 1. This implies the graph of codewords of partial weight 2 were all the clusters have maximum degree 4. So, to construct an asymptotically good code family, the fraction $\tilde{\lambda}_2$ must not exceed 2/5. If the degree of clusters in the graph of codewords of partial weight 2 is chosen at random given $\tilde{\lambda}_2 = 0.4$, we have a relatively large fraction of clusters of degrees 3 in comparison with code families presented above:

$$a_0 = \frac{81}{625}; \quad a_1 = \frac{216}{625}; \quad a_2 = \frac{216}{625}; \quad a_3 = \frac{96}{625}; \quad a_4 = \frac{16}{625}.$$

To find a structure of the graph of codewords of partial weight 2 without cycles becomes a harder task but is still feasible. Its structure in this case becomes more involved and contains large "stars" formed by clusters of degree 1,3 and 4, "chains" formed by clusters of degree 1 and 2 as well as clusters of degree 0.

Simulation results

In Fig.3.31 we present simulation results on Gaussian channel for family-F codes of rates 1/10 and of different lengths, for every of which the degree distribution obtained by optimisation was adapted to the given length.

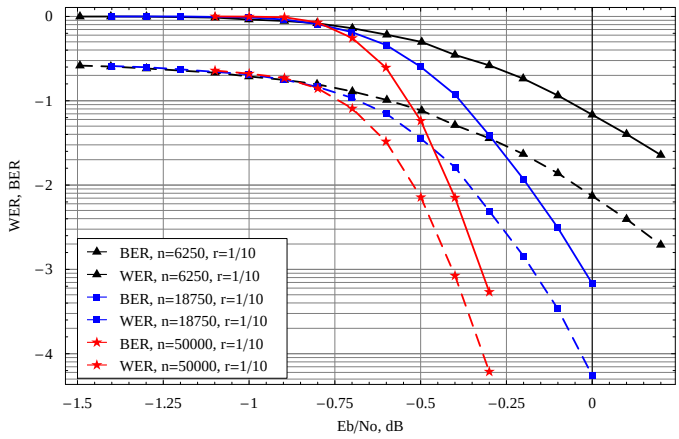


Figure 3.31: Performance of family-F codes of rate 1/10 and of lengths 6250, 18750 and 50000 with following degree distributions: $\tilde{\Lambda}(x) = 0.4x + 0.2644x^2 + 0.09x^4 + 0.2376x^8 + 0.008x^9$ for $n = 6250$, $\tilde{\Lambda}(x) = 0.4x + 0.2646x^2 + 0.0902x^4 + 0.2364x^8 + 0.008x^9$ for $n = 18750$ and $\tilde{\Lambda}(x) = 0.4x + 0.2642x^2 + 0.0905x^4 + 0.2365x^8 + 0.008x^9$ for $n = 50000$.

Chapter 4

Trapping sets of TLDPC codes

4.1 Introduction: trapping sets of LDPC codes

Recent simulations of LDPC codes on hardware platforms have shown that LDPC codes, even those with good distance properties such as $(3, 6)$ Gallager codes, do exhibit error-floors. However, the error-floor does not typically arise from low-weight codewords.

It is well known that for LDPC codes, iterative decoding techniques used on the BEC fail if the errors occur on subsets of variable nodes forming a *stopping set* [28], while for the AWGN channel a similar phenomenon can be observed with respect to *trapping sets* [56], sometimes referred as to *pseudocodewords* [8] or *near-codewords* [51].

The influence of trapping sets on the onset of error-floors in LDPC codes is attributed to the following phenomenon, related to the properties of the code graph, decoding algorithm and the probability of realization of certain special channel-noise configurations. In the initial stage of belief-propagation, due to abnormal large values of noise on a certain trapping set, variable nodes internal to it experience a large increase in the reliability estimates for the incorrect bit value. This information gets propagated to other variable nodes in the trapping set, some of which already have very low-reliability channel values. After the initial increase in reliability, external variables usually start adjusting the incorrect estimates towards the correct bit values. But by that time, the variable nodes in a trapping set may have already significantly biased their decisions towards the incorrect values. Since there are very few unsatisfied check nodes with odd degrees in the case of LDPC codes which are capable to detect error within the trapping set, the unreliable information remains unchanged until the end of decoding process.

More formally, let us bring in following definitions and notations of trapping sets for LDPC codes. First we define general trapping sets which englobe all possible configurations of trapping sets for a given code. However, it was shown by simulations that harmful trapping sets are those with small a and very small b (typically b is equal to 1 or 2). We define them to be elementary trapping sets.

Definition 17 *A general (a, b) trapping set is a configuration of a variable nodes in V for which the induced subgraph in G contains $b > 0$ odd-degree check nodes.*

Definition 18 *An elementary (a, b) trapping set is a trapping set for which all check nodes in the induced subgraph have either degree one or two, and there are exactly b degree-one check nodes.*

It was shown in [52] that for (3,5) and (3,6) regular LDPC codes the exponential growth of average number of elementary trapping sets is very close to the exponential growth of average number of general trapping sets. In other words, this gives some theoretical evidence that trapping sets with higher degrees of check nodes (and thus having a more complicated configuration) are possible but very unlikely for mentioned code ensembles.

It was also noticed in [56] that small trapping sets contribute more to the appearance of the error-floor than large ones. This paper also suggests an importance sampling methodology to predict error-floors of LDPC codes. We describe it by the example of the AWGN channel with noise variance σ^2:

1. first, trapping sets in the given code have to be detected;

2. an abnormal large noise is applied to a trapping set $\mathcal{T}$ (that is Gaussian noise with variance $\sigma'^2 > \sigma^2$) and "normal" Gaussian noise of variance σ^2 is applied elsewhere, iterative decoding is performed and the probability of decoding failure $P(\mathcal{T}, \sigma', \sigma)$ is estimated;

3. let $Q(\mathcal{T}, \sigma', \sigma)$ be the probability that the empirical variance of the noise on the trapping set is indeed σ'^2 when the bits are subject to an AWGN channel with noise variance σ^2. The value $\sigma_{\mathcal{T}}^2$ of σ'^2 which maximizes the product $P(\mathcal{T}, \sigma', \sigma)Q(\mathcal{T}, \sigma', \sigma)$ is computed and the error floor is then estimated by the sum $\sum_{\mathcal{T}} P(\mathcal{T}, \sigma_{\mathcal{T}}, \sigma)Q(\mathcal{T}, \sigma_{\mathcal{T}}, \sigma)$.

For the sake of complexity issues it was also proposed in [23] to put a certain kind of deterministic noise on a trapping set and thus to estimate its contribution to the error-floor.

The phase of trapping set detection can be accomplished using one of the following approaches:

- search by iterative decoding: we simply run decoding simulations in the error-floor region to look for trapping sets. This approach may not lead to the identification of all possible trapping sets but it does not need to know the combinatorial structure of trapping sets and thus can be performed directly.

- focus on detection of elementary trapping sets of small size: knowing their structure, we perform a search of such configurations in the Tanner graph of the code which could potentially give us the elementary trapping set. This approach is intended to find all potential elementary trapping sets up to some maximum size.

During decoding simulations for TLDPC code families D and E we observed that the error-floor starts at word error-rates 10^{-6}. These two code families are conjectured to be asymptotically good and thus the error-floor is unlikely to be caused by low-weight codewords. Studying incorrect bits of decoding failures, we found out that they belong to special configurations of the Tanner graph which prevent the iterative decoder from converging to a valid codeword.

We call these configurations trapping sets of TLDPC codes.

We begin with describing properties of decoding failures obtained by simulations of a family-E code.

4.2 Experimental results for a family-E code

In order to reveal the trapping sets structure and to collect different statistics, we run simulations for a family-E code of length 15000 and of rate 1/3 at 0.55 dB which corresponds to its error-floor region. Because of the code linearity the all-zero codeword is always transmitted. The maximal number of iterations is taken to be 200.

The data were obtained in the following way. If after 22 decoding iterations the frame still contains errors (the value of iteration number corresponds to the average number of performed iterations for the family-E code at 0.55 dB), then we begin to do the following:

- trace the number of erroneous bits at every iteration,
- for every bit compute the number of iterations when it was erroneous,
- if numbers of erroneous bits for the next 2 iterations are larger than the present one, the present configuration of erroneous bits is saved.

If the frame still contains errors after 200 iterations, it is declared to be erroneous and we output the saved configurations, the list of numbers of erroneous bits per iteration and the list of bits which were erroneous most frequently altogether with their grades (number of iterations during which bits were estimated incorrectly). The minimal number of incorrect iterations to be taken into account is fixed to be 60.

The results given below were obtained for 77 million frames, 93 frames being in error under belief propagation decoding.

4.2.1 Properties of decoding failures

Decoding behaviour

First we study the behaviour of decoding failures during decoding iterations. Plotting the number of erroneous bits with respect to the iteration number, three principal behaviours were determined (see Fig. 4.2.1): random-like oscillations show the case when a trapping set is covered by an error configuration received from the channel (we say that we got a trapping frame), the smooth lower curve represents the decoder convergence to a low-weight codeword and the smooth upper curve corresponds to a decoding failure when the decoder did not converge. 75 erroneous frames from 93 behaved accordingly to the first example, 5 - to the second one and 13 - to the third one.

Remark that in the case of trapping frame the number of erroneous bits at the end of decoding can be quite large as it oscillates continuously.

Variable nodes and assignments of the base code of observed trapping sets

It was observed that trapping frames have a large fraction of degree-2 variable nodes and a quite small number of variable nodes of higher degrees: observed trapping frames contained in average 91.95% of degree-2 variable nodes, 8.02 % of degree 3 variable nodes and 1.03 % of variable nodes of higher degrees. If we look at grades of most frequently erroneous bits we see that variable nodes of degree 2 have in average higher grades than other variable nodes. This is due to the fact that they are less protected and, having been trapped once, they have more difficulty to turn their estimations to correct bit values.

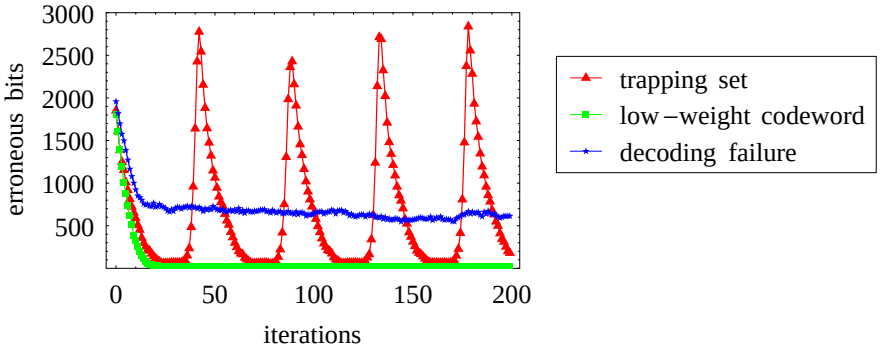


Figure 4.1: Examples of decoding behaviour.

Let us describe trapping frames from the base code side, namely we are interested in the location of erroneous positions in the base code. To do this, for a trapping frame take the list of most frequently erroneous bits, assign them to 1 and others to 0 (as we work under all-zero codeword assumption, such an assignment will cover the trapping set of the given trapping frame). We noticed that in most of cases the induced assignments of bits of the base code are not codewords.

Assignments of the base code can be decomposed into smaller sets of bits, each of them inducing either a minimal codeword (this means that its support does not contain a support of another nonzero codeword) or a word being close to a minimal codeword of the base code. We are particularly interested in codewords and near-codewords of the base code of partial weight 2 and 3. By near-codewords of partial weight 2 and 3 we understand words with such a support in the base code that there is a position of degree greater than 1 in the base code that if we put it to 1 and add to a given support, we will get a support of a codeword of the base code of partial weight 2 or 3 respectively.

In observed trapping frames there were in average 53.93 % of near-codewords of partial weight 2, 30.55 % of codewords of partial weight 2, 7.36 % of near-codewords of partial weight 3, 6.4 % of codewords of partial weight 3 and 1.76 % of others. Of course, the percentage depends on how we output grades of bits after erroneous decodings, but still we can grade that they contain a large fraction of codewords and near-codewords of the base code of partial weight 2 and 3.

By using the list of grades of the most frequently erroneous bits we are able to calculate what type of codewords of near-codewords of weight 2 and 3 is trapped more often. To do this, we pass grades of erroneous bits to codewords and near-codewords in the base code connected to corresponding variable nodes, we take an average on all grades received by a codeword or a near-codeword and thus we obtain a grade for each of them. This grade is an estimation of the number of iterations during which the codeword or near-codeword was trapped. Having obtained such grades, we compute the mean on grades for every type of given codewords and near-codewords. Obtained results are presented in Table 4.2.1 and witness that from the base code view codewords of small weight, once they have been trapped, are harder to reestimate correctly than near-codewords of small weight.

To conclude, observed trapping sets contain a large fraction of degree-2 variable nodes and the assignments of bits of the base code, induced by trapping sets form codewords or

type	mean number of iterations
near-codewords of weight 2	77
codewords of weight 2	104
near-codewords of weight 3	84
codewords of weight 3	114

Table 4.1: Mean number of iterations during which a codeword or near-codeword in the base code of weight 2 or 3 is trapped.

near-codewords of the base code and have a very small weight (mostly 2 and 3).

Clusters in observed trapping sets

Now let us study codewords and near-codewords of partial weight 2 and 3 by focusing on statistical properties of clusters. Clusters[1] were defined in Section 3.7.1.

Definition 19 *For a given assignment of base code bits, the degree of a cluster is the number of its positions equal to 1.*

Remark family-E codes contain 40 % of clusters of possible maximum degrees 2 and that 60 % of clusters of possible maximum degree 4.

It is possible to make a correspondence between clusters and codewords and near-codewords of weight 2 and 3 discussed above. Codewords of partial weight 2 will correspond to clusters of degree 2 and near-codewords of partial weight 2 will correspond to clusters of degree 1. Due to the trellis structure of the family-E base code, codewords of partial weight 3 correspond to three consecutive clusters of degree 1 having respective maximum degrees 4,2 and 4. Similarly, near-codewords of weight 3 correspond to three consecutive clusters of maximum degrees 4, 2 and 4, the two of which have degree 1 and one - degree 0.

As incorrect bits of trapping frames induce many near-codewords of weight 2 and codewords and near-codewords of weight 3 and they correspond to clusters of degree 1, we expect them to be the most numerous between all the clusters corresponding to induced codewords and near-codewords of the base code in trapping frames. And indeed, we obtain that in average there are 70.05 % of clusters of degree 1, 29.68 % clusters of degree 2 and others are clusters of higher degrees.

If we look at maximum possible degree of connected clusters, erroneous bits of trapping frames connect in average 76.5 % of clusters of maximum degree 4 and 23.5 % of clusters of degree 2. These proportions are different from proportions of clusters for family-E codes in favour of clusters of maximum degree 4, so we conclude that they appear more frequently in trapping set configurations than clusters of maximum degree 2.

These figures, put together with the fact that a large fraction of degree-2 variable nodes corresponds to incorrect bits, imply that the more frequent erroneous bits must correspond to the configuration of a variable node of degree 2 connected to two clusters of maximum degree 4.

[1]Recall that clusters are equivalent classes of positions in the base code of degree > 1 such that they form a support of a codeword of the base code of partial weight 2.

max.degrees of clusters	mean grade
2 and 2	96.93
2 and 4	97.64
4 and 4	97.41

Table 4.2: Mean grade of degree-2 variable nodes connected to two clusters of different maximum degrees.

Now we want to know if bits in such configurations are trapped more often than other bits corresponding to degree-2 variable nodes. To estimate this, we use grades of most erroneous bits and we compute mean grades for bits corresponding to degree-2 variable nodes connected to clusters with different maximum degrees. Results presented in Table 4.2.1 show that all bits corresponding to degree-2 variable nodes have almost the same mean grade (recall it corresponds to the mean number of iterations during which a given bit was estimated incorrect) not depending on maximum degree of connected clusters.

4.3 Generalized definition of trapping sets

In this section we generalise the notion of trapping sets of LDPC codes to a general class of codes and in particular to TLDPC codes.

Let us consider the general class of codes including LDPC codes, turbo codes as well as TLDPC codes which was defined in Section 3.2. Then we can define a general trapping set as follows:

Definition 20 *A general (a, b) trapping set is a set of a variable nodes in V such that if all the positions in the base code connected to this set are put to 1 then we need to put to 1 b another positions in the base code in order to obtain a valid codeword of the base code.*

In particular case of LDPC codes this definition is equivalent to Definition 17, as the base code of LDPC codes is simply a juxtaposition of small parity-check codes.

In Fig.4.3 we give some examples of trapping sets.

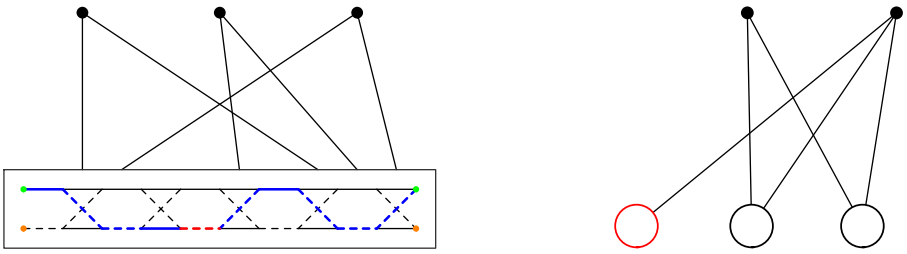


Figure 4.2: Examples of trapping sets.

In the figure on the left we show a code whose base code is represented by a two-state tail-biting trellis (straight lines correspond to “0” labels, dashed - to “1”). Dark circles represent variables nodes of the trapping set. These variable nodes put to 1 induce a word

(0110010111) which is not a codeword of the base code. But if we put the fifth position to 1 we obtain (0110110111) which is a codeword of the base code. So, for the given trapping set $a = 3$ and $b = 1$.

The figure on the right represents a subgraph of an LDPC code induced by variables nodes of a trapping set put to 1. One check node in the subgraph receives only one "1" which is not a codeword of a parity-check code. To obtain a codeword, we need to put another bit of the parity-check code to 1. Thus, we have a (2,1) trapping set.

The definition of elementary trapping sets, however, will depend on properties of the particular underlying code. In the case of TLDPC codes it is a convolutional code such that the labels of its branches for one trellis section belong either to the parity-check code C_0 either to its coset C_1 whose parity is equal to 1.

Let us define generalized elementary trapping sets. For doing this we first define a certain graph derived from degree-2 variable nodes, clusters defined in Section 3.7.1 and codewords of the base code. We begin by the simple case with no codewords of the base code of partial weight higher than 2 and we describe what elementary trapping sets look like in this case:

Definition 21 *The graph of type 2 is a bipartite graph G with V to be the set of variable nodes and W to be the set of clusters where a cluster c in W is connected to a variable node v in V if c contains a position of the base code connected to the given variable node in the Tanner graph of the code.*

Definition 22 *An elementary trapping set (a, b) of type 2 is a set of a variable nodes such that the induced graph only contains clusters connected once or twice and there are exactly b clusters connected once.*

It can be seen that for LDPC codes, clusters of the base code coincide with parity-check nodes in the bipartite graph. So, Definition 18 of elementary trapping sets coincides with Definition 22 and elementary trapping sets of LDPC codes are elementary trapping sets of type 2 in our notation.

As trapping sets of TLDPC codes display more complicated structures than those of LDPC codes involving codewords of the base code of higher weights in their structure, the given definition is insufficient to describe them. This motivates us to define an elementary trapping set of type i with the help of the graph of type i, $i > 2$:

Definition 23 *For $i \geq 3$, the graph of type i is a graph G_i with set V of variable nodes, set W_1 of clusters and set W_2 of nodes of codewords of partial weight j, $3 \leq j \leq i$. A node of codewords of weight j in W_2 (a cluster in W_1) is connected to a node in V if there is a position belonging to the support of the codeword of weight j in the base code (belonging to given cluster) connected to the corresponding variable node.*

Remark that for the Tanner graph of the code a variable node and a position of the base code determine only one edge in the graph. It is also true for the graph of codewords of partial weight 2 for which every edge is defined by a variable node and a position in the base code. For a graph of type i a node in V and a position in the base code determine not one edge but a set of edges. This comes from the fact that the same position in the base code belong to several supports of codewords of weight j, $2 \leq j \leq i$, in the base code.

For a node v in V and for a position k in the base code we denote $E_{v,k}$ to the set of edges determined by v and k.

Definition 24 *An elementary trapping set (a, b) of type i is a set of a nodes in V such that*

1. *for a node v and a position k in the base code one edge is only active in $E_{v,k}$,*
2. *the induced graph is the graph containing a nodes, active edges and clusters or codewords activated by these edges,*
3. *codewords of weight j in the graph, $3 \leq j \leq i$, are connected at least $(j-1)$ times.*

If the number of clusters connected once is b_1 and the number of codewords of weight j of degree $(j-1)$ is b_2, then $b = b_1 + b_2$.

4.4 Error-floor estimation

In this section, we describe how to estimate the error-floor of a TLDPC code. As an example we estimate the error-floor of the family-E code of length 15000 and of rate 1/3 described in Section 3.8.6.

Generally, if $\xi_{\mathcal{T}} \in \{0,1\}^n$ denotes the set of bits of the global code that gives rise to a failure on a trapping set $\mathcal{T}$, then the block error rate P_B of a code in the error-floor region can be evaluated as

$$P_B = \sum_{\mathcal{T}} \mathbf{P}\{\xi_{\mathcal{T}}\}.$$

To estimate the lower bound of the error-floor of the given code, we can use trapping sets obtained during simulations.

As we would like to have a general method to estimate trapping sets of all families of TLDPC codes, we restrict ourselves to the estimation of the error-floor by considering small trapping sets only up to some size k. In this case, we will obtain a lower bound on the block error rate:

$$P_B \geq \sum_{i=1}^{k} \mathbf{P}\{\xi_{\mathcal{T}_i}\}. \tag{4.1}$$

Knowing that small trapping sets are responsible for the beginning of the error-floor (and this is exactly our region of interest), this bound is supposed to be tight. Moreover, we also assume that, as for LDPC codes, most of the general trapping sets of TLDPC codes are elementary ones, that other, more complicated configurations are possible, but unlikely, and we only take into account elementary trapping sets.

So, we begin by searching small elementary trapping sets in the given code and, more exactly, by searching typical configurations of such trapping sets. If these configurations exist, we can find all these configurations in the Tanner graph of the code and then to look for elementary trapping sets between them.

4.4.1 Evaluation of $\mathbf{P}\{\xi_{\mathcal{T}}\}$ in AWGN case

Consider a trapping set $\mathcal{T}$ with K variable nodes.

Under zero-codeword assumption and BPSK modulation the channel output can be written as $y_i = \frac{2}{\sigma^2}(1 + n_i)$, $i = 1, \ldots, n$, n_i are independent identically distributed (i.i.d.)

random variables with distribution $\mathcal{N}(0, \sigma^2)$. Consider the subvector $(n_1, \ldots, n_k)$ of noise applied to $\mathcal{T}$ and suppose that each element of it is subject to noise s with distribution $\mathcal{N}(0, K^{-1}\sigma^2)$. Supposing that the failure rate is dependent on noise s applied to $\mathcal{T}$, we consider conditioning on s in order to evaluate it. Then $\mathbf{P}\{\xi_{\mathcal{T}}\}$ can be expressed as follows:

$$\mathbf{P}\{\xi_{\mathcal{T}}\} = \mathbf{E_s}(\mathbf{P}\{\xi_{\mathcal{T}|s}\}) = \frac{\sqrt{K}}{\sqrt{2\Pi}\sigma} \int_{-\infty}^{\infty} \mathbf{P}\{\xi_{\mathcal{T}|s=x}\} e^{-\frac{-x^2 K}{2\sigma^2}} dx. \tag{4.2}$$

In practice, we try to determine the function to be integrated in a neighborhood of its maximum and it is supposed to decay quickly with s. The factor $\mathbf{P}\{\xi_{\mathcal{T}|s=x}\}$ is estimated by simulations, we count only failures that occur precisely in $\mathcal{T}$. Typically, it is sufficient to simulate down to 10^{-3}, which does not take much time and thus makes the technique efficient.

4.5 Error-floor estimation of the family-E code over AWGN channel

To be able to use (4.1), we do the following:

- study of typical trapping set configurations,
- search of all typical configurations in the Tanner graph of the code,
- simulation of the failure rate for each of found configurations,
- evaluation of the error-floor by (4.2).

4.5.1 Trapping sets configurations

To select trapping frames corresponding to small trapping sets, let us consider their minimum number of incorrect bits occurred during iterations. As these small sets of bits cannot be decoded correctly, they must cover some trapping sets of small size.

During simulations of the given family E code we saved lists of erroneous bits for such iterations of the decoding that the number of erroneous bits increased during next two iterations. They are exactly small sets of bits in which we are interested in.

As we are interested in trapping sets of small size, from all saved lists we choose those whose size do not exceed 50, there are 50 of them.

Now, for every of such lists we construct the graph of type 3,4 or 5, depending which kind of minimal codewords and near-codewords of the base code are induced by the given list, and we study their structure. Our aim is to find out what types of small configurations the constructed graphs consist of and if there exist such configurations that they are present in every studied trapping structure. We call them dominant configurations.

Examples of graphs of type 3 and 4 are presented in Appendix C.1.

Considering 50 graphs of type 3,4 or 5, three dominant configurations were found. Surprisingly, their structure is quite simple, all of them only consist of codewords of weight 2 and near-codewords of weight 3 connected through degree-2 variable nodes.

The first dominant configuration has $b = 3$ and is composed of three near-codewords of weight 3 connected through degree-2 variable nodes and clusters. It is present in 19

trapping structures from 50 studied. Let us denote this dominant configuration by $(a, 3)$. The second dominant configuration has $b = 2$, it is composed of two near-codewords of weight 3 connected through degree-2 variable nodes and clusters and it was found in 29 studied trapping structures. We denote the dominant configuration by $(a, 2)$. The third dominant configuration has $b = 1$, it is composed of only one near-codeword of weight 3 connected through degree-2 variable nodes and clusters and it was found in 3 studied trapping structures. We denote the dominant configuration by $(a, 1)$.

First two dominant configurations cover together 47 trapping structures from 50 and other three configurations belong to the third type.

We represent dominant configurations graphically in Fig.4.3. Near-codewords of the base code of weight 3 are depicted by circles. and they are connected by labelled edges. An edge with label k between two codewords exists if in the the corresponding graph the codewords are connected through exactly k variable nodes of degree 2 and $k - 1$ clusters. Remark that when clusters are connected by degree-2 variable nodes in the graph, they form a chain of clusters where all the positions belonging to inner clusters of the chain are satisfied. Red arrows represent unsatisfied positions of near-codewords of weight 3.

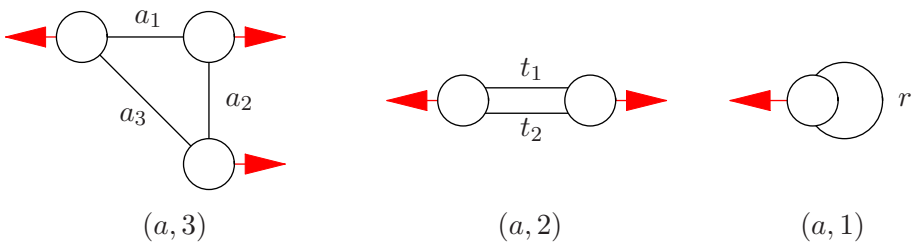


Figure 4.3: Dominant configurations with 1) $a = a_1 + a_2 + a_3 + 3$; 2) $a = t_1 + t_2 + 2$; 3) $a = r + 1$.

4.5.2 Detection of dominant configurations

To detect potential trapping sets in the given code, we make the search of three dominant configurations described above.

Let n_3 be the number of codewords of weight 3 (and thus, of possible near-codewords of weight 3) in the base code. Let us describe the proposed detection procedure of $(a, 3)$ configurations. We do it in a simplified way, by describing how to detect three near-codewords of weight 3 in the base code which make the base of an $(a, 3)$ configuration. Then, to obtain numbers of bits of global code corresponding to a given configuration we have to make the list of clusters trough which the near-codewords of weight 3 are connected. This can be done either by saving number of clusters in a list during the procedure of near-codewords detection either by performing an additional step during which the clusters are determined knowing what near-codewords of weight 3 they connect.

The procedure of detection of numbers of near-codewords of weight 3 is as follows:
Fix the depth of search l_{max}.

- Fill the distance (triangular) table D of dimensions $n_3 \times n_3$. An element $d_{i,j}$ is equal to the minimal length l of the chain connecting the given support of codeword of

weight 3 i with the support of codeword of weight 3 j if $l \leq l_{max}$ and ∞ otherwise. For fixed i let D_i be an ensemble of numbers such that $d_{i,j} \neq \infty$ for every $j \in D_i$.

- For i from 0 to $n_3 - 1$ do the following:
  - For every $j \in D_i$ look for k such that $k \notin D_i$ and $d_{i,k} \neq \infty$ as well as $d_{j,k} \neq \infty$
  - Save $\{i, j, k\}$ and $d_{i,j} + d_{i,k} + d_{j,k}$.

We detect $(a, 2)$ and $(a, 1)$ configurations with the following procedure:

- For a set of clusters $\{c_i^k\}_{k=0}^2$ corresponding to a codeword w_i of base code of weight 3 , $0 \leq i < n_3$, we create a list of connections G^k with elements $g_j^k = (w_j^k, c_j^k, l_j^k)$ where w_j^k is the codeword of weight 3 of the base code, c_j^k is a cluster corresponding to this codeword and l_j^k is the length of the chain of clusters and degree-2 variable nodes connecting the cluster c_j^k and the initial cluster c_i^k. The list includes all the clusters up to a maximum depth l_{max}.
- If $w_t^k = w_t^{k'}$ and $c_j^k = c_t^{k'}$ when $k \neq k'$, then a $(l_j^k + l_t^{k'}) + 1, 1)$ configuration is detected. Save $(c_i^k, c_j^k, c_i^{k'})$.
- If $w_t^k = w_t^{k'}$ and $c_j^k \neq c_t^{k'}$ when $k \neq k'$, then a $(l_j^k + l_t^{k'}) + 2, 2)$ configuration is detected. Save $(c_i^k, c_j^k, c_t^{k'}, c_i^{k'})$.

By performing such a search we found 37 connected couples and 7042 triangles.

4.5.3 Error-floor estimation results

In order to obtain the lower bound on the error-floor for a TLDPC code, we need to calculate $\mathbf{P}\{\xi_{\mathcal{T}_i}\}$, where $\xi_{\mathcal{T}_i}$ represents a decoding failure on (a, i) configurations, $i = 1, 2, 3$, and to sum these probabilities. By doing this, we obtain an estimation of $8.5 \cdot 10^{-7}$ for 0.56 dB which is comparable to the estimation of $1.1 \cdot 10^{-6}$ obtained by simulationg the decoding failure of trapping sets revealed by simulations.

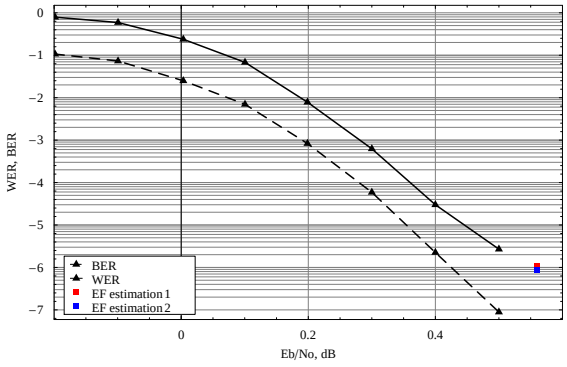


Figure 4.4: Estimations of error floor 1) by simulations results; 2) by using trapping configurations.

Chapter 5

Codes over $\mathbb{F}_q$

5.1 Motivation

So far, only binary codes were considered.

However, in some cases it is very natural to use non-binary codes. Moreover, this may improve the decoding performances, even over binary channels. The benefits of non-binary codes constructions were demonstrated on turbo-codes [12, 13, 15, 62] as well as on LDPC codes [26, 39, 77, 40] and on RA codes [78]. Note that LDPC codes on non-binary alphabets have already been considered by Gallager in Chapter 5 of his PhD thesis [36]. Further, Hu showed that even the performance of 2-regular LDPC (cycle) codes can be improved considerably with non-binary alphabets [39].

For a graph-based code, increasing the alphabet size given a fixed degree distribution increases the density of the underlying binary Tanner graph and the chances of finding better performance and achieving the channel capacity. Note though that iterative decoding is performed over q-ary bipartite graph in this case and the performance improvement will be achieved at the cost of higher decoding complexity. The computation complexity of iterative decoding (Sum-Product or Min-Sum) applied directly over $\mathbb{F}_q$ is of order $0(q^2)$ by symbol. However, it was first observed in [50] that using Fourier transforms for computing extrinsic messages from check nodes to variables nodes decreases the complexity of the decoding and can be reduced to be of order $O(q \log_2 q)$. Later, this result was rediscovered many times.

The bad news is that increasing the alphabet size does not necessarily improve the performance. The investigation performed in [55] on the erasure channel strongly suggests that for a given degree distribution, the asymptotic noise threshold sustained by the family is a unimodal function of the alphabet size : the performances either always decrease with the alphabet size or increase up to a certain alphabet size and decrease afterwards.

The first step to understand the behaviour of non-binary codes is to study their asymptotic performance. Note that in general the messages of the decoder are vectors of dimension $q-1$ and it becomes a difficult task to track their densities. However, it was shown in [2] that, under Gaussian approximation, the $q-1$ dimensional distribution of the messages of a non-binary coset LDPC code can be described by a single parameter. Another interesting result was presented in [74], where it was shown that for non-binary LDPC codes on the BEC the densities of the messages can be computed recursively.

Recall that for cycle codes (these are LDPC codes where all variable nodes are of degree 2) a dramatic improvement in the performances was observed by increasing the alphabet size [39]. It was also put forward in [40], that unlike the binary case, where one has to choose very large degrees to approach capacity, there are very good degree distributions for LDPC codes for large alphabet sizes with variable nodes which are all of small degree and with a very large proportion of variable nodes of degree 2. All these investigations seem to indicate that the case of $(2, d)$ Gallager (cycle) codes over large fields is particularly interesting.

Density evolution operations in [74] are also applied to cycle codes. In this chapter we study density evolution for a non-binary LDPC code family with a small constant fraction of symbols of degree 1 obtained by a slight modification of cycle codes. In this case, the error probability will be constant but it is interesting to see the influence of symbols of degree 1 on the iterative decoding threshold.

We continue investigations on analysis of non-binary codes by extending density evolution operations of [74] over non-binary TLDPC codes. Similarly for the binary case, non-binary TLDPC codes allow symbols of degree 1 in their structure and, by computing thresholds for some particular families with symbols of degree 1, we find that they are equal to thresholds of non-binary cycle codes! This motivates us to study if there is any correspondence between codewords of cycle codes and ones of TLDPC codes with symbols of degree 1. We show in Section 5.3.7 that a more general result holds: an irregular LDPC code, binary or not, having at least 2 degree-2 variable nodes per parity-check equation can be represented as a TLDPC code with symbols of degree 1 and vice versa. The fact to present the LDPC codes as TLDPC codes with symbols of degree 1 reveals another, a more efficient way of decoding of LDPC codes [32].

5.2 State of art

In this section we present general definitions and then density evolution operations for non-binary LDPC codes from [74].

5.2.1 General definitions

Assume that the transmission takes place over the binary erasure channel with erasure probability p and that the all-zero codeword is transmitted. We perform a belief propagation decoding at the receiver end. During an iteration t extrinsic messages $E^{(t)}$ of symbols of the base code are computed and then intrinsic messages $I^{(t)}$ are updated. Note that the messages in the belief propagation decoder are vectors of length q[1]. For a given symbol i of the base code the k-th component of the message $E_i^{(t)}$ ($I_i^{(t)}$) is denoted as $E_i^{(t)}(k)$ ($I_i^{(t)}(k)$), $k = 0, \ldots, q-1$, and this component is equal to the a posteriori probability that the symbol is k.

In the following sections of this chapter, we often consider $q = 2^m$ and in this case all the operations are performed over $\mathbb{F}_{2^m}$.

[1]The decoder message can be represented by a vector of length $q-1$ without any loss of information. However, it will be convenient to consider them to be of length q.

When $q = 2^m$, the vectors $E^{(t)}$ and $I^{(t)}$ have a special form as they can be interpreted as subspaces of the vector space $(\mathbb{F}_2)^m$. Due to transmission over the BEC, the number of non-zero positions of the vectors is a power of 2 and, if the number of non-zero positions is 2^k, then all these positions are equal to 2^{-k}.

Based on messages $E^{(t)}$ and $I^{(t)}$ and supposing $q = 2^m$, we define the following probability vectors $\epsilon^{(t)}$ and $\iota^{(t)}$:

Definition 25 *Let $\iota^{(t)}(k)$ be the probability that the intrinsic message $I^{(t)}$ at iteration t has 2^k non-zero positions, $k = 0, \ldots, m$. Let $\epsilon^{(t)}(k)$ denote the similar probability for message $E^{(t)}$.*

5.2.2 Definition of some useful operations over $\mathbb{R}^q$ and $\mathbb{R}^{m+1}$

In this part we define some operations over $\mathbb{R}^q$ and $\mathbb{R}^{m+1}$ which will be used to define decoding and density evolution operations for non-binary codes.

Convolution and multiplication over $\mathbb{R}^q$

The convolution operator $\otimes$ is defined as an operator between two vectors $a = (a_i)_{i\in\mathbb{F}_q} \in \mathbb{R}^q$ and $b = (b_i)_{i\in\mathbb{F}_q} \in \mathbb{R}^q$, which produces a vector whose k-th component is given by

$$[a \otimes b]_k = \sum_{\alpha\in\mathbb{F}_q} a_\alpha \cdot b_{k-\alpha}, \quad k \in \mathbb{F}_q, \tag{5.1}$$

where the subtraction $k - \alpha$ is evaluated over $\mathbb{F}_q$.

We denote the element-wise multiplication over $\mathbb{R}^q$ by $\odot$:

$$[a \odot b]_k = a_k \cdot b_k, \quad k \in \mathbb{F}_q. \tag{5.2}$$

Both $\odot$ and $\otimes$ operations are associative and commutative over $\mathbb{R}^q$. It is easy to verify that the following property also holds:

$$[a \odot (b \otimes c)]_k = [(a_k \cdot b) \otimes c)]_k = [b \otimes (a_k \cdot c)]_k \,, \tag{5.3}$$

where $a_k \cdot b$ denotes the vector $(a_k \cdot\ b_i)_{i\in\mathbb{F}_q}$.

p-Gaussian binomial coefficient

As we write density evolution equations using p-Gaussian binomial coefficients, it is essential to provide a definition.

Definition 26 *The p-Gaussian binomial coefficient $\begin{bmatrix} n \\ k \end{bmatrix}_p$ denotes the number of vector subspaces of dimension k of a vector space of dimension n over a field of p elements which can be written as*

$$\begin{bmatrix} n \\ k \end{bmatrix}_p = \Pi_{i=0}^{k-1} \frac{p^{n-i} - 1}{p^{k-i} - 1}$$

or

$$\begin{bmatrix} n \\ k \end{bmatrix}_p = \Pi_{i=0}^{k-1} \frac{p^n - p^i}{p^k - p^i}.$$

Let us mention several properties of Gaussian binomial coefficients. One of the properties we will use is the symmetry property:

$$\left[\begin{array}{c} n \\ k \end{array} \right]_p = \left[\begin{array}{c} n \\ n-k \end{array} \right]_p.$$

Define $[n]_p$ by

$$[n]_p = \Pi_{i=0}^{n-1}(p^{n-i}-1),$$

then

$$\left[\begin{array}{c} n \\ k \end{array} \right]_p = \frac{[n]_p}{[k]_p\,[n-k]_p}. \tag{5.4}$$

In what follows, we work with the vector space $(\mathbb{F}_2)^m$ for convenience, we choose $p=2$ and thus we only use 2-Gaussian binomial coefficients.

$a \boxdot b$ and $a \boxtimes b$ over $\mathbb{R}^{m+1}$

For two vectors $a = (a_i)_{0\le i\le m} \in \mathbb{R}^{m+1}$ and $b = (b_i)_{0\le i\le m} \in \mathbb{R}^{m+1}$ we define operations $a \boxdot b$ and $a \boxtimes b$ so that

$$\begin{aligned} [a \boxdot b]_k &= \sum_{i=k}^{m} \sum_{j=k}^{k+m-i} C_{\boxdot}(m,k,i,j) a_i b_j, \quad k = 0,\dots,m, \\ [a \boxtimes b]_k &= \sum_{i=0}^{k} \sum_{j=k-i}^{k} C_{\boxtimes}(m,k,i,j) a_i b_j, \quad k = 0,\dots,m, \end{aligned}$$

with the following expressions for $C_{\boxdot}$ and $C_{\boxtimes}$

$$\begin{aligned} C_{\boxdot}(m,k,i,j) &= \frac{\left[\begin{array}{c} i \\ k \end{array} \right]_2 \left[\begin{array}{c} m-i \\ j-k \end{array} \right]_2}{\left[\begin{array}{c} m \\ j \end{array} \right]_2} 2^{(i-k)(j-k)}, \\ C_{\boxtimes}(m,k,i,j) &= \frac{\left[\begin{array}{c} m-i \\ m-k \end{array} \right]_2 \left[\begin{array}{c} i \\ k-j \end{array} \right]_2}{\left[\begin{array}{c} m \\ m-j \end{array} \right]_2} 2^{(k-i)(k-j)}. \end{aligned}$$

$C_{\boxdot}(m,k,i,j)$ represents the probability of choosing a subspace of dimension j in a space of dimension m so that its intersection with a fixed subspace of dimension i has dimension k. $C_{\boxtimes}(m,k,i,j)$ is the probability to choose a subspace of dimension j in a space of dimension m so that it forms a subspace of dimension k together with a fixed subspace of dimension i.

Operations $\boxdot$ and $\boxtimes$ are commutative over $\mathbb{R}^{m+1}$ as shown in Appendix D.1.

5.2.3 LDPC codes over $\mathbb{F}_q$

An $\mathbb{F}_q$-LDPC code is defined by its q-ary bipartite graph where to each variable node a symbol from $\mathbb{F}_q$ is assigned, a linear bijective mapping $f : \mathbb{F}_q \mapsto \mathbb{F}_q$ is associated to each edge and each check node corresponds to a parity-check equation over $\mathbb{F}_q$.

Let k be the number of edges of the q-ary bipartite graph and $\lambda = (\lambda_2, \ldots, \lambda_s)$ be a probability distribution over the set of integers $\{2, \ldots, s\}$ so that $\lambda_i k/i$ is an integer for any $i \in \{2, \ldots, s\}$, then the number of edges of left degree i is $\lambda_i k$ and the number of variable nodes of degree i is $\lambda_i k/i$. Similarly, let $\rho = (\rho_3, \ldots, \rho_l)$ be a probability distribution over the set of integers $\{3, \ldots, l\}$ so that $\rho_j k/j$ is an integer for any $j \in \{3, \ldots, l\}$, then the number of edges of right degree j is $\rho_j k$ and the number of check nodes of degree j is $\rho_j k/j$.

Note that the total number of vertices of the q-ary bipartite graph is $n = k \sum_{i=2}^{s} \lambda_i/i$ and the total number of check nodes is $r = k \sum_{j=3}^{l} \rho_j/j$.

Let $x = (x_0, \ldots, x_{n-1}) \in (\mathbb{F}_q)^n$ be a codeword of an $\mathbb{F}_q$-LDPC code. When $q = 2^m$, each symbol x_i can be represented as a binary m-tuple and x can be seen as a binary codeword of length mn.

For the analysis, it is essential that the edge mapping f is linear bijective, this provides that after the edge action a message of same dimension gets mapped to any message of same dimension with equal probability.

We make the infinite tree hypothesis, i.e. that for a fixed iteration number t the computation graph is a tree and the density at iteration t is only determined by the distributions λ and ρ. In fact, this hypothesis holds for infinite block lengths (asymptotic case), but it can be shown that in the finite-length case the average densities of the non-binary LDPC code ensemble converge to densities of the asymptotic case for a fixed number of iterations when $n \to \infty$.

We present density evolution operations for $\mathbb{F}_q$-LDPC codes in the asymptotic case, the average is taken over all permutations and all linear bijective mappings as it was defined in [74].

Lemma 17 [74] *For an $\mathbb{F}_q$-LDPC code density evolution operations at the t-th iteration are the following:*

1. *on the check node size*

$$\epsilon^{(t+1)} = \sum_{i=3}^{l} \rho_i \epsilon_i^{(t)}$$

where the $\epsilon_i^{(t)}$'s are computed recursively

$$\begin{aligned} \epsilon_3^{(t)} &= \iota^{(t)} \boxtimes \iota^{(t)} \\ \epsilon_j^{(t)} &= \epsilon_{j-1}^{(t)} \boxtimes \iota^{(t)}, \quad j = 4, \ldots, i. \end{aligned}$$

2. *on the variable node side*

$$\iota^{(t)} = \sum_{i=2}^{s} \lambda_i \iota_i^{(t)},$$

where the $q_i^{(t)}$'s are computed recursively

$$\begin{aligned} \iota_2^{(t)} &= \iota^{(0)} \boxdot \epsilon^{(t)} \\ \iota_j^{(t)} &= \iota_{j-1}^{(t)} \boxdot \epsilon^{(t)}, \quad j = 3, \ldots, i. \end{aligned}$$

with $\iota^{(0)}(k) = \binom{m}{k} p^k (1-p)^{m-k}$, $0 \leq k \leq m$, p is the erasure channel probability.

5.3 TLDPC codes over $\mathbb{F}_q$

In this section we define a family of non-binary TLDPC codes and density evolution operations for this family.

5.3.1 Definition

Similarly as for the binary case, a general construction for graph-based code ensembles over $\mathbb{F}_q$ can be defined by means of a non-binary base code and of a q-ary bipartite graph. We consider $q = 2^m$.

Definition 27 [Non-binary code ensemble] *A non-binary graph-based code ensemble of length n over $\mathbb{F}_q$ is defined by a non-binary base code C_b of length $n_b \geq n$ and a random q-ary bipartite graph between a set V of variable nodes of size n, each of them being assigned a symbol from $\mathbb{F}_q$, and a set W formed by positions of C_b. To each edge of the q-ary bipartite graph a linear bijective mapping $f : (\mathbb{F}_2)^m \mapsto (\mathbb{F}_2)^m$ is assigned.*

Let $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_s)$ be a probability distribution over the set of integers $\{1, 2, \ldots, s\}$ so that $\lambda_i n_b / i$ is an integer for any $i \in \{1, 2, \ldots, s\}$ and such that $n = n_b \sum_{i=1}^{s} \lambda_i / i$, then the number of edges of degree i is $\lambda_i n_b$ and the number of vertices in V of degree i is $\lambda_i n_b / i$.

The bipartite graph together with the base code C_b specifies a code of length n as the set of assignments $\{v_i\}_{1 \leq i \leq n} \in (\mathbb{F}_q)^n$ of V such that the induced assignments [2] $\{w_j\}_{1 \leq j \leq n_b} \in (\mathbb{F}_q)^{n_b}$ of vertices of W belong to C_b.

We consider a particular case of this general construction called TLDPC code family over $\mathbb{F}_q$ defined by the its non-binary base code:

Definition 28 [Non-binary $((a, b))$ TLDPC code] *Let T be a trellis having two sections, parallel (P) and cross-like (X), as shown in Fig.5.1, for which*

- *states $s_0, s_1, s_2 \in \mathbb{F}_q$,*
- *the branch label of the parallel section $x^P = (x_1^P, \ldots, x_a^P) \in (\mathbb{F}_q)^a$ and the branch label of the cross-like section $x^X = (x_1^X, \ldots, x_b^X) \in (\mathbb{F}_q)^b$,*

[2] a vertex in W receives the assignment of the vertex in V to which it is connected through a linear bijective mapping f.

- *the parallel and the cross-like sections are defined by following parity equations*

$$s_1 = \sum_{j=1}^{a} x_j^P = s_0,$$

$$s_2 = s_1 + \sum_{j=1}^{b} x_j^X,$$

the sums being computed over $\mathbb{F}_q$.

Then the trellis of the base code of an $((a, b))$ *TLDPC code over* $\mathbb{F}_q$ *is a concatenation of* r *copies of* T *for which the initial* s_0 *and the last* s_{2r} *trellis states are identified. The base code has length* $n_b = r(a + b)$.

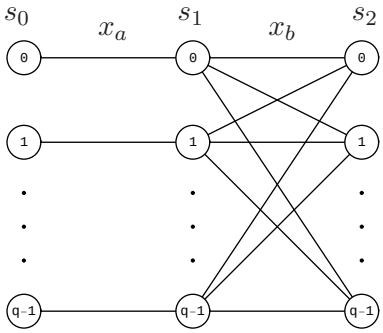


Figure 5.1: Trellis T with two sections, parallel (P) and cross-like (X).

As in the case of binary codes, the inclusion of a constant fraction of symbols of degree 1 in the structure of a non-binary code ensemble may improve the convergence of the iterative decoding and performances in the waterfall region. However, if we have such a constant fraction of symbols of degree 1 in our structure and we choose at random positions of symbols of degree 1 in the base code, the probability of having a codeword of constant weight is non-zero. This comes from many reasons, one of them is that there is a non-zero probability to have three consecutive sections X, P and X, each of them carrying a symbol of degree 1, which gives a valid configuration for a codeword of weight 3. Moreover, if symbols of degree 1 are put on sections of length > 1, there is a non-zero probability that a variable of degree i is connected to i sections each of them carrying a symbol of degree 1 and this is a valid configuration for a codeword of weight $i + 1$. So, in order not to harm the minimum distance of the code ensemble by allowing symbols of degree 1 in its structure, symbols of degree 1 should be placed on the trellis sections of the base code in a structured way in order to prevent that two consecutive sections carry symbols of degree 1 and sections carrying symbols of degree 1 should be of length 1.

With the aim to allow the maximum possible fraction of symbols of degree 1, we can choose all the parallel sections to be of length 1 and to carry symbol of degree 1. Thus every second trellis section will carry a symbol of degree 1. Note that we cannot choose cross-like sections to be carriers of symbols of degree 1, in this case the minimum distance is upper bounded by $1 + 2i$, where i is the minimum degree > 1. This comes from the fact

that a non-zero position on a parallel section forms a codeword of weight 3 together with symbols of degree 1 of two neighbouring cross-like sections.

In following sections we consider two particularly interesting families of non-binary TLDPC codes with parameters $((b,b))$ and $((1,b))_1$. The $((b,b))$ TLDPC code family is an example of a code family with no symbols of degree 1 in its structure. By $((1,b))_1$ we denote a TLDPC code family which has symbols of length 1 carried by parallel sections of the trellis of the base code of length 1.

5.3.2 Decoding operations for $((a,b))$ TLDPC code family

We assume that the transmission takes place over the binary erasure channel with erasure probability p and we perform a belief propagation decoding at the receiver end. During an iteration t we compute extrinsic messages $E^{(t)}$ of symbols of the base code and then we update intrinsic ones $I^{(t)}$.

Similarly, we also define state messages $S^{(t)}$ of trellis sections of the base code. We denote by $S_l^{(t)}$ the state vector on the input of the trellis section l at iteration t, where $l = 0, \ldots, 2r$, r being the number of concatenated trellises T in the trellis of the base code. We distinguish forward and backward state messages.

Let us define operations during the belief propagation decoding. Before iterating we initialize the intrinsic messages $I^{(0)}$ from variable nodes to the base code by values received from the channel.

An iteration of decoding consists of the following steps:

1. edge mapping
   Because of invertibility of the edge mapping, elements in vectors $I^{(t)}$ are permuted, i.e. for an intrinsic message vector $I_i^{(t)} = (I_i^{(t)}(0), \ldots, I_i^{(t)}(q-1))$ and a mapping f we get
   $$\tilde{I}^{(t)} = (I_i^{(t)}(f(0)), \ldots, I_i^{(t)}(f(q-1))).$$

2. base code decoding
   The decoding on the tail-biting trellis of the base code is performed. It computes outgoing messages $\tilde{E}^{(t)}$ as a function of the incoming messages $\tilde{I}^{(t)}$.

   The tail-biting trellis of the base code has two types of sections, P and X, having respectively a and b symbols per branch.

   On forward (backward) stage we compute state messages S^f (S^b) as a function of the previous (future) state messages and intrinsic ones. Let $d = a + b$. For the forward stage we have

   $$\begin{aligned} S^f_{2j+1} &= S^f_{2j} \odot (\otimes_{i=0}^{a-1} \tilde{I}^{(t)}_{dj+i}) \\ S^f_{2j+2} &= S^f_{2j+1} \otimes (\otimes_{i=0}^{b-1} \tilde{I}^{(t)}_{dj+a+i}) \end{aligned}$$

   and for the backward stage

   $$\begin{aligned} S^b_{2j+1} &= S^b_{2j+2} \otimes (\otimes_{i=0}^{b-1} \tilde{I}^{(t)}_{dj+a+i}) \\ S^b_{2j} &= S^b_{2j+1} \odot (\otimes_{i=0}^{a-1} \tilde{I}^{(t)}_{dj+i}). \end{aligned}$$

with $j = 0, \ldots, r-1$. We fix initial condition to be $S_0^f = S_{2r}^b = \frac{1}{q}(1\ 1\ \ldots\ 1)^{\mathrm{T}}$. In the following step we compute extrinsic messages $\tilde{E}^{(t+1)}$ as follows:

$$\begin{aligned}
\tilde{E}_{dj+i}^{(t+1)} &= (S_{2j}^f \odot S_{2j+1}^b) \otimes (\otimes_{s=0,s\neq i}^{a-1} \tilde{I}_{dj+s}^{(t)}), \qquad i = 0, \ldots, a-1, \\
\tilde{E}_{dj+a+i}^{(t+1)} &= (S_{2j+1}^f \otimes S_{2j+2}^b) \otimes (\otimes_{s=0,s\neq i}^{b-1} \tilde{I}_{dj+a+s}^{(t)}), \quad i = 0, \ldots, b-1.
\end{aligned}$$

3. inverse edge mapping
   Messages $\tilde{E}^{(t+1)}$ are normalised and inversely permuted, i.e. for an extrinsic message vector $\tilde{E'}_i^{(t+1)} = (\tilde{E'}_i^{(t+1)}(0), \ldots, \tilde{E'}_i^{(t+1)}(q-1))$ obtained by normalisation of the vector $\tilde{E}_i^{(t+1)}$, and an edge mapping f we get

$$E_i^{(t+1)} = (\tilde{E'}_i^{(t+1)}(f^{-1}(0)), \ldots, \tilde{E'}_i^{(t+1)}(f^{-1}(q-1))).$$

4. message updating at variable nodes
   We calculate new intrinsic messages $I^{(t+1)}$ by element-wise multiplication of incoming messages of neighbouring edges $E^{(t+1)}$ and of the initial message $I^{(0)}$. Thus for an edge l connected with a degree-n variable node incoming extrinsic messages $E_1^{(t+1)}, \ldots, \tilde{E}_n^{(t+1)}$ we write

$$I_l^{(t+1)} = (\odot_{k=1,k\neq l}^{n} \tilde{E}_k^{(t+1)}) \odot I_l^{(0)}. \tag{5.5}$$

5.3.3 Density evolution equations

To define density evolution operations, we make the following assumptions:

- by linearity, we assume without loss of generality that the all-zero codeword was transmitted.
- we assume that that the code is of infinite length and, therefore, that the base code is of infinite length, so any two distant positions of the base code are located far from each other and the extrinsic probabilities calculated for these positions are independent as if they came from two separate base codes.
- similarly as for $\mathbb{F}_q$ LDPC codes, we make the infinite tree assumption by assuming that that for a fixed iteration number t the computation graph of the TLDPC code is tree-like.

Thus we will define density evolution operations for the asymptotic (infinite code length) case. In the finite-length case the last two assumptions do not hold, but the average densities of the TLDPC code ensemble will converge to densities of the asymptotic case as the code length will tend to infinity.

We describe density evolution operations for the base code side separately for families $((b,b))$ and $((1,b))_1$. Note that the edge mapping and the inverse edge mapping do not change values of $\iota^{(t)}$ and $\epsilon^{(t)}$ messages, so, we can write density evolution equations for the base code side in terms of these quantities (working now with $E^{(t)}$ instead of $\tilde{E}^{(t)}$).

Recall that the number of symbols per trellis section of the base code for $((b,b))$ TLDPC family is equal to b.

Variable nodes side

Note that as the transmission is made over the BEC, initial intrinsic messages probabilities are equal to

$$\iota^{(0)}(k) = \binom{m}{k} p^k (1-p)^{m-k}, \quad 0 \le k \le m.$$

Density evolution operations can be rewritten in terms of probabilities $\epsilon^{(t)}$ and $\iota^{(t)}$.

Lemma 18 *The density evolution over variable nodes of degree ≥ 2 is rewritten in terms of probabilities $\epsilon^{(t)}$ and $\iota^{(t)}$ in the following way:*

$$\iota^{(t)} = \iota^{(0)} \boxdot \sum_{k=2}^{s} \frac{\lambda_k(\boxdot_{i=1}^{k-1} \epsilon^{(t)})}{\sum_{i=2}^{s} \lambda_i}, \quad t \geq 1.$$

For variable nodes of degree 1 the probability vector $\iota^{(t)}$ is always equal to $\iota^{(0)}$.

Base code side operations for family $((b, b))$

- Forward/backward stage
  First we define state probability vectors for the trellis of the base code

  Definition 29 *For a given trellis section i of the base code, let $\xi_i^{(t)}(k)$ be the probability that the corresponding state message $\xi_i^{(t)}$ at iteration t has k non-zero positions, $0 \le k \le m$, $i = 0, \ldots, 2r$, r being the number of concatenated trellises T in the trellis of the base code. We distinguish forward ξ_i^f and backward ξ_i^b probability vectors*[3].

  As example, in Fig. 5.2 we depict the factor graph of the l-th concatenated trellis T of $((2, 2))$ code family. It contains a parallel (P) and a cross-like (X) section. Black circles represent state edges of the trellis. Boxes represent symbols connected to every trellis section.

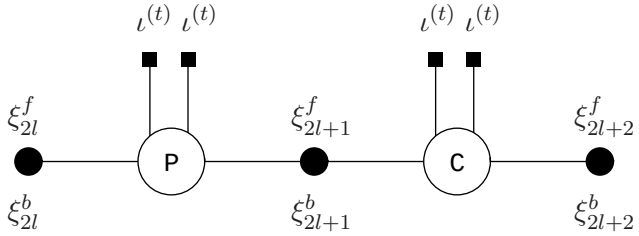


Figure 5.2: A l-th concatenated trellis T of $((2, 2))$ code family and the location of probability vectors.

In the case of $((b, b))$ family, for a trellis section we denote $\tilde{\iota}^{(t)}$ and $\hat{\iota}^{(t)}$ to be

$$\tilde{\iota}^{(t)} = \boxtimes_{i=0}^{b-1} \iota_i^{(t)},$$
$$\hat{\iota}^{(t)} = \boxtimes_{i=0}^{b-2} \iota_i^{(t)}.$$

[3]the iteration number t is omitted for simplicity

Then for the forward/backward stage we can write the following sets of equations:

$$\begin{array}{l} \xi^f_{2l+1} = \tilde{\iota}^{(t)} \boxdot \xi^f_{2l} \\ \xi^f_{2l+2} = \tilde{\iota}^{(t)} \boxtimes \xi^f_{2l+1} \end{array} \tag{5.6}$$

and

$$\begin{array}{l} \xi^b_{2l} = \tilde{\iota}^{(t)} \boxdot \xi^b_{2l+1} \\ \xi^b_{2l+1} = \tilde{\iota}^{(t)} \boxtimes \xi^b_{2l+2} \end{array} \tag{5.7}$$

where $l = 0, \ldots, r-1$.

Lemma 19 *The following expressions hold for forward state probability vectors ξ^f_{2l+1} and ξ^f_{2l+2}:*

$$\begin{array}{l} \xi^f_{2l+1} = M^l_{XP} \xi^f_1, \\ \xi^f_{2l+2} = M^l_{PX} \xi^f_0, \end{array}$$

where the elements of matrices M_{PX} and M_{XP} are the following:

$$M_{PX}(i,j) = \begin{cases} \sum_{u=0}^{j} A(m,i,u)B(m,u,j), & \text{if } \ 0 \leq \mathrm{j} \leq \mathrm{i}, \\ \sum_{u=0}^{k} A(m,i,u)B(m,u,j), & \text{if } \ \mathrm{i}+1 \leq \mathrm{j} \leq \mathrm{m}, \end{cases} \tag{5.8}$$

$$M_{XP}(i,j) = \begin{cases} \sum_{u=i}^{m} B(m,i,u)A(m,u,j) & \text{if } \ 0 \leq \mathrm{j} \leq \mathrm{i}, \\ \sum_{u=j}^{m} B(m,i,u)A(m,u,j) & \text{if } \ \mathrm{i}+1 \leq \mathrm{j} \leq \mathrm{m}, \end{cases} \tag{5.9}$$

$A(m,i,u) = \sum_{v=i-u}^{i} C_{\boxtimes}(m,i,u,v)\iota^{(t)}(v)$
and $B(m,u,j) = \sum_{v=u}^{m+u-j} C_{\boxdot}(m,u,j,v)\iota^{(t)}(v)$.

Proof : By rewriting expressions (5.6) and using commutativity, we obtain

$$\begin{array}{lll} \xi^f_{2l+1} & = \tilde{\iota}^{(t)} \boxdot (\tilde{\iota}^{(t)} \boxtimes \xi^f_{2l-1}) & = (\xi^f_{2l-1} \boxtimes \tilde{\iota}^{(t)}) \boxdot \tilde{\iota}^{(t)} \\ \xi^f_{2l+2} & = \tilde{\iota}^{(t)} \boxtimes (\tilde{\iota}^{(t)} \boxdot \xi^f_{2l}) & = (\xi^f_{2l} \boxdot \tilde{\iota}^{(t)}) \boxtimes \tilde{\iota}^{(t)}. \end{array}$$

The mapping $\xi^f_{2l-1} \mapsto (\xi^f_{2l-1} \boxtimes \tilde{\iota}^{(t)}) \boxdot \tilde{\iota}^{(t)}$ from $\mathbb{R}^{m+1}$ into $\mathbb{R}^{m+1}$ is linear and can be written as $\xi^f_{2l-1} \mapsto M_{XP}\xi^f_{2l-1}$, where M_{XP} is defined by (5.8), and thus $\xi^f_{2l+1} = M_{XP}\xi^f_{2l-1}$. By similar considerations, $\xi^f_{2l+2} = M_{PX}\xi^f_{2l}$, M_{PX} being defined by (5.8). The details about about how the expression (5.8)was obtained are presented in Appendix D.2.

We obtain that

$$\begin{array}{lll} \xi^f_{2l+1} & = & M^l_{XP}\xi^f_1, \\ \xi^f_{2l+2} & = & M^l_{PX}\xi^f_0. \end{array}$$

∎

We give without proof a similar lemma for backward state probability vectors:

Lemma 20 *The following expressions hold for backward state probability vectors* ξ^b_{2l+1} *and* ξ^b_{2l}*:*

$$\begin{aligned} \xi^b_{2l+1} &= (M_{PX})^{r-l}\xi^b_{2r-1}, \\ \xi^b_{2l} &= (M_{XP})^{r-l}\xi^b_{2r}, \end{aligned}$$

where M_{PX} *and* M_{XP} *are defined by (5.8).*

We look for a solution of systems of equations in the asymptotic case assuming that the base code is of infinite length, i.e. we want to calculate state probability vectors ξ^f_{2l}, ξ^f_{2l+1}, ξ^b_{2l} and ξ^b_{2l+1}. This can be done using the theory of nonnegative matrices. For this we recall some well-known definitions:

Definition 30 [Stochastic matrix] *A stochastic matrix* M *is a nonnegative matrix such that its column sums are equal to 1.*

Lemma 21 M_{PX} *and* M_{XP} *are stochastic matrices.*

Proof : The matrix M_{PX} (M_{XP}) is the matrix of linear mapping which maps any probability vector ξ^f_{2l} (ξ^f_{2l-1}) into a probability vector ξ^f_{2l+2} (ξ^f_{2l+1}). By (5.8), the coefficients of M_{PX} (M_{XP}) are nonnegative. These two facts imply the stochasticity of M_{PX} (M_{XP}). ∎

Definition 31 [Irreducible matrix, period] *An irreducible matrix* M *is a square nonnegative matrix such that for every* i, j *there exists* $p_{i,j} > 0$ *such that* $[M^{p_{i,j}}]_{i,j} > 0$*. Let* $k_{i,j} = \min\{p_{i,j} : [M^{p_{i,j}}]_{i,j} > 0\}$*. The period of an irreducible matrix* M *is the greatest common divider for all the* $k_{i,i}$*'s.*

Definition 32 [Aperiodic matrix] *An aperiodic matrix* M *is an irreducible matrix with period one.*

Note that M_{PX} and M_{XP} are irreducible and aperiodic when $0 < p < 1$. This comes from the fact that in this case the vector $\iota^{(t)}$ is positive and so is $\tilde{\iota}^{(t)}$. Matrices defined by $C_{\boxdot}$ and $C_{\boxtimes}$ operations being always positive, we obtain that all the elements of M_{PX} and M_{XP} are positive and, therefore, they are irreducible and aperiodic.

Let $\Pi = (\pi_0, \ldots, \pi_m)$ be a stationary distribution vector[4] associated to a stochastic matrix M, if $M\Pi = \Pi$ and $\sum_i \pi_i = 1$. Now we draw upon the Perron-Frobenius theorem:

Theorem 12 *If* M *is an irreducible aperiodic stochastic matrix then*

- *one of its eigenvalues is equal to 1,*
- *all other eigenvalues are smaller than 1 in absolute value,*
- M *has a unique positive stationary distribution vector* Π*,*
- *for any non-zero probability vector* U*,* $\lim_{k\to\infty} M^k U = \Pi$*.*

[4] Π is a column vector

Definition 33 [Stationary distribution vectors] *We define by* $\xi^f_{2\infty+1}$ *and* $\xi^f_{2\infty}$ *the stationary distribution vectors of respective matrices* M_{XP} *and* M_{PX}*.*

To compute $\xi^f_{2\infty+1}$ and $\xi^f_{2\infty}$, we use the properties of Π and solve the following systems of equations:

$$\begin{cases} \xi^f_{2\infty} = M_{PX}\xi^f_{2\infty} \\ \sum_k \xi^f_{2\infty}(k) = 1 \end{cases}$$

and

$$\begin{cases} \xi^f_{2\infty+1} = M_{XP}\xi^f_{2\infty+1} \\ \sum_k \xi^f_{2\infty+1}(k) = 1 \end{cases} .$$

Similarly for the backward step, it can be shown that probability vectors $\xi^b_{2\infty}$ and $\xi^b_{2\infty+1}$ are respectively stationary distribution vectors of M_{XP} and M_{PX} and

$$\begin{aligned} \xi^b_{2\infty} &= \xi^f_{2\infty+1}, \\ \xi^b_{2\infty+1} &= \xi^f_{2\infty}. \end{aligned}$$

- Computation of $\epsilon^{(t)}$

  We have the following theorem:

  Theorem 13 *Given* $\xi^f_{2\infty}$*,* $\xi^f_{2\infty+1}$*,* $\xi^b_{2\infty}$ *and* $\xi^b_{2\infty+1}$*, the probability vector* $\epsilon^{(t)}$ *of extrinsic messages under the infinite base code assumption is calculated as follows*

$$\epsilon^{(t)} = \frac{1}{2}\left[(\xi^f_{2\infty} \boxdot \xi^b_{2\infty+1}) \boxtimes \hat{\imath}^{(t)} + (\xi^f_{2\infty+1} \boxtimes \xi^b_{2\infty}) \boxtimes \hat{\imath}^{(t)}\right]$$

  Proof : Let $\epsilon^{(t)}_P$ and $\epsilon^{(t)}_X$ be the probability vectors of extrinsic messages for the parallel trellis section and for the cross-like trellis section respectively. Then they are computed as follows

$$\begin{aligned} \epsilon^{(t)}_P(k) &= (\xi^f_{2\infty} \boxdot \xi^b_{2\infty+1}) \boxtimes \hat{\imath}^{(t)}, \\ \epsilon^{(t)}_X(k) &= (\xi^f_{2\infty+1} \boxtimes \xi^b_{2\infty}) \boxtimes \hat{\imath}^{(t)}, \quad k = 0, \ldots, m. \end{aligned}$$

  The $((b, b))$ base code containing the equal number of parallel and cross-like sections, the probability vector $\epsilon^{(t)}$ is simply

$$\epsilon^{(t)} = \frac{\epsilon^{(t)}_P + \epsilon^{(t)}_X}{2}$$

  ∎

Base code side operations for the family $((1, b))_1$

- Forward/backward stage
  We define forward ξ^f and backward ξ^b state probability vectors as it is done in Definition 29.

As an example, in Fig.5.3 we depict the factor graph of the l-th concatenated trellis T of the $((1,2))_1$ code family containing a parallel (P) section of length 1 and a cross-like (X) section of length 2 where parallel sections carry symbols of degree 1. Black circles represent state edges of the trellis. Boxes represent symbols connected to every trellis section.

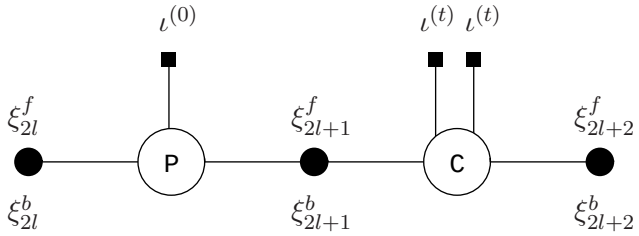


Figure 5.3: A l-th concatenated trellis T of the $((1,2))_1$ code family and the location of probability vectors.

For a cross-like trellis section we denote $\tilde{\iota}^{(t)}$ and $\hat{\iota}^{(t)}$ to be

$$\tilde{\iota}^{(t)} = \boxtimes_{i=0}^{b-1} \iota_i^{(t)},$$
$$\hat{\iota}^{(t)} = \boxtimes_{i=0}^{b-2} \iota_i^{(t)}.$$

Then on the forward/backward stage we can write following sets of equations:

$$\xi^f_{2l+1} = \tilde{\iota}^{(0)} \boxdot \xi^f_{2l}$$
$$\xi^f_{2l+2} = \tilde{\iota}^{(t)} \boxtimes \xi^f_{2l+1}$$

and

$$\xi^b_{2l} = \tilde{\iota}^{(0)} \boxdot \xi^b_{2l+1}$$
$$\xi^b_{2l+1} = \tilde{\iota}^{(t)} \boxtimes \xi^b_{2l+2}$$

As we compute extrinsic messages for symbols of degrees > 1, we are only interested in odd forward and even backward probability vectors.

Lemma 22 *The following expressions hold for odd forward and even backward state probability vectors*

$$\xi^f_{2l+1} = M^l \xi^f_1, \tag{5.10}$$
$$\xi^b_{2l} = M^{r-l} \xi^b_{2r}, \tag{5.11}$$

the elements of M being the following:

$$M(i,j) = \begin{cases} \sum_{u=i}^{m} B_0(m,i,u) A(m,u,j) & \text{if } \ 0 \le \mathrm{j} \le \mathrm{i}, \\ \sum_{u=j}^{m} B_0(m,i,u) A(m,u,j) & \text{if } \ \mathrm{i}+1 \le \mathrm{j} \le \mathrm{m}, \end{cases} \tag{5.12}$$

where $A(m,i,u) = \sum_{v=i-u}^{i} C_{\boxtimes}(m,i,u,v) \iota^{(t)}(v)$
and $B_0(m,u,j) = \sum_{v=u}^{m+u-j} C_{\boxdot}(m,u,j,v) \iota^{(0)}(v)$.

Proof : We have

$$\begin{aligned}\xi^f_{2l+1} &= \tilde{\iota}^{(0)} \boxdot (\tilde{\iota}^{(t)} \boxtimes \xi^f_{2l-1}) = (\xi^f_{2l-1} \boxtimes \tilde{\iota}^{(t)}) \boxdot \tilde{\iota}^{(0)} = M\xi^f_{2l-1},\\ \xi^b_{2l} &= \tilde{\iota}^{(0)} \boxdot (\tilde{\iota}^{(t)} \boxtimes \xi^b_{2l+2}) = M\xi^b_{2l+2},\end{aligned}$$

where M is the matrix of linear mapping $\mathbb{R}^{m+1} \mapsto \mathbb{R}^{m+1}$ and is defined by (5.12). We obtain (5.10) and (5.11) by recursion. ∎

Using the same considerations as for the family $((b,b))$, one can show that the matrix M is irreducible, aperiodic and stochastic.

Lemma 23 *Let $\xi^f_{2\infty+1}$ and $\xi^b_{2\infty}$ are stationary distribution vectors of M. Then $\xi^f_{2\infty+1} = \xi^b_{2\infty}$.*

We compute $\xi^f_{2\infty+1}$ and $\xi^b_{2\infty}$ by solving the system of equations:

$$\begin{cases}\xi^f_{2\infty+1} = M\xi^f_{2\infty+1}\\ \sum_k \xi^f_{2\infty+1}(k) = 1\\ \xi^f_{2\infty+1} = \xi^b_{2\infty}\end{cases}$$

- Computation of $\epsilon^{(t)}$

Theorem 14 *Given $\xi^f_{2\infty+1}$ and $\xi^b_{2\infty}$, the extrinsic probability vector $\epsilon^{(t)}$ under the infinite base code assumption is calculated as follows*

$$\epsilon^{(t)}(k) = \epsilon^{(t)}_X(k) = (\xi^f_{2\infty+1} \boxtimes \xi^b_{2\infty}) \boxtimes \hat{\iota}^{(t)}.$$

5.3.4 Results on thresholds

In this part we present obtained results of thresholds for families $((b,b))$ and $((1,b))_1$.

We compute thresholds for $((b,b))$-family with $\Lambda(x) = x$ for different values of b and m and present the obtained results in the table below as well as rates and capacity limits p^{Sh} for different values of b.

For $b = 2$ the threshold gets worse with increasing the size of the alphabet. But for $b = 3$ and $b = 4$ we have a slight improvement of the thresholds for $m = 2$ and after that the threshold values begin to decrease.

In Table 5.2 we present thresholds of $((1,b))_1$-TLDPC code family with $\Lambda(x) = 1/(b+1) + bx/(b+1)$ for different values of b and m.

In Table 5.3 we present the thresholds for (2,3) non-binary LDPC codes taken from [74] and for (2,4) non-binary LDPC codes. Comparing them with the results of Table 5.2 we see that the thresholds for $((1,1))_1$-TLDPC code family coincide with the thresholds of (2,3) LDPC codes up to the third digit and so do the $((1,2))_1$-TLDPC codes with the (2,4) LDPC codes. At the same time, it can be easily verified that in the binary case the threshold of the (2,3) LDPC code ensemble coincides with the threshold of the $((1,1))_1$-TLDPC code ensemble as well as the threshold of the (2,4) LDPC code ensemble coincides with the threshold of the $((1,2))_1$-TLDPC code ensemble. Thus, the difference

m \ b	2	3	4
1	0.4529	0.2986	0.2215
2	0.4489	0.3005	0.2252
3	0.4372	0.2932	0.2200
4	0.4227	0.2828	0.2121
rate	1/2	2/3	3/4
p^{Sh}	0.5	1/3	0.25

Table 5.1: Thresholds of $((b,b))$-TLDPC code family over $\mathbb{F}_{2^m}$ with $\Lambda(x) = x$ for different values of b and m.

m \ b	1	2
1	0.5	1/3
2	0.5774	0.4096
3	0.6184	0.4506
4	0.6369	0.4681
5	0.6445	0.4742
6	0.6465	0.4747
7	0.6453	0.4724
rate	1/3	1/2

Table 5.2: Thresholds of $((1,b))_1$-TLDPC code family over $\mathbb{F}_{2^m}$ with $\Lambda(x) = 1/(b+1) + bx/(b+1)$ for different values of b and m.

m \ (c,d)	(2,3)	(2,4)
1	0.5	1/3
2	0.5775	0.4094
3	0.6183	0.4506
4	0.6369	0.4680
5	0.6446	0.4742
6	0.6464	0.4742
7	0.6453	0.4723
rate	1/3	1/2

Table 5.3: Thresholds of non-binary (2,3) and (2,4) LDPC codes for different values and m.

in thresholds of LDPC and TLDPC codes can be due to the slow convergence in threshold computations. To verify this, we fix $m = 4$ and do 10000 of iterations for the (2,3) LDPC cycle code ensemble and 1000 iterations for the $((1,1))_1$ TLDPC code ensemble; the computations are made with the Maple precision of 30 digits for floating-point numbers. We obtain the same value of threshold $p^{th} = 0.63684$ for both of them.

We conjecture that the thresholds of (2,3) LDPC and $((1,1))_1$-TLDPC code ensembles, as well as of (2,4) LDPC and $((2,1))$-TLDPC code ensembles are the same.

5.3.5 EXIT chart for the (2,3) LDPC code ensemble and the $((1,1))_1$-TLDPC code ensemble

The important difference between ensembles of 2-regular non-binary LDPC codes and ensembles of $((1,b))_1$-TLDPC codes is the existence of symbols of degree 1 in the structure of $((1,b))_1$-TLDPC codes which has a positive effect on the entropy curve of the base code and on iterative decoding performances of the code ensemble. In this subsection we illustrate the positive effect of symbols of degree 1 on entropy curves of a code ensemble comparing entropy curves of the (2,3) LDPC binary code ensemble and the $((1,1))_1$-TLDPC binary code ensemble in the binary case.

For the binary (2,3) LDPC code ensemble the entropy curve of the base code (which is the parity code [3,2,2]) is described by the simple expression

$$g_{[3,2,2]}(x) = 1 - (1-x)^2.$$

The expression for the entropy curve of the $((1,1))_1$-TLDPC base code can also be found analytically, and it is done in Appendix D.3. Finally, if p is the erasure probability of the transmission channel, then the entropy curve of the $((1,1))_1$-TLDPC base code is expressed as

$$g_B(x,p) = 1 - \left(\frac{1-p}{1-p(1-x)}\right)^2.$$

The entropy curve of variable nodes for both code ensembles is simply

$$h(x) = x/p.$$

$h(x) = x/p$, $g_{[3,2,2]}(x)$ and $g_B(x,p)$ for $p = 1/2$ are presented in Fig.5.4.

It can be shown that entropy curves of both base codes are concave, and, given that the entropy curve of variable nodes for 2-regular code ensembles is a straight line, in the limit case of $p = p^{th}$ it is a tangent to the corresponding entropy curve of the base code. It can be also shown that the tangent of entropy curves of both base codes is $y = 2x$ and thus, both code ensembles have threshold $p^{th} = 1/2$ in the binary case.

We can see that the entropy curve of the base code of the $((1,1))_1$ TLDPC family

- is equal to $1 - (1-p)^2 < 1$ for $x = 1$ and $p < 1$,

- is always below the entropy curve of the base code of the (2,3) LDPC family (see Fig.5.4), which will imply the faster convergence of the iterative decoding for the TLDPC codes,

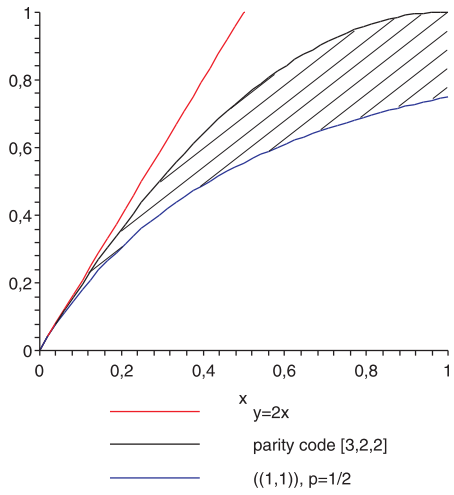


Figure 5.4: Entropy curves for base codes of the (2,3) LDPC code ensemble and of the $((1,1))_1$-TLDPC code ensemble and its entropy curves of variable nodes for $p = 1/2$.

- depends on the channel noise so that the area $\mathcal{A}$ between the entropy curves of the base code and of variable nodes gets larger when the channel noise decreases and that the area derivative $d\mathcal{A}/dp$ for TLDPC entropy curves is strictly greater than for LDPC entropy curves. Roughly speaking, this means that for channel noise values close to the threshold (waterfall region) the bottleneck between entropy curves for TLDPC ensemble is larger than for LDPC one which implies that the number of successful decodings for TLDPC codes is greater. This is demonstrated in Fig.5.5: not only the entropy curve of variable nodes moves away by enlarging the bottleneck (and which is also the case for LDPC code ensembles) but the entropy curve of the base code does the same.

5.3.6 Convergence of the (2,3) LDPC code ensemble and of the $((1,1))_1$-TLDPC code ensemble

In this subsection we demonstrate that the convergence of iterative decoding of binary TLDPC codes with bits of degree 1 is faster than the convergence for cycle codes.

To compare the convergence of a 2-regular LDPC code ensemble and a corresponding TLDPC code ensemble with bits of degree 1, we take as an example the (2,3) LDPC ensemble and the $((1,1))_1$ TLDPC code ensemble and we compute the number of iterations of the density evolution necessary for each of them to attain a fixed bit error probability P_b. The comparison is performed for p at which several dozens of iterations are needed for TLDPC decoding (in our examples $p = 0.6 \ldots 0.64$) and for p close to the iterative decoding threshold.

In Fig.5.6 the comparison is made for $m = 4$ and $P_b = 10^{-4}$. Recall that the threshold of both code ensembles is equal to 0.63684.

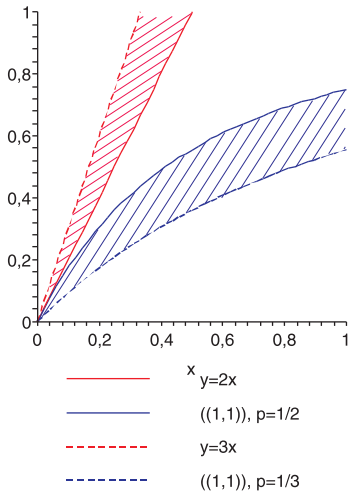


Figure 5.5: Entropy curves for the base code of the $((1,1))_1$-TLDPC code ensemble and their entropy curve of variable nodes for $p = 1/2$ and for $p = 1/3$.

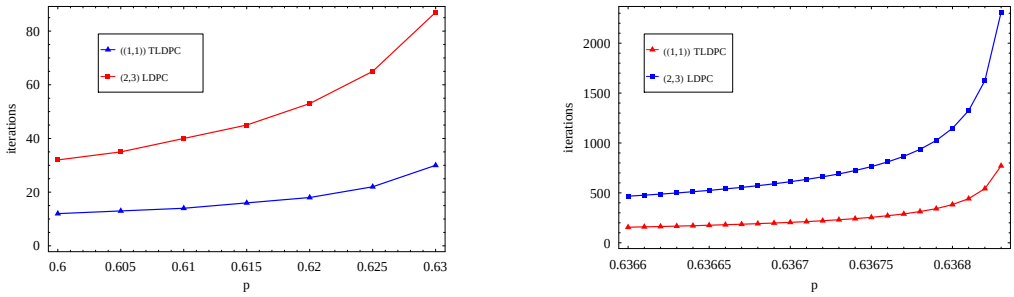


Figure 5.6: Average number of iterations necessary to attain the bit error probability $P_b = 10^{-4}$ as a function of the erasure channel probability p for the $((1,1))_1$ TLDPC and (2,3) LDPC code ensembles, $m = 4$.

In Fig.5.7 we compare two code ensembles for $m = 7$ and $P_b = 10^{-5}$. The threshold in this case is 0.64530. For $m = 4$ and $p = 0.6 \ldots 0.63$ the number of iterations of LDPC decoding is from 2.67 to 2.9 times greater than for TLDPC decoding. For $m = 7$ and $p = 0.6 \ldots 0.64$ the respective figures are 2.57 and 2.8. In both examples, for p close to the iterative decoding threshold the TLDPC code ensemble needs at least three times less iterations than the LDPC one. The smaller number of iterations for TLDPC codes is due to the fact that the entropy curve of the TLDPC base code is below the entropy curve of the LDPC base code and, moreover, the entropy curve of the TLDPC base code moves below with decreasing of the channel noise.

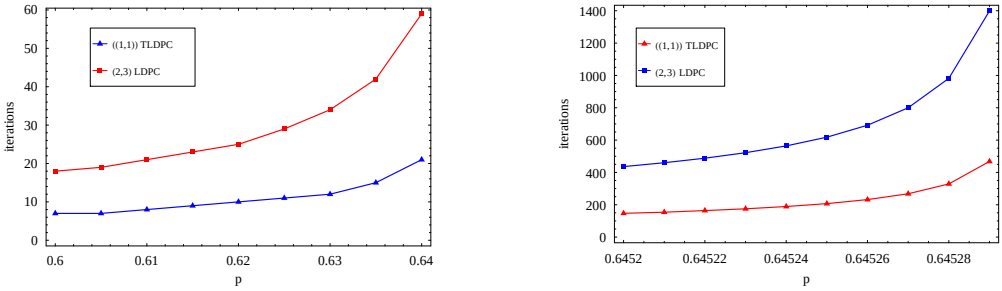


Figure 5.7: Average number of iterations necessary to attain the bit error probability $P_b = 10^{-5}$ as a function of the erasure channel probability p for the $((1,1))_1$ TLDPC and (2,3) LDPC code ensembles, $m = 7$.

5.3.7 $((1,b))_1$ TLDPC codes as irregular LDPC codes with a large fraction of symbols of degree 2

In this section we show that binary TLDPC codes with bits of degree 1 can be presented as binary LDPC codes which have at least two degree-2 bits per each parity-check equation. The same result holds in the non-binary case. The fact that the same code can be represented both in LDPC and TLDPC form gives us another decoding algorithm for irregular LDPC codes of this structure, they can be decoded like TLDPC codes and thus can have a faster convergence and a better performance in the waterfall region.

We have the following theorem:

Theorem 15 *Consider a binary $((1,R(x)))_1$ TLDPC code*[5] *with degree distribution of variable nodes $L(x) = \sum_{i=1}^{s} l_i x^{i-1}$, where $R(x) = \sum_{j=1}^{t} r_j x^{j-1}$, r_j is the fraction of cross-like sections of length j. Let the TLDPC base code be of length m and thus consists of $l_1 m$ local codes. Then it can be represented as an irregular LDPC code with respective degree distributions of variable nodes and of check nodes $L'(x) = \sum_{i=2}^{s} l'_i x^{i-1}$ and $R'(x) = \sum_{j=1}^{t} r_j x^{j+1}$, where $l'_2 = l_1 + l_2$, $l'_i = l_i$ for $i > 2$ and the Tanner graph of which contains a cycle formed by $l_1 m$ variable nodes of degree 2.*

Proof : Consider the tail-biting trellis of the $((1,R(x)))_1$ TLDPC base code. Let the parity of bits of a cross-like section of length j be P_j and let u_1 and u_2 be the bits of two parallel sections of length 1, neighboring with the given cross-like section. Then the following parity holds

$$u_1 + P_j = u_2.$$

By writing similar expressions for all cross-like trellis sections, we obtain a set of $l_1 m$ parity equations which define the TLDPC base code.

We associate a parity node to each parity equation and we construct the corresponding Tanner graph. Note that each of $l_1 m$ bits of degree 1 (the bits being carried by parallel sections) participate in exactly two parity equations, so their corresponding variable nodes have degree 2 and form a cycle of length $l_1 m$ in the Tanner graph. Any other bit of the TLDPC code of degree i, $2 \leq i \leq s$, participates in i different parity equations and thus, its corresponding variable node is of degree i and is connected to different parity nodes.

[5] by abuse of notation $R(x)$ denotes the distribution of cross-like sections of different lengths and not the length of cross-like sections

In other words, we obtained the Tanner graph of an irregular LDPC code with the degree distributions $L'(x)$ and $R'(x)$, which contains a cycle of length $l_1 m$ formed by variable nodes of degree 2 and all $l_1 m$ parity nodes. ∎

It can be easily proven that the inverse is hold: a binary irregular LDPC code with at least two bits of degree 2 per parity-check equation can be represented as a TLDPC code with bits of degree 1.

It is straightforward to show that the same result holds in the non-binary case.

Recall that an TLDPC code is decoded in two stages, one of which consists to calculate extrinsic messages of the symbols of the base code and another one consists to update intrinsic messages at variable nodes. The fact that an irregular LDPC code with a cycle with degree-2 variable nodes in its Tanner graph can be represented as a TLDPC code with symbols of degree 1 implies that the corresponding LDPC code can be also decoded in similar, TLDPC-like, way: first, the decoding on the cycle is performed and extrinsic messages for symbols not included in the cycle are calculated, then the updating of intrinsic messages of these symbols are performed at variable nodes [32].

For binary $(2, b+2)$-regular LDPC codes the following theorem holds:

Theorem 16 *The entropy curve of the base code of an $(2, b+2)$-regular LDPC code is always under the entropy curve of its corresponding $((1,b))_1$ TLDPC code with bits of degree 1.*

Proof I: t easy to show that the entropy curve of the base code of the $(2, b+2)$-regular LDPC code ensemble is described by

$$h(x) = 1 - (1-x)^{b+1}.$$

It can be also shown that the entropy curve of the $((1,b))_1$ TLDPC code ensemble is

$$g(x) = 1 - \left(1 - \frac{px\sum_{i=0}^{b-1}(1-x)^i}{1-p(1-x)^b}\right)^2,$$

$0 < x < 1$ and p is the erasure channel probability.

We have

$$(1-x)^{b+1} \leq \left(1 - p\frac{1-(1-x)^{b-1}}{1-p(1-x)^b}\right)^2 \leq \left(\frac{1-p}{1-p(1-x)^b}\right)^2 \leq \left(\frac{1-p}{1-p(1-x)^{b+1}}\right)^2.$$

By setting $y = (1-x)^{b+1}$ and solving the cubic equation, we obtain that $y \leq z$, where $z = \frac{1}{p}\left(1 - 0.5p + 0.5\sqrt{4p - 3p^2}\right)$. So, $b + 1 \geq \frac{\ln z}{\ln(1-x)}$. Note that $\ln z \geq 0$ for $0 \leq p \leq 1$ and $\ln(1-x) \leq 0$ for $0 \leq x \leq 1$. Thus, $b \geq C$ with $C \leq -1$. ∎

Theorem 16 implies better convergence and performances for the TLDPC code, the decoding of $(2, d)$-regular LDPC codes using cycles in its Tanner graph is always the best strategy of decoding to obtain better performances and convergence.

It is also curious to remark that thresholds of non-binary cycle codes decoded in LDPC- and turbo-like way coincide for any alphabet size while two decoding algorithms are different.

5.3.8 Performance of some families of non-binary TLDPC codes

We study the performances of $((2,2))$-TLDPC codes with $\Lambda(x) = x$ over two alphabets, $\mathbb{F}_4$ and $\mathbb{F}_8$. For the linear bijective mapping we have taken the multiplicative operation over $\mathbb{F}_q$ i.e. $x \mapsto g \cdot x$.

Over $\mathbb{F}_4$

For a trellis section of the $((2,2))$-TLDPC code let $(g_i x_i,\ g_{i+1}x_{i+1})$ be the pair of symbols on its branches, x_i being the symbols from variables nodes and g_i - the edge labels, i is the number of trellis section. Assume without loss of generality that $g_i = 1$.

With the aim to study performance of different $((2,2))$-TLDPC base codes, we study the performances of $((2,2))$-TLDPC codes as a function of the g_{i+1}'s. Note that the case $g_{i+1} = 1$ corresponds to the binary code of length twice the length of the code over $\mathbb{F}_4$.

We present the performance of codes of length 1000 and of rate 1/2 having different values of g_{i+1} and we compare it with the performances of the binary code of length 2000 and of rate 1/2 as if it were in the case $g_{i+1} = 1$. In our simulations g_{i+1} can be constant for every section or belongs to a pattern repeated periodically. We can see that in average there is $0.2 - 0.25$ dB of improvement in comparison with the binary code twice as long. The best $((2,2))$-TLDPC code over $\mathbb{F}_q$ seems to be the one with $g_{i+1} = 2$.

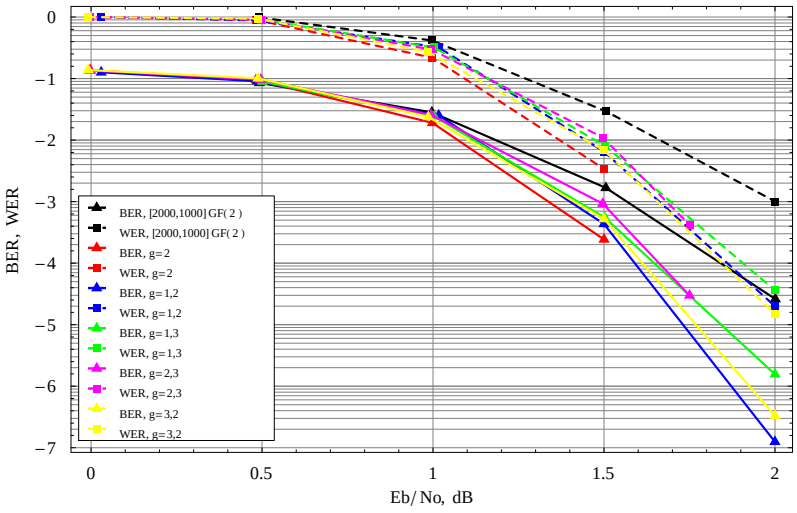


Figure 5.8: Performance of codes with parameters [1000,500] and $\Lambda(x) = x$ over $\mathbb{F}_4$ with different coefficients $g_{i+1} = g$ on trellis sections.

The maximum number of iterations was fixed to 100. However, the average number of iterations for a family-$((2,2))$ code of length 1000 and of rate 1/2 over $\mathbb{F}_4$ is 19 for $WER = 0.2034$ and 9 for $WER = 7.1 \cdot 10^{-3}$.

Then we plot WER vs. SNR for codes with parameters [1000,500], [672,336] and [502,252] over $\mathbb{F}_4$, $g_{i+1} = 2$ in Fig.5.9. We can see that the threshold of this family seems to be about 0.7 dB.

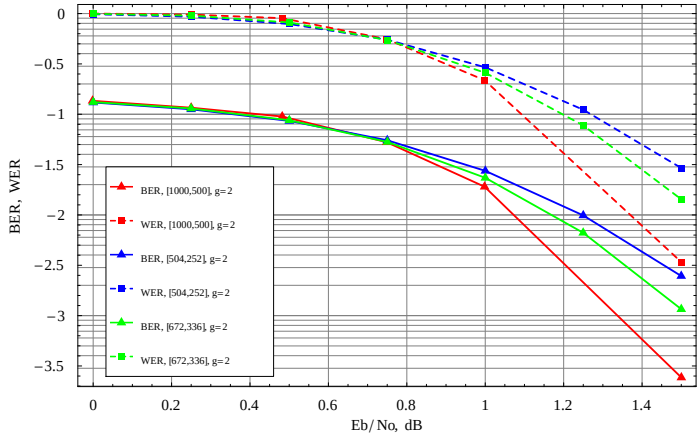


Figure 5.9: WER vs. SNR of codes [1000,500], [672,336] and [502,252] over $\mathbb{F}_q$, $g_{i+1} = 2$, $\Lambda(x) = x$.

In Fig. 5.10 we present the reference curves from [77] and we compare them with the performances of [504,252] over $\mathbb{F}_q$. We can see that in the waterfall region this simple $((2, 2))$-TLDPC code with only degree-2 variable nodes has a similar behaviour to the best irregular binary PEG code from [77]. It also outperforms by 0.4 dB the equal length binary code of Mackay as well as the PEG cycle code over $\mathbb{F}_4$.

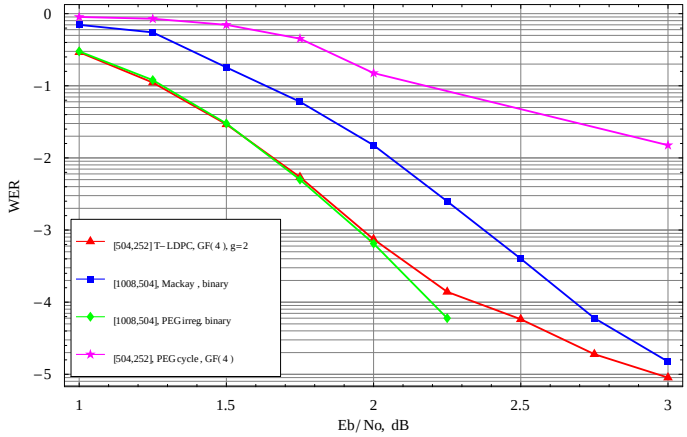


Figure 5.10: WER vs. SNR of the code [502,252] over $\mathbb{F}_q$, $g_{i+1} = 2$, $\Lambda(x) = x$ compared with the binary code of Mackay [1008,504], PEG cycle [504,252] code over $\mathbb{F}_q$ and PEG binary irregular code with parameters [1008,504].

Over $\mathbb{F}_8$

We present the performances of codes over $\mathbb{F}_8$ of rate 1/2 and of lengths 1000, 672, 336 and with $g_{i+1} = 2$ in Fig.5.11. We can see that the threshold is 0.75 dB.

In Fig.5.12 we compare the $((2, 2))$-TLDPC code [672,336] over $\mathbb{F}_8$ with the cycle

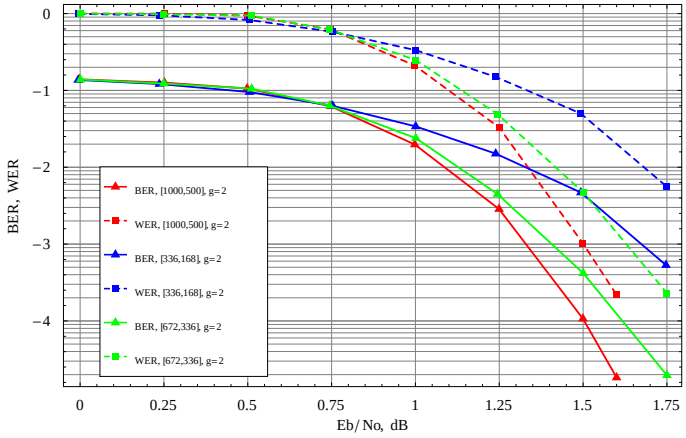


Figure 5.11: Performances of codes of rate 1/2 and of length 1000, 672 et 336 over $\mathbb{F}_q$, $\Lambda(x) = x$.

PEG code of the same parameters and with irregular binary PEG code [1008,504]. The performance of the 2-regular $((2,2))$-TLDPC code [336,168] constructed over $\mathbb{F}_8$ is only 0.05 dB worse in the waterfall region than performances of the irregular binary PEG code [1008,504] and of the irregular PEG code [336,168] over $\mathbb{F}_8$. Comparing the 2-regular

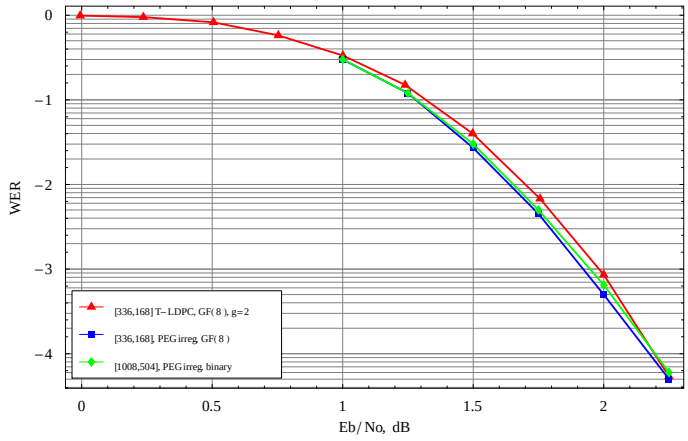


Figure 5.12: WER vs. SNR of the $((2,2))$-TLDPC code [672,336] with $\Lambda(x) = x$ and of the cycle [672,336] PEG code over $\mathbb{F}_8$.

$((2,2))$-TLDPC code [672,336] and the PEG cycle code [672,336] both constructed over $\mathbb{F}_8$ (see Fig.5.13), we observe that the performance of the $((2,2))$-TLDPC code is 0.8 dB better at $WER = 10^{-2}$, and, in contrast to the PEG cycle code, the $((2,2))$-TLDPC code does not show the error-floor till $WER = 2 \cdot 10^{-4}$.

We compare a $((2,2))$-TLDPC [336,168] code over $\mathbb{F}_8$ and a $((2,2))$-TLDPC code [504,252] over $\mathbb{F}_4$ in Fig.5.14, $\alpha_2 = 2$, $\Lambda(x) = x$. Performances of both codes in the waterfall region are similar, however, the code over $\mathbb{F}_8$ does not suffer from an error-floor which begins at $WER = 10^{-4}$ for the code over $\mathbb{F}_4$.

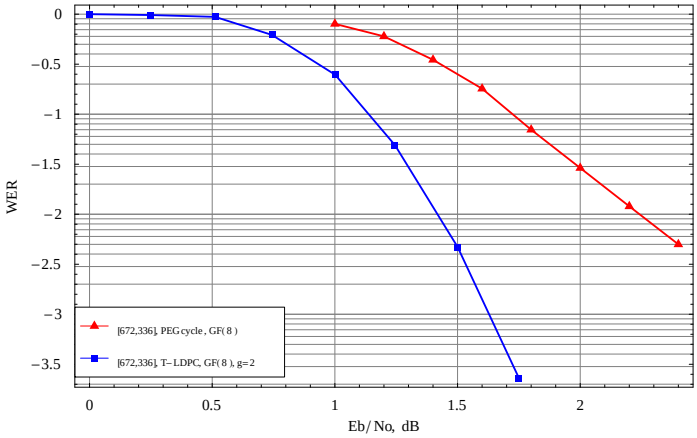


Figure 5.13: WER vs. SNR of the $((2,2))$-TLDPC [336,168] code over $\mathbb{F}_8$ with $\Lambda(x) = x$ compared to the irregular PEG [336,168] code over $\mathbb{F}_8$.

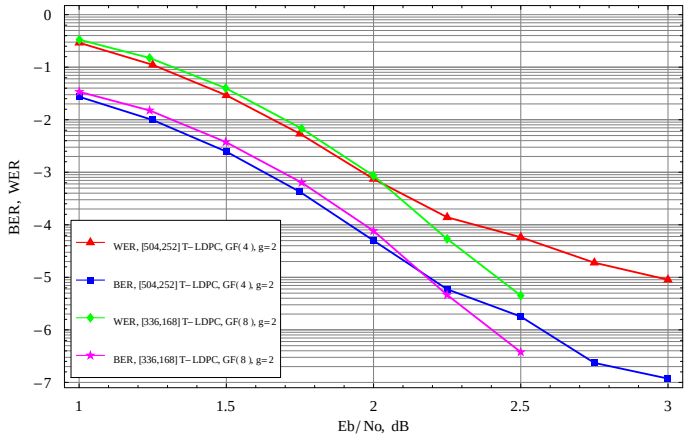


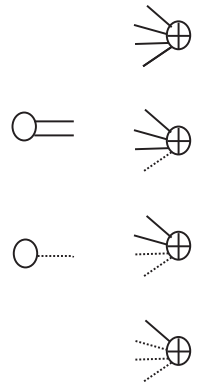
Figure 5.14: WER vs. SNR of codes $((2,2))$-TLDPC [336,168] over $\mathbb{F}_8$ and [504,252] over $\mathbb{F}_4$, $\Lambda(x) = x$.

5.4 Analysis of LDPC codes over $\mathbb{F}_q$ having a small constant fraction of symbols of degree 1

We study an ensemble of family of non-binary LDPC codes defined over $\mathbb{F}_q$ with a small constant fraction of symbols of degree 1. We suppose $q = 2^m$ and we consider that a linear bijective mapping f: $\mathbb{F}_q \mapsto \mathbb{F}_q$ is associated to each edge of the corresponding bipartite graph.

Consider a non-binary LDPC code ensemble defined by left and right degree distributions $\lambda(x) = \lambda_1 + (1 - \lambda_1)x$ and $\rho(x) = x^{d-1}$. Note that the fraction λ_1 of edges of degree 1 is non-zero. The minimum distance of such a code ensemble is typically constant since there is typically a constant number of variable nodes of degree 1 and any path in the q-ary Tanner graph connecting two variable nodes of degree 1 between them is a valid configuration of a codeword. However, the interest of this code ensemble is that on large fields the iterative decoding threshold for a given bit erasure probability (say, of order $10^{-3} \dots 10^{-4}$) is better than the threshold of the cycle code family with $\omega \in \mathbb{F}_{2^m} \setminus \{0\}$ from [55].

Note that if the permutation is chosen at random in this case, there will be λ_1^d check nodes connected to variable nodes of degree 1 only, so messages of the variable nodes will not change during decoding process. To avoid such configurations one need to make a structured choice of permutation. This gives us a multi-edge structure with two types of edges, those connected to variable nodes of degree 1 and those connected to variable nodes of degree 2. In the figure at right we present an example of the structure for $d = 4$. Circles in the figure are variable nodes and $\oplus$ are check nodes. Edges of two types are showed by dotted and straight lines correspondingly.



Thus, check nodes are of d types, the check node of type i being connected to i variable nodes of degree 2. We choose the proportion a_i of check nodes of type i to be

$$a_i = \frac{\binom{d}{i}\lambda_1^{d-j}(1-\lambda_1)^j}{1-\lambda_1^d}$$

which corresponds to the expected number of check nodes of degree i in the case when the edge matching is chosen at random and the configurations of check nodes connected to d variable nodes of degree 1 are forbidden. Proportions of variable nodes of degree 1 and 2 are $\frac{2\lambda_1}{1+\lambda_1}$ and $\frac{1-\lambda_1}{1+\lambda_1}$.

We consider the density evolution for the multi-edge code ensemble. Assume the transmission over binary erasure channel with erasure probability p and perform a belief propagation decoding at the receiver end. During an iteration t we compute extrinsic messages $E^{(t)}$ of symbols and then we update intrinsic ones $I^{(t)}$. We note that the messages in the belief propagation decoder are vectors of length q. For a given symbol i of the base code we denote the k-th component of the message $E_i^{(t)}$ ($I_i^{(t)}$) as $E_i^{(t)}(k)$ ($I_i^{(t)}(k)$), $k = 0, \dots, q-1$, and this component is equal to the posteriori probability that the symbol is k.

5.4.1 Code rate as a function of λ_1

For a non-binary LDPC code ensemble with a fraction λ_1 of symbols of degree 1, a fraction $\lambda_2 = 1-\lambda_1$ of symbols of degree 2 and with edge labels taken at random from $\{0, \ldots, 2^m - 1\}$, we define left and right degree distributions $\lambda(x)$ and $\rho(x)$ as following:

$$\begin{aligned} \lambda(x) &= \lambda_1 + (1-\lambda_1)x \\ \rho(x) &= x^{d-1} \end{aligned}$$

So, the design rate of the code ensemble is a function of λ_1 and is equal

$$R = 1 - \frac{2}{d(\lambda_1 + 1)}.$$

5.4.2 Density evolution equations

The transmission is made over the BEC with the erasure probability p, initial intrinsic messages probability vector $\iota^{(0)}$ is equal to

$$\iota^{(0)}(k) = \binom{m}{k} p^k (1-p)^{m-k}, \quad 0 \leq k \leq m.$$

Lemma 24 *For a non-binary LDPC code with a constant fraction λ_1 of symbols of degree 1, a fraction $\lambda_2 = 1 - \lambda_1$ of symbols of degree 2 and the constant degree d of parity nodes, the average extrinsic probability vector $\epsilon^{(t)}$ and the average intrinsic probability vector $\iota^{(t)}$ at the t-th iteration are computed as follows*

$$\begin{aligned} \epsilon^{(t)} &= \frac{d+1}{2(1-\lambda_1^d)} \sum_{j=1}^{d} \frac{j}{d} \binom{d}{j} \lambda_1^{d-j} (1-\lambda_1)^j \, (\boxtimes^{j-1} \iota^{(t-1)}) \boxtimes (\boxtimes^{d-j} \iota^{(0)}), \\ \iota^{(t)} &= \epsilon^{(t)} \boxdot \iota^{(0)}, \end{aligned}$$

where $\boxtimes^j g = \underbrace{g \boxtimes g \ldots \boxtimes g}_{j \text{ times}}$.

The a posteriori probability vector $\hat{\iota}^{(l)}$ after l iterations is

$$\hat{\iota}^{(l)} = \epsilon^{(l)} \boxdot \epsilon^{(l)} \boxdot \iota^{(0)}.$$

Recall that the i-th element of $\hat{\iota}^{(l)}$ denotes the probability that the a posteriori message of m bits has k non-zero bits after iterative decoding. Having $\hat{\iota}^{(l)}$, the bit erasure probability after l decoding iterations is given by

$$P_e^{(l)} = \frac{1}{m} \sum_{i=1}^{m} n_m(i) \hat{\iota}^{(l)}(i),$$

where $n_m(i)$ is the average number of erased bits in a symbol of m bits given that it belongs to a random space of dimension i,

$$n_m(i) = \sum_{j=i}^{m} \frac{b_m(i,j)}{a_m(i)} (m-j),$$

$a_m(i)$ being the total number of vector subspaces of dimension i of a vector space of dimension m, $a_m(i) = \begin{bmatrix} m \\ i \end{bmatrix}_2$, and $b_m(i,j)$ being the number of vector subspaces of dimension i with support of size j of a vector space of dimension m.

Below we give several examples for $n_m(i)$.

Example

For $i = 1$, $a_m(1) = 2^m - 1$ and $b_m(1,j) = \binom{m}{j}$. Thus, we have

$$n_m(i) = \frac{m}{2}(1 - \frac{1}{2^m - 1}).$$

◇

Example

For $i = 2$, $a_m(2) = \begin{bmatrix} m \\ 2 \end{bmatrix}_2$ and

$$b_m(2,j) = \frac{\sum_{i=1}^{j-1} \binom{m}{i}\binom{m-i}{j-i}2^i + \binom{m}{j}(2^j - 2)}{(2^2-1)(2^2-2)}.$$

We obtain that

$$n_m(2) = m\frac{4^{m-1} + 2m}{4^m - 3 \cdot 2^m + 2}.$$

◇

5.4.3 Obtained thresholds

As $n_m(i) \leq m$, we can denote

$$\hat{P}_e = \frac{\hat{\iota}(1)}{2}(1 - \frac{1}{2^m - 1}) + \frac{4^{m-1} + 2m}{4^m - 3 \cdot 2^m + 2}\hat{\iota}(2) + \sum_{i=3}^{m} \hat{\iota}(i)$$

to be an upper bound on the bit erasure probability after iterative decoding. This bound is often tight as in practice the values of $\hat{\iota}(i)$ decay quickly with i.

In this section we compute iterative decoding thresholds of modified cycle codes for different values of λ_1 and for $\hat{P}_e = 10^{-3}$.

In Fig.5.15 we present an example of how the iterative decoding threshold of an LDPC code family changes with the fraction of symbols of degree 1. In this example the LDPC code family is over $\mathbb{F}_2^7$ and has degree distributions $\lambda(x) = \lambda_1 + (1 - \lambda_1)x$ and $\rho(x) = x^3$. We see that the iterative decoding threshold p^{th} decreases gradually up to some critical value of λ_1 and than goes down rapidly. Compared to the maximum achievable threshold p^{Sh} as a function of λ_1 (and R [6]), p^{th} is more closer to p^{Sh} for the critical value of λ_1.

In Table 5.4 we present rates R and iterative decoding thresholds p^{th} of non-binary LDPC codes over $\mathbb{F}_2^5$, $\mathbb{F}_2^6$ and $\mathbb{F}_2^7$ for their corresponding critical values of λ_1. Presented code families have $d = 3$ and $d = 4$.

[6] see Appendix E.1

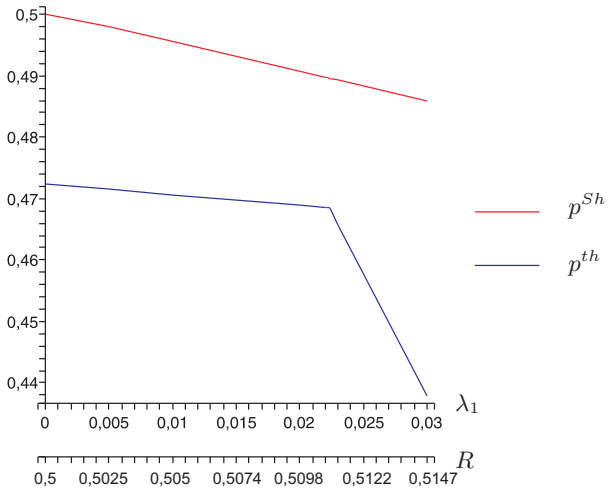


Figure 5.15: Thresholds of non-binary LDPC codes with $\lambda(x) = \lambda_1 + (1-\lambda_1)x$ and $\rho(x) = x^3$ for $\hat{P}_e = 10^{-3}$ and over $\mathbb{F}_2^7$ as a function of λ_1.

$d = 3$

m	λ_1	R	p^{th}	p^{Sh}
5	0.0055	0.337	0.6423	0.6644
6	0.0075	0.3383	0.6442	0.662
7	0.009	0.3393	0.6432	0.6611

$d = 4$

m	λ_1	R	p^{th}	p^{Sh}
5	0.0122	0.506	0.4708	0.4945
6	0.0173	0.5085	0.471	0.492
7	0.0223	0.5112	0.4685	0.4896
8	0.0273	0.5133	0.4647	0.467
9	0.0332	0.5161	0.4602	0.4626

Table 5.4: Thresholds p^{th} of non-binary LDPC codes with $\lambda(x) = \lambda_1 + (1-\lambda_1)x$ and $\rho(x) = x^{d-1}$ for $\hat{P}_e = 10^{-3}$ over $\mathbb{F}_2^m$.

Chapter 6

Conclusions and perspectives

6.1 Conclusions

This thesis is a contribution on how close we can approach the channel capacity with asymptotically good code ensembles which is known to be a non-trivial problem [53].

We focus on the construction of a code family having good performances in both high and low SNR regions under iterative decoding of low complexity. Our approach for this purpose is to construct a family of asymptotically good codes with large fraction of bits (symbols) of degree 2. We have also tried to include bits (symbols) of degree 1 which seems to improve the performances in the waterfall region and decrease the number of decoding iterations.

We begin with a general construction which covers both turbo codes and LDPC codes and we optimise its parameters in order to obtain good performances under iterative decoding. The structure of the code family which has been studied is defined using two components: a base code and a bipartite graph. In the particular case when the base code of a code family is represented by a tail-biting trellis, it is called Tail-biting LDPC (TLDPC) code family.

There are two constraints to be set if we want to construct a family of asymptotically good codes:

- a special graph called the graph of codewords of (partial) weight 2 must have no cycles of sublinear size,

- one has to pay a special attention to the number of codewords of weight 2 and 3 in the base code.

For one of presented TLDPC codes families (namely, family A) the property of being asymptotically good was proven under condition that its corresponding graph of codewords of weight 2 has no cycles. All the other presented code families are conjectured to be asymptotically good.

Binary TLDPC codes have a low complexity of iterative decoding due to their simple base code and a large fraction of degree-2 variable nodes in their structure. Iterative decoding thresholds of TLDPC codes with bits of degree 1, the degree distributions of which were optimised for rates 1/3 on the Gaussian channel, are located at most 0.2 dB from the channel capacity. The performances of TLDPC codes at rates 1/3 and 1/2 beat

the ones of standard turbo-codes for code lengths of several thousand and are comparable with the performances of multi-edge LDPC codes.

It is a difficult task to construct LDPC codes of low rates. The proposed TLDPC code of rate 1/10 has high performances under iterative decoding and is one of the best low-rate code presented by now.

An error-floor region was observed for one of the TLDPC codes at $WER = 10^{-6}$. It was found out that it is due to trapping sets in its structure. The study showed that typical trapping configurations have more complicated structure that configurations for LDPC codes, and that they also have greater size. A method to estimate the error-floor of TLDPC codes with the help of typical trapping configurations was proposed.

We also explored non-binary TLDPC codes with and without symbols of degree 1. We calculated the iterative decoding thresholds of several code ensembles using density evolution.

We noticed that any irregular LDPC code, binary or not, having at least two symbols of degree 2 in each parity-check equation can be represented as a TLDPC code with a non-zero fraction of symbols of degree 1. Such a TLDPC representation gives us another way of decoding of the LDPC codes when their parity checks nodes in the Tanner graph can be arranged in cycles containing only degree-2 variable nodes. The advantage of the new decoding algorithm is its very fast convergence. Typically, it requires from 2 to 3 times less iterations than a standard decoding algorithm for LDPC codes.

We also noticed that the thresholds of some non-binary TLDPC code families coincide with thresholds of cycle codes for different alphabets.

We present the performances of several TLDPC codes over $\mathbb{F}_4$ and $\mathbb{F}_8$. These codes, having a very simple structure, have very good performances and have a significantly steeper waterfall region in comparison to binary codes.

We study iterative decoding thresholds of modified non-binary cycle codes with a small constant fraction of symbols of degree 1. The bit error probability of such codes is kept below some fixed value, say 10^{-3}. The presence of symbols of degree 1 increases the rate of the code ensemble. Even if the minimum distance of modified cycle codes is constant, their iterative decoding threshold becomes closer to the theoretical limit. For example, for modified (2,4) cycle codes over $\mathbb{F}_{64}$ with the fraction of symbols of degree 1 equal to 0.0173, the rate of the code ensemble is 0.5085 the obtained iterative decoding threshold is 0.021 from the theoretical limit.

6.2 Future work

In continuation of this work, there are a number of problems that can be the subject of future research. Here is a short list of some of the possible directions.

- We explored the code families of rates $1/10 - 1/2$. It remains to investigate TLDPC code families of higher rates.

- We did not discuss the encoding of TLDPC codes. It is easy to show that the codes of family A with $\lambda_2 = 1$ are linearly encodable. It is interesting to study if the linear encoding is possible for other TLDPC code families.

- To find simple criteria of being asymptotically good for codes with more complicated structures than LDPC codes.
- To show formally the impact of bits of degree 1 in the structure of a code.
- To show formally the impact of larger alphabets on the code performances.
- To construct good codes with symbols of degree 1 over larger alphabets.

Appendix A

A.1 APP modified decoding algorithm

The Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [9, 75], is used to decode a code of length n when it is presented by a trellis. The input of the algorithm is conditional probabilities $p(y_i|x_i)$, $0 \leq i \leq n-1$, computed at the channel output, x_i represents the value of the i-th bit sent to the channel and y_i represents the corresponding received value. The output of the algorithm is A Posteriori Probabilities of bits. Because of it the algorithm is also called APP.

In this section we propose a modification of the algorithm which outputs the extrinsic probabilities $p_i = \mathbf{P}(x_i|y_0, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_{n_1})$ of the i-th bit, $0 \leq i \leq n-1$. We also suppose that the trellis of a given code is tail-biting with constant number of states and that each trellis section contains n_b bits, $n_b > 1$. Let $b_i = \left\{x_i^1, \ldots, x_i^{n_b}\right\}$ to be the ensemble of bits of the i-th section.

We define the variables used during the algorithm description:

Input variables:
n_b: number of bits in a trellis section;
$n_{sections} = \frac{n}{n_b}$: number of trellis sections;
n_{states}: number of trellis states;
$p(y_i^j|x_i^j)$: conditional probability of the j-th bit of the i-th trellis section on the channel output, $j = 1, \ldots, n_b$, $i = 1, \ldots, n_{sections}$;
$q_i(m, m^{'})$: 1 if for the i-th section there exists the transition between states m and $m^{'}$, 0 otherwise, $m = 1, \ldots, n_{states}$, $m' = 1, \ldots, n_{states}$.

Output variables:
$\sigma_{u,i}^j(m, m')$: extrinsic probability of the j-th bit for the transition $m \to m'$ in the i-th trellis section given the initial trellis state u;
p_i^j: extrinsic probability of the bit x_i^j.

Intermediate variables:
j: number of the bit within the section, $j = 1, \ldots, n_b$;
i: section number, $i = 1, \ldots, n_{sections}$;
S_i: trellis state at the i-th section output;
$\mathbf{x}_i = (x_i^1, \ldots, x_i^{n_b})$: ensemble of bits of the i-th trellis section;
$\mathbf{x}_i^j = (x_i^1, \ldots, x_i^{j-1}, x_i^j, \ldots, x_i^{n_b})$: ensemble of bits of the i-th trellis section excluding the

j-th bit;
$\gamma_i(m, m^{'}) = \mathbf{P}(S_i = m^{'}, \mathbf{x}_i | Si - 1 = m)$: $m \to m'$ transition probability of the i-th trellis section;
$\bar{\gamma}_i^j(m, m^{'}) = \mathbf{P}(S_i = m^{'}, \mathbf{x}_i^j | S_{i-1} = m)$: extrinsic probability of the transition $m \to m'$ for the i-th trellis section;
$\alpha_{u,i}(m) = \mathbf{P}(S_i = m | \{\mathbf{x}_k\}_{k=1}^i, S_0 = u)$: probability of the state m given observations from 1 to i and the initial trellis state u;
$\beta_{u,i}(m) = \mathbf{P}(S_i = m | \{\mathbf{x}_k\}_{k=i+1}^{N-1}, S_{n_{sections}} = u)$: probability of the state m given observations from $i+1$ to $N-1$ and the final trellis state u.

The algorithm proceeds in four stages: on the first stage initial values of α and β are fixed. As the trellis is tail-biting, $S_0 = S_{n_{sections}}$.

A principal part of the BCJR algorithm contains forward and backward procedures to compute α and β. We do this using transition probabilities γ. On the next stage, *sigma* are evaluated using corresponding probabilities α and β and extrinsic transition probabilities $\bar{\gamma}$. Finally, bit extrinsic probabilities are computed.

The operations performed during the stages of the modified APP algorithm are presented below:

Initialisation

$$\alpha_{u,i}(m) = \begin{cases} 1 & \text{if } m = 0 \text{ and } i = u, \\ 0 & \text{otherwise.} \end{cases} \qquad u = 1 \dots n_{states}$$

$$\beta_{u,i}(m') = \begin{cases} 1 & \text{if } m' = n_{sections} - 1 \text{ and } i = u, \\ 0 & \text{otherwise.} \end{cases} \qquad u = 1 \dots n_{states}$$

Transition probabilities computation:

$\gamma_i(m, m') = q_i(m, m') \prod_{k=1}^{n_b} P(y_i^k | x_i^k)$
$\bar{\gamma}_i^j(m, m') = q_i(m, m') \prod_{k=1, k \neq j}^{n_b} P(y_i^k | x_i^k)$

Forward procedure

$\alpha_{u,i}(m') = \sum_m \gamma_i(m, m') \alpha_{u,i-1}(m)$ $\qquad i = 1 \dots n_{sections}$ $\qquad u = 1 \dots n_{states}$

Backward procedure

$\beta_{u,i}(m) = \sum_{m'} \gamma_{i+1}(m, m') \beta_{u,i+1}(m')$ $\qquad i = n_{sections} - 1 \dots 0$ $\qquad u = 1 \dots n_{states}$

Computation of $\sigma_{u,i}^j(m, m')$

$\sigma_{u,i}^j(m, m') = \alpha_{u,i-1}(m) \bar{\gamma}_i^j(m, m') \beta_{u,i}(m')$

Extrinsic probabilities computation

$p_i^j = \sum_{u,m,m' | x_i^j} \sigma_{u,i}^j(m, m')$ $\qquad u, m, m' = 1, \dots, n_{states}$

Appendix B

B.1 Optimisation of the degree distribution $\Lambda(x)$ to maximise the iterative decoding threshold

It is well known that such irregular codes may perform much better with respect to iterative decoding than their regular counterparts. This holds for LDPC codes [47, 48, 57, 22] as well as for other families of graph-sparse codes such as TLDPC codes, turbo-codes or for repeat-accumulate codes [42, 34, 15]. We discuss the $\Lambda(x)$ optimisation of the general construction (see Section 3.2). Two principal points of the optimisation are

1. To fix the level of noise and to maximise the code rate of given code ensemble as a function of λ_i.

2. Such an optimisation can be written as a linear programming problem.

To understand the second point, let us consider that the entropy curve of the base code can be described by the function $y = f(x)$ and that different entropy curves associated to degree distributions of constant degrees $\Lambda_d(x) \stackrel{\text{def}}{=} x^{d-1}$ can be described by equation $x = g_i(y)$. So, the entropy curve associated to the degree distribution $\Lambda(x) = \sum_i \lambda_i x^{i-1}$ can be expressed as $x = \sum_i \lambda_i g_i(y)$. We make a hypothesis that the iterative decoding converges with the probability tending to 1 with the codelength if and only if the entropy curve of variable nodes is above the entropy curve of the base code. From this it follows that when the iterative decoding converges, we have (see Fig.B.1)

$$\forall x \in [0,1] \quad \sum_i \lambda_i g_i(f(x)) \leq x \tag{B.1}$$

The maximum code rate is equal to

$$R = 1 + \frac{R_b - 1}{\sum_i \frac{\lambda_i}{i}}$$

with R_b to be the rate of the base code. Thus, we want to maximise the quantity

$$\rho(\Lambda) \stackrel{\text{def}}{=} \sum_i \frac{\lambda_i}{i}.$$

This constitutes a linear programming problem in infinite space.

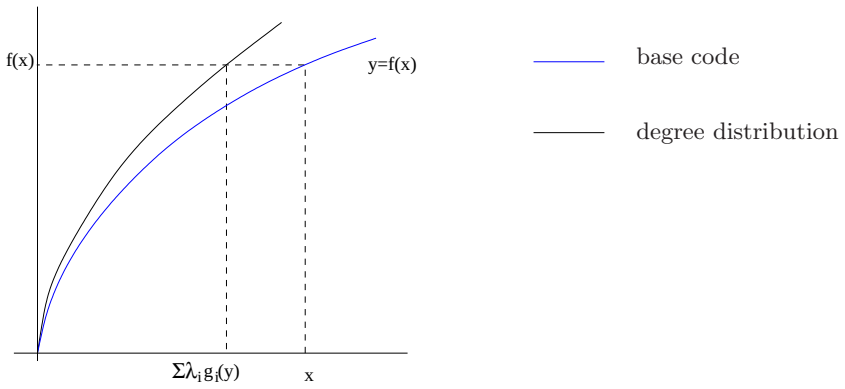


Figure B.1: Illustration of inequality (B.1)

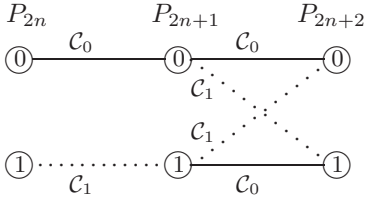
In general, the entropy curve of the base code will be obtained experimentally and we will only have in our disposition an approximation f_{approx} of the function $f(x)$ for a small number E of discrete values. So, we have to solve the following problem

$$\begin{array}{ll} & \text{Maximise} \quad \sum_i \frac{\lambda_i}{i} \qquad\qquad \text{given} \\ \forall x \in E & \sum_i \lambda_i g_i(f_{\text{approx}}(x)) \leq x. \end{array}$$

B.2 Optimisation of degree distributions for the binary erasure channel

B.2.1 Entropy curves of base codes for families A and B

Both local codes of families A and B can be presented by two trellis sections, parallel-like and cross-like, each of length 2 and 3 respectively. Denote the length of one trellis section by a. Thus, we have $m/(2a)$ local codes if the base code of family A or B is of length m. Consider n-th local code in the base code (see Fig.B.2.1).



A: $a = 2$, $\mathcal{C}_0 = \{00, 11\}$, $\mathcal{C}_1 = \{10, 01\}$

B: $a = 3$, $\mathcal{C}_0 = \{000, 011, 101, 110\}$, $\mathcal{C}_1 = \{001, 010, 100, 111\}$

Figure B.2: n-th local code of the base code of the family A or B.

Let x the bit erasure probability, so the probability p that all the bits of a section are erased is equal to $p = 1 - (1 - x)^a$. In the same way, the probability q that for a given bit all its neighbour bits in the section are erased is $q = 1 - (1 - x)^{a-1}$. For a local code n denote the pair of probabilities $\{P_{2n+1}, P_{2n+2}\}$, $n = 0, 1, \ldots, m/(2a) - 1$, where P_i is the state erasure probability on the output of the section $i - 1$ of the base code.

Using BCJR decoding procedure on the trellis of the base code, we compute analytical expressions for extrinsic probabilities of bits of the base code. On the forward stage we

obtain that

$$\begin{cases} P^{(f)}_{2n+1} = pP_{2n} \\ P^{(f)}_{2n+2} = (1-p)P_{2n+1} + p \end{cases} \qquad P_0 = 0,\ n = 0, 1, \ldots, \frac{m}{2a} - 1.$$

We can rewrite these expressions as follows

$$\begin{cases} P^{(f)}_{2n+2} = p(1-p)P_{2n} + p & n = 0, 1, \ldots, \frac{m}{2a} - 1 \\ P^{(f)}_{2n+3} = p(1-p)P_{2n+1} + p^2 & n = 0, 1, \ldots, \frac{m}{2a} - 1 \end{cases}$$

Note that $P^{(f)}_{2n+2} = p(1-p)P_{2n} + p$ is a linear recursion of the first order with the zero first coefficient. To find the solution of the recursion by using, for example, Theorem 2.1 of [63], we obtain that

$$P^{(f)}_{2n+2} = p \sum_{1 \le i \le n+1} [p(1-p)]^{n+1-i} = p_1,$$

where we denote $p_1 = p\frac{1-[p(1-p)]^{n+1}}{1-p(1-p)}$. Similarly for $P^{(f)}_{2n+3}$ we get

$$P^{(f)}_{2n+3} = p^2 \sum_{1 \le i \le n+1} [p(1-p)]^{n+1-i} = p_2,$$

with $p_2 = p^2\frac{1-[p(1-p)]^{n+1}}{1-p(1-p)}$.

In the same way we obtain for the backward stage

$$\begin{cases} P^{(b)}_{2n+2} = p_2, & n = 1, \ldots, \frac{m}{2a} - 1 \\ P^{(b)}_{2n+3} = p_1 & n = 1, \ldots, \frac{m}{2a} - 1 \end{cases}$$

Let $E_{2n,2n+1}$ and $E_{2n+1,2n+2}$ be bit extrinsic probabilities for the $n+1$-th local code. Then

$$E_{2n,2n+1} = 1 - (1-p_1)^2(1-q) - 2p_1(1-p_1)(1-q) = p_1^2 + (1-p_1^2)q,$$
$$E_{2n+1,2n+2} = 1 - (1-p_2)^2(1-q) = 2p_2 - p_2^2 + (1-p_2)^2 q.$$

Thus, the average extrinsic probability of the base code is equal to

$$\bar{E} = \frac{1}{m}\sum_{i=0}^{m} E_i = \frac{E_{2n,2n+1} + E_{2n+1,2n+2}}{2} = q\left[1 - p_1^2 + (1-p_2)^2\right] - (1-p_1^2) + (1-p_2)^2.$$

B.3 Comparison of different types of channel

To compare iterative decoding performance for different types of channel, we use the following assumption: iterative decoding performance on different types of channel are similar for the same value of the entropy of noise. The entropy of noise is defined by $\mathbf{Ent}(X)$, X being the probability of a bit to be equal to 0 where the bit was transmitted over the associated channel with the corresponding noise level.

As an example, the following table presents values of the channel parameter for different types of channel (BEC, BSC, Gaussian channel) for the code rate 1/2 which give the same entropy of noise.

BEC p_e	BSC p_s	Gauss. SNR, dB
0.5	0.11003	0.1871
0.4914	0.1072	0.3
0.4838	0.1047	0.4
0.4761	0.1022	0.5
0.4684	0.098	0.6
0.4607	0.0974	0.7
0.4529	0.095	0.8
0.4451	0.0926	0.9
0.4372	0.0902	1.0
0.4294	0.0879	1.1
0.4214	0.0856	1.2
0.4135	0.0833	1.3

Note that the dependence between two of these parameters is quasi-linear in the large range of values, as shown in the following figures:

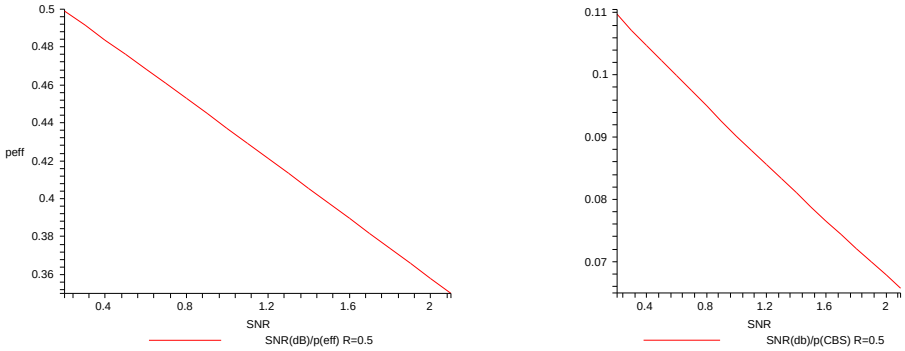


Figure B.3: Erasure probability (left) and error probability of the BSC (right) as functions of the SNR (in dB) of the Gaussian channel

We present another table in which we show limit values of noise which can be corrected with probability $\to 1$ when the codelength $n \to \infty$ (Shannon limit) for different code rates:

Code rate	4/5	3/4	2/3	3/5	1/2	1/3
p_e	0.2	0.25	1/3	0.4	0.5	2/3
p_s	0.0311	0.042	0.0615	0.0794	0.11003	0.1740
SNR (dB)	2.0400	1.6264	1.0595	0.6787	0.1871	-0.4954

Appendix C

C.1 Trapping set configurations observed during simulations

In the appendix we present structures of graphs of type 3 and 4 for trapping sets of the family E code of length 15000 and of rate 1/3. We use the following graphical representation. Codewords of the base code of weight 3 are depicted by circles and codewords of the base code of weight 3 - by squares. Variable nodes of degrees greater than 2 are denoted by filled circles. Codewords and variable nodes are connected by labelled edges. An edge with label k between two codewords exists if in the the corresponding graph the codewords are connected through exactly k variable nodes of degree 2 and $k-1$ clusters. Similarly, An edge with label k connects a codeword and a variable node of degree > 2 if in the the corresponding graph they are connected through exactly k variable nodes of degree 2 and k clusters.

We depict unsatisfied connections of codewords and of variable nodes by red arrows. If, for example, a codeword of weight 3 has a red arrow, it represents a near-codewords of weight 3 - a word of weight 2 which is not a codeword, but there is such a position in the base code that if we put it to 1 we get a codeword of weight 3.

The number of unsatisfied connections is equal exactly to b.

C.2 Low-weight codewords

During decoding simulations some low-weight codewords were detected. In this part we represent graphically their structure.

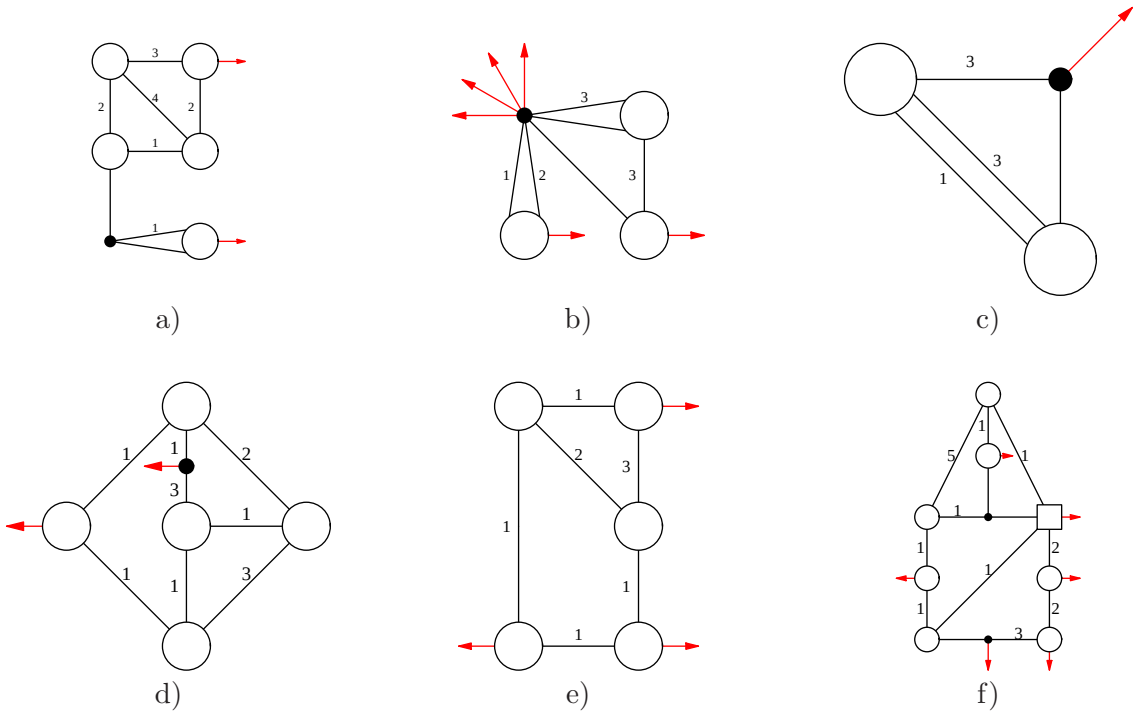


Figure C.1: Examples of trapping set configurations.

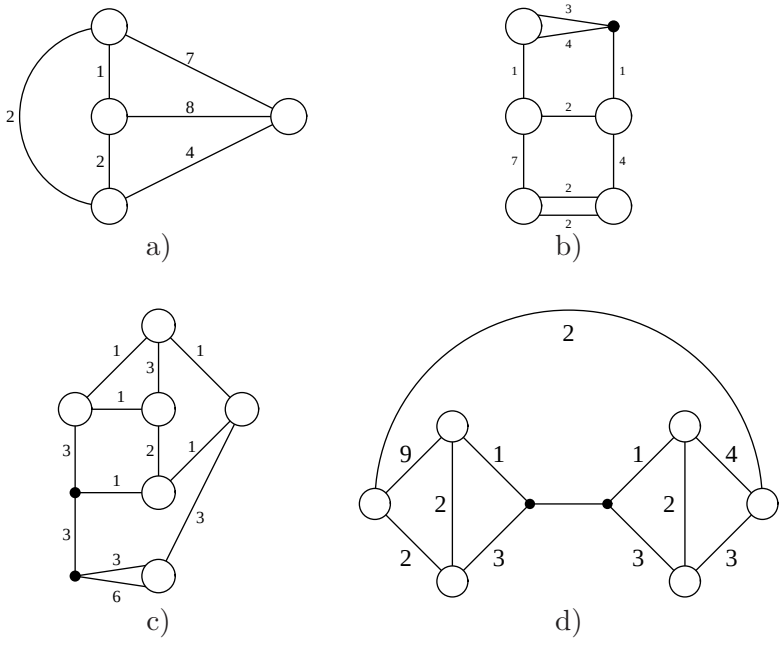


Figure C.2: Examples of codewords of weight a) 24; b) 27; c) 30; d) 35.

Appendix D

D.1 Properties of ⊠ and ⊡ operations

1. Commutativity:

   Using Property 5.4, we show the symmetry of $C_{\boxdot}(m,k,i,j)$ and $C_{\boxtimes}(m,k,i,j)$:

$$C_{\boxdot}(m,k,i,j) = \frac{[i]_2\,[m-i]_2\,[j]_2\,[m-j]_2}{[k]_2\,[i-k]_2\,[j-k]_2\,[m-i-j+k]_2\,[m]_2} 2^{(i-k)(j-k)} = C_{\boxdot}(m,k,j,i),$$
$$C_{\boxtimes}(m,k,i,j) = \frac{[m-i]_2\,[i]_2\,[m-j]_2\,[j]_2}{[m-k]_2\,[k-i]_2\,[k-j]_2\,[i+j-k]_2\,[m]_2} 2^{(k-i)(k-j)} = C_{\boxtimes}(m,k,j,i).$$

$$[a \boxdot b]_k = \sum_{i=k}^{m} \sum_{j=k}^{k+m-i} C_{\boxdot}(m,k,i,j) a_i b_j = \sum_{j=k}^{m} \sum_{i=k}^{k+m-j} C_{\boxdot}(m,k,j,i) b_j a_i = [b \boxdot a]_k\,,$$
$$[a \boxtimes b]_k = \sum_{i=0}^{k} \sum_{j=k-i}^{k} C_{\boxtimes}(m,k,i,j) a_i b_j = \sum_{j=0}^{k} \sum_{i=k-j}^{k} C_{\boxtimes}(m,k,j,i) b_j a_i = [b \boxtimes a]_k\,.$$

D.2 Computation of (5.8) in Lemma 19

$$\begin{aligned}
\xi^f_{2l}(i) &= \sum_{u=0}^{i} \left(\sum_{v=i-u}^{i} C_{\boxtimes}(m,i,u,v) \iota^{(t)}(v) \right) \left(\sum_{j=u}^{m} \xi^f_{2l}(j) \sum_{n=u}^{m+u-j} C_{\boxdot}(m,u,j,n) \iota^{(t)}(n) \right) \\
&= \sum_{u=0}^{i} A(m,i,u) \left(\sum_{j=u}^{m} \xi^f_{2l}(j) B(m,u,j) \right) \\
&= \sum_{j=0}^{i} \xi^f_{2l}(j) \sum_{u=0}^{j} A(m,i,u) B(m,u,j) + \sum_{j=i+1}^{m} \xi^f_{2l}(j) \sum_{u=0}^{i} A(m,i,u) B(m,u,j)
\end{aligned}$$

where $A(m,i,u) = \sum_{v=i-u}^{i} C_{\boxtimes}(m,i,u,v)\iota^{(t)}(v)$
and $B(m,u,j) = \sum_{n=u}^{m+u-j} C_{\boxdot}(m,u,j,n)\iota^{(t)}(n)$.

Similarly,

$$\xi^f_{2l+1}(i) = \sum_{u=i}^{m} \left(\sum_{v=i}^{m+i-u} C_{\boxdot}(m,i,u,v)\iota^{(t)}(v) \right) \left(\sum_{j=0}^{u} \xi^f_{2l-1}(j) \sum_{n=u-j}^{u} C_{\boxtimes}(m,u,j,n)\iota^{(t)}(n) \right) =$$

$$\sum_{u=i}^{m} B(m,i,u) \left(\sum_{j=0}^{u} \xi^f_{2l-1}(j) A(m,u,j) \right) =$$

$$\sum_{j=0}^{i} \xi^f_{2l-1}(j) \sum_{u=i}^{m} B(m,i,u)A(m,u,j) + \sum_{j=i+1}^{m} \xi^f_{2l-1}(j) \sum_{u=j}^{m} B(m,i,u)A(m,u,j).$$

D.3 Entropy curve of the $((1,1))_1$-TLDPC base code

The local code of the $((1,1))_1$-TLDPC non-binary family is presented by two trellis sections, parallel-like and cross-like, both of length 1. The length-1 parallel section carries the bit of degree 1, and the cross-like section carries bits of degrees > 1.

Let x be the bit erasure probability for bits of degrees > 1, so the probability that the bit of a cross-like section is erased is equal to x. Let p be the erasure probability of the transmission channel. For a local code n denote the pair of probabilities $\{P_{2n+1}, P_{2n+2}\}$, $n = 0, 1, \ldots, m/2-1$, where P_i is the state erasure probability on the output of the section $i-1$ of the base code of length m (Fig.D.1).

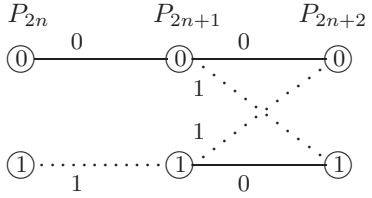


Figure D.1: n-th local code of the $((1,1))_1$-TLDPC base code.

Using BCJR decoding procedure on the trellis of the base code, we compute analytical expressions for extrinsic probabilities of bits of the base code. On the forward stage we obtain that

$$\begin{cases} P^{(f)}_{2n+1} = pP_{2n} \\ P^{(f)}_{2n+2} = (1-x)P_{2n+1} + x \end{cases}$$

The expression for $P^{(f)}_{2n+1}$ can be rewritten as follows

$$P^{(f)}_{2n+3} = px + p(1-x)P_{2n+1} \tag{D.1}$$

This is a linear recursion of the first order. We define $P^{(f)}_{2\infty+1}$ to be the stationary solution ? of (D.1). By using Theorem 2.1 of [63] and letting n tend to infinity, we obtain that

$$P^{(f)}_{2\infty+1} = \frac{px}{1-p(1-x)}.$$

Let $P_{2\infty}^{(b)}$ be the stationary solution of the backward recursion for even state stages. It can be shown that the expression $P_{2\infty}^{(b)}$ is the same,

$$P_{2\infty}^{(b)} = P_{2\infty+1}^{(f)}.$$

Let $E_C(x,p)$ be the extrinsic probability for a bit of the cross-like section. Then

$$E_C(x,p) = 1 - (1 - P_{2\infty+1}^{(f)})(1 - P_{2\infty}^{(b)}) = 1 - (1 - P_{2\infty+1}^{(f)})^2.$$

Thus, the average extrinsic probability of the base code is equal to

$$\bar{E} = E_C(x,p) = 1 - \left(\frac{1-p}{(1-p(1-x))}\right).$$

Appendix E

E.1 Noisy channel coding theorem for binary erasure channel

The noisy channel coding theory was first presented by Shannon in [64]. In this section we formulate it for binary erasure channel for a non-zero bit erasure probability P_e.

Theorem 17 *For binary erasure channel with capacity C, the following holds:*

- *If a bit erasure probability P_e is acceptable, the maximum achievable rate $R(P_e)$ is given by*
$$R(P_e) = \frac{C}{1 - P_e}.$$
- *all the rates no greater than $R(P_e)$ are achievable.*

Proof :

- Maximum achievable rate.
The source, encoder, noisy channel and decoder form (see Fig.E.1) a Markov chain

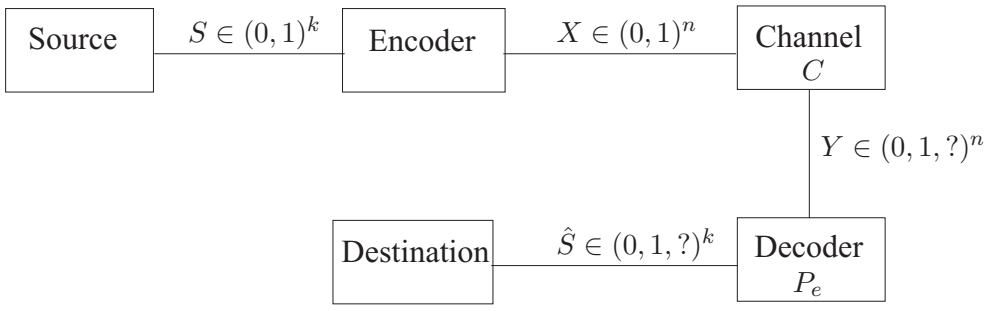


Figure E.1: Communication chain.

$S \to X \to Y \to \hat{S}$. For this chain we have that

$$I(S;\hat{S}) \leq I(X;Y) \leq \sum_i I(x_i;y_i) \leq nC.$$

Assume that a system achieves a rate R and that $\hat{S}$ is estimated with a maximum bit erasure probability P_e. Then

$$\begin{aligned} I(S;\hat{S}) &= H(S) - H(S|\hat{S}) = \sum_i H(s_i) - \sum_i H(s_i|\hat{s}_i, s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_k) \\ &\geq \sum_i H(s_i) - \sum_i H(s_i|\hat{s}_i) = \sum_i I(s_i;\hat{s}_i) \\ &\geq k(1-P_e) = nR(1-P_e). \end{aligned}$$

We obtain that $nR(1-P_e) \leq I(S;\hat{S}) \leq nC$ and thus

$$R \leq \frac{C}{1-P_e}.$$

- Achievable region.
  When a non-zero bit erasure probability P_e is acceptable, one can present the communication chain as consisting of an uniform source, a lossy compressor with distortion, an encoder, a noisy channel with capacity C, a decoder and a decompressor with error P_e as shown in Fig.E.2.

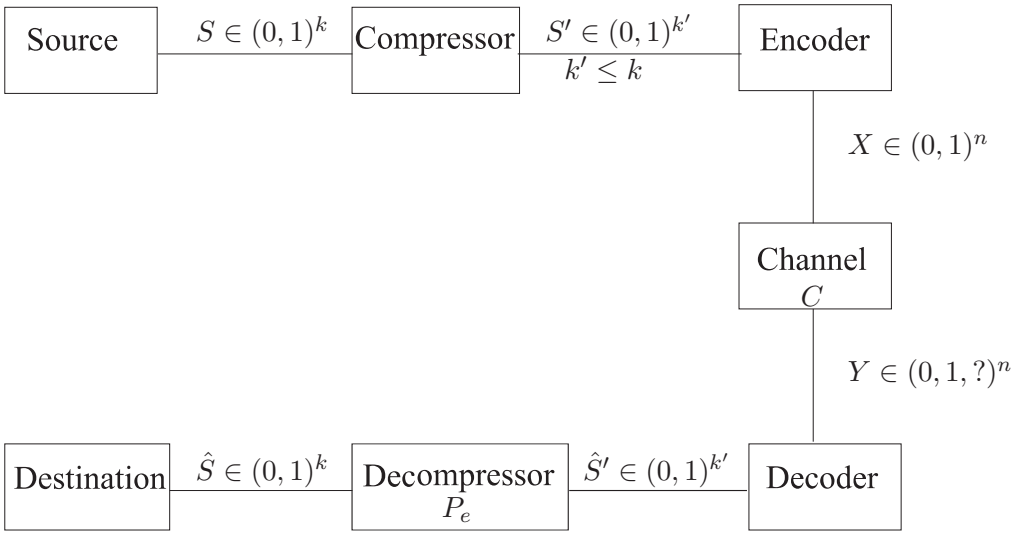


Figure E.2: Presentation of the communication chain with a lossy compressor.

The distortion function of the compressor between it input s and output s' bits is

$$d(s,s') = \begin{cases} 0, & \text{if s}' = \text{s} \\ 1, & \text{if s}' = ? \end{cases}$$

where ? denotes an erasure.

We use Theorem 13.2.1 in [24], p.342, to calculate the rate distortion function $R_{comp}(P_e)$ for the i.i.d. uniform source and the lossy compressor with the distortion

function $d(s, s')$ and maximum erasure probability P_e:

$$\begin{aligned} R_{comp}(P_e) &= \min_{p(s'|s):\sum_{(s,s')} p(s)p(s'|s)d(s,s')\leq P_e} I(S;S') \\ &= \min_{p(?|s):p(?|0)=p(?|1)\leq P_e} k - H(s|?) \\ &= 1 - P_e. \end{aligned}$$

The rate of the communication chain $R(P_e)$ is given by the multiplication

$$R(P_e) = R_{comp}R_{cod},$$

where R_{comp} is the rate of the source coding part and P_{cod} is the rate of the channel coding part. Note that $R_{comp} \leq \frac{1}{1-P_e}$ and $R_{cod} \leq C$. By choosing the source (channel) coding part so that $R_{comp} = \frac{1-\epsilon_1}{1-P_e}$ ($R_{cod} = C(1-\epsilon_2)$) for some $\epsilon_1 > 0$ ($\epsilon_2 > 0$), we obtain that

$$R(P_e) = C\frac{1-\epsilon}{1-P_e}$$

with $1 - \epsilon \geq (1-\epsilon_1)(1-\epsilon_2)$.

∎

By the theorem above, we compute the threshold p^{Sh} on the BEC for fixed code rate R and bit erasure probability P_e as follows

$$p^{Sh} = 1 - R(1 - P_e).$$

This result is used in Section 5.4.3.

Bibliography

[1] EN 301 790. Digital video broadcasting: Interaction channel or satellite distribution systems, December 2000.

[2] D. Burshtein A. Bennatan. Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels. *IEEE Trans. Inform. Theory*, 52(2):549–583, February 2006.

[3] N. Alon, S. Hoory, and N. Linial. The Moore bound for irregular graphs. *Graphs Combin.*, 18:53–57, 2002.

[4] I. Andriyanova, J.P. Tillich, and J.C. Carlach. Asymptotically good codes with high iterative decoding performances. In *ISIT'05*, pages 850–854. IEEE, September 2005.

[5] I. Andriyanova, J.P. Tillich, and J.C. Carlach. A new family of asymptotically good codes with high iterative decoding performances. In *ICC'06*, June 2006.

[6] M. Ardakani and F. R. Kschischang. A more accurate one-dimensional analysis and design of LDPC codes,. *IEEE Trans. Communications*, 52(12):2106–2114, December 2004.

[7] A. Ashikhmin, G. Kramer, and S. ten Brink. Extrinsic information transfer functions : model and erasure channel properties. *IEEE Trans. Inform. Theory*, 50(11):2657–2673, November 2004.

[8] R. Koetter B. Frey and A. Vardy. Skewness and pseudocodewords in iterative decoding. In *ISIT 1998*, August 2006.

[9] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimazing symbol error rate. *IEEE Trans. on Information Theory*, 20:284–287, March 1974.

[10] S. Benedetto and G. Montorsi. Unveiling turbo codes: some results on parallel concatenated coding schemes. *IEEE Trans. on Information Theory*, 42:409–429, March 1996.

[11] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding. In *ICC'93*, pages 1064–1070, Genève, Switzerland, May 1993.

[12] C. Berrou and M. Jézéquel. Non binary convolutional codes for turbo-coding. *Electronic Letters*, 35(1):39–40, January 1999.

[13] C. Berrou, M. Jézéquel, C. Douillard, and S. Kerouédan. The advantages of non-binary turbo-codes. In *Information Theory Workshop ITW'01*, pages 61–63, Cairns, Australia, September 2001.

[14] K. Bhattad and K. R. Narayanan. An MSE based transfer chart to analyze iterative decoding schemes. submitted to *IEEE Inform. Theory*, June 2005.

[15] J. Boutros, G. Caire, E. Viterbo, H. Sawaya, and S. Vialle. Turbo code at 0.03 db from capacity limit. In *Proceedings of the IEEE Int. Symp. Inform. Theory*, page 56, Lausanne, Switzerland, July 2002.

[16] J. Boutros, O. Pothier, and G. Zémor. Generalized low density (Tanner) codes. In *Proceedings of ICC'99*, pages 441–445, Vancouver, June 1999.

[17] M. Breiling. A logarithmic upper bound on the minimum distance of turbo codes. *IEEE Transactions on Information Theory*, 50(8):1692–1710, 2004.

[18] D. Burshtein and G. Miller. Asymptotic enumeration methods for analyzing LDPC codes. *IEEE Trans. Inform. Theory*, 50(6):1115–1131, June 2004.

[19] A. Montanari C. Measson and R. Urbanke. Maxwell construction: the hidden bridge between iterative and maximum a posteriori decoding. Submitted to *IEEE Inform. Theory*, June 2005.

[20] T. Richardson C. Measson, A. Montanari and R. Urbanke. Life above threshold: from list decoding to area theorem and MSE. In *2004 IEEE Information Theory Workshop*, October 2004.

[21] C. Chaikalis and J. M. Noras. Reconfigurable turbo decoding for 3G applications. *Signal Processing*, 84:1957–1972, 2004.

[22] S. Y. Chung, G. D. Forney Jr., T. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 db from the Shannon limit. *IEEE Communication Letters*, 5:58–60, 2001.

[23] C. Cole and E. Hall. Analysis and design of moderate length regular LDPC codes with low error floors. In *Conference on Information Sciences and Systems*, March 2006. URL: http://www288.pair.com/ciss/ciss/numbered/120.pdf.

[24] T.M. Cover and J.M. Thomas. *Elements of information theory*. A Wiley-Interscience publication, 1991.

[25] S. Dolinar D. Divsalar and C. Jones. Construction of protograph LDPC codes with linear minimum distance. In *ISIT 2006*, July 2006.

[26] M.C. Davey and D.J.C. MacKay. Low density parity check codes over GF(q). *IEEE Communications Letters*, 2:165–167, June 1998.

[27] A. Dembo and O. Zeitouni. *Large deviations techniques and applications*. Applications of Mathematics, Stochastic modelling and applied probability. Spinger Verlag, 1998.

[28] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke. Finite length analysis of low-density parity-chack codes on the binary erasure channel. *IEEE Trans. Inform. Theory*, 48(6):1570–1579, June 2002.

[29] C. Di, T. Richardson, and R. Urbanke. Weight distributions : how deviant can you be? In *Proceedings of the IEEE Int. Symp. Inform. Theory*, page 50, Washington, USA, July 2001.

[30] C. Di, T. Richardson, and R. Urbanke. Weight distribution of low-density parity-check codes. preprint, 2004.

[31] D. Divsalar, H. Jin, and R. J. McEliece. Coding theorems for 'turbo-like' codes. In *Proc. 36th Allerton Conf. on Coomunication, Control, and Computing.*, pages 201–210, Allerton, Illinois, September 1998.

[32] T. Lehnigk-Emden F. Kienle and N. Wehn. Fast convergence algorithm for LDPC codes. In *VTC 2006-Spring*, volume 5, pages 2393– 2397, June 2006.

[33] B. J. Frey F. R. Kschischang and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Inform. Theory*, 47(2):498–519, February 2001.

[34] J. Frey and J.C.MacKay. Irregular turbo-codes. In *Proceedings of the 37th annual Allerton conference on Communications, Control and Computing*, October 1999.

[35] F.R.Gantmacher, editor. *The theory of matrices*, volume 2. Chelsea, 1959.

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

[37] H. El Gamal and A. R. Hammons. Analyzing the turbo decoder using the Gaussian approximation. *IEEE Trans. Inform. Theory*, 47(2):671–686, February 2001.

[38] C. H. Hsu and A. Anastasopoulos. Asymptotic weight distributions of irregular repeat-accumulate codes. In *2005 IEEE Global Telecommunications Conference*, volume 3, pages 1147–1151, November 2005.

[39] X. Hu. *Low-delay low-complexity error-correcting codes on sparse graphs.* PhD thesis, EPFL, Lausanne, Switzerland, 2002.

[40] X. Hu, E. Eleftheriou, and D.M. Arnold. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Trans. on Inform. Theory*, 51(1):386–398, January 2005.

[41] E. Telatar I. Sason and R. Urbanke. On the asymptotic input-output weight distributions and thresholds of convolutional and turbo-like encoders. *IEEE Trans. Inform. Theory*, 48:3052–3061, December 2002.

[42] H. Jin, A. Khandekar, and R. McEliece. Irregular repeat-accumulate codes. In *Proceedings of the 2nd Int. Conf. on Turbo codes and related topics*, pages 1–8, September 2000.

[43] J.Pearl. *Probabilistic reasoning in intelligent systems: networks of Plausible Inference*. Morgan Kaufmann, San Meteo, CA, 1988.

[44] J. W. Lee and R. E. Blahut. Bit error rate estimate of finite length turbo codes. In *ICC 2003 - IEEE International Conference on Communications*, pages 2728–2732, May 2003.

[45] W.K.Raymond Leung, Guosen Yue, Li Ping, and Xiaodong Wang. Concatenated zigzag Hadamard codes. *IEEE Trans. Inform. Theory*, 52(4):1711–1723, April 2006.

[46] S. Litsyn and V. Shevelev. On ensembles of low-density parity-check codes: asymptotic distance distributions. *IEEE Trans. Inform. Theory*, 48(4):887–908, 2002.

[47] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Practical loss resilent codes. In *Proceedings of the 30th annual ACM Symposium on Theory of Computing (STOC)*, pages 249–258, 1998.

[48] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inform. Theory*, 47(2):569–584, February 2001.

[49] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Trans. Inform. Theory*, 47(2):585–598, February 2001.

[50] D. Mackay and M. Davey. Evaluation of Gallager codes for short block length and high rate applications. *IMA Volumes in Mathematics and its Applications*, 123, 2000.

[51] D. MacKay and M. Postol. Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes. In *Electronic Notes in Theoretical Computer science*, volume 74, 2003. URL: http:/www.elsevier.nl/locate/entcs/volume74.html.

[52] 0. Milenkovic, E. Soljanin, and P. Whiting. Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles. Submitted for publication, *IEEE Trans. Inform. Theory*.

[53] H. Pishro-Nik and F. Fekri. Performance of low-density parity-check codes with linear minimum distance. *IEEE Trans. Inform. Theory*, 52(1):292–300, January 2006.

[54] O. Pothier. *Compound codes based on graphs and their iterative decoding*. PhD thesis, ENST Paris, 2000.

[55] V. Rathi and R. Urbanke. Density evolution, thresholds and the stability condition for non-binary LDPC codes. *IEEE Trans. on Comm.*, December 2005.

[56] T. Richardson. Error-floors of LDPC codes. In *Proceedings of the 41st Annual Conference on Communications*, pages 1426–1435, September 2003.

[57] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of capacity-approaching irregular low-density parity check codes. *IEEE Trans. Inform. Theory*, 47:619–637, February 2001.

[58] T. Richardson and R. Urbanke. Modern coding theory. In preparation. see http://lthcwww.epfl.ch/papers/ics.ps.

[59] T. Richardson and R. Urbanke. Multi-edge LDPC codes. Available at:http://lthcwww.ep.ch/papers/multiedge.ps.

[60] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inform. Theory*, 47:599–618, February 2001.

[61] G. Montorsi S. Benedetto, D. Divsalar and F. Pollara. Serial concatenation of interleaved codes: performance analysis, design and iterative decoding. *IEEE Trans. on Information Theory*, 44:909–926, May 1998.

[62] H.E. Sawaya and J. Boutros. Irregular turbo-codes with symbol-based iterative decoding. In 3^{rd} *International Symposium on Turbo-codes*, pages 407–410, Brest, France, September 2003.

[63] R. Sedgewick and P. Flajolet. *Introduction à l'analyse des algorithmes*. International Thomson Publishing France, Paris, 1996.

[64] C. E. Shannon. A mathematical theory of communication. In *Bell Syst. Tech. J.*, volume 27, pages 379–423,623–656, July/October 1948.

[65] A. Shokrollahi. Capacity-achieving sequences. In B. Marcus and J. Rosenthal, editors, *Codes,Systems, and Graphical Models*, number 123 in IMA vol. in Mathematics and Applications, pages 153–166. Springer, 2000.

[66] G.W. Stewart and J.G. Sun. *Matrix perturbation Theory*. Computer science and scientific computing. Academic Press, 1990.

[67] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27:533–547, September 1981.

[68] S. ten Brink. Convergence behaviour of iteratively decoded parallel concatenated code. *IEEE Trans. Commun.*, 49:1727–1737, Oct. 2001.

[69] S. ten Brink, G. Kramer, and A. Ashikhmin. Design of low-density parity-check codes for modulation and detection. *IEEE Transactions on Communications*, 52(4):670–678, April 2004.

[70] J. Thorpe. Low-density parity-check (LDPC) codes constructed from protographs. IPN Progress Report 42-154, 2003.

[71] J. P. Tillich. The average weight distribution of Tanner code ensembles and a way to modify them to improve their weight distribution. In *Proceedings of ISIT'04*, page 7, Chicago, Illinois, 2004.

[72] J.-P. Tillich. *Habilitation à Diriger des Recherches*. INRIA Rocquencourt, France, 2006. in preparation.

[73] J.P. Tillich and G. Zémor. On the minimum distance of structured LDPC codes with two variable nodes of degree-2 per parity-check equation. In *Proceedings of ISIT 2006*, Seattle, USA, 2006.

[74] R. Urbanke V. Rathi. Density evolution, thresholds and the stability condition for non-binary LDPC codes. *IEEE Trans. on Communications*, December 2005.

[75] Yi-Pin Wang, R. Ramesh, A. Hassan, and H. Koorapaty. On MAP decoding for tail-biting convolutional codes. In *Proceedings of ISIT'97*, Ulm, Germany, March 1997.

[76] N. Wiberg. *"Codes and Decoding on General Graphs"*. PhD thesis, Linköping University, Sweden, April 1996.

[77] E. Eleftheriou X. Hu. Binary representation of cycle Tanner-graph GF(2^b) codes. In *ICC'04*, Paris, France, June 2004.

[78] Kyeongcheol Yang. A nonbinary extension of RA codes: weighted nonbinary repeat accumulate codes. In *14th IEEE 2003 International Symposium on Personal, Indoor and Mobile Radio Communication Proceedings*, 2003.