Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Ingegneria Elettronica, Elettrotecnica e delle Telecomunicazioni

---

Design and Analysis of Spatially Coupled LDPC Convolutional Codes

Ph.D. Dissertation of:
Massimo Battaglioni

Advisor:
Prof. Giovanni Cancellieri

Coadvisor:
Dr. Marco Baldi

Curriculum Supervisor:
Prof. Francesco Piazza

XVII edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Ingegneria Elettronica, Elettrotecnica e delle Telecomunicazioni

---

Design and Analysis of Spatially Coupled LDPC Convolutional Codes

Ph.D. Dissertation of:
Massimo Battaglioni

Advisor:
Prof. Giovanni Cancellieri

Coadvisor:
Dr. Marco Baldi

Curriculum Supervisor:
Prof. Francesco Piazza

XVII edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

To my family

Acknowledgments

Now that my PhD course is over, I wish to spend a few words to heartily thank the people who contributed to make it special.

I would like to thank my supervisor, Prof. Giovanni Cancellieri, for his guidance throughout the years. His enthusiasm and fresh ideas have been an invaluable source of motivation for me.

A special thank to Dr. Marco Baldi, for always being in the forefront with me when I needed advice, help, care and support. I thank him for having guided me into a long process of personal and professional growth.

A giant thank to Prof. Franco Chiaraluce, for representing a model of rigour and relentlessness. His meticulous comments on my reports and papers, along with his precious advice on everything I required, inestimably contributed to my formation.

I would also like to thank Prof. Michael Lentmaier, for making my period in Lund a wonderful experience and for teaching me, during our endless brainstorming sessions, how to attack a problem from many different perspectives.

I heartily thank Dr. David Mitchell, for hosting me at New Mexico State University and providing assistance and help at work and beyond. Thanks to him, in the three months in Las Cruces I have been productive and happy.

A great thank to Prof. Mark Flanagan, for inviting me at UCD and for laying the basis of a new collaboration. I hope the future reserves us more opportunities to work together.

A sincere thank to Alireza, Torleiv and Enrico, for the insightful discussions on coding.

Thanks to the other fellows in the research group: to Nicola and Giacomo, the gentle giants, which assisted me at the beginning of my PhD course and have been true friends then; to Linda, for sharing with me many adventures and for tolerating my apathy at home; to Paolo, for the passionate discussions on beautiful mathematical topics.

I would like to thank also Terenz and Pol, never boring colleagues for many years!

Michele, LauraF, LauraM, Manola and Lorenzo brought daily happiness to the office, thank you!

Stefano has been a lovely friend and housemate, I would not have made it without his dinners and bridge discussions.

Thanks to Edoardo for being my punching bag, for the nonsense discussions on life and for sharing his personal life with me.

It was thanks to my friends Wei in Lund and Ahmad in Las Cruces that I understood how easy is to overcome racial and cultural barriers.

A sincere thank to my parents and Leonardo, for making home a wonderful place and for their unconditional encouragement.

The final thank to Ludovica for always standing by my side, even when I was a thousand leagues away. Her love has been fuel throughout the hard moments of this period.

Ancona, November 2018

Massimo Battaglioni

Abstract

Nowadays, communication systems require data to be transmitted from a source to a destination with sufficiently low latency. On the other hand, the stringent requirements on data reliability enforce the use of efficient error correcting codes guaranteeing a low probability of erroneous reception and retrieval. Low-density parity-check (LDPC) codes are a family of codes based on graphs which are able to provide with reduced complexity the same error correction performance as other codes. Thanks to their excellent error rate performance and better adaptability to high throughput requirements, LDPC codes have been chosen to replace turbo codes in the new radio access technology of the 5G standard, in the third generation partnership project (3GPP) framework.

One of the most promising families of LDPC codes is that of spatially coupled LDPC (SC-LDPC) codes, which are capacity achieving codes. It is well-known that these codes can outperform their block counterparts. In this dissertation, the properties of SC-LDPC codes are analyzed and new design approaches are proposed.

After an overview of channel coding, LDPC block and convolutional codes are introduced. The most common iterative decoding techniques of these codes are also discussed. Particular attention is devoted to the analysis and design of *compact* SC-LDPC codes, as they comply with the severe rules on complexity and latency imposed by many modern applicative scenarios. Keeping in mind these requirements, the design of codes without structural flaws in their graphical representation is addressed, eventually yielding improved error rate performance and minimum distance features.

Foreword

During my period at Dipartimento di Ingegneria dell'Informazione of Università Politecnica delle Marche as Ph.D. student, I had the pleasure to work with the research group leaded by Prof. Giovanni Cancellieri, Prof. Franco Chiaraluce and Dr. Marco Baldi.

In our work we have analyzed and proposed new techniques able to increase reliability in modern communication systems. Nowadays, communication systems endeavor to achieve reliable and efficient information transmission with low complexity and our team focused on error correction coding, with particular attention to LDPC convolutional codes.

During these years I had the possibility to collaborate with very important international academic organizations, such as Lund University, New Mexico State University and University College Dublin. In particular, I spent three months at the Department of Electrical and Information Technology in Lund, Sweden, with the research group led by Prof. Michael Lentmaier. I also spent three months at Klipsch School of Electrical and Computer Engineering at New Mexico State University, Las Cruces, USA, under the supervision of Dr. David Mitchell. Finally, I visited the research group led by Prof. Mark Flanagan at the School of Electrical and Electronic Engineering, University College Dublin, Ireland, for one month. Part of this thesis was developed during these visiting periods.

The contents of this thesis have been partially included in the following publications:

- **M. Battaglioni**, A. Tasdighi, M. Baldi, M. H. Tadayon and F. Chiaraluce, "Compact QC-LDPC Block and SC-LDPC Convolutional Codes for Low-Latency Communications", in *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)* 2018, Sep. 2018, Bologna, Italy.

- **M. Battaglioni**, M. Baldi, and G. Cancellieri, "Connections Between Low-Weight Codewords and Cycles in Spatially Coupled LDPC Convolutional Codes", *IEEE Trans. Commun.*, vol. 66, no. 8, pp. 3268–3280, Aug. 2018, DOI: 10.1109/TCOMM.2018.2820702.

- M. H. Tadayon, A. Tasdighi, **M. Battaglioni**, M. Baldi, and F. Chiaraluce, "Efficient Search of Compact QC-LDPC and SC-LDPC Convolu-

tional Codes with Large Girth", *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1156–1159, Jun. 2018, DOI: 10.1109/LCOMM.2018.2827959.

- **M. Battaglioni**, A. Tasdighi, G. Cancellieri, F. Chiaraluce, and M. Baldi, "Design and Analysis of Time-Invariant SC-LDPC Convolutional Codes With Small Constraint Length", *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 918–931, Mar. 2018, DOI: 10.1109/TCOMM.2017.2774821.

- **M. Battaglioni**, M. Baldi, and E. Paolini, "Complexity-Constrained Spatially Coupled LDPC Codes Based on Protographs", in *Proc. International Symposium on Wireless Communication Systems (ISWCS)*, pp. 49–53, Bologna, Italy, Sep. 2017, DOI: 10.1109/ISWCS.2017.8108160.

- **M. Battaglioni**, M. Baldi, and G. Cancellieri, "Design of Spatially Coupled LDPC Codes Based on Symbolic Hyper-Graphs", in *Proc. International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1–5, Split, Croatia, Sep. 2016, DOI: 10.1109/SOFTCOM.2016.7772132.

- M. Baldi, **M. Battaglioni**, F. Chiaraluce, and G. Cancellieri, "Time-Invariant Spatially Coupled Low-Density Parity-Check Codes with Small Constraint Length", in *Proc. IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–5, Varna, Bulgary, Jun. 2016, DOI: 10.1109/BlackSeaCom.2016.7901543.

Contents

Foreword xiii

List of abbreviations xxiii

1 Introduction 1
- 1.1 Outline of the thesis . . . 2
- 1.2 Main contributions of the thesis . . . 3

2 Background 5
- 2.1 Channel models . . . 6
  - 2.1.1 Binary additive white gaussian noise channel . . . 6
- 2.2 Low-density parity-check codes . . . 6
  - 2.2.1 Definitions and notation for LDPC block codes . . . 6
- 2.3 Classical and modern constructions of LDPC convolutional and block codes . . . 8
  - 2.3.1 Classical construction of LDPC convolutional codes . . . 8
  - 2.3.2 Quasi-cyclic LDPC block codes from LDPC convolutional codes . . . 13
  - 2.3.3 Protograph based LDPC codes . . . 14
- 2.4 Summary . . . 16

3 Finite length and asymptotic performance of LDPC codes 17
- 3.1 Iterative decoding techniques: the LLR-SPA . . . 18
  - 3.1.1 Initialization . . . 19
  - 3.1.2 Left semi-iteration . . . 19
  - 3.1.3 Right semi-iteration . . . 19
  - 3.1.4 Decision . . . 19
  - 3.1.5 Decoding complexity and latency . . . 20
- 3.2 Protograph extrinsic information transfer analysis . . . 21
  - 3.2.1 PEXIT analysis for block codes . . . 21
- 3.3 Sliding window . . . 23
  - 3.3.1 Decoding latency and complexity of the sliding window decoder . . . 23
- 3.4 Summary . . . 24

4 SC-LDPC convolutional codes with small constraint length 25
4.1 Notation . . . 26
4.2 Lower bounds on the constraint length . . . 26
4.2.1 Type-1 codes . . . 26
4.2.2 Bounds on the constraint length of type-z codes, $\forall z$ . . . 32
4.3 Design methods . . . 35
4.3.1 Type-1 codes . . . 35
4.3.2 Type-2 codes . . . 35
4.3.3 Type-3 codes . . . 37
4.4 Code examples and numerical results . . . 38
4.4.1 Code design based on exhaustive searches . . . 38
4.4.2 IP model . . . 39
4.4.3 Code design based on integer programming . . . 41
4.4.4 Numerical simulations of coded transmissions . . . 41
4.5 Design based on sequentially multiplied columns . . . 45
4.5.1 Sequentially multiplied columns . . . 46
4.5.2 Latency and complexity performance . . . 47
4.5.3 Numerical simulations of coded transmissions . . . 50
4.6 Summary . . . 51

5 SC-LDPC codes in complexity-constrained scenarios 53
5.1 Finding the optimal trade-off between window size and number of iterations . . . 53
5.2 Summary . . . 57

6 Connection between cycles and codewords of SC-LDPC convolutional codes 59
6.1 Notation . . . 60
6.2 Codewords as superposition of component codewords . . . 62
6.2.1 Types of component codes . . . 67
6.3 Analysis of some ensembles of codes . . . 69
6.3.1 Low rate codes . . . 70
6.3.2 Medium and high rate codes . . . 79
6.4 Performance assessment . . . 80
6.5 Summary . . . 82

7 Conclusions 83

Bibliography 85

List of Figures

2.1 Block scheme of a digital communication system 5
2.2 Tanner Graph of the parity-check matrix in Example 2.2.1 . . . 8
2.3 Protograph corresponding to the base matrix in Example 2.3.2 15
2.4 Example of a lifted protograph 15

3.1 Representation of sliding window decoding over coupled protographs and the corresponding semi-infinite parity-check matrix. 24

4.1 Bounds on L_h and values found through exhaustive searches as a function of a for some values of c and: (a) $d_v = 2$, $g = 6$ (b) $d_v = 2$, $g = 8$ (c) $d_v = 3$, $g = 6$ (d) $d_v = 3$, $g = 8$. 39
4.2 Bounds on m_s and values found through the design method described in Section 4.3 as a function of a, for $d_v = c = 3$, $g = 6$. 40
4.3 Simulated performance of the codes in Table 4.4. 43
4.4 BER performance of the codes in Table 4.5. 44
4.5 Minimum lifting degree (L) of new and previously designed QC-LDPC codes versus a, for $c = 3, 4$. 49
4.6 Minimum syndrome former memory order (m_s) of new and previously designed SC-LDPC-CCs versus a, for $c = 3, 4$. 49
4.7 Simulated performance of SC-LDPC-CCs with: (a) $g = 10$ and (b) $g = 12$ as a function of the signal-to-noise ratio. 51

5.1 Thresholds $\frac{E_b}{N_0}^*$ vs W for $R_\infty = \frac{3}{6}$ codes when $C = 6000$, $M = 1$. 55
5.2 Thresholds $(\frac{E_b}{N_0})^*$ vs W for $R_\infty = \frac{3}{6}$ codes when $C = 24000$, $M = 1$. 55
5.3 BER performance and protograph EXIT (PEXIT) thresholds of the proposed codes, lifted with circulant permutation matrices of size $M = 25$. 56

6.1 Cycle of length 10, associated to $\mathbf{u}_0(D)$, forming (6.15). It involves columns with indexes $\in \{0, 1, 2, 6, 7\}$. 75
6.2 Cycle of length 12, associated to $\mathbf{u}_1(D)$, forming (6.15). It involves columns with indexes $\in \{0, 4, 5, 6, 11, 16\}$. 76
6.3 Cycle of length 18, associated to $\mathbf{u}_2(D)$, forming (6.15). It involves columns with indexes $\in \{0, 1, 2, 4, 5, 6, 7, 11, 16\}$. 76

6.4 BER performance of classical and optimized SC-LDPC-CCs with: (a) $v_s = 272$ and (b) $v_s = 156$ as a function of the SNR per bit, E_b/N_0. 82

List of Tables

4.1 Minimum values of m_s for Type-1 codes with $d_v = c = 3$ and $g = 8$. 39
4.2 Minimum values of m_s for codes with $d_v = c = 3, 4, 5, 6$ and $g = 8$ obtained through the Min-Max algorithm, for several code rates. 42
4.3 Minimum values of m_s for codes with $d_v = c = 3$ and $g = 10$ obtained through the Min-Max algorithm, for several code rates. 42
4.4 Values of a, c, d_v, m_s, v_s and g of the considered codes with $R = \frac{14}{17}$. 43
4.5 Values of a, c, d_v, m_s, v_s and g of the considered codes with $R = \frac{2}{3}$. 44
4.6 Values of a, c, m_s, v_s and g of the considered SC-LDPC-CCs with $R = \frac{4}{7}$. 50

5.1 Optimal values of the window size and number of decoding iterations for $R_\infty = \frac{3}{6}$ and $C = 6000, 24000$. 56
5.2 Values of the PEXIT thresholds for the codes in Fig. 5.3, $C \leq 24000$. 57

6.1 Component codewords forming the unavoidable codeword (6.10). 71
6.2 Component codewords forming the unavoidable codeword (6.14). 74
6.3 Component codewords forming the unavoidable codeword (6.15). 75
6.4 Component codewords forming the codeword (6.17). 79

List of abbreviations

3GPP third generation partnership project.

AWGN additive white gaussian noise.

BEC binary erasure channel.

BER bit error rate.

BP belief propagation.

BPSK binary phase shift keying.

CPM circulant permutation matrices.

DE density evolution.

DVB-S digital video broadcasting - satellite.

DVB-T digital video broadcasting - terrestrial.

eMBB enhanced mobile broadband.

EXIT extrinsic information transfer.

IoT internet of things.

IP integer programming.

LDPC low-density parity-check.

LDPC-CC LDPC convolutional code.

LLR log likelihood ratio.

LLR-SPA log-likelihood ratio sum product algorithm.

M2M machine-to-machine.

MI mutual information.

PDF perfect difference family.

PEXIT protograph EXIT.

QC-LDPC quasi-cyclic LDPC.

SC-LDPC spatially coupled LDPC.

SC-LDPC-CC spatially coupled LDPC convolutional code.

SMC sequentially multiplied column.

SNR signal-to-noise ratio.

SPA sum product algorithm.

SW sliding window.

Chapter 1

Introduction

We are in the middle of the *third industrial revolution*: an on-going social and economical process relying on the manipulation of information. Nowadays, people are able to communicate with each other, access and exchange digital information almost without time and space constraints. One of the breakthroughs that permitted the advent of the *information age* we live in is the pioneering paper "A Mathematical Theory of Communication", by Claude E. Shannon [1]. The main contributions of Shannon's work were to define the *bit* as the basic unit of information and to describe a novel digital *communication model*. The contributions built upon this new model laid the basis of information and coding theory.

Since then, schemes for error correction have been increasingly adopted to improve the *reliability* of communication systems. The use of coding schemes has been recommended in the most modern communication systems, such as satellite communications, digital video broadcasting - satellite (DVB-S) , digital video broadcasting - terrestrial (DVB-T), mobile networks, Wi-Fi, WiMax, 10GBase-T Ethernet, etc.

In this dissertation we deal with low-density parity-check (LDPC) codes and their properties. Owing to their excellent performance, they have been chosen as the standard codes for the new radio access technology of the next generation of mobile systems (5G). In particular, LDPC codes meet the stringent requirements of the enhanced mobile broadband (eMBB) scenario: low computation complexity, low latency, low cost and high flexibility [2, 3].

LDPC block codes, first introduced by Gallager in [4], are a class of capacity approaching codes. Their convolutional counterparts were first proposed in [5] and are also known as spatially coupled LDPC (SC-LDPC) codes; these codes can achieve capacity under belief propagation decoding [6], thanks to a threshold saturation phenomenon [7–9].

Nevertheless, these outstanding properties have been proved to hold in asymptotic regime, i.e., they can be obtained with a decoding complexity and latency tending to infinity. Practical scenarios usually demand low decoding latency, which can only be achieved using codes characterized by a finite value of the

memory. Moreover, the finite length performance of these codes cannot be predicted theoretically, and intensive simulations are required to assess it. For these reasons, in this dissertation we address the design and analysis of *compact* block and convolutional LDPC codes, more suitable to practical systems. Classically, LDPC convolutional codes are designed starting from block codes [5]. We show that, if the design of SC-LDPC codes does not start from that of their block counterparts, extremely compact solutions can be found. These compact codes are also well fit in applicative scenarios beyond 5G, such as, for example, machine-to-machine (M2M) communications, widely employed in the wider framework of internet of things (IoT), as they are based on the transmission of short packets [10].

The iterative decoding algorithms for LDPC codes can get trapped in error patterns, which are caused by structural imperfections in their graphical representation. These objects may cause a severe degradation of the code error correction performance. This behavior was observed for the first time in [11]. During the years, these harmful objects have assumed different names (stopping sets [12], trapping sets [13], absorbing sets [14], etc.), depending on the considered channel and the type of identified decoding error. The "nucleus" of these harmful objects is a cycle, or a cluster of cycles, in the graphical representation of the code.

Part of this dissertation is focused on the design of compact SC-LDPC codes which do not contain the most harmful objects and, thus, achieve a good performance with reasonable decoding latency and complexity. New theoretical lower bounds on the value of the memory of SC-LDPC codes are derived, under constraints on the minimum size of their harmful objects.

SC-LDPC codes are also analyzed from a minimum distance perspective and a new method for deriving low weight codewords is proposed. In particular, we show that the codewords of the main code can be expressed as a combination of codewords of component codes. The design of codes free of codewords up to a certain weight is also addressed. We show that low-weight codewords in the main code can be avoided by removing cycles in its graphical representation.

1.1 Outline of the thesis

The document is organized as follows.

Chapter 2

In Ch. 2 the definitions and notation used throughout the dissertation are introduced. The classical and modern construction methods of spatially coupled LDPC convolutional codes (SC-LDPC-CCs) and quasi-cyclic LDPC (QC-LDPC) codes are recalled.

Chapter 3
In Ch. 3 a commonly used iterative decoding algorithm for LDPC codes is described. In particular, we focus our attention on the log-likelihood ratio sum product algorithm (LLR-SPA)-based decoder. We consider this decoder in the Monte Carlo simulations of transmissions we use to assess the finite length performance of LDPC codes. Moreover, we recall the basic concepts of protograph EXIT (PEXIT) analysis, through which it is possible to estimate the performance of code ensembles in the asymptotic regime. Finally, we introduce the concept of sliding window and show how this tool facilitates the analysis of LDPC convolutional codes.

Chapter 4
In Ch. 4 compact SC-LDPC-CCs are studied. Firstly, theoretical lower bounds on the constraint length of these codes are derived. Then, many construction methods are devised. Monte Carlo simulations confirm the effectiveness of our approach.

Chapter 5
Chapter 5 is devoted to the analysis of SC-LDPC-CCs in complexity-constrained scenarios. A heuristic approach, based on PEXIT analysis, shows how to properly choose the parameters of the sliding window decoder, in order to have the best possible performance, when computing and memory are limited.

Chapter 6
In Ch. 6 we describe the relationship between codewords and cycles of SC-LDPC-CCs. In particular, we show that removing some class of cycles in the Tanner graph, better minimum distance properties can be obtained. Furthermore, the removal of such harmful configurations has a positive impact on the error rate performance, too.

Chapter 7
Finally, Ch. 7 concludes the thesis.

1.2 Main contributions of the thesis

In the following list the main contributions of this thesis are reported.

Chapter 4

- Lower bounds on the constraint length of the most important families of SC-LDPC-CCs are derived, which permit to avoid cycles up to a given length.

- Several design methods permitting to achieve, or at least approach these bounds are proposed.
- An optimization model is proposed, which permits to find very compact codes.
- A construction method based on sequentially multiplied columns is introduced.

Chapter 5

- A heuristic method to find the best possible trade-off between window size and number of iterations of sliding window decoders is proposed.
- Given a specific code and an upper bound on the maximum complexity allowed by the communication system, our procedure permits to find the best asymptotic conditions under which the decoder can work.

Chapter 6

- We prove that the codewords of a given code can be expressed as a combination of codewords of some component codes.
- We show that each codeword of a component code can be associated to a number of cycles in the code Tanner graph.
- We propose a heuristic method based on the removal of some cycles, which yields better minimum distance properties and error rate performance compared to previous solutions.

Chapter 2

Background

According to Shannon's definition [1], "the fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point". A basic block scheme of the digital communication system he proposed is shown in Fig. 2.1. According to Shannon's separation principle, the processing of a sequence prior to transmission can be separated into two steps without losing optimality: the source encoding and the channel encoding. The analysis of source encoding goes beyond the scope of this work and, for this reason, the source and the source encoder are combined into a single block, whose output is an *information sequence*. Similarly, the source decoder and the destination are grouped in a single block, as well.

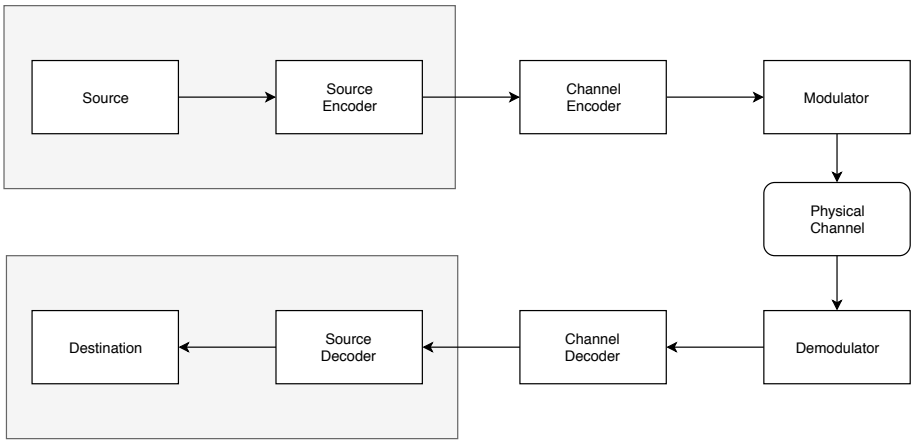


Figure 2.1: Block scheme of a digital communication system

In the same paper, Shannon enunciated the *noisy channel theorem*, which states that absolutely reliable transmission is possible as long as the transmission rate is less than the channel capacity. This outstanding result motivates the presence of two blocks: the channel encoder, whose function is to add redundancy to the information sequence in such a way that the effect of the

channel is mitigated and the channel decoder, whose function is to retrieve the initial information sequence.

2.1 Channel models

In this section the additive white gaussian noise (AWGN) channel is briefly described.

2.1.1 Binary additive white gaussian noise channel

A realistic channel model is the binary input AWGN channel. Let us consider a binary sequence $\mathbf{b}$, whose symbols are mapped into antipodal waveforms with energy E_s by the modulator, according to the binary phase shift keying (BPSK) scheme. The received symbol can be expressed as

$$r_t = (1 - 2b_t)\sqrt{E_s} + n_t,$$

where b_t denotes the tth entry of $\mathbf{b}$, n_t is a Gaussian random variable with zero mean and variance $\sigma_n^2 = \frac{N_0}{2}$ and N_0 is the noise spectral density.

The transition probability density function can hence be expressed as

$$p(r_t|b_t) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(r_t - (1-2b_t)\sqrt{E_s})^2}{N_0}}.$$

2.2 Low-density parity-check codes

This section is focused on the definition of a class of binary linear block codes: binary LDPC codes.

2.2.1 Definitions and notation for LDPC block codes

A binary code C of length n and 2^k codewords is linear if and only if its codewords form a k-dimensional vector subspace of the vector space of all the binary n-tuples. We define n as the length of the code, k as the dimension of the code and $R = \frac{k}{n}$ as the rate of the code.

A binary linear code $\mathcal{C}$ can also be described as the null space of its $r \times n$ parity-check matrix $\mathbf{H}$, where $r \geq n - k$, that is,

$$\mathcal{C} = \{\mathbf{T} \in \{0,1\}^n : \mathbf{H}\mathbf{T}^T = \mathbf{0} \pmod 2\}$$

where $\mathbf{T}$ is a codeword of length n and ^{T} denotes transposition.

The Hamming distance between two vectors $\mathbf{c}_1$ and $\mathbf{c}_2$ belonging to $\{0,1\}^n$, is the number of positions in which they differ. The *minimum* distance $d_{\min}$ of

a block code is defined as the minimum of the Hamming distances between two codewords. The sum of any two codewords of a linear code is also a codeword, and $d_{\min}$ is equal to the minimum Hamming weight over all non-zero codewords.

LDPC codes are among the most promising channel codes. They were first introduced by Gallager in his pioneering PhD thesis [4] and set aside for some decades, except for few notable works [15,16]. After the discovery of turbo codes [17], a great deal of attention was dedicated to iterative decoding techniques. Few years later, Spielman [18, 19] and MacKay and Neal [20], independently rediscovered LDPC codes.

An LDPC code is defined by a $r \times n$ sparse parity-check matrix containing relatively few 1's. The rate of such a code is $R \geq \frac{n-r}{n}$. If $\mathbf{H}$ contains exactly d_v ones in each column, the code is said to be *column-regular*; if it contains exactly d_c ones in each row, the code is *row-regular*. A code which is either row- and column-regular is said to be (d_v, d_c)-regular. If the code is irregular, we denote the weight of the jth column (ith row) of the parity-check matrix as $d_v^{(j)}$ ($d_c^{(i)}$).

The parity-check matrix is the bi-adjacency matrix of a bipartite graph, called Tanner graph [16]. The Tanner graph has n *variable* nodes and r *check* nodes; an edge connects a variable node to a check node if and only if the corresponding entry in $\mathbf{H}$ is 1. We denote the set of variable nodes as $\mathcal{V} = \{v_0, v_1, \ldots, v_{n-1}\}$ and the set of check nodes as $\mathcal{U} = \{u_0, u_1, \ldots, u_{r-1}\}$.

Example 2.2.1 Let us consider the following parity check matrix, not referred to an LDPC code

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

The corresponding Tanner graph is shown in Fig. 2.2, where circles and squares denote variable and check nodes, respectively.

Let us also introduce the following definitions:

- a closed walk in a graph consists of a sequence of nodes and edges, starting from and ending to the same node;
- a *simple cycle* (simply called cycle in the following) in a graph is a closed walk where repeated edges and nodes are not allowed;
- the *girth* g of a code is the length of the shortest cycle(s) in its Tanner graph.

Since the parity-check matrix is the bi-adjacency matrix of the Tanner graph of a code, cycles can be defined over such a matrix as well.

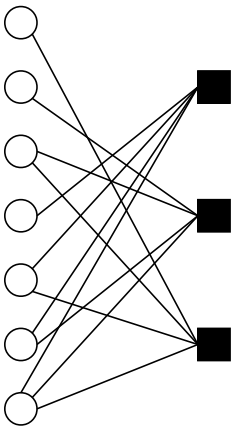


Figure 2.2: Tanner Graph of the parity-check matrix in Example 2.2.1

An additional constraint of LDPC codes is that their graph cannot contain cycles of length 4 which, according to the above definition, are the shortest possible cycles in a bipartite graph. In fact, as we shall discuss in the following chapters, short cycles limit the decoding performance of LDPC codes [21–23].

2.3 Classical and modern constructions of LDPC convolutional and block codes

In this section we discuss how LDPC convolutional codes (LDPC-CCs) can be obtained, with classical and modern methods and how they are linked to their block counterparts. LDPC-CCs were introduced in [5] and the analysis of their favorable characteristics endures to these days.

2.3.1 Classical construction of LDPC convolutional codes

In general, an LDPC-CC with rate $R_\infty = \frac{a-c}{a}$ can be defined as the set of sequences $\mathbf{c}_{[\infty]}$ satisfying $\mathbf{H}_{[\infty]}\mathbf{c}_{[\infty]}^T = \mathbf{0}$, where $\mathbf{c}_{[\infty]} = [\ldots, \mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_i, \ldots]$, being $\mathbf{c}_i$ a binary vector with length a, and

$$\mathbf{H}_{[\infty]}^T = \begin{bmatrix} \ddots & & \ddots & & \\ \mathbf{H}_0^T(0) & \ldots & \mathbf{H}_{m_s}^T(m_s) & & \\ & \ddots & \vdots & \ddots & \\ & & \mathbf{H}_0^T(\tau) & \ldots & \mathbf{H}_{m_s}^T(\tau+m_s) \\ & & \ddots & & \ddots \end{bmatrix}, \tag{2.1}$$

where each block $\mathbf{H}_i(\tau)$, $i = 0, 1, 2, \ldots, m_s$, is a binary matrix with size $c \times a$, having the following structure[1]

$$\mathbf{H}_i = \begin{bmatrix} h_i^{(0,0)} & h_i^{(0,1)} & \ldots & h_i^{(0,a-1)} \\ h_i^{(1,0)} & h_i^{(1,1)} & \ldots & h_i^{(1,a-1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_i^{(c-1,0)} & h_i^{(c-1,1)} & \ldots & h_i^{(c-1,a-1)} \end{bmatrix}.$$

We define a as the *period length* of the code, m_s as the syndrome former memory order and $v_s = (m_s + 1)a$ as the syndrome former constraint length. The following properties have to be satisfied:

- $\mathbf{H}_i(j) = 0$ for $i > m_s$, $\forall j$
- There exists a j such that $\mathbf{H}_{m_s}(j) \neq \mathbf{0}$
- $\mathbf{H}_0(j) \neq \mathbf{0}$ and has full rank, $\forall j$

We define any $\mathbf{H}_s(\tau) = \left[\mathbf{H}_0^T(\tau)|\mathbf{H}_1^T(\tau+1)|\ldots|\mathbf{H}_{m_s}^T(\tau+m_s)\right]$ as the τth syndrome former matrix. If $\mathbf{H}_i(\tau) = \mathbf{H}_i(\tau + \mathcal{T})$ for a finite value of $\mathcal{T}$, the corresponding code is said to be *periodically time-varying* with period $\mathcal{T}$. If the code is periodically time-varying with period $\mathcal{T} = 1$, we say that it is *time-invariant*. Time-invariant codes are characterized by a unique syndrome former matrix $\mathbf{H}_s$. We also define L_h, as the maximum spacing between any two ones appearing in any column of $\mathbf{H}_s^T$. The free Hamming distance of LDPC-CCs is denoted as d_{free}.

An obvious observation is that time-varying codes can be seen as time-invariant codes with larger syndrome former matrix. In fact, we can consider $\mathbf{H}_s$ as the collection of $\mathbf{H}_s^T(\tau)$, $\tau = 0, 1, \ldots, \mathcal{T} - 1$, as follows

$$\mathbf{H}_s = \begin{bmatrix} \mathbf{H}_0^T(0) & \ldots & \mathbf{H}_{m_s}^T(m_s) & & \\ & \ddots & \vdots & \ddots & \\ & & \mathbf{H}_0^T(\mathcal{T}-1) & \ldots & \mathbf{H}_{m_s}^T(\mathcal{T}-1+m_s) \end{bmatrix}$$

This way, an equivalent time-invariant code with period length $a\mathcal{T}$ and $c\mathcal{T}$ control symbols per period is obtained.

The symbolic representation of the syndrome former matrix of time-invariant LDPC-CCs exploits polynomials $\in F_2[D]$, where $F_2[D]$ is the ring of polynomials with coefficients in the binary Galois field. In this case, the code is described

[1] For the sake of readability, τ has been omitted.

by a $c \times a$ *symbolic matrix* having polynomial entries, that is

$$\mathbf{H}(D) = \begin{bmatrix} h_{0,0}(D) & h_{0,1}(D) & \dots & h_{0,a-1}(D) \\ h_{1,0}(D) & h_{1,1}(D) & \dots & h_{1,a-1}(D) \\ \vdots & \vdots & \ddots & \vdots \\ h_{c-1,0}(D) & h_{c-1,1}(D) & \dots & h_{c-1,a-1}(D) \end{bmatrix}, \tag{2.2}$$

where each $h_{i,j}(D)$, $i = 0, 1, 2, \dots, c-1$, $j = 0, 1, 2, \dots, a-1$, is a polynomial $\in F_2[D]$. The code representation based on $\mathbf{H}_s$ can be converted into that based on $\mathbf{H}(D)$ as follows

$$h_{i,j}(D) = \sum_{m=0}^{m_s} h_m^{(i,j)} D^m. \tag{2.3}$$

A binary codeword of the LDPC-CC is a semi-infinite binary row vector

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_0 & \mathbf{T}_1 & \dots \end{bmatrix},$$

where each $\mathbf{T}_i$ is an a-bit vector of the type

$$\mathbf{T}_i = \begin{bmatrix} T_i^{(0)} & T_i^{(1)} & \dots & T_i^{(a-1)} \end{bmatrix}$$

and such that

$$\mathbf{H}\mathbf{T}^T = \mathbf{0}.$$

A polynomial representation of the codewords is easily obtained using (2.3). The polynomial codeword corresponding to $\mathbf{T}$ is denoted as a row vector of polynomials $\mathbf{T}(D) = \begin{bmatrix} T_0(D) & \dots & T_{a-1}(D) \end{bmatrix}$, whose entries can be obtained as

$$T_j(D) = \sum_{i=0}^{\infty} T_i^{(j)} D^i \tag{2.4}$$

and such that

$$\mathbf{T}(D)\mathbf{H}^T(D) = \mathbf{0}.$$

The *support* $I_{\mathbf{T}}$ of a binary codeword $\mathbf{T}$ is the set of positions of the non-zero bits in $\mathbf{T}$. We also define the *support matrix* $\tilde{\mathbf{H}}$ as the sub-matrix containing only the columns of $\mathbf{H}$ whose indexes are in $I_{\mathbf{T}}$.

The symbolic representation can also be used if the code is periodically time-varying with period $\mathcal{T}$ [24], by placing $\mathcal{T}$ symbolic matrices, properly delayed, side-by-side, as in

$$\mathbf{H}(D) = \left[\mathbf{H}_0(D) | D\mathbf{H}_1(D) | \dots | D^{\mathcal{T}-1}\mathbf{H}_{\mathcal{T}-1}(D)\right],$$

where each $\mathbf{H}_i(D)$ is the symbolic matrix of $\mathbf{H}_s(i)$. In this case, the symbolic matrix contains, at most, $\mathcal{T}ca$ polynomial entries.

We define $\mathcal{W}(\cdot)$ as the function that returns the weight of its argument. If the argument of the function is a polynomial matrix (or vector), $\mathcal{W}(\cdot)$ results in a matrix (or vector) containing the number of non-zero coefficients of its entries. On the other hand, if the argument of $\mathcal{W}(\cdot)$ is a binary vector, the function results in its total weight.

We also define $\mathbf{W}$ as a diagonal square matrix, with size a, such that its entry at position (i,i) is $\sum_{j=0}^{c-1}\mathcal{W}(h_{j,i}(D))$, $\forall i = 0,\ldots,a-1$, that is,

$$\mathbf{W} = \begin{bmatrix} w_0 & 0 & \ldots & 0 \\ 0 & w_1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & w_{a-1} \end{bmatrix}. \tag{2.5}$$

An *ensemble of codes* is the collection of all the codes characterized by the same $\mathcal{W}(\mathbf{H}(D))$.

A code whose symbolic matrix contains only monomial or null entries is called here Type-1 code, a code having also binomial entries is called Type-2 code, and so on. A code whose symbolic matrix contains only monomial entries is called *monomial* code.

The matrix $\mathbf{H}(D)$ can be described by the exponents of D without loss of information. This permits us to define the exponent matrix $\mathbf{P}$ of Type-1 codes such that $p_{i,j} = \log_D(h_{i,j}(D))$; conventionally, $\log_D(0)$ is denoted as -1.

It is shown in [25, Lemma 6] that, given a SC-LDPC-CC described by $\mathbf{H}(D)$, there exist $\binom{a}{c+1}$ codewords, denoted as $\mathbf{U}^{(l)}(D) = \begin{bmatrix} U_0^{(l)}(D) & \ldots & U_{a-1}^{(l)}(D) \end{bmatrix}$, $l \in [0,\ldots,\binom{a}{c+1}-1]$, whose entries are

$$U_i^{(l)}(D) = \begin{cases} \mathrm{perm}(\mathbf{H}_{\mathcal{L}\setminus i}(D)) & \text{if } i \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where $\mathcal{L}$ is the lth size-$(c+1)$ subset of $[0,\ldots,a-1]$ and $\mathrm{perm}(\cdot)$ returns the permanent of its argument; $\mathbf{H}_{\mathcal{L}\setminus i}(D)$ is the submatrix of $\mathbf{H}(D)$ that contains only the columns of $\mathbf{H}(D)$ with indexes $\in \mathcal{L}\setminus i$. Notice that these codewords always exist, hence are called *unavoidable codewords*.

Based on unavoidable codewords, an upper bound on the free Hamming distance of the code can be derived as

$$d_{\mathrm{free}} \leq \min_{\substack{\mathcal{L}\subseteq[0,a-1] \\ |\mathcal{L}|=c+1}}{}^{*} \sum_{i\in\mathcal{L}} \mathrm{perm}(\mathcal{W}(\mathbf{H}_{\mathcal{L}\setminus i}(D))), \tag{2.7}$$

where the $\min^*$ operator returns the smallest positive entry of the list if the

list contains positive entries, otherwise it returns $+\infty$ [25].

Given two polynomials $p(D)$ and $z(D)$, we say that $p(D) \subseteq z(D)$ if $p(D)$ contains only monomials that are included in $z(D)$. For a given $\mathbf{H}(D)$, we define *component code* any code described by a column-regular matrix $\mathbf{H}_c(D)$ with $d_v = 2$ such that any of its entries $h_{c_{i,j}}(D) \subseteq h_{i,j}(D)$. The codewords of any component code are denoted as *component codewords*.

LDPC-CCs can be obtained by *unwrapping* LDPC block codes. Even though the term "unwrapping" was introduced in [26], we discuss here the unwrapping procedure described in [5]. Let us consider an $r \times n$ parity-check matrix and let $\eta = \gcd(r, n)$. A *diagonal cut* is performed by alternatively moving $a = \frac{n}{\eta}$ units to the right and $c = \frac{r}{\eta}$ units down. The resulting LDPC-CC has asymptotic rate $R_\infty = \frac{a-c}{a}$, $m_s = \eta - 1$ and $v_s = n$. This unwrapping procedure has been later generalized to obtain more general classes of LDPC-CCs in [27].

Example 2.3.1 Let us consider a $(3, 6)$-regular block code, defined by the following parity-check matrix

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

We have $\eta = 6$, so the cut is performed by moving 2 units to the right and 1 down, and the following submatrix is obtained

$$\mathbf{H}_s^T = \begin{bmatrix} 1 & 0 & & & & & & & & & & \\ 0 & 1 & 0 & 0 & & & & & & & & \\ 0 & 1 & 0 & 1 & 0 & 0 & & & & & & \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & & & & \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & & \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ & & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ & & & & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ & & & & & & 0 & 1 & 1 & 1 & 1 & 0 \\ & & & & & & & & 0 & 0 & 0 & 1 \\ & & & & & & & & & & 1 & 1 \end{bmatrix},$$

which can be replicated in order to obtain a time-varying LDPC-CC with period $\mathcal{T} = 6$.

2.3.2 Quasi-cyclic LDPC block codes from LDPC convolutional codes

A time-invariant LDPC-CC can be terminated in a *tail-biting* fashion by adding the rows last $m_s c$ rows of a truncated version of the parity-check matrix to its first $m_s c$ rows [28, 29], thus obtaining

$$\mathbf{H}^{\text{tb}}_{[0,L-1]} = \begin{bmatrix} \mathbf{H}_0 & & & & \mathbf{H}_{m_s} & \dots & \mathbf{H}_1 \\ \vdots & \mathbf{H}_0 & & & & \ddots & \vdots \\ \mathbf{H}_{m_s-1} & \vdots & \ddots & & & & \mathbf{H}_{m_s} \\ \mathbf{H}_{m_s} & \mathbf{H}_{m_s-1} & & \mathbf{H}_0 & & & \\ & \mathbf{H}_{m_s} & \ddots & \vdots & \mathbf{H}_0 & & \\ & & \ddots & \mathbf{H}_{m_s-1} & \vdots & \ddots & \\ & & & \mathbf{H}_{m_s} & \mathbf{H}_{m_s-1} & \dots & \mathbf{H}_0 \end{bmatrix}. \quad (2.8)$$

Equation (2.8) defines a QC-LDPC code, according to the definition given in [30]. The most common representation of the parity-check matrix of a QC-LDPC code was independently given in [31] and is as follows

$$\mathbf{H}_{\text{QC}} = \begin{bmatrix} \mathbf{Q}^{(0,0)} & \dots & \mathbf{Q}^{(0,a-1)} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}^{(c-1,0)} & \dots & \mathbf{Q}^{(c-1,a-1)} \end{bmatrix}, \quad (2.9)$$

where each $\mathbf{Q}^{(i,j)}$, $i \in [0, 1, \dots, c-1]$, $j \in [0, 1, \dots, a-1]$ is a square circulant matrix of size L.

In order to pass from (2.8) to (2.9) it is possible to adopt the following procedure:

1) Apply a permutation $\pi_1 : \{0, 1, \dots aL-1\} \rightarrow \{0, 1, \dots aL-1\}$ to the columns of $\mathbf{H}^{\text{tb}}$ such that $\pi_1(j) = \lfloor \frac{j}{L} \rfloor + (j \mod L)a$, $j = 0, 1, \dots, aL-1$.

2) Apply a permutation $\pi_2 : \{0, 1, \dots n-1\} \rightarrow \{0, 1, \dots n-1\}$ to the columns of the parity-check matrix obtained in step 1) such that $\pi_2(i) = \lfloor \frac{i}{L} \rfloor + (i \mod L)c$, $i = 0, 1, \dots, cL-1$.

The notation can be further simplified if we consider that any circulant matrix of size L is isomorphic to a polynomial $\in F_2[x]_{(x^L+1)}$, the ring of polynomials with coefficients in the binary Galois field, modulo $(x^L + 1)$. The resulting symbolic representation is as in (2.10)

$$\mathbf{H}(x) = \begin{bmatrix} h_{0,0}(x) & h_{0,1}(x) & \dots & h_{0,a-1}(x) \\ h_{1,0}(x) & h_{1,1}(x) & \dots & h_{1,a-1}(x) \\ \vdots & \vdots & \ddots & \vdots \\ h_{c-1,0}(x) & h_{c-1,1}(x) & \dots & h_{c-1,a-1}(x) \end{bmatrix}, \tag{2.10}$$

where $h_{i,j}(x)$ is a polynomial $\in F_2[x]_{(x^L+1)}$.

The similarities between (2.2) and (2.10) show how evanescent is the difference between QC-LDPC block codes and LDPC-CCs: L is finite for the former ones, $L \to \infty$ for the latter ones. This was first observed in [26] and later reformulated in [32].

2.3.3 Protograph based LDPC codes

In this section we describe how LDPC codes can be obtained from protographs [33]. In the following we consider the design of both block and convolutional binary codes from this basic graphical structure.

Protograph based LDPC block codes

Let us consider a small bipartite Tanner graph, called *protograph*, with n_v variable nodes and n_c check nodes, connected each other by edges. A protograph based LDPC block code with blocklength $n = Mn_v$ can be obtained by taking an $M-$*lifting* of the protograph. When lifting the protograph, each node becomes a group of M nodes, and so for each edge. These edges are then permuted according to some rule. The rate of the graph obtained with such a procedure is that of the initial protograph, that is, $R = \frac{n_v - n_c}{n_v}$. The protograph can be unambiguously described by a *base matrix* $\mathbf{B}$, such that each of its entries $B_{i,j}$ is the number of edges connecting jth variable node to the ith check node. The $M-$lifting operation corresponds to the substitution of all-zero $M \times M$ matrices to the zeros of $\mathbf{B}$, $M \times M$ permutation matrices to the ones of $\mathbf{B}$, and sum of $M \times M$ non-overlapping permutation matrices to the entries of $\mathbf{B}$ which are larger than 1. We remark that, if circulant permutation matrices are used instead of random permutation matrices, QC-LDPC codes are obtained.

Example 2.3.2 Let us consider the base matrix

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

The corresponding protograph is shown in Fig. 2.3.

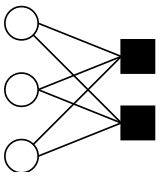


Figure 2.3: Protograph corresponding to the base matrix in Example 2.3.2

The protograph obtained after a generic lifting operation is shown in Fig. 2.4, where the permutations applied to the bundles of edges are denoted as π_i, $i = 1, \ldots, 6$.

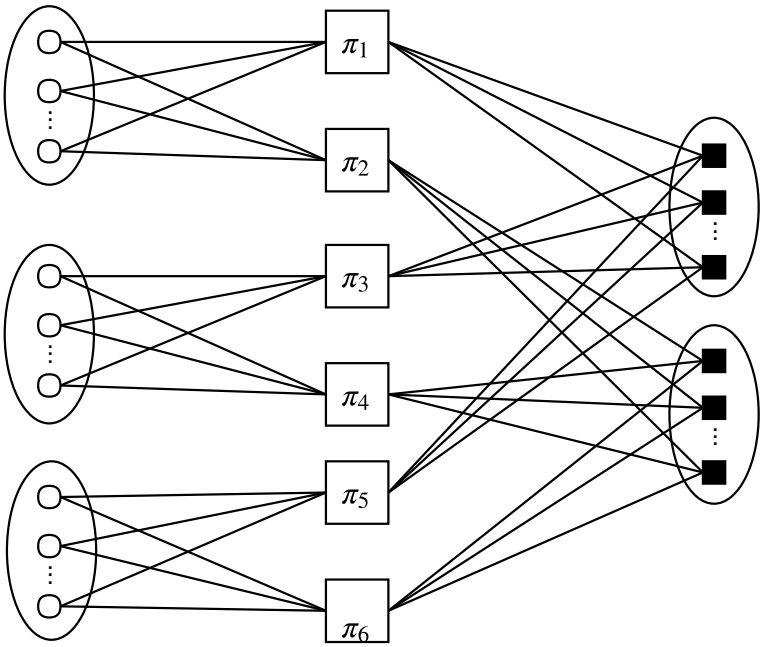


Figure 2.4: Example of a lifted protograph

Spatially coupled LDPC convolutional codes

A *convolutional* protograph can be obtained by coupling together a number of disjoint protographs. We follow the notation given in [34].

Let us consider a protograph with b_c check nodes and b_v variable nodes, described by base matrix $\mathbf{B}$, and let us copy it L times. We can associate each of the copies to a time instant τ. Suppose that the jth variable node is connected to the ith check node by $B_{i,j}$ edges in each protograph, where $i \in [0, 1, \ldots, b_c - 1]$ and $j \in [0, 1, \ldots, b_v - 1]$. These $B_{i,j}$ edges are spread from

the jth variable node at time instant τ to the check nodes of ith type at time instants $\tau, \tau+1, \ldots, \tau+w$, where w is the memory of the code. This procedure is usually addressed to as *edge spreading* procedure.

SC-LDPC-CCs are obtained by taking $M-$liftings of convolutional protographs. The new base matrix, denoted as $\mathbf{B}_{[-\infty,+\infty]}$, corresponds to a convolutional protograph (*convolutional base matrix*) and has the following form

$$\mathbf{B}_{[-\infty,+\infty]} = \begin{bmatrix} \ddots & \ddots & & \ddots & & & \\ & \mathbf{B}_w & \mathbf{B}_{w-1} & \cdots & \mathbf{B}_0 & & \\ & & \mathbf{B}_w & \mathbf{B}_{w-1} & \cdots & \mathbf{B}_0 & \\ & & & \ddots & \ddots & & \ddots \end{bmatrix},$$

where each $\mathbf{B}_i$, $i = 0, \ldots, w$ is a $b_c \times b_v$ matrix such that

$$\sum_{i=0}^{w} \mathbf{B}_i = \mathbf{B}.$$

Notice that if the elements in $\mathbf{B}_i$, $i = 0, \ldots, w$, belong to $\{0, 1\}$, the convolutional base matrix can be seen as the parity-check matrix of a convolutional code, with no need for a further lifting stage. The asymptotic rate of the corresponding convolutional graph is $R_\infty = \frac{b_v - b_c}{b_v}$.

2.4 Summary

In this chapter we reminded some fundamentals of error coding schemes for data reliability. The main definitions and the basic notation for the families of LDPC codes we consider have been recalled. The most common construction methods for block and convolutional LDPC codes have been discussed.

Chapter 3

Finite length and asymptotic performance of LDPC codes

In this chapter we describe the decoder we consider in the Monte Carlo simulations through which the finite-length performance of LDPC codes is assessed. Gallager [4] firstly proposed a message-passing algorithm for decoding of LDPC codes, which later became known as the sum product algorithm (SPA). It was later recognized in [35] that the SPA is a special case of belief propagation (BP) [36, 37]. We focus here on the log likelihood ratio (LLR) version of the SPA.

The average performance of LDPC block code ensembles can be estimated by means of techniques, such as density evolution (DE) [38] and extrinsic information transfer (EXIT) analysis [39]. Unfortunately, these tools may not be able to catch the structural properties of the protograph, and more advanced techniques, such as, for example, PEXIT analysis [40] represent a fitting instrument to estimate more tightly and eventually optimize the average performance of LDPC code ensembles based on protographs. Nevertheless, these tools are accurate only if the blocklength is very large, ideally $n \to \infty$[1], and if the number of decoding iterations is also very large, ideally tending to infinity. A further assumption on which these tools rely is that the code Tanner graph contains no cycles, i.e. it is a tree. This is the reason why, when we consider practical codes, PEXIT analysis (as well as the other techniques) only permits to estimate where the so-called waterfall of the error rate performance is going to happen, and gives no indications on the performance in the error-floor region, which heavily depends on the peculiar characteristics of each individual code in the ensemble.

Finally, we recall the concept of sliding window decoding and describe how it can be used to perform LLR-SPA decoding on SC-LDPC-CCs; the extension to PEXIT analysis is straightforward. In fact, due to the infinite length of these codes, the tools which are conventionally used to decode LDPC codes and to predict their asymptotic performance must be revisited, in such a way to work

[1] According to the notation in Section 2.3.3, this happens when $M \to \infty$.

on a sliding window, processing a relatively small portion of the parity-check matrix at a time [5, 6, 41].

3.1 Iterative decoding techniques: the LLR-SPA

Owing to the sparse nature of their parity-check matrix, LDPC codes can be efficiently decoded with *message-passing* algorithms working on the code Tanner graph. We focus on the LLR version of the SPA, as the logarithmic metric allows computational and performance advantages over the linear version.

Let us introduce few more definitions:

- $\hat{\mathbf{x}} = [\hat{x}_0, \ldots, \hat{x}^{n-1}]$ is the estimated codeword
- $\mathbf{y} = [y_0, \ldots, y^{n-1}]$ is the received sequence
- $\mathcal{V}^{(j)}$ is the set of variable nodes connected to u_j
- $\mathcal{V}^{(j)} \backslash i$ is the set of variable nodes connected to u_j, except for v_i
- $\mathcal{U}^{(i)}$ is the set of check nodes connected to u_i
- $\mathcal{U}^{(i)} \backslash j$ is the set of check nodes connected to v_i, except for u_j
- $q_{i \to j}(x)$, $x \in 0, 1$, is the information sent by v_i to u_j
- $r_{j \to i}(x)$, $x \in 0, 1$, is the information sent by u_j to v_i
- the LLR of a random variable X is defined as

$$L(X) = \ln \frac{P(X = 0)}{P(X = 1)},$$

  where $P(X = x)$ denotes the probability that X assumes the value x.

In the LLR-SPA, the messages sent from the the ith variable node to the jth check node are defined as

$$\Gamma_{i \to j}(x_i) = \ln \frac{q_{i \to j}(0)}{q_{i \to j}(1)},$$

whereas the messages sent from the jth check node to the ith variable node are defined as

$$\Lambda_{j \to i}(x_i) = \ln \frac{r_{j \to i}(0)}{r_{j \to i}(1)}.$$

In the following, the four main steps of the decoding process are described.

3.1.1 Initialization

As a first step, the initial values of the messages are initialized as follows

$$\Gamma_{i\to j}(x_i) = L(x_i),$$

$$\Lambda_{j\to i}(x_i) = 0,$$

where

$$L(x_i) = \ln \frac{P(x_i = 0|y_i = y)}{P(x_i = 1|y_i = y)}$$

and $P(x_i = x|y_i = y)$, $x \in \{0, 1\}$ is the probability that x_i is equal to x, given that y_i is equal to y.

3.1.2 Left semi-iteration

In the second step of the algorithm, the messages sent from the check nodes to the variable nodes are updated as follows

$$\Lambda_{j\to i}(x_i) = 2\tanh^{-1}\left\{\prod_{k\in\mathcal{V}^{(j)}\setminus i} \tanh\left[\frac{1}{2}\Gamma_{k\to j}(x_k)\right]\right\}$$

3.1.3 Right semi-iteration

In the penultimate step, the messages sent from the variable nodes to the check nodes are computed as follows

$$\Gamma_{i\to j}(x_i) = L(x_i) + \sum_{k\in\mathcal{U}^{(i)}\setminus j} \Lambda_{k\to i}(x_i).$$

An additional quantity, to be used in the last step of the procedure, is calculated as

$$\Gamma_i(x_i) = L(x_i) + \sum_{k\in\mathcal{U}^{(i)}} \Lambda_{k\to i}(x_i).$$

3.1.4 Decision

In this final step, the bits of the received codeword are estimated according to the following rule

$$\hat{x}_i = \begin{cases} 0 & \text{if} \quad \Gamma_i(x_i) \geq 0 \\ 1 & \text{if} \quad \Gamma_i(x_i) < 0. \end{cases}$$

If $\hat{\mathbf{x}}\mathbf{H}^T = \mathbf{0}$ decoding stops and outputs the estimated received codeword. Otherwise, the algorithm goes back to step 2, and the procedure is repeated, until the parity-check equations system is satisfied or a prefixed number of iterations is reached.

3.1.5 Decoding complexity and latency

The decoding complexity of any decoder can be measured as the number of binary iterations required per decoding instance. We now define a complexity metric for the decoder described in the previous subsection, referring to the implementation proposed in [42]. The maximum decoding complexity is

$$C = I_{\max}(C_{\text{LSI}} + C_{\text{RSI}} + C_{\text{PC}})$$

where $I_{\max}$ is the maximum number of decoding iterations and C_{LSI}, C_{RSI} and C_{PC} represent the number of binary operations required by the left semi-iteration, the right semi-iteration and the decision and parity-check steps, given by the following expressions

$$\begin{cases} C_{\text{LSI}} = 6d_c^{\text{avg}} - 2qn(1-R) \\ C_{\text{RSI}} = (2d_v^{\text{avg}} + 1)qn \\ C_{\text{PC}} = d_v^{\text{avg}} n \end{cases}$$

where q is the number of quantization bits used inside the decoder (we will consider $q = 8$) and, with reference to the general case of irregular codes, d_v^{avg} and d_c^{avg} are the average column and row weight, respectively.

Considering that $d_v^{\text{avg}} = d_c^{\text{avg}}(1-R)$, we derive

$$C = nI_{\max}[q(8d_v^{\text{avg}} + 12R - 11) + d_v^{\text{avg}}] = nI_{\text{avg}} f(q, d_v^{\text{avg}}, R),$$

where f states for a function of its arguments. The per-bit decoding complexity can be obtained as $\frac{C}{n}$.

Another fundamental quantity that will be considered in the following is the decoding latency, expressed as the number of bits that must be awaited before the decoding process starts, that is

$$\Lambda = n.$$

3.2 Protograph extrinsic information transfer analysis

In this section we recall how the PEXIT analysis can be performed on LDPC block codes, considering the AWGN channel. This tool allows to analyze the convergence behavior of protograph-based codes, based on mutual information metrics.

3.2.1 PEXIT analysis for block codes

Following the notation in [40], we define

- $\mathbf{B}$ as an $n_c \times n_v$ base matrix; the rate of the corresponding code is $R = \frac{n_v - n_c}{n_v}$.
- $I_{E_v}(i,j)$ ($I_{E_c}(i,j)$) as the extrinsic mutual information (MI) between the message sent by variable node v_j (check node u_i) to check node u_i (variable node v_j) and the codeword bit associated to the variable node sending (receiving) the message.
- $I_{A_v}(i,j)$ ($I_{A_c}(i,j)$) as the a priori MI between the message sent to v_j (u_i) by u_i (v_j) and the codeword bit associated to the variable node receiving (sending) the message.
- $I_{\mathrm{APP}}(j)$ as the MI between the a posteriori probability log-likelihood ratio

$$\ln \frac{\Pr(x_j = +1|\mathbf{r})}{\Pr(x_j = -1|\mathbf{r})},$$

  where $\mathbf{r}$ is the received vector, evaluated by the v_j and the associated codeword bit x_j.
- $I_{ch} = J(\sigma_n)$ as the MI between a binary random variable X, with $\Pr(X = +\mu) = \Pr(X = -\mu) = 1/2$, and a continuous Gaussian-distributed random variable Y with mean X and variance $\sigma_n^2 = 2\mu$. I_{ch} basically represents the capacity of a binary-input AWGN.

The functions $J(\cdot)$ and $J^{-1}(\cdot)$ are calculated in the following using the polynomial approximation proposed in [43].

Initialization

The first step consists in assigning a value to $I_{ch}^{(j)}$, $\forall j = 0, 1, \ldots, n_v - 1$ as

$$I_{ch}^{(j)} = J(\sigma_{ch,j}),$$

with

$$\sigma_{ch,j} = 8R\frac{E_b}{N_0}\Big|_{v_j}$$

where $\frac{E_b}{N_0}\Big|_{v_j}$ represents the signal-to-noise ratio associated to v_j.

Then, set $I_{A_v}(i,j) = I_{ch}^{(j)}$, $\forall i,j$ such that $B_{i,j} \neq 0$.

Variable to check update

In the left to right semi-iteration, $\forall i,j$ such that $B_{i,j} \neq 0$ compute $I_{E_v}(i,j)$ as follows

$$I_{E_v}(i,j) = J\left(\sqrt{\sum_{s\neq i} B_{s,j}\left[J^{-1}(I_{A_v}(s,j))\right]^2 + (B_{i,j}-1)\left[J^{-1}(I_{A_v}(i,j))\right]^2 + \left[J^{-1}(I_{ch}^{(j)})\right]^2}\right).$$

Then, set $I_{A_c}(i,j) = I_{E_v}(i,j)$.

Check to variable update

In this step compute $I_{E_c}(i,j)$, $\forall i,j$ such that $B_{i,j} \neq 0$, as follows

$$I_{E_c}(i,j) = 1 - J\left(\sqrt{\sum_{s\neq i} B_{i,s}\left[J^{-1}(1-I_{A_c}(i,s))\right]^2 + (B_{i,j}-1)\left[J^{-1}(1-I_{A_c}(i,j))\right]^2}\right)$$

and, then, set $I_{A_v}(i,j) = I_{E_c}(i,j)$.

Evaluation

Compute the following quantity,

$$I_{\mathrm{APP}}(k) = J\left(\sqrt{\sum_{s} B_{s,k}\left[J^{-1}(I_{A_v}(s,k))\right]^2 + \left[J^{-1}(I_{ch}^{(k)})\right]^2}\right)$$

with $k \in [0, n_v - 1]$.

If all the elements in I_{APP} are greater than or equal to $1-\xi$, where ξ is a small enough quantity (e.g., $\xi \approx 10^{-2}$), we say that the algorithm converges, and the analysis is terminated.

Otherwise, repeat the previous steps until the above condition is met, or a prefixed number of iterations is reached.

The PEXIT threshold $\left(\frac{E_b}{N_0}\right)^*$ is defined as the lowest value of $\frac{E_b}{N_0}$ such that the algorithm converges.

3.3 Sliding window

Using only a portion of the parity-check matrix to assess the asymptotic and finite-length performance of SC-LDPC-CCs is feasible because, due to the diagonal stair-like structure of the parity-check matrix, variable nodes that are $m_s + 1$ time instants apart from each other cannot be connected to the same check node. We need to be aware that, when a sliding window decoder is used, there is a trade-off between decoding performance and complexity. However, the performance loss can be overcome by choosing a sufficiently large window size.

Decoding can be performed over a window of size W, measured in blocks, at a time; when decoding over the current window terminates, a decision on the first $a = b_v M$ symbols is taken. In the next step, decoding restarts over a new window, obtained by translating the previous window downwards by $c = b_c M$ positions and towards right by a positions in the parity check matrix representation of the code. Experimental results show that values of $W \approx (5 \div 10)(m_s + 1)$ results in significant savings in terms of latency and memory with minimal performance degradation [44]. The decoding algorithm can be summarized as follows. At the beginning, the decoder executes the LLR-SPA over the first window with the aim of decoding the first a symbols in the window, called *target symbols*. Decoding is performed starting from the vector of LLR values for the received symbols. A right message vector and a left message vector are iteratively updated according to the LLR-SPA check and variable node update rules. If no early termination criteria are used, after a fixed number of iterations, the window slides down by c rows and right by a columns, and decoding continues over the next window. Any decoded block is kept in the memory for the next m_s window configurations, but the corresponding variable nodes are no longer updated. A representation of sliding window (SW) decoding performed over the parity-check matrix of a protograph-based time invariant SC-LDPC-CC is provided in Fig. 3.1.

3.3.1 Decoding latency and complexity of the sliding window decoder

Following the approach given in Section 3.1.5, we express the decoding latency (Λ_{SW}) and average per-output-bit complexity (Γ_{SW}) of a SW decoder as

$$\begin{cases} \Lambda_{\mathrm{SW}} = Wa = \alpha(m_s+1)a, \\ \Gamma_{\mathrm{SW}} = \dfrac{WaI_{\mathrm{avg}} f(q, d_v^{\mathrm{avg}}, R_\infty)}{a} \\ \qquad\quad = \alpha(m_s+1) I_{\mathrm{avg}} f(q, d_v^{\mathrm{avg}}, R_\infty). \end{cases} \tag{3.1}$$

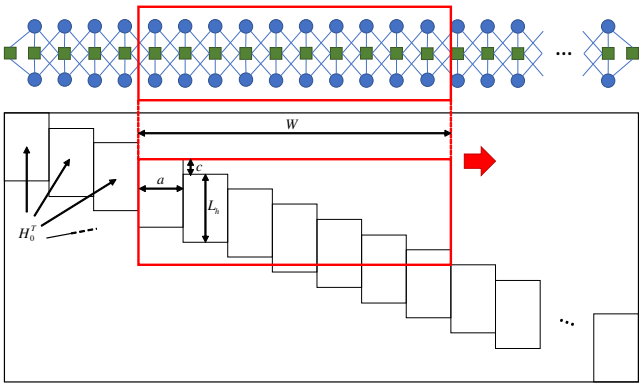


Figure 3.1: Representation of sliding window decoding over coupled protographs and the corresponding semi-infinite parity-check matrix.

In fact, the SW decoder has to wait for Wa bits before it can start decoding. Then, even though the SW decoder works on Wa bits at a time, it returns a bits per window, independently of the window size W.

Note that SC-LDPC-CCs characterized by small values of m_s can be decoded with small window sizes. According to (3.1), this results in a reduction of both the decoding latency and per-output-bit complexity. Motivated by this, in the next chapter we address the design of SC-LDPC-CCs characterized by small values of m_s.

3.4 Summary

In this section we have described the iterative decoder we consider in the Monte Carlo simulations we perform in order to assess the finite-length performance of LDPC codes. The complexity and latency required by this tool has been derived. We have also considered PEXIT analysis which, under asymptotic assumptions, allows efficient estimation of the code error rate performance in the waterfall region.

Chapter 4

SC-LDPC convolutional codes with small constraint length

In this chapter we address the construction of SC-LDPC-CCs with small constraint length. According to (3.1), codes with small constraint length can be decoded with reduced latency and complexity. We also provide theoretical lower bounds on their constraint length, under constraints on the minimum length of cycles in their Tanner graphs.

According to the discussion in Sections 2.3.1 and 2.3.2, SC-LDPC-CCs can be obtained starting from QC-LDPC block codes [45] and applying suitable unwrapping techniques [5,26,27]. Freedom in the code design can be increased by avoiding this intermediate step and designing SC-LDPC-CCs directly. Such an approach has been followed, for example, in [46, 47] for codes having rates $R = \frac{a-1}{a}$, and extended in [48] to codes with rate $R = \frac{a-c}{a}$. As we mentioned, the performance of iterative decoders is adversely affected by the presence of cycles with short length and, in particular, of length 4 in the code Tanner graph. Therefore, code design approaches often aim at increasing the minimum length of cycles. In this chapter we show how SC-LDPC-CCs with small syndrome former constraint length and without short cycles can be effectively designed. Monte Carlo simulations show that good error rate performance can be achieved with reduced complexity with respect to previous solutions. The proposed method uses the symbolic parity-check matrix that, in the form commonly considered in the literature [49–51], has some fixed entries. We show that the removal of this constraint makes investigations more general and yields significant reductions in terms of constraint length. We also derive theoretical lower bounds on the constraint length for many families of SC-LDPC-CCs, under constraints on the girth, and we provide several explicit examples of codes that are able to reach, or at least approach, these bounds [52].

4.1 Notation

Let us consider the parity-check matrix of a time invariant SC-LDPC-CC. We describe the matrix $\mathbf{H}_s^T$ through a set of integer values representing the position differences between each pair of ones in its columns. We denote the position difference between a pair of ones in a column of $\mathbf{H}_s^T$ as $\delta_{i,j}$, where i is the column index of $\mathbf{H}_s^T$ ($i = 0, 1, 2, \ldots, a-1$) and j is the row index of $\mathbf{H}_s^T$ corresponding to the first of the two symbols 1 involving the difference ($j \in [0, 1, 2, \ldots, L_h - 2]$). The index of the second symbol 1 involved in the difference is simply found as $j + \delta_{i,j}$. For each difference, we also compute the values of two *levels* which are relative to the parameter c. The *starting level* is defined as $s_l = j \mod c$, while the *ending level* is defined as $e_l = (j + \delta_{i,j}) \mod c$. Both levels obviously take values in $\{0, 1, 2 \ldots, c-1\}$. Based on this representation of $\mathbf{H}_s^T$, a cycle is associated to a sum of the type $\delta_{i_1,j_1} \pm \delta_{i_2,j_2} \pm \ldots \pm \delta_{i_\lambda,j_\lambda} = 0$, and the length of the cycle is 2λ, with λ being an integer > 1. Not all the possible sums or differences of $\delta_{i,j}$ generate cycles. In fact, δ_{i_1,j_1} can be added to δ_{i_2,j_2} in order to form a cycle if and only if the starting level of the latter coincides with the ending level of the former. Instead, δ_{i_1,j_1} can be subtracted to δ_{i_2,j_2} in order to form a cycle if and only if their ending levels coincide. In addition, the starting level of the first difference and the starting level of the last difference in $\delta_{i_1,j_1} \pm \delta_{i_2,j_2} \pm \ldots - \delta_{i_\lambda,j_\lambda}$ must coincide. Similarly, the starting level of the first difference and the ending level of the last difference in $\delta_{i_1,j_1} \pm \delta_{i_2,j_2} \pm \ldots + \delta_{i_\lambda,j_\lambda}$ must coincide. Based on these considerations, it is possible to obtain lower bounds on the constraint length which is needed to avoid cycles up to a given length, as described in the next sections.

4.2 Lower bounds on the constraint length

For the sake of convenience, according to the definition given in Section 2.3.1, we distinguish between the case of Type-1 codes and Type-z codes, with $z > 1$.

The bounds are described in terms of m_s when the analysis is performed on $\mathbf{H}(D)$ and in terms of L_h when $\mathbf{H}_s^T$ is studied. We remind that these parameters are directly related to the syndrome former constraint length v_s. So, the bounds on v_s can be easily computed as well.

4.2.1 Type-1 codes

Lower bounds on m_s

The following lemma provides a necessary condition on m_s for fulfilling the condition $g \geq 6$. Longer girths are studied next.

Lemma 4.2.1 A Type-1 code with with $g \geq 6$, has

$$m_s \geq \left\lceil \frac{a-1}{2} \right\rceil. \tag{4.1}$$

Proof. The difference between any two exponents in a column of $\mathbf{H}(D)$ takes values in $[-m_s, m_s]$. Any difference must appear only once, or $g = 4$. Exploiting all values in $[-m_s, m_s]$ to design the matrix columns, we obtain $a = 2m_s+1$, from which (4.1) is derived. ■

A very important property of Type-1 codes with parity-check matrix column weight $d_v = c = 2$ is that cycles of length $2(2k + 1)$, $k = 1, 2, \ldots$, cannot exist because of structural characteristics. In fact, in Type-1 codes with $d_v = c = 2$, any difference starting from the first level ends in the second level, and vice versa; this means that

$$s_{li} \neq e_{li} \quad \forall i. \tag{4.2}$$

In order to form a cycle of length 2λ, the starting level of the first difference and the starting level of the last difference in $\delta_{i_0,j_0}^{(s_{l0} \to e_{l0})} \pm \delta_{i_1,j_1}^{(s_{l1} \to e_{l1})} \pm \ldots - \delta_{i_\lambda,j_\lambda}^{(s_{l\lambda} \to e_{l\lambda})} = 0$ must coincide. Similarly, the starting level of the first difference and the ending level of the last difference in $\delta_{i_0,j_0}^{(s_{l0} \to e_{l0})} \pm \delta_{i_1,j_1}^{(s_{l1} \to e_{l1})} \pm \ldots + \delta_{i_\lambda,j_\lambda}^{(s_{l\lambda} \to e_{l\lambda})} = 0$ have to be the same. If $\lambda = 2k + 1$, (4.2) forces $s_{l0} \neq s_{l(2k+1)}$ in the first case, and $s_{l0} \neq e_{l(2k+1)}$ in the second case, thus preventing the occurrence of the cycle. Therefore, cycles of length 6 cannot exist when $d_v = c = 2$.

In order to find a lower bound on m_s for $d_v = c = 3$, we can proceed as follows. Let us consider $\mathbf{H}(D)$ and its corresponding exponent matrix $\mathbf{P}$. From the latter we can compute the matrix $\Delta\mathbf{P}$, where $\Delta p_{0,j} = p_{1,j} - p_{0,j}$, $\Delta p_{1,j} = p_{2,j} - p_{1,j}$, $\Delta p_{2,j} = p_{2,j} - p_{0,j}$. Obviously, $\Delta p_{0,j} + \Delta p_{1,j} = \Delta p_{2,j}$. To satisfy $g \geq 6$ it must be $\Delta p_{i,j_1} \neq \Delta p_{i,j_2}$, $\forall i$, $\forall j_1 \neq j_2$. It is proved in [50] that a cycle of length six must involve three columns and three rows of $\mathbf{H}(D)$, this means that the equality $\Delta p_{0,j_1} + \Delta p_{1,j_2} = \Delta p_{2,j_3}$ covers all possible length-6 cycles.

Lemma 4.2.2 The elements of $\Delta\mathbf{P}$ for a monomial code with $d_v = c = 3$ whose Tanner graph has $g \geq 8$ satisfy the following properties:

i. $\Delta p_{i,j} \in [-m_s, m_s]$, $\forall i, j$

ii. $\Delta p_{i,j_1} \neq \Delta p_{i,j_2}$, $\forall i$, $\forall j_1 \neq j_2$

iii. $\Delta p_{0,j_1} + \Delta p_{1,j_2} \neq \Delta p_{2,j_3}$, $\forall j_1 \neq j_2, j_3$, $\forall j_2 \neq j_3$

Proof.

i. Any $p_{i,j} \in [0, m_s]$, $\forall i, j$, by definition. Since any $\Delta p_{i,j}$ is a difference between two entries of $\mathbf{P}$, it can take values in $[-m_s, m_s]$.

ii. In order to have $g \geq 6$, it must be $p_{i_1,j_1} - p_{i_2,j_1} \neq p_{i_1,j_2} - p_{i_2,j_2}$, $\forall i_1 \neq i_2$, $\forall j_1 \neq j_2$, that is, $\Delta p_{i,j_1} \neq \Delta p_{i,j_2}$, $\forall i$, $\forall j_1 \neq j_2$.

iii. A cycle of length 6 occurs if and only if an equation of the type

$$(p_{1,j_1} - p_{0,j_1}) + (p_{2,j_2} - p_{1,j_2}) = (p_{2,j_3} - p_{0,j_3})$$

is verified, for some j_1, j_2, j_3, with $j_1 \neq j_2, j_3$ and $j_2 \neq j_3$. Hence, for a given j_3, this may occur $\forall j_2 \neq j_3$ and $\forall j_1 \neq j_2, j_3$.

■

Lemma 4.2.3 A necessary condition to have $g \geq 8$ in monomial codes with $d_v = c = 3$ is

$$m_s \geq \left\lceil \frac{a(a-1)}{8} \right\rceil. \tag{4.3}$$

Proof. Let us consider a generic entry in the third row of $\Delta\mathbf{P}$, namely $\Delta p_{2,j}$. In order to satisfy condition 3) of Lemma 4.2.2, $\Delta p_{2,j_3}$ must be different from all the sums of an element $\Delta p_{0,j_1}$, $\forall j_1 \neq j_3$ with an element $\Delta p_{1,j_2}$, $\forall j_2 \neq j_1, j_3$. Furthermore, for condition 2) of Lemma 4.2.2, $\Delta p_{2,j_1} = \Delta p_{0,j_1} + \Delta p_{1,j_1} \neq \Delta p_{2,j_3}$, $\forall j_1 \neq j_3$. Thus, there must be $\sum_{i=0}^{a-2} (a-i-1) = \binom{a}{2}$ different sums which are also different from $\Delta p_{2,j}$. Since $(\Delta p_{0,j_1} + \Delta p_{1,j_2}) \in [-2m_s, 2m_s]$, the different sums can assume $4m_s$ values (all values except $\Delta p_{2,j}$). It follows that

$$\binom{a}{2} \leq 4m_s,$$

from which (4.3) is obtained. ■

Definition 4.2.1 We define a generic difference $\Delta p_j^{(i_1 \leftrightarrow i_2)} = p_{i_1,j} - p_{i_2,j}$. A difference of differences is defined as

$$\begin{aligned} &\Delta p_{j_1}^{(i_1 \leftrightarrow i_2)} - \Delta p_{j_2}^{(i_1 \leftrightarrow i_2)} = p_{i_1,j_1} - p_{i_2,j_1} - p_{i_1,j_2} + p_{i_2,j_2}, \\ &\forall j_1 \neq j_2, i_1 \neq i_2. \end{aligned} \tag{4.4}$$

Lemma 4.2.4 A necessary and sufficient condition for a monomial SC-LDPC-CC with $d_v = c = 3$ to have girth $g \geq 10$ is that its symbolic parity-check matrix is free of repeated differences of differences.

Proof. For $d_v = 3$ the following equation holds

$$\begin{aligned} &\Delta p_j^{(i_1 \leftrightarrow i_2)} + \Delta p_j^{(i_2 \leftrightarrow i_3)} = \Delta p_j^{(i_1 \leftrightarrow i_3)} \\ &\forall j, \forall i_3, \forall i_2 \neq i_3, \forall i_1 \neq i_2, i_3. \end{aligned} \tag{4.5}$$

A generic cycle having length 6 can be described as

$$\begin{aligned} &\Delta p_{j_1}^{(i_1 \leftrightarrow i_2)} + \Delta p_{j_2}^{(i_2 \leftrightarrow i_3)} = \Delta p_{j_3}^{(i_1 \leftrightarrow i_3)} \\ &\forall i_3, \forall i_2 \neq i_3, \forall i_1 \neq i_2, i_3, \forall j_3, \forall j_2 \neq j_3, \forall j_1 \neq j_2, j_3. \end{aligned} \tag{4.6}$$

Substituting (4.5) for $j = j_1$ in (4.6) we obtain the alternative form

$$\Delta p_{j_1}^{(i_1 \leftrightarrow i_3)} - \Delta p_{j_3}^{(i_1 \leftrightarrow i_3)} = \Delta p_{j_1}^{(i_2 \leftrightarrow i_3)} - \Delta p_{j_2}^{(i_2 \leftrightarrow i_3)}. \tag{4.7}$$

On the other hand, a cycle of length 8 can be characterized, by definition, as

$$\Delta p_{j_1}^{(i_1 \leftrightarrow i_2)} - \Delta p_{j_2}^{(i_1 \leftrightarrow i_2)} = \Delta p_{j_3}^{(i_1 \leftrightarrow i_3)} - \Delta p_{j_4}^{(i_1 \leftrightarrow i_3)}. \tag{4.8}$$

From (4.7) and (4.8) we see that, avoiding coincident differences of differences, we avoid cycles of both lengths 6 and 8. It is also true that (4.7) and (4.8) cover all the cases of equalities that can be formed from (4.4). Therefore, the converse holds as well. ■

Corollary 4.2.1 For $d_v = c = 3$, in order to avoid coincident differences of differences and achieve $g \geq 10$ we need

$$2m_s \geq 3\binom{a}{2}. \tag{4.9}$$

Proof. The absolute value of any difference of differences lies in the range $[1, 2, \ldots, 2m_s]$. For $d_v = c = 3$, The total number of differences of differences is $3\binom{a}{2}$. By imposing that the cardinality of the set of possible values is greater than or equal to the total number of possibilities, we obtain (4.9). ■

The same reasoning can be extended to the case of $d_v = c = 2$ but, as we have seen before, in such a case cycles of length 10 do not exist. Therefore the following corollary holds.

Corollary 4.2.2 For $d_v = c = 2$, in order to avoid equal differences of differences and achieve $g \geq 12$ we need

$$2m_s \geq \binom{a}{2}.$$

Proof. Same as Corollary 4.2.1. ■

Comparison with QC-LDPC block code matrices

As shown in Section 2.3, we can describe an SC-LDPC-CC with a symbolic parity-check matrix $\mathbf{H}(D)$. Such a representation can also be used for QC-LDPC block codes, the difference being in the procedure that converts $\mathbf{H}(D)$ into the binary parity-check matrix $\mathbf{H}$. For QC-LDPC block codes, the latter involves expansion of each entry of a symbolic matrix into a binary circulant block, while for SC-LDPC-CCs (2.3) is used. In the following, for the sake of readability, the symbolic matrix of SC-LDPC-CCs will be denoted as $\mathbf{H}(D)$; that of QC-LDPC codes, instead, as $\mathbf{H}(x)$. We can consider symbolic matrices designed for QC-LDPC block codes and use them to obtain SC-LDPC-CCs. However, as we show next, this approach does not produce optimal results from the syndrome former constraint length standpoint.

Fossorier in [50] provides two bounds for the symbolic matrix of QC-LDPC codes. Both of them relate the minimum size of the circulant permutation blocks forming the code parity-check matrix to the code girth. The symbolic matrices considered in [50] define monomial codes and contain all one entries in their first row and column, i.e.,

$$\mathbf{H}(x) = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & x^{p_{1,1}} & \dots & x^{p_{1,a-1}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{p_{c-1,1}} & \dots & x^{p_{c-1,a-1}} \end{bmatrix}. \tag{4.10}$$

In (4.10), $p_{i,j}$ represents the cyclic shift to the right of the elements in each row of an identity matrix, and defines the circulant permutation matrix at position (i, j). This representation has also been used in [49] and [51].

Theorem 2.2 in [50] states that a necessary condition to have $g \geq 6$ is $p_{i,j_1} \neq p_{i,j_2}$ and $p_{i_1,j} \neq p_{i_2,j}$. This theorem is also valid for SC-LDPC-CCs and we have used a similar approach to demonstrate Lemma 4.2.1. A corollary of this theorem claims that a necessary condition to have $g \geq 6$ in the Tanner graph representation of a (J, K)-regular QC-LDPC code is $L \geq K$ if L is odd and $L > K$ if K is even (L is the size of the circulant permutation matrices). The bounds on symbolic matrices resulting from this analysis can be extended to SC-LDPC-CCs. In fact, for a QC-LDPC code the maximum exponent appearing in $\mathbf{H}(x)$ is equal to $L - 1$. Therefore, if we use the symbolic matrix to define an SC-LDPC-CC, the condition for $g = 6$ becomes $\max\{m_s\} = L - 1$. Hence we find that $m_s \geq a - 1$ if a is odd and $m_s > a - 1$ if a is even. By comparing these conditions with (4.1), we see that having removed the border of ones in $\mathbf{H}(D)$ has allowed us to obtain values of m_s which are about half those needed in the

presence of this constraint as in [50]. Examples of matrices with the smallest achievable m_s, $d_v = c = 3$ and $g = 6$ with and without the all one border of ones are as follows

$$\mathbf{H}_1(D) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^4 \\ 1 & D^3 & D & D^2 \end{bmatrix},$$

$$\mathbf{H}_2(D) = \begin{bmatrix} 1 & D^2 & 1 & 1 \\ 1 & 1 & D & D^2 \\ D^2 & D & D & 1 \end{bmatrix}.$$

Indeed, the maximum exponent appearing in $H_1(D)$ is twice that in $H_2(D)$. According to Theorem 2.4 in [50], for $J \geq 3$ and $K \geq 3$, a necessary condition to have $g \geq 8$ is $p_{j_1,k_1} \neq p_{j_2,k_2}$ for $0 < j_1 < j_2$ and $0 < k_1 < k_2$. A corollary of this theorem states that a necessary condition is $L > (J-1)(K-1)$. It is shown in [49] that this bound is usually not tight when $J \geq 4$.

Extending these results to SC-LDPC-CCs, we can calculate a necessary condition to avoid cycles of length 4 and 6 in codes with the all one border, thus obtaining $m_s \geq (a-1)(c-1)$. Also in this case, the all one border causes an increase in m_s, as it is evident from the following example

$$\mathbf{H}_1(D) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & D^6 & D^2 & D^5 \\ 1 & D^3 & D^4 & D \end{bmatrix},$$

$$\mathbf{H}_2(D) = \begin{bmatrix} 1 & D^3 & 1 & D^3 \\ D^3 & D^2 & D^2 & 1 \\ D^2 & 1 & D^3 & D^2 \end{bmatrix}.$$

An intermediate step for the generalization of the codes in [50] is the removal of the constraint of having the first column filled with ones, that is,

$$\mathbf{H}(x) = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x^{p_{1,0}} & x^{p_{1,1}} & \dots & x^{p_{1,a-1}} \\ \vdots & \vdots & \ddots & \vdots \\ x^{p_{c-1,0}} & x^{p_{c-1,1}} & \dots & x^{p_{c-1,a-1}} \end{bmatrix}. \tag{4.11}$$

As we show next, this modification leads to a reduction in the lower bounds on m_s.

Codes described by a symbolic matrix as in (4.11) are referred to as Type-1c codes.

Lemma 4.2.5 A necessary condition for a Type-1c SC-LDPC-CC with $d_v = c = 3$ to achieve $g \geq 6$ is $m_s \geq a - 1$, $\forall a$.

Proof. Similar to the proof of Corollary 2.2 in [50]. ■

Lemma 4.2.6 A necessary condition for a Type-1c SC-LDPC-CC with $d_v = c = 3$ to achieve $g \geq 8$ is $m_s \geq \frac{a(c-1)}{2} - 1$.

Proof. A cycle of length 6 follows from equalities having the following form: $p_{j_1,k_1} + p_{j_2,k_2} = p_{j_1,k_2} + p_{j_2,k_3}$. If there exists some $p^*_{j_1,k_1} = p^*_{j_2,k_1}$, then to avoid length-6 cycles there must not be any $p^d_{j_1,k_1} = p^d_{j_2,k_2}$ in the remaining entries of the matrix. Otherwise, either a cycle of length 6 resulting from $p^d_{j_1,k_1} + p^*_{j_1,k_1} = p^d_{j_2,k_2} + p^*_{j_2,k_1}$ or a cycle of length 4 resulting from $p^d_{j_1,k_1} - p^d_{j_2,k_2} = p^*_{j_1,k_1} - p^*_{j_2,k_1}$ would appear. In this case, we would obtain $m_s \geq (a-1)(c-1)$, showing that the case studied in [50] is only a particular case of the above construction. If $p_{j_1,k_1} = p_{j_2,k_1}$ is never verified, we can fill the $(c-1)a$ possible values with $\frac{(c-1)a}{2}$ couples of exponents (also 0 is admitted), leading to $m_s \geq \frac{a(c-1)}{2} - 1$. ■

The lower bounds provided by Lemma 4.2.5 and Lemma 4.2.6 remain worse than those obtained with the general structure of $\mathbf{H}(D)$, thus confirming that removal of the constraints in (4.10) is preferable in view of minimizing m_s.

4.2.2 Bounds on the constraint length of type-z codes, $\forall z$

In order to meet the condition $g \geq 6$, we must ensure that cycles of length 4 do not exist. Such short cycles occur when, for some i_1, j_1, i_2, j_2, $j_1 \neq j_2$,

$$\delta_{i_1,j_1} = \delta_{i_2,j_2} \quad \text{and} \quad s_{l_1} = s_{l_2}, \tag{4.12}$$

where s_{l_1} and s_{l_2} are the starting levels of δ_{i_1,j_1} and δ_{i_2,j_2}, respectively. So, in order to avoid cycles of length 4, there must not be any two equal differences starting from the same level. We observe that such two differences may even be in the same column of $\mathbf{H}^T_s$.

Lemma 4.2.7 A Type-z code with $\mathbf{H}^T_s$ having $d_v = 2$, free of length-4 cycles, has

$$a \leq \sum_{i=0}^{c-1} (L_h - i - 1) = cL_h - \binom{c+1}{2}, \tag{4.13}$$

that is

$$L_h \geq \left\lceil \frac{a + \binom{c+1}{2}}{c} \right\rceil.$$

Considering that by construction it must be $L_h > c$, the above constraint should be rewritten as

$$L_h \geq \max\left\{c+1, \left\lceil \frac{a + \binom{c+1}{2}}{c} \right\rceil \right\}. \tag{4.14}$$

Proof. For column weight $d_v = 2$, each column of $\mathbf{H}_s^T$ only contains one difference $\delta_{i,j}$ and each difference can be used up to c times without incurring cycles of length 4 (by using all the possible c starting levels). For a given L_h, the number of possible differences starting from level l is $L_h - l - 1$. Since the differences corresponding to any two of the a columns of $\mathbf{H}_s^T$ must be different in value and/or starting level, summing all the contributions we obtain (4.13), from which (4.14) is eventually derived. ■

We can extend (4.13) to the case of a regular $\mathbf{H}_s^T$ with $d_v > 2$ by considering that, in such a case, each column of $\mathbf{H}_s^T$ provides $\binom{d_v}{2}$ differences that must meet condition (4.12). Hence, (4.13) becomes

$$a\binom{d_v}{2} \leq cL_h - \binom{c+1}{2},$$

while (4.14) becomes

$$L_h \geq \max\left\{c+1, \left\lceil \frac{a\binom{d_v}{2} + \binom{c+1}{2}}{c} \right\rceil \right\}. \tag{4.15}$$

When we have an irregular $\mathbf{H}_s^T$ having different column weights $d_v^{(i)}$, $i = 0, 1, 2, \ldots, a-1$, each column of $\mathbf{H}_s^T$ corresponds to $\binom{d_v^{(i)}}{2}$ differences. Therefore (4.15) becomes

$$L_h \geq \max\left\{c+1, \left\lceil \frac{\sum_{i=0}^{a-1}\binom{d_v^{(i)}}{2} + \binom{c+1}{2}}{c} \right\rceil \right\}.$$

In order to find the conditions which permit us to have $g \geq 8$, let us first consider the case with $c = 1$ and $\mathbf{H}_s^T$ with column weight $d_v^{(i)} = 2$. Since the sum of two odd integers is even, the following proposition easily follows.

Proposition 4.2.1 For $c = 1$ and $d_v = 2$, if all the $\delta_{i,j}$ are different and odd, then $g \geq 8$.

From Proposition 4.2.1 it follows that, if we wish to minimize L_h, we can choose the values of $\delta_{i,j}$ equal to $\{1, 3, 5, \ldots, 2a-1\}$ and the code will be free of cycles of length smaller than 8.

Lemma 4.2.8 For $c = 1$ and $d_v = 2$, cycles having length smaller than 8 can be avoided if and only if

$$L_h \geq 2a. \tag{4.16}$$

Proof. From Proposition 4.2.1 we see that the maximum value of a difference that is needed to avoid cycles of length smaller than 8 is $2a - 1$. Therefore, we have $L_h \geq 1 + 2a - 1 = 2a$. In order to prove the converse, let us consider that from the set $[1, 2, \ldots, y]$ we can select at most $\frac{y}{2}$ values which may be summed pairwise resulting in other values in the same set. If we choose the values of the differences from the set $[1, 2, \ldots, 2a - 2]$, we only have $a - 1$ values which may be summed pairwise resulting in other values in the same set. Therefore, we can only allocate $a - 1$ differences without introducing cycles of length 6, which is not sufficient to cover all the a rows of $\mathbf{H}_s$. So, $[1, 2, \ldots, 2a - 1]$ is the smallest set of difference values able to avoid cycles of length 4 and 6, from which (4.16) follows. ■

Equation (4.16) can be extended to the case $c > 1$ by considering that, in such a case, each difference value can be repeated up to c times (by exploiting all the c available values as starting levels). Therefore, for $d_v = 2$ and $c > 1$ we have

$$L_h \geq \max\left\{c + 1, \frac{2a}{c}\right\}.$$

Let us consider larger values of d_v, i.e., $d_v \geq 3$. For $c = 1$, each column of $\mathbf{H}_s^T$ has one or more cycles of length 6, since at least 3 symbols one are at the same level.

Instead, for $d_v \geq 3$ and $c > 1$ we can follow the same approach used for the case with $g = 6$. This way, we obtain that to have $g \geq 8$ we need

$$L_h \geq \max\left\{c + 1, \frac{2a\binom{d_v}{2}}{c}\right\}. \tag{4.17}$$

Finally, when the columns of $\mathbf{H}_s^T$ are irregular with weights $d_v^{(i)}$, $i = 0, 1, 2, \ldots, a-1$, as done for the case with $g = 6$, we can consider that each column of $\mathbf{H}_s^T$ contains to $\binom{d_v^{(i)}}{2}$ differences and (4.17) becomes

$$L_h \geq \max\left\{c + 1, \left\lceil \frac{2\sum_{i=0}^{a-1}\binom{d_v^{(i)}}{2}}{c} \right\rceil\right\}.$$

4.3 Design methods

In this section we introduce new methods for the design of Type-1, Type-2 and Type-3 SC-LDPC-CCs with constraint length approaching the bounds described in Section 4.2. These design approaches have general validity. However, for the sake of simplicity we focus on the case $d_v = c = 3$.

4.3.1 Type-1 codes

As we have already demonstrated in Lemma 4.2.1, Type-1 codes having $g \geq 6$ satisfy (4.1). For any integer value k, a Type-1 code with $a = 2k + 1$ able to achieve such a bound is defined by the following symbolic matrix

$$\mathbf{H}_e(D,k) = \begin{bmatrix} D^k & D^k & D^k & \dots & D^k & D^k & 1 & 1 & 1 & \dots & 1 & 1 \\ 1 & D & D^2 & \dots & D^{k-1} & D^k & D & D^2 & D^3 & \dots & D^{k-1} & D^k \\ D^k & D^{k-1} & D^{k-2} & \dots & D & 1 & D^k & D^{k-1} & D^{k-2} & \dots & D^2 & D \end{bmatrix} \tag{4.18}$$

when k is even and by

$$\mathbf{H}_o(D,k) = \begin{bmatrix} D^k & D^k & D^k & \dots & D^k & D^k & 1 & 1 & 1 & \dots & 1 & 1 \\ 1 & D & D^2 & \dots & D^{k-2} & D^{k-1} & 1 & D & D^2 & \dots & D^{k-1} & D^k \\ D^{k-1} & D^{k-2} & D^{k-3} & \dots & D & 1 & D^k & D^{k-1} & D^{k-2} & \dots & D & 1 \end{bmatrix} \tag{4.19}$$

when k is odd. Starting from (4.18) and (4.19), the removal of one column permits us to also cover the case of $a = 2k$.

4.3.2 Type-2 codes

In the case of Type-2 codes with $d_v = c = 3$, we can jointly use monomial and binomial entries in the same column of the symbolic parity-check matrix. The bound given by (4.15), expressed in terms of m_s, becomes

$$m_s \geq \left\lceil \frac{a-1}{3} \right\rceil. \tag{4.20}$$

Let k be an integer and $\mathbf{H}_e^1(D,k)$, $\mathbf{H}_e^2(D,k)$ and $\mathbf{H}_e^3(D,k)$ be the following $3 \times k$ symbolic matrices

$$\mathbf{H}_e^1(D,k) = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ D + D^k & D^2 + D^{k-1} & \dots & D^{\frac{k}{2}} + D^{\frac{k+2}{2}} & 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 & 1 + D^k & D + D^{k-1} & \dots & D^{\frac{k-2}{2}} + D^{\frac{k+2}{2}} \end{bmatrix}, \tag{4.21}$$

$$\mathbf{H}_e^2(D,k) = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 + D^k & D + D^{k-1} & \dots & D^{\frac{k-2}{2}} + D^{\frac{k+2}{2}} \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ D + D^k & D^2 + D^{k-1} & \dots & D^{\frac{k}{2}} + D^{\frac{k+2}{2}} & 1 & 1 & \dots & 1 \end{bmatrix},$$

$$\mathbf{H}_e^3(D,k) = \begin{bmatrix} D+D^k & D^2+D^{k-1} & \dots & D^{\frac{k}{2}}+D^{\frac{k+2}{2}} & 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 & 1+D^k & D+D^{k-1} & \dots & D^{\frac{k-2}{2}}+D^{\frac{k+2}{2}} \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix}.$$

Also, let $\mathbf{H}_o^1(D,k)$, $\mathbf{H}_o^2(D,k)$ and $\mathbf{H}_o^3(D,k)$ be the following $3 \times k$ symbolic matrices:

$$\mathbf{H}_o^1(D,k) = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ D+D^k & D^2+D^{k-1} & \dots & D^{\frac{k-1}{2}}+D^{\frac{k+3}{2}} & 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 & 1+D^k & D+D^{k-1} & \dots & D^{\frac{k-1}{2}}+D^{\frac{k+1}{2}} \end{bmatrix},$$

$$\mathbf{H}_o^2(D,k) = \begin{bmatrix} 1 & 1 & \dots & 1 & 1+D^k & D+D^{k-1} & \dots & D^{\frac{k-1}{2}}+D^{\frac{k+1}{2}} \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ D+D^k & D^2+D^{k-1} & \dots & D^{\frac{k-1}{2}}+D^{\frac{k+3}{2}} & 1 & 1 & \dots & 1 \end{bmatrix},$$

$$\mathbf{H}_o^3(D,k) = \begin{bmatrix} D+D^k & D^2+D^{k-1} & \dots & D^{\frac{k-1}{2}}+D^{\frac{k+3}{2}} & 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 & 1+D^k & D+D^{k-1} & \dots & D^{\frac{k-1}{2}}+D^{\frac{k+1}{2}} \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{4.22}$$

It is easy to verify that each one of the matrices in (4.21) to (4.22) results in a code with girth 6. For even and odd values of k, codes with $a = 3k$ are defined by, respectively,

$$\mathbf{H}_e(D,k) = \begin{bmatrix} \mathbf{H}_e^1(D,k) & \mathbf{H}_e^2(D,k) & \mathbf{H}_e^3(D,k) \end{bmatrix},$$

$$\mathbf{H}_o(D,k) = \begin{bmatrix} \mathbf{H}_o^1(D,k) & \mathbf{H}_o^2(D,k) & \mathbf{H}_o^3(D,k) \end{bmatrix}.$$

Also in this case, it is possible to remove one or more columns to obtain the required values of a. In fact, both $\mathbf{H}_e(D,k)$ and $\mathbf{H}_o(D,k)$ define codes with girth 6. Any reduction of the girth with respect to their component matrices is avoided owing to the order of the rows of $\mathbf{H}_e^i(D,k)$ and $\mathbf{H}_o^i(D,k)$ $(i = 1,2,3)$, which ensures that the supports of any two columns belonging to two different component matrices do not overlap in more than one position.

Example 4.3.1 Let $d_v = c = 3$ and $a = 3k = 12$ (i.e., $k = 4$). Based on (4.20), $m_s \geq \left\lceil \frac{12-1}{3} \right\rceil = 4$. It is possible to construct a syndrome former matrix $\mathbf{H}_e(D,4)$ with $m_s = 4$ by concatenating the three matrices below, each of size 3×4

$$\mathbf{H}_e^1(D,k) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ D+D^4 & D^2+D^3 & 1 & 1 \\ 1 & 1 & 1+D^4 & D+D^3 \end{bmatrix},$$

$$\mathbf{H}_e^2(D,4) = \begin{bmatrix} 1 & 1 & 1+D^4 & D+D^3 \\ 0 & 0 & 0 & 0 \\ D+D^4 & D^2+D^3 & 1 & 1 \end{bmatrix},$$

$$\mathbf{H}_e^3(D,4) = \begin{bmatrix} D+D^4 & D^2+D^3 & 1 & 1 \\ 1 & 1 & 1+D^4 & D+D^3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

4.3.3 Type-3 codes

Finally, let us consider the use of trinomial entries in the columns of the symbolic parity-check matrices. Since $d_v = 3$, we cannot use more than one trinomial entry per column but it is possible to use every trinomial entry three times, one per each row of the symbolic matrix. For different values of a, some groups of trinomials with the smallest possible degrees that permit us to avoid cycles of length 4 are as follows

$$\begin{cases} 1+D+D^4 \quad 1+D^2+D^7 \quad a=6 \\ 1+D+D^5 \quad 1+D^6+D^8 \quad 1+D^3+D^{10} \quad a=9 \\ 1+D^2+D^9 \quad 1+D^3+D^8 \quad 1+D^4+D^{10} \quad 1+D+D^{12} \quad a=12 \\ 1+D^{11}+D^{12} \quad 1+D^{13}+D^{15} \quad 1+D^7+D^{10} \quad 1+D^5+D^9 \quad 1+D^8+D^{14} \quad a=15 \\ 1+D^6+D^7 \quad 1+D^{13}+D^{15} \quad 1+D^{14}+D^{17} \quad 1+D^8+D^{12} \quad 1+D^{11}+D^{16} \quad 1+D^{10}+D^{19} \quad a=18 \\ \dots \end{cases} \tag{4.23}$$

To see why these trinomials are those with smallest degrees, we refer the interested reader to [53], where the authors exploit a combinatorial notion known as Perfect Difference Families (PDFs). They show that Type-3 QC-LDPC block codes with $d_v = 3,\ c = 1,\ a = 3k$ and girth 6 can be constructed by using the entries of each block of a $(v, 3, 1)$ PDF to define a trinomial in the syndrome former matrix. In fact, if our target syndrome former matrix is only formed by trinomials, PDFs give us the smallest trinomial degrees. Those reported in (4.23) and many other groups of trinomials can be found starting from [54, Table I].

We also notice that we can combine trinomial and monomial entries to design codes with $g \geq 6$. In fact, trinomial entries introduce "horizontal" separations whereas monomial entries introduce "vertical" separations. This is also the reason why, in general, trinomial and binomial entries cannot be combined in the configurations described in Section 4.3.2. We provide next an example in which we concatenate two matrices containing only monomial and trinomial entries to construct a syndrome former matrix with $d_v = c = 3$, $a = 21$ and

girth 6, achieving the smallest possible m_s according to (4.20)

$$\mathbf{H}(D,7)_{odd} = \left[\begin{array}{cccccc|c} p_1(D) & p_2(D) & 0 & 0 & 0 & 0 & \\ 0 & 0 & p_1(D) & p_2(D) & 0 & 0 & \\ 0 & 0 & 0 & 0 & p_1(D) & p_2(D) & \end{array}\right.$$

$$\left|\begin{array}{ccccccc} D^7 & D^7 & D^7 & D^7 & D^7 & D^7 & D^7 \\ 1 & D & D^2 & D^3 & D^4 & D^5 & D^6 \\ D^6 & D^5 & D^4 & D^3 & D^2 & D & 1 \end{array}\right|$$

$$\left.\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^3 & D^4 & D^5 & D^6 & D^7 \\ D^7 & D^6 & D^5 & D^4 & D^3 & D^2 & D & 1 \end{array}\right],$$

where $p_1(D) = 1 + D + D^4$ and $p_2(D) = 1 + D^2 + D^7$.

4.4 Code examples and numerical results

In this section we provide some examples of code design and their comparison with analytical bounds. We also assess the codes performance through numerical simulations of coded transmissions.

4.4.1 Code design based on exhaustive searches

In Fig. 4.1 we report the bounds on L_h obtained as described in Section 4.2 as a function of a, for some values of d_v, g and c. We compare these bounds with the results obtained through exhaustive searches over all the possible choices of $\mathbf{H}_s^T$. The matching between the bound and the values found through exhaustive searches is perfect for all the considered values of c when $d_v = 2$ and $g = 6$, as shown in Fig. 4.1a. We note in Fig. 4.1b that the results of exhaustive searches are well matched with the bounds also for the case with $d_v = 2$ and $g = 8$. When $d_v > 2$, the bound may not be achievable in practical terms. We observe in Fig. 4.1c that the deviations of the heuristic values from the theoretical curves are rather small when $g = 6$. The gap to the bound increases for increasing values of c when $g = 8$, as can be noticed in Fig. 4.1d.

In Fig. 4.2 we consider $d_v = c = 3$, $g = 6$, and compare the values of m_s achievable through the design methods described in Section 4.3 with the corresponding lower bounds.

In Table 4.1 we instead provide the results of an exhaustive search of Type-1 codes with $d_v = c = 3$ and $g = 8$, and their comparison with the corresponding lower bound.

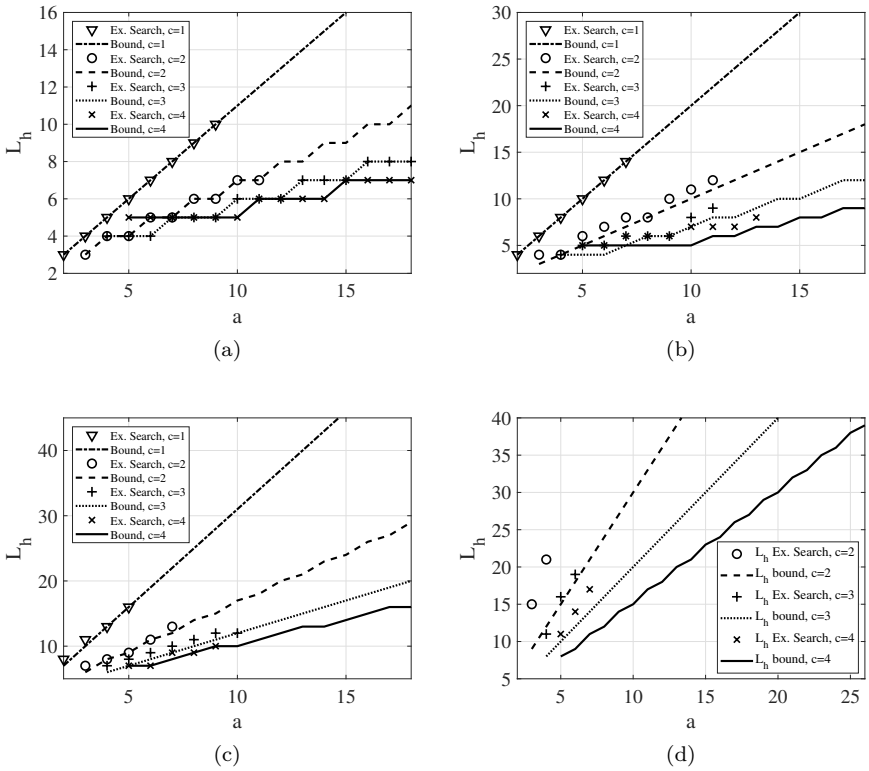


Figure 4.1: Bounds on L_h and values found through exhaustive searches as a function of a for some values of c and: (a) $d_v = 2$, $g = 6$ (b) $d_v = 2$, $g = 8$ (c) $d_v = 3$, $g = 6$ (d) $d_v = 3$, $g = 8$.

Table 4.1: Minimum values of m_s for Type-1 codes with $d_v = c = 3$ and $g = 8$.

CODE RATE	1/4	2/5	3/6	4/7
Exhaustive Search	3	5	6	8
Bound (4.3)	2	3	4	6

4.4.2 IP model

We also propose to use an optimization integer programming (IP) model [55] in order to find the minimum possible m_s for each exponent matrix of monomial codes. We call this method Min-Max. Such a model, instead of performing an exhaustive search, takes benefit of a heuristic optimization approach. This process significantly decreases the search time. The input exponent matrices of our Min-Max model were selected from [49, 51, 56].

Let us describe the IP optimization model we use. As inputs, it takes a big

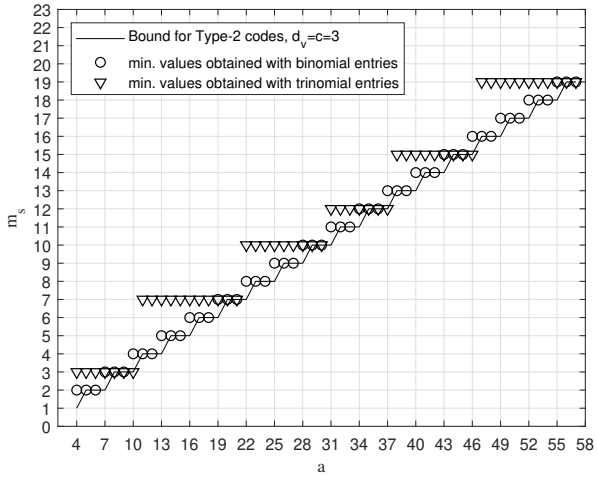


Figure 4.2: Bounds on m_s and values found through the design method described in Section 4.3 as a function of a, for $d_v = c = 3$, $g = 6$.

enough penalty M, all the entries of an exponent matrix $\mathbf{P}$, a positive integer $L \to \infty$ to represent the maximum allowed exponent in $\mathbf{P}$, and the set of relatively prime numbers to L (this set has cardinality $\phi(L)$, where ϕ is the Euler function) and it finds Min-Max of the elements of all the matrices $\mathbf{P}_t$'s, where $\mathbf{P}_t$ is the transformed exponent matrix obtained by applying [52, Lemmas 1,2,3] on $\mathbf{P}$. Note that for specific parameters of d_v, c, a and g, we picked the exponent matrix $\mathbf{P}$ from [49,51,56], where $\mathbf{P}$ has the smallest possible circulant size L. Thus, if $\mathbf{P}_t^*$ is one of the optimal transformed exponent matrices, our model explicitly finds the linear transformations based on [52, Lemmas 1,2,3] and, by applying them on $\mathbf{P}$, we achieve new instances of $\mathbf{P}_t^*$. Furthermore, the model finds the maximum syndrome former memory order m_s in $\mathbf{P}_t^*$ as output; the final m_s is the minimum possible m_s of all the transformed matrices $\mathbf{P}_t$.

In the following list, we enumerate the steps of our model:

1. minimize $Z = \sum_{i=0}^{a-1} \sum_{j=0}^{c-1} x_{ij}$
s.t.
2. $b_{ij} = \left(\sum_{g=1}^{\phi(L)} k_g T_g \right) p_{ij} + r_i + c_j$
3. $\sum_{g=1}^{\phi(L)} k_g = 1$
4. $L\psi_{ij} \leq b_{ij}$
5. $b_{ij} + 0.5 \leq (1 + \psi_{ij})\, L$
6. $d_{ij} = b_{ij} - L\psi_{ij}$
7. $d_{mn} \leq d_{ij} + M\,(1 - y_{ij}) \quad (m, n) \neq (i, j)$
8. $\sum_{i=0}^{a-1} \sum_{j=0}^{c-1} y_{ij} = 1$
9. $x_{ij} \leq M y_{ij}$
10. $d_{ij} \leq x_{ij} + M\,(1 - y_{ij})$
11. $k_g, y_{ij} \in \{0, 1\},\ 0 \leq r_i, c_j < L,$ and $r_i, c_j, \psi_{ij}, d_{ij}, x_{ij}$ are integers.

A brief description of the steps of the model is as follows. Each element p_{ij} of $\mathbf{P}$ is transformed to b_{ij} by multiplying it to a relatively prime number T_g, as well as by adding two decision variables r_i and c_j (Constraint 2). Constraint 3 indicates that just one of the relatively prime number to L should be selected. Constraints 4 and 5 determine the quotient ψ_{ij} of element b_{ij} divided by L. Constraint 6 is the residual of subtracting $p\psi_{ij}$ from b_{ij}. The two constraints 7 and 8 are used to detect the maximum element of the transformed exponent matrix modulo L, where y_{ij}'s are identification binary variables. Clearly, just one of these variables can be chosen. Variables x_{ij}'s in constraints 9 and 10 are created in such a way that just one of them is greater than zero, which is the maximum among all of the elements in $\mathbf{P}_t^*$. This output element is considered to be Min-Max or our desired m_s.

4.4.3 Code design based on integer programming

When the girth takes values $g \geq 8$, the values of m_s or, equivalently, L_h are so high that exhaustive searches become unfeasible even for small values of a and c. Thus, we resort to the Min-Max algorithm and we show in Tables 4.2 and 4.3 the minimum values of m_s for $g = 8, 10$.

4.4.4 Numerical simulations of coded transmissions

Concerning the bit error rate (BER) performance of the considered codes, as reasonable and expected, there is a trade-off with their constraint lengths.

We have assessed the performance of our codes through Monte Carlo simulations of BPSK modulated transmission over the AWGN channel, and compared it with that of some SC-LDPC-CCs constructed using the technique described

Table 4.2: Minimum values of m_s for codes with $d_v = c = 3, 4, 5, 6$ and $g = 8$ obtained through the Min-Max algorithm, for several code rates.

$d_v = c = 3$	a	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	25
	m_s	3	6	7	9	10	12	15	17	21	24	26	29	32	37	39	43	48	76
$d_v = c = 4$	a	–	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	25
	m_s	–	10	11	14	17	23	27	31	36	48	53	64	78	87	97	110	122	230
$d_v = c = 5$	a	–	–	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	25
	m_s	–	–	14	22	31	39	49	58	77	91	101	129	150	174	207	232	279	458
$d_v = c = 6$	a	–	–	–	7	8	9	10	11	12	13	14	15	16	17	18	19	20	–
	m_s	–	–	–	31	42	65	81	97	113	155	185	226	259	301	348	377	446	–

Table 4.3: Minimum values of m_s for codes with $d_v = c = 3$ and $g = 10$ obtained through the Min-Max algorithm, for several code rates.

Code rate	1/4	2/5	3/6	4/7	5/8	6/9	7/10	8/11	9/12
m_s	11	19	31	53	76	127	222	307	388

in [32], denoted as Tanner codes. A full size BP decoder and a BP-based SW decoder, both performing 100 iterations, have been used in the simulations. The memory of the sliding window decoder is periodically reset in order to interrupt the catastrophic propagation of some harmful decoding error patterns. The decoder memory reset period has been heuristically optimized for each code. Let us consider a code with $a = 17$, $d_v = c = 3$, $g = 8$ and $v_s = 646$, denoted as $\mathcal{C}_1$. Its exponent matrix is as follows

$$\mathbf{P}_{\mathcal{C}_1} = \begin{bmatrix} 5 & 17 & 6 & 12 & 30 & 0 & 7 & 37 & 11 & 20 & 2 & 33 & 0 & 16 & 0 & 4 & 21 \\ 29 & 0 & 21 & 24 & 15 & 34 & 0 & 0 & 0 & 0 & 8 & 9 & 14 & 0 & 36 & 37 & 7 \\ 0 & 28 & 0 & 0 & 0 & 32 & 30 & 29 & 0 & 21 & 0 & 0 & 37 & 36 & 2 & 0 & 0 \end{bmatrix}.$$

We also consider a Tanner code with the same values of a, c and g, but $v_s = 5185$; an equivalent code of the Tanner code with $v_s = 4641$ has been found and will be called $\mathcal{C}_{t_1}$ in the following. This way, a smaller sliding window can be used also for decoding of the Tanner code. In order to decode $\mathcal{C}_1$ with a SW decoder, the minimum required window size in blocks is $W = m_s + 1 = 38$, whereas $\mathcal{C}_{t1}$ requires at least $W = m_s + 1 = 273$. Their performance comparison, obtained by fixing $W = 273$, is shown in Fig. 4.3, where the BER is plotted as a function of the signal-to-noise ratio per bit E_b/N_0. In the figure we also report the performance of the codes simulated using a very large window. We notice that $\mathcal{C}_{t_1}$ outperforms $\mathcal{C}_1$ when $W \to \infty$, thanks to its larger syndrome former constraint length which likely entails better structural properties, but for small window sizes $\mathcal{C}_1$ significantly outperforms $\mathcal{C}_{t_1}$. This happens because,

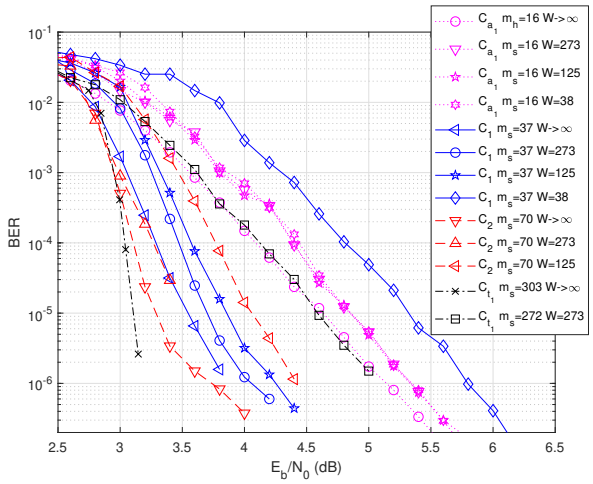


Figure 4.3: Simulated performance of the codes in Table 4.4.

as mentioned in Chapter 2, values of W roughly 5 to 10 times as large as $(m_s + 1)$ result in minimal performance degradation, compared to using the full window size. In this case, $W_{\mathcal{C}_1} > 8m_s$, and $\mathcal{C}_1$ approaches its full length performance, whereas $W_{\mathcal{C}_{t_1}} = m_s + 1$, and $\mathcal{C}_{t_1}$ still incurs some loss due to the small window size.

In order to understand the role of the cycles length, another code, noted as $\mathcal{C}_2$, has been designed with a heuristic search. Its exponent matrix is as follows

$$\mathbf{P}_{\mathcal{C}_2} = \begin{bmatrix} 9 & 59 & 30 & 44 & 0 & 55 & 0 & 0 & 65 & 0 & 21 & 0 & 58 & 37 & 24 & 0 & 41 \\ 0 & 67 & 26 & 60 & 53 & 0 & 18 & 32 & 0 & 59 & 0 & 57 & 0 & 0 & 0 & 38 & 13 \\ 5 & 0 & 0 & 0 & 9 & 55 & 70 & 42 & 27 & 14 & 43 & 16 & 68 & 57 & 56 & 41 & 0 \end{bmatrix}.$$

Considering only its first 7 columns, there are no cycles of length 8. Finally, the performance of an array code [57] is shown for $W \to \infty$ and some finite values of W. The parameters of all the considered codes are summarized in Table 4.4.

Table 4.4: Values of a, c, d_v, m_s, v_s and g of the considered codes with $R = \frac{14}{17}$.

Code	a	c	d_v	m_s	v_s	g
$\mathcal{C}_1$	17	3	3	37	646	8
$\mathcal{C}_{t1}$	17	3	3	272	4896	8
$\mathcal{C}_2$	17	3	3	70	1207	6
$\mathcal{C}_{a_1}$	17	3	3	16	289	6

We notice that $\mathcal{C}_2$ has a very steep curve in the waterfall region when $W \to \infty$ but its performance is affected by an error floor. However, under SW decoding

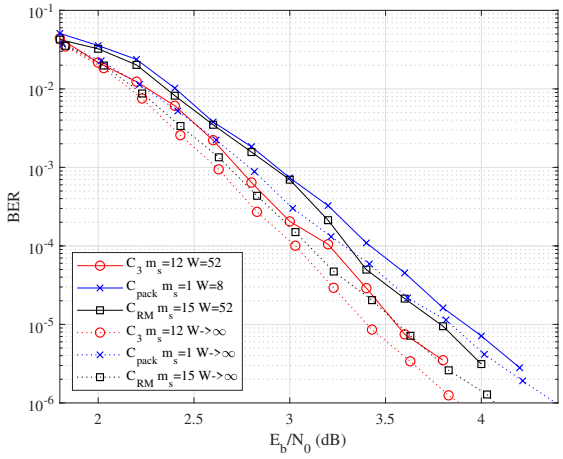


Figure 4.4: BER performance of the codes in Table 4.5.

with $W = 273$, $\mathcal{C}_2$ notably outperforms $\mathcal{C}_{t_1}$ and $\mathcal{C}_1$. Despite its performance is worse than the other ones when $W \to \infty$, the low value of its constraint length allows it to achieve good performance for very small window sizes, namely $W = 38$.

In order to provide a comparison with other design approaches, we have used our technique to design a rate-2/3 code, denoted as $\mathcal{C}_3$, having the following exponent matrix

$$\mathbf{P}_{\mathcal{C}_3} = \begin{bmatrix} 12 & 10 & 9 & 7 & 3 & 0 & 0 & 5 & 0 & 4 & 1 & 10 \\ 6 & 0 & 0 & 11 & 0 & 1 & 11 & 12 & 8 & 10 & 0 & 6 \\ 0 & 4 & 10 & 0 & 5 & 0 & 4 & 12 & 9 & 0 & 11 & 9 \\ 12 & 9 & 5 & 0 & 1 & 9 & 10 & 0 & 3 & 8 & 12 & 0 \end{bmatrix}.$$

The performance of this code has been compared with that of two codes having similar syndrome former constraint length, but designed through other techniques: an SC-LDPC-CC based on packings (see [58] for details) denoted as $\mathcal{C}_{\text{pack}}$, and a replicate and mask SC-LDPC-CC based on algebraic methods (see [59]) denoted as $\mathcal{C}_{\text{RM}}$. The parameters of the three codes are summarized in Table 4.5.

Table 4.5: Values of a, c, d_v, m_s, v_s and g of the considered codes with $R = \frac{2}{3}$.

Code	a	c	d_v	m_s	v_s	g
$\mathcal{C}_3$	12	4	4	12	156	6
$\mathcal{C}_{\text{pack}}$	78	26	4	1	156	6
$\mathcal{C}_{\text{RM}}$	12	4	4	15	192	6

Their BER performance under SW decoding is shown in Fig. 4.4, where the performance achievable with very large windows ($W \to \infty$) is also considered as a benchmark. We notice that the code $\mathcal{C}_3$ outperforms the other two codes with both $W \to \infty$ and $W_{\text{bits}} = Wa = 624$, confirming the effectiveness of our design approach. Note that $W_{\mathcal{C}_3} = 4(m_s+1)$ and $W_{\mathcal{C}_{\text{pack}}} = 4(m_s+1)$, whereas, in order to make the window size in bits comparable, it must be $3 < \frac{W_{\mathcal{C}_{\text{RM}}}}{m_s+1} < 4$. $\mathcal{C}_{\text{RM}}$ has a value of m_s similar to $\mathcal{C}_3$, and its performance is also not far from that of $\mathcal{C}_3$. Instead, $\mathcal{C}_{\text{pack}}$ has a very small m_s and apparently this reflects into a significant performance loss with respect to $\mathcal{C}_3$.

4.5 Design based on sequentially multiplied columns

In this section we propose a low-complexity method to find QC-LDPC block codes with shorter length than those designed through classical approaches. The method is extended to time-invariant SC-LDPC-CCs, permitting to achieve small syndrome former constraint lengths. Several numerical examples are given to show its effectiveness. In the following, we focus on codes with symbolic matrix containing only monomials, so the analysis is less general than that in the previous sections. Nevertheless, for this family of codes the new design method is much more effective and thus worth investigating.

We aim at designing QC-LDPC block codes with smaller blocklength and SC-LDPC-CCs with smaller m_s than those with comparable girth available in the literature, and we use the notion of *compact codes* to encompass block and convolutional LDPC codes with these features in one word.

In order to design compact codes, we resort to a construction exploiting sequentially multiplied columns (SMCs). The latter are obtained starting from a base column and consecutively multiplying it by suitably chosen coefficients. These columns are used to form the parity-check matrix of a QC-LDPC code (SC-LDPC-CC) and then a greedy search algorithm is used to find compact codes. The SMC assumption significantly reduces the search space and permits us to exhaustively explore it. Moreover, we use the IP optimization model described in Section 4.4.2, in order to find the minimum possible m_s.

A necessary and sufficient condition for the existence of a cycle of length $2k$ in the Tanner graph of monomial codes is [50]

$$\sum_{i=0}^{k-1} \left(p_{c_i a_i} - p_{c_i a_{i+1}}\right) = 0 \mod L, \tag{4.24}$$

where $a_k = a_0$, $c_i \neq c_{i+1}$, $a_i \neq a_{i+1}$ and L is the size of the circulant permuta-

tion matrices.

To achieve a certain girth g, for given values of c and a, and for a fixed value of L, one has to find a matrix $\mathbf{P}$ whose entries do not satisfy (4.24) for any value of $k < g/2$, and any possible choice of the row and column indexes c_i and a_i.

We define an *avoidable cycle* in the Tanner graph of a QC-LDPC block code as a cycle for which $\sum_{i=0}^{k-1} \left(p_{c_i a_i} - p_{c_i a_{i+1}}\right) = \beta L$, $\beta > 0$. A *strictly avoidable cycle* is defined as a cycle for which $\sum_{i=0}^{k-1} \left(p_{c_i a_i} - p_{c_i a_{i+1}}\right) = 0$.

4.5.1 Sequentially multiplied columns

It is shown in [49] that the complexity of exhaustively checking equations of the type (4.24) goes exponentially high by increasing each one of the parameters a, c or L.

Solutions with reduced complexity are proposed, for example, in [56,60], but the corresponding design methods result in $g = 8$. Next, we prove that by using the SMCs assumption we can instead design codes with girth up to 12.

Let $c < a \leq L$ $(a, c, L \in \mathbb{N})$ and consider the exponent matrix $\mathbf{P}$ for a QC-LDPC code in the form (SMCs assumption)

$$\mathbf{P}_{c\times a}^{\mathrm{SMC}} = \left[\begin{array}{c|c|c|c|c|c} \vec{0} & \vec{P}_1 & \gamma_2 \otimes \vec{P}_1 & \gamma_3 \otimes \vec{P}_1 & \ldots & \gamma_{a-1} \otimes \vec{P}_1 \end{array} \right], \tag{4.25}$$

where $\vec{0}$ and $\vec{P}_1$ are column vectors of size c. $\vec{0}$ is an all zero vector and $\vec{P}_1$ is a vector with first (i.e., top most) entry equal to zero, second entry equal to one, while its remaining entries are selected from $\{2, \ldots, L-1\}$, in an increasing order. Vectors $\gamma_j \otimes \vec{P}_1$ $(j = 2, \ldots, a-1;\ \gamma_j \in \{2, \ldots, L-1\}$ and $\gamma_j < \gamma_{j+1})$ are obtained from the base vector $\vec{P}_1$ through sequential multiplications, and $\otimes$ represents multiplication mod L. Let us denote by $\mathcal{I}_{0,1,2,\ldots,j}^{k}$, $j = 2, \ldots, a-1$ a set containing all relations (4.24) corresponding to the potential cycles having lengths ranging between 4 and $2k$ $(k = 2, 3, 4, 5)$ in the Tanner graph of (4.25). The following proposition holds.

Proposition 4.5.1 Let $\mathbf{P}_{c\times a}^{\mathrm{SMC}}$ be the exponent matrix of a QC-LDPC block code $\mathcal{C}$ as defined in (4.25). Suppose that the Tanner graph associated to the submatrix $\left[\vec{0}|\vec{P}_1\right]$ contains no strictly avoidable cycles of length up to λ, $\lambda \in \{4, \ldots, 10\}$. Then, the Tanner graph of $\mathcal{C}$ has no strictly avoidable cycle of length up to λ for sufficiently large L and a proper choice of γ_j's.

Proof. Demonstration is conducted inductively, which means that γ_2 is determined first, followed sequentially by $\gamma_3, \gamma_4, \ldots, \gamma_{a-1}$. Since we are considering cycles of length up to 10, for each element in $\mathcal{I}_{0,1,\ldots,s-1}^{k}$ we can write a relation of type (4.24) consisting of, at most, five parts (depending on the overall length

of a considered cycle), namely

$$\alpha_i\gamma_i + \alpha_j\gamma_j + \alpha_h\gamma_h + \alpha_k\gamma_k + \alpha_{s-1}\gamma_{s-1} \tag{4.26}$$

where each two successive indexes are distinct, i.e. $i \neq j \neq h \neq k \neq s-1$. According to (4.25), the coefficients α_l's include only the elements $p_{i0} = 0$ and p_{i1}, $i = 0, \ldots, c-1$. Hence, $a_0 = 0$, while, if present in (4.26), $\gamma_1 = 1$. Having assumed that the Tanner graph relative to the submatrix $\left[\vec{0}|\vec{P}_1\right]$ has no strictly avoidable cycles of length up to 10, it has to be $a_i \neq 0$, $\forall i \in [1, \ldots, s-1]$. In order to ensure that the whole expression is different from 0 as well, it is sufficient to choose

$$\gamma_{s-1} > \left| \frac{-\alpha_i\gamma_i - \alpha_j\gamma_j - \alpha_h\gamma_h - \alpha_k\gamma_k}{\alpha_{s-1}} \right|$$

with values $\in \{\gamma_{s-2}+1, \ldots, L-1\}$. This condition must hold for any element of $\mathcal{I}^k_{0,1,\ldots,s-1}$. So, setting $\lambda^k_{0,1,\ldots,s-1} = \max\{|x| \mid x \in \mathcal{I}^k_{0,1,\ldots,s-1}\}$, and $L > \lambda^k_{0,1,\ldots,s-1}$, all the elements in $\mathcal{I}^k_{0,1,\ldots,s-1}$ are non-zero mod L. The final value of L results at the end of this analysis, that is for $s = a$. ■

Based on Proposition 4.5.1, we have developed a search algorithm that finds the smallest possible $\gamma_j \in \{\gamma_{j-1}+1, \ldots, L-1\}$, $j = 2, \ldots, a-1$, that leads to $g = 12$. From the complexity viewpoint, we can estimate the advantage resulting from the SMCs assumption by considering that the exhaustive search of $\mathbf{P}$ requires to find ac elements. Instead, with our method, we only need to find $a+c-4$ values, namely, $c-2$ entries of the vector $\vec{P}_1$ and $a-2$ multiplication factors. A formal description of the proposed greedy search procedure is given in Algorithm 1. As inputs, it takes a, c, L ($c < a \leq L$), with $c, a, L \in \mathbb{N}$, k ($= 2, 3, 4, 5$), and an all zero matrix $\mathbf{P}$ of dimension $c \times a$. As output, it returns 0 if there is no feasible solution, or an exponent matrix with girth $g \geq 2k$, otherwise. Moreover, the minimum possible m_s for each exponent matrix of SC-LDPC-CCs has been found through the min-max optimization model, described in 4.4.2.

4.5.2 Latency and complexity performance

By applying the method presented in the previous sections, we have designed several codes with values of L and m_s in many cases significantly smaller than those of classical codes with the same code rate and girth. The method illustrated for the case of $g = 12$ has been applied also for the case of $g = 10$. In particular, we have considered $c = 3, 4$ and $a = 4, \ldots, 12$. The obtained values of L and m_s have been compared with those resulting from the application of classical design approaches reported in [49, 51, 52, 61, 62], which, to the best of

Algorithm 1 Greedy search algorithm

Input: $c,\ a,\ L,\ k$ and zero matrix $\mathbf{P} = \left[\vec{0}|\vec{P}_1|\vec{P}_2|\ldots|\vec{P}_{a-1}\right]$
Output: Exponent matrix $\mathbf{P}$ with girth $2k$

$p_{10} \leftarrow 0,\ p_{11} \leftarrow 1,\ \mathcal{S}_2 \leftarrow (c-2)$-combinations of $\{2,\ldots,L-1\}$
top:
Pick $(p_{21},\ldots,p_{(c-1)1})$ from $\mathcal{S}_2$ in a way that $p_{21} < p_{31} < \ldots < p_{(c-1)1}$
$\mathcal{S}_2 \leftarrow \mathcal{S}_2 \setminus \{(p_{21},\ldots,p_{(c-1)1})\}$
if at least one of the relations in $\mathcal{I}_{0,1}^{k-1}$ results in a cycle and $|\mathcal{S}_2| > 0$ **then**
    goto *top*
else if $|\mathcal{S}_2| = 0$ **then**
    return 0
for $j:\ 2$ to $a-1$ **do**
    $\mathcal{L}_j \leftarrow \{\gamma_{j-1}+1,\ldots,L-1\}$
loop:
    Pick γ_j from the set $\mathcal{L}_j$
    $\mathcal{L}_j \leftarrow \mathcal{L}_j \setminus \{\gamma_j\}$
    $\vec{P}_j \leftarrow \gamma_j \otimes \vec{P}_1$
    if at least one of the relations in $\mathcal{I}_{0,1,\ldots,j}^{k-1}$ results in a cycle and $|\mathcal{L}_j| > 0$ **then**
        goto *loop*
    else if $|\mathcal{L}_j| = 0$ **then**
        return 0
return P

our knowledge, are those producing the codes with the minimum values of L and m_s. The comparison with our results is shown in Figs. 4.5 and 4.6.

We see that the values obtained through our approach are everywhere smaller (often significantly) than those derived with the previous solutions. Let us denote as $\tilde{L}$ ($\tilde{m}_s$) the smallest lifting degree (syndrome former memory order) found with our approach, and as L^* (m_s^*) the minimum value found through previous approaches. The ratio of the decoding latency of the newly designed QC-LDPC block codes over that of the classical ones is

$$\Theta_L = \frac{\tilde{L}}{L^*}.$$

Based on (3.1), we can also assess the ratio of decoding complexity (per output bit) and latency achieved by the newly designed SC-LDPC-CCs over the classical ones, as

$$\Theta_{m_s} = \frac{\tilde{m}_s + 1}{m_s^* + 1}. \tag{4.27}$$

The values of Θ_L and Θ_{m_s} should be kept as small as possible if we aim at minimizing the decoding latency and complexity. We have obtained values of Θ_L as small as 0.47, which means a reduction in decoding latency by more than 50%, and values of Θ_{m_s} as small as 0.23, yielding a reduction of Λ_{SW} and Γ_{SW} by more than 75%, with respect to previous solutions.

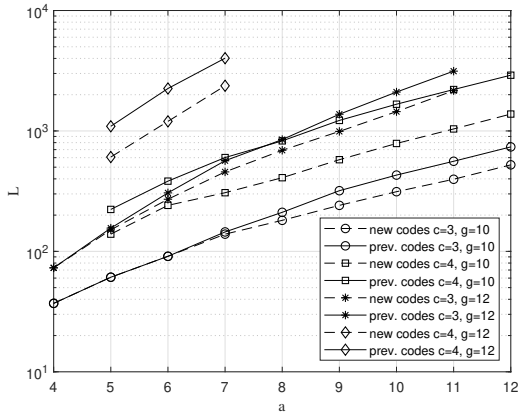


Figure 4.5: Minimum lifting degree (L) of new and previously designed QC-LDPC codes versus a, for $c = 3, 4$.

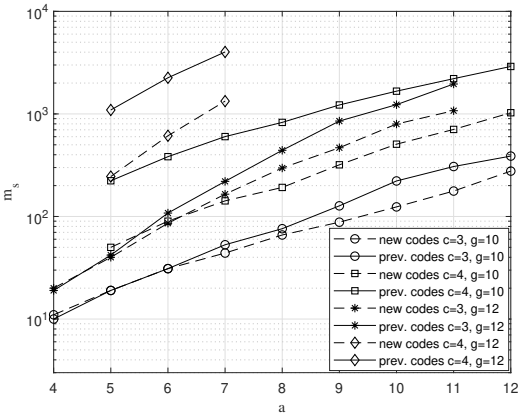


Figure 4.6: Minimum syndrome former memory order (m_s) of new and previously designed SC-LDPC-CCs versus a, for $c = 3, 4$.

Table 4.6: Values of a, c, m_s, v_s and g of the considered SC-LDPC-CCs with $R = \frac{4}{7}$.

Code	a	c	m_s	v_s	g	Θ_{m_s}
C_1	7	3	44	315	10	–
C_2	7	3	165	1162	12	–
C_{B1}	7	3	53	378	10	0.83
C_{B2}	7	3	220	1547	12	0.75
C_{B3}	7	3	88	623	10	0.51
C_{B4}	7	3	432	3031	12	0.38

4.5.3 Numerical simulations of coded transmissions

As a further benchmark of the newly designed codes, we have estimated the BER of our SC-LDPC-CCs through Monte Carlo simulations of BPSK modulated transmissions over the AWGN channel, and compared it with that of some codes constructed following [51, 52]. A full-size BP decoder and a SW decoder, both performing 100 iterations, have been used in the simulations. Let us consider two of our codes ($\mathcal{C}_1$ and $\mathcal{C}_2$) with $R = \frac{4}{7}$ and $g = 10$ and 12, two previous codes ($\mathcal{C}_{B1}$ and $\mathcal{C}_{B2}$) designed following [52] and two codes ($\mathcal{C}_{B3}$ and $\mathcal{C}_{B4}$) obtained by unwrapping QC-LDPC block codes designed as in [51]. The parameters of these codes are summarized in Table 4.6. Their BER performance is shown in Fig. 4.7. We notice that the performance degradation is minimal for both very large and small window sizes. So, we can conclude that the new SC-LDPC-CCs do not exhibit any significant loss with respect to the classical codes, while they enjoy reduced latency and complexity. The value of Θ_{m_s}, according to (4.27), is also shown in Table 4.6.

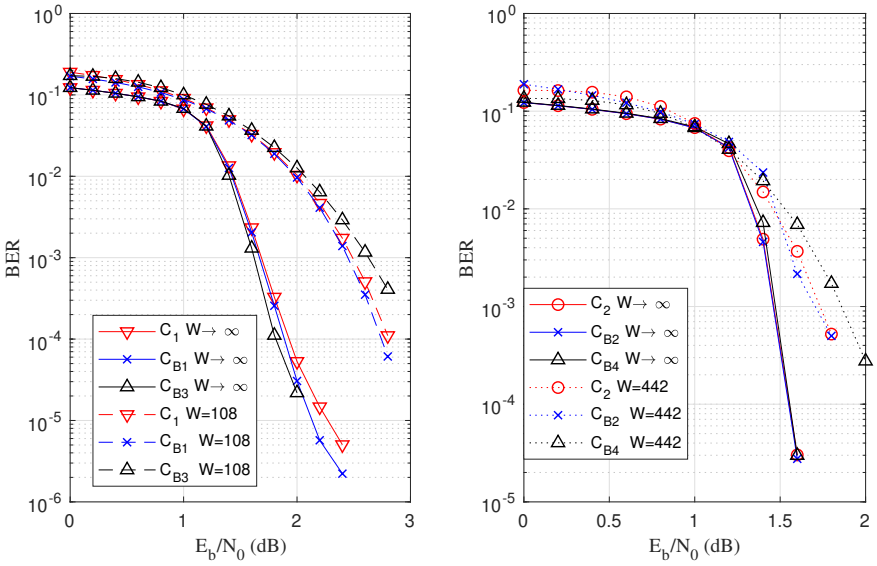


Figure 4.7: Simulated performance of SC-LDPC-CCs with: (a) $g = 10$ and (b) $g = 12$ as a function of the signal-to-noise ratio.

4.6 Summary

In this chapter we have investigated the design of compact SC-LDPC-CCs. We have shown that a direct design, which does not start from block codes, permits to obtain codes with shorter constraint length with respect to that of the best codes found in previous literature. Several examples and simulations show the effectiveness of our method. The savings in terms of latency and complexity yielded by the newly designed codes are often significant. We have incorporated in the code design the removal of imperfections of the codes' Tanner graphs, which has led to an additional improvement of their error rate performance.

Chapter 5

SC-LDPC codes in complexity-constrained scenarios

LDPC code ensembles are usually designed for an asymptotically large codeword length and letting the number of decoding iterations tend to infinity. Nevertheless, practical applications may require a finite, and constrained, number of iterations. Code ensembles whose *asymptotic decoding thresholds* are optimized for an infinite number of iterations may result in sub-optimal codes in an iteration-constrained setting. Some efforts have been devoted to the optimization of block LDPC code ensembles for a finite number of iterations, over the binary erasure channel (BEC) and the AWGN channel [63–66]. We propose in this chapter an analysis in this regime for SC-LDPC-CCs. The complexity of the SW decoder increases linearly both with the number of decoding iterations and the window size, i.e., ultimately, the code syndrome former constraint length, as shown in Section 3.3.1. In resource constrained devices, complexity must be kept limited due to scarcity of hardware and software resources. So, we propose a method that can be used to find the best trade-off between the SW size and the number of decoding iterations. For this purpose, we define a simplified complexity metric capturing both aspects and we aim at finding optimized codes under complexity constraints. As usual, we then validate the results of our analysis through Monte Carlo simulations. We remark that the approach we propose is tailored to finding an optimal trade-off between the SW size and the number of decoding iterations for a given code, and not to designing new codes.

5.1 Finding the optimal trade-off between window size and number of iterations

As an example of application of the proposed method, we consider three SC-LDPC-CCs codes with asymptotic rate $R_\infty = \frac{3}{6}$; however, the analysis is independent of the code rate and can be performed for any values of a and c.

The binary and symbolic parity-check matrices of the considered codes have been designed in [52] and they are characterized by small syndrome former constraint lengths v_s. For the sake of completeness we report the symbolic parity-check matrices which are

$$\mathbf{H}_1(D) = \begin{bmatrix} 1 & D^{33} & 1 & D^{17} & D^{30} & D^{11} \\ D^{16} & D^{8} & D^{33} & 1 & 1 & D^{33} \\ D^{38} & 1 & D^{34} & D^{20} & D^{4} & 1 \end{bmatrix},$$

$$\mathbf{H}_2(D) = \begin{bmatrix} 1 & D^{100} & 1 & D^{100} & D^{100} & D^{100} \\ D^{54} & D^{91} & D^{108} & D^{24} & D^{62} & D^{52} \\ D^{47} & D^{18} & D^{108} & D^{22} & D^{75} & 1 \end{bmatrix},$$

$$\mathbf{H}_3(D) = \begin{bmatrix} 1 & D^{50} & D^{58} & 1 & D^{40} & 1 \\ D^{3} & 1 & D^{66} & D^{2} & D & D^{53} \\ D^{73} & D^{67} & 1 & D^{36} & 1 & D^{56} \end{bmatrix}.$$

Their thresholds have been assessed through PEXIT analysis, with target mutual information 0.99, under a fixed value of complexity C, which, for the sake of simplicity, is written as $C = \alpha v_s I_{\max} = aMWI_{\max} = a'\alpha(m_s+1)I_{\max}$, where α usually takes values in the set $5, 6, \ldots, 10$ and M is the size of the matrices we lift the convolutional protograph with. Note that, since W and $I_{\max}$ are integers, it may not always be possible to obtain exactly a given and fixed integer value of C, therefore we consider its nearest value $C_{act} = a\lfloor \frac{C}{aI_{\max}} \rfloor I_{\max}$, where $\lfloor \frac{x_1}{x_2} \rfloor$ represents the integer value of $\frac{x_1}{x_2}$. Obviously, $C_{act} \leq C$, which complies with the fact that values of complexity higher than C are not allowed.

In Fig.s 5.1 and 5.2 the optimal thresholds $(\frac{E_b}{N_0})^*$ so found are shown as a function of the window size, for some fixed values of complexity. The corresponding number of decoding iterations can be easily derived as $I_{\max} = \lfloor \frac{C}{Wa} \rfloor$. It is known that a window size as large as 5 to 10 times (m_s+1) usually results in an optimal choice when the number of decoding iterations is unconstrained, or rather high. We notice that, under a stringent complexity constraint, the optimal window size tends to be smaller than 5 times (m_s+1). In fact, the optimal values of α found through the aforementioned optimization procedure are $\frac{3}{2} \leq \alpha^* \leq 4$. Table 5.1 summarizes the results of the threshold optimization for $R_\infty = \frac{3}{6}$ and $C = 6000, 24000$.

From Fig.s 5.1 and 5.2 we observe that the analysis based on PEXIT indeed results in a convex function for each of the considered codes. Therefore, we are able to find the optimal trade-off between W and $I_{\max}$ by minimizing those functions, for integer values of both W and $I_{\max}$. This way, we are able to design

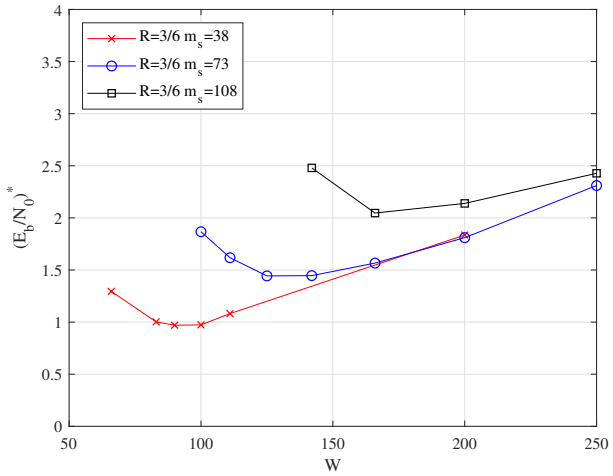


Figure 5.1: Thresholds $\frac{E_b}{N_0}^*$ vs W for $R_\infty = \frac{3}{6}$ codes when $C = 6000$, $M = 1$.

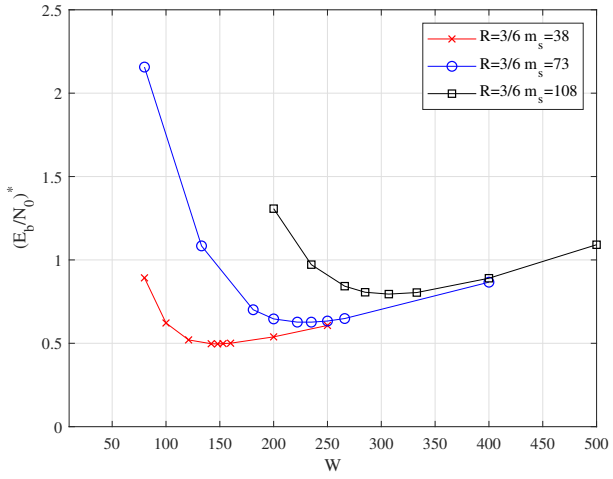


Figure 5.2: Thresholds $(\frac{E_b}{N_0})^*$ vs W for $R_\infty = \frac{3}{6}$ codes when $C = 24000$, $M = 1$.

the decoder parameters in such a way that the best asymptotic performance is achieved under the given complexity constraint.

These results are also confirmed in non-asymptotic conditions, through Monte Carlo simulations of BPSK modulated transmissions. Figure 5.3 reports some examples of BER curves obtained for the three codes. A lifting procedure, exploiting random circulant permutation matrices with size $M = 25$, has been used. Even though the lifting procedure results in upscaling the decoding

Table 5.1: Optimal values of the window size and number of decoding iterations for $R_\infty = \frac{3}{6}$ and $C = 6000, 24000$.

	$C = 6000$			$C = 24000$		
m_s	38	73	108	38	73	108
W	90	125	166	148	235	307
$I_{\max}$	11	8	6	27	17	13
C_{act}	5940	6000	5976	23976	23970	23946
$(\frac{E_b}{N_0})^*$	0.97	1.4432	2.0463	0.4964	0.6269	0.7950

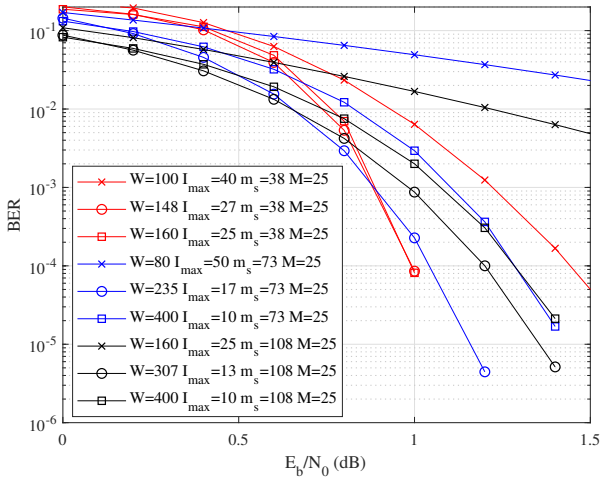


Figure 5.3: BER performance and PEXIT thresholds of the proposed codes, lifted with circulant permutation matrices of size $M = 25$.

complexity by a factor $M = 25$, it is needed to approach the asymptotic performance assessed through PEXIT analysis that, eventually, would be achieved for $M \to \infty$ in the waterfall region. This means that the complexity constraint used in the PEXIT analysis shall take into account such an expansion by a factor M when real codes are designed.

The performance of the three codes has been simulated for the parameters corresponding to the minimum values of the convex functions of Fig. 5.2 and for two other values, on both sides of the minimum. We highlight that, when the values of W and $I_{\max}$ are chosen to be equal to the best trade-offs or are close to them, the BER curves are coherent with the PEXIT analysis, as expected. Instead, when decoding is performed with an excessively small window size,

Table 5.2: Values of the PEXIT thresholds for the codes in Fig. 5.3, $C \leq 24000$.

	$m_s = 38$			$m_s = 73$			$m_s = 108$		
W	100	148	160	80	235	400	160	307	400
$I_{\max}$	40	27	25	50	17	10	25	13	10
$(\frac{E_b}{N_0})^*$	0.6	0.5	0.5	2.2	0.6	0.9	1.9	0.8	0.9

considerable performance losses are experienced. For completeness, Table 5.2 reports the asymptotic thresholds found through PEXIT for the codes and parameters in Fig. 5.3.

On the other hand, from Fig.s 5.1 and 5.2 it apparently results that codes with smaller syndrome former constraint length outperform those with larger syndrome former constraint length, which may seem counterintuitive. However, we cannot neglect the stringent complexity constraints under which we are developing our analysis. We indeed remark that an increase in the syndrome former constraint length does not necessarily imply an improvement of the asymptotic performance, although this generally occurs when the codes are designed through approaches that aim at optimizing their syndrome former constraint length. If we relax such constraints and allow a fixed and large number of decoding iterations, as well as a large window size, from the PEXIT analysis we find that the codes with larger syndrome former constraint lengths asymptotically achieve better performance than those with shorter ones.

5.2 Summary

In this chapter we have provided a method to find optimal trade-offs between the decoding window size and the number of decoding iterations, considering a SW decoder. The theoretical asymptotic analysis has been validated through simulations of finite-length performance.

Chapter 6

Connection between cycles and codewords of SC-LDPC convolutional codes

In this chapter we consider time-invariant SC-LDPC-CCs, for which we study the connections existing between their low-weight codewords and cycles in their Tanner graphs. Using the polynomial representation of these codes, we show that parity-check matrices having columns with weight ≥ 2 can be analyzed considering a certain number of parity-check sub-matrices having regular columns with weight 2. These sub-matrices are associated to cycles in the code Tanner graph. Based on this observation, we find that codewords of the main code can be expressed as a combination of codewords of the *component codes*. The design of codes free of codewords up to a certain weight is also addressed. We show that low-weight codewords in the main code can be avoided by removing cycles in its Tanner graph.

According to the definition given in Section 2.3.1, the structure of the symbolic matrix $\mathbf{H}(D)$ defines ensembles of codes which can have unavoidable codewords, which can computed by using an algebraic method [25]. The minimum weight of an unavoidable codeword obviously provides an upper bound on the code free Hamming distance, as shown in Section 2.3.1. It can be easily demonstrated that, in codes having a parity-check matrix, and thus symbolic matrix, with all columns with weight 2, a single cycle can be associated to a codeword. Instead, if $\mathbf{H}(D)$ and $\mathbf{H}$ have at least one column with weight greater than 2, then more than one cycle has to be properly combined in order to define a codeword. In this chapter we show how this can be done for some common forms of $\mathbf{H}(D)$, but the approach is general and can be extended to other ensembles of codes not considered here.

The analysis in this chapter is related to that of trapping sets [14, 67–69], which are known to strongly influence the performance of LDPC codes in the error floor region [11].

In [70], the authors propose a search method for the low-weight codewords

of structured LDPC codes with constant column weight. Such a method relies on dissecting trapping sets of a code into smaller trapping sets belonging to the same code. In order to perform efficient searches, the knowledge of the topologies of these smaller trapping sets is required.

We instead show here that codewords can always be seen as the superposition of a number of lower weight component codewords which do not belong to the original code, but to component codes defined by parity-check matrices with column weight $d_v = 2$. This way, any codeword can be represented as a combination of a number of component codewords, associated to cycles having a length that, in general, cannot be established a priori.

Differently from previous approaches, the analysis carried out in this chapters is also valid for irregular codes and does not require any knowledge about trapping sets.

6.1 Notation

In this section we give a representation of cycles slightly different from those of the previous chapters, but is more suitable for the analysis in this chapter.

Let us suppose that one of the entries of $\mathbf{H}(D)$ is a polynomial with weight w; then, there exist $\binom{w}{2}$ vertical separations between symbols 1 in the binary parity-check matrix, associated to such a polynomial. For the sake of clarity, let us consider $h_{i,j}(D) = D^{q_1} + D^{q_2}$, with $q_2 > q_1$; this implies the existence of the following vertical separation between the corresponding symbols 1 in the matrix $\mathbf{H}_s^T$

$$d_l = (q_2 - q_1)c. \tag{6.1}$$

In fact, the jth column of $\mathbf{H}_s^T$ contains these two symbols 1 in the blocks $\mathbf{H}_{q_1}$ and $\mathbf{H}_{q_2}$, at the ith row of these two blocks.

Let us consider two entries in the same column of $\mathbf{H}(D)$ having weight w_A and w_B, respectively; then, there exist $\binom{w_A}{2}+\binom{w_B}{2}+w_Aw_B = \binom{w_A+w_B}{2}$ vertical separations between symbols 1 in $\mathbf{H}$, associated to the two polynomials. For better clarity, let us consider two polynomials in the jth column of $\mathbf{H}(D)$, in the i_1th and i_2th row, respectively, having the following form: $h_{i_1,j}(D) = D^{q_1}$ and $h_{i_2,j}(D) = D^{q_2}$, with $q_2 > q_1$. This implies the existence of the following vertical separation between the corresponding symbols 1 in the matrix $\mathbf{H}_s^T$

$$d_l = (q_2 - q_1)c + (i_2 - i_1). \tag{6.2}$$

In fact, the jth column of the matrix $\mathbf{H}_s^T$ contains these two symbols 1 in the blocks $\mathbf{H}_{q_1}$ and $\mathbf{H}_{q_2}$, at their rows i_1 and i_2, respectively.

A cycle of length λ corresponds to an equality of the following type, involving

differences of the type (6.1) and/or (6.2)

$$d_{l_0} \pm d_{l_1} \pm \ldots \pm d_{l_{\frac{\lambda}{2}-1}} = 0. \tag{6.3}$$

In general, as already mentioned in Section 4.5, cycles can be grouped in two categories: the avoidable ones and the unavoidable ones. Avoidable cycles are associated to particular d_l values; so, in principle, they can be avoided by properly choosing the exponents of the polynomials in $\mathbf{H}(D)$. Unavoidable cycles, instead, only depend on the structure of $\mathcal{W}(\mathbf{H}(D))$, where $\mathcal{W}(\cdot)$ has been introduced in Section 2.3.1, no matter what the exponents of the polynomials are, and can be split, in their turn, in two categories:

i. unavoidable cycles associated to single elements of $\mathcal{W}(\mathbf{H}(D))$ larger than 1

ii. unavoidable cycles associated to submatrices of $\mathcal{W}(\mathbf{H}(D))$

An unavoidable cycle of the first type can be associated, for example, to a polynomial $h_{i,j}(D) = D^{q_1} + D^{q_2} + D^{q_3}$, which has weight 3. In such a case, the following vertical separations always exist

$$\begin{aligned} d_{l_0} &= (q_2 - q_1)c, \\ d_{l_1} &= (q_3 - q_1)c, \\ d_{l_2} &= (q_3 - q_2)c \end{aligned}$$

and $d_{l_0} - d_{l_1} + d_{l_2} = 0$ represents an unavoidable cycle of length 6.

An example of unavoidable cycle of the second type is that with length 12 occurring when $\mathcal{W}(\mathbf{H}(D))$ contains the following submatrix, or its transposed version,

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Such a cycle is described in [71, Fig. 1]; the corresponding equation is $d_{l_0} + d_{l_1} + d_{l_2} - d_{l_3} - d_{l_4} - d_{l_5} = 0$, where $d_{l_0} = d_{l_3}$, $d_{l_1} = d_{l_4}$, $d_{l_2} = d_{l_5}$.

We define $s_{k,i}$ as the number of different cycles involving the kth symbol 1 of the ith column of $\mathbf{H}$. Restricting to the columns of the syndrome former matrix, which describe all the parity-check matrix columns due to the code time-invariancy, we also define the matrix of cycle superposition degrees as $\mathbf{S} = s_{k,i}$, $k = 0, \ldots, w_i - 1$, $i = 0, \ldots, a - 1$, where w_i is the entry at position (i, i) in $\mathbf{W}$, whose structure is described in (2.5). We will prove in Theorem 6.2.1 that $s_{k,i} = s$, $\forall k, i$, thus showing that the cycle superposition degree indeed is a constant (s).

6.2 Codewords as superposition of component codewords

The following lemmas concerning the binary and symbolic parity-check matrices of SC-LDPC-CCs hold.

Lemma 6.2.1 Given an SC-LDPC-CC described by $\mathbf{H}(D)$, let us consider the SC-LDPC-CC described by $\mathbf{Q}(D) = q(D)\mathbf{H}(D)$, where $q(D)$ is a non-null polynomial $\in F_2[D]$. Then, the codes described by $\mathbf{H}(D)$ and $\mathbf{Q}(D)$ are coincident [72, Definition 1].

Proof. Let us consider the symbolic parity-check matrix obtained by multiplying all the entries of $\mathbf{H}(D)$ by $q(D)$, noted by $\mathbf{Q}(D)$. It can be easily proved that the corresponding binary parity-check matrix $\mathbf{Q}$ contains rows which are obtained as linear combinations of the rows of $\mathbf{H}$ and that $\text{rank}(\mathbf{H}) = \text{rank}(\mathbf{Q})$. Thus, the code described by $\mathbf{Q}(D)$ coincides with the code described by $\mathbf{H}(D)$. ■

Lemma 6.2.2 Given an SC-LDPC-CC described by $\mathbf{H}(D)$, let us consider a coincident code described by $\mathbf{E}(D) = (1 + D^e)\mathbf{H}(D)$, where $e \in \mathbb{Z}_{>0}$; then, all the non-zero entries of $\mathbf{E}(D)$ have even weight.

Proof. Let us consider a polynomial $p(D) \in F_2[D]$ with weight w_p, then $(1 + x^D)p(D)$ results in a polynomial $z(D) \in F_2[D]$ with weight $w_z = 2w_p - 2C = 2(w_p - C)$, where C is the number of monomials appearing both in $p(D)$ and in $D^e p(D)$. ■

Lemma 6.2.3 Given an SC-LDPC-CC described by $\mathbf{H}$ with all even-weight columns, let us consider any of its codewords, say $\mathbf{T}$, and its support $I_{\mathbf{T}}$, from which $\tilde{\mathbf{H}}$ can be readily obtained. Then, any of the symbols 1 of $\tilde{\mathbf{H}}$ is involved in at least one cycle in the Tanner graph associated to $\tilde{\mathbf{H}}$.

Proof. $\tilde{\mathbf{H}}$ has $|I_{\mathbf{T}}|$ columns, whose sum modulo 2 results in the all-zero vector, because $\mathbf{H}\mathbf{T}^T = 0$. Let us denote as r the number of rows of $\tilde{\mathbf{H}}$. Let us consider a symbol 1 in the (i, j)th entry of $\tilde{\mathbf{H}}$. According to the above consideration, there always exists a symbol 1 in the (i, j')th entry of $\tilde{\mathbf{H}}$, for some $j' \neq j$. On the other hand, $\tilde{\mathbf{H}}$ contains a subset of the columns of $\mathbf{H}$, which have even weight by definition. Hence, given a symbol 1 in the (i, j)th entry of $\tilde{\mathbf{H}}$, there always exists a symbol 1 in the (i', j)th entry of $\tilde{\mathbf{H}}$, for some $i' \neq i$. It follows that, given a pair of symbols 1 in the i_0th row of $\tilde{\mathbf{H}}$, there always exists a row i_1 containing a pair of symbols 1 such that at least one of the symbols 1 in row i_0 belongs to the same column of one of the symbols 1 in row i_1. There are two cases:

- both the symbols 1 in row i_1 are in the same columns as both the symbols 1 in row i_0 and a cycle of length 4 exists
- a symbol 1 in row i_0 and a symbol 1 in row i_1 are not in the same column.

In the second case, there always exists a row i_2 containing a pair of symbols 1 such that the remaining symbol 1 in row i_0 is in the same column of one of the symbols 1 in row i_2. At this point,

- both the symbols 1 in row i_2 are in the same columns as both the symbols 1 in row i_0 and a cycle of length 4 exists
- a symbol 1 in row i_2 is in the same column of the remaining symbol 1 in row i_1 and a cycle of length 6 exists
- a symbol 1 in row i_1 and a symbol 1 in row i_2 are not in the same column

In the last case, we can apply iteratively the same arguments to the remaining symbols of each row. In the worst case scenario, there remain:

i. $r-3$ rows containing at least two symbols 1 which have been associated to other symbols 1 in different rows;

ii. two rows, denoted as i_{f-2} and i_{f-1}, containing a symbol 1 which has been associated to another symbol 1 in a different row and at least a symbol 1 which has not been associated to any other symbol 1;

iii. a row, i_f, containing symbols 1 which have not been associated to any other symbol 1.

It follows from the initial considerations that there must exist a row containing a symbol 1 in the same column of the remaining symbol of i_{f-2} and a row containing a symbol 1 in the same column of the remaining symbol of i_{f-1}. Nevertheless, there is only an available row, i_f, which necessarily contains both these symbols 1. When a cycle involving some pairs of ones in the parity-check matrix has been found, it can be ignored (together with the corresponding ones) and the same analysis can be applied on the remaining symbols, until all the ones in the matrix are included in some cycle. In fact, by definition, a cycle involves exactly two symbols 1 in each column; if we ignore these symbols, we obtain a column with an even or null weight. Therefore, when the resulting weight is not null, the above analysis focused on pairs of ones can be applied again, whereas this is not possible if odd-weight columns are allowed. ■

As highlighted in the proof of Lemma 6.2.3, the existence of odd-weight columns is a necessary (but not sufficient) condition for the existence of ones in $\tilde{\mathbf{H}}$ which are not involved in any cycle. The case of columns of $\mathbf{H}(D)$ with weight 1 is a trivial example. Therefore, in such cases Lemma 6.2.3 does not hold.

However, in those cases it is easy to find an alternative though equivalent form of $\mathbf{H}(D)$ with all even-weight columns, which hence allows applying Lemma 6.2.3. In order to obtain such an equivalent form of $\mathbf{H}(D)$, it is sufficient to multiply all the entries of $\mathbf{H}(D)$ by a binomial $\in F_2[D]$ such as, for example, $(1 + D)$. This way, a coincident code described by $\mathbf{E}(D) = (1 + D)\mathbf{H}(D)$ is obtained. Then, according to Lemma 6.2.2, Lemma 6.2.3 holds for $\mathbf{E}(D)$.

The symbolic matrices for which all the ones of $\tilde{\mathbf{H}}$ are involved in at least one cycle are named *cycle-coverable* matrices. We remark that, according to the above considerations, symbolic matrices which are not cycle-coverable can always be transformed into cycle-coverable matrices with a simple linear transformation. Notice that cycle-coverable symbolic matrices cannot have columns with weight 1.

Lemma 6.2.4 Provided that there are no null columns in $\mathbf{H}(D)$, $\mathbf{W}^{-1}$ always exists.

Proof. Any column of $\mathbf{H}(D)$ contains at least one element, that is, $w_i \neq 0$, $\forall i = 0, \ldots, a-1$. Thus, owing to the diagonal form of $\mathbf{W}$, we have $\det(\mathbf{W}) = \prod_{i=0}^{a-1} w_i \neq 0$. So, $\mathbf{W}^{-1}$ always exists over the rational positive numbers $\mathbb{Q}_{>0}$, and we have $\mathbf{W}^{-1} = \mathbf{diag}(w_0^{-1}, \ldots, w_{a-1}^{-1}) = \mathbf{diag}(\frac{1}{w_0}, \ldots, \frac{1}{w_{a-1}})$. ■

Theorem 6.2.1 Any codeword $\mathbf{T}(D)$ of an SC-LDPC-CC described by a cycle-coverable symbolic matrix can be written as

$$\mathbf{T}(D) = \left[\sum_{j=0}^{N-1} t_j(D) \right] \mathbf{W}^{-1} \frac{2}{s}, \tag{6.4}$$

where N is the number of component codewords $\mathbf{t}_j(D)$, $j = 0, \ldots, N-1$ and s is an integer representing the number of cycles involving the ones in $\tilde{\mathbf{H}}$.

Notice that the sum in (6.4) is performed over the set of integers $\mathbb{Z}$.

Proof. Let us consider a codeword $\mathbf{T}(D) = [T_0(D), \ldots, T_{a-1}(D)]$ and the corresponding weight vector $\mathcal{W}(\mathbf{T}(D))$.

We denote the lth monomial in $T_i(D)$ as $T_i^{(l)}(D)$, $l = 0, \ldots, \mathcal{W}(T_i(D)) - 1$, assuming ascending order of the powers. According to Lemma 6.2.3, $\tilde{\mathbf{H}}$ contains a number of cycles N, corresponding to as many codewords $\mathbf{t}_j(D) = [t_{j,0}(D), \ldots, t_{j,a-1}(D)]$, $j = 0, \ldots, N-1$, of component codes, such that

$$N_{l_i} T_i(D) = \sum_{f=0}^{N_i - 1} t_{f,i}(D) \quad \forall i = 0, \ldots, a-1, \tag{6.5}$$

where $t_{f,i}(D) \subseteq T_i(D)$, N_i is the number of component codewords such that $t_{f,i}(D) \neq 0$ and N_{l_i} is the number of component codewords such that $T_i^{(l)}(D) \subseteq$

$t_{f,i}(D)$.

The component codewords cannot have unitary weight; in fact, in a code defined by a column-regular parity-check matrix with column weight 2, a codeword with weight w is associated to a cycle of length $2w$, and the minimum cycle length is 4.

So, N_{l_i} is the number of cycles with a vertical edge in the mth column of $\mathbf{H}$, which corresponds to the ith column of $\mathbf{H}_s^T$ ($i = m \mod a$); notice that any vertical edge touches exactly two symbols 1 of $\mathbf{H}_s^T$. Furthermore, any of the w_i ones in a column of $\mathbf{H}_s^T$ is involved, by definition, in $s_{k,i}$ cycles. It follows that $2N_{l_i}$ must be equal to $s_{k,i}w_i$, that is $s_{k,i} = \frac{2N_{l_i}}{w_i}$, $\forall k = 0, \ldots, w_i - 1$. $\frac{2N_{l_i}}{w_i}$ does not depend on k, hence $s_{k,i} = s_i \ \forall k$. Thus, (6.5) can be rewritten as

$$s_i w_i T_i(D) = 2 \sum_{f=0}^{N_i-1} t_{f,i}(D) \quad \forall i = 0, \ldots, a-1.$$

Generalization to all the columns can be done if $s_i = s \ \forall i$, i.e., the ones in all the columns have the same superposition degree. Let us consider $\tilde{\mathbf{H}}^T$, which contains only even-weight columns (any row of $\tilde{\mathbf{H}}$ has even weight since it sums to zero) and none of its rows has unitary or null weight (since we have assumed that $\mathbf{H}(D)$ is cycle-coverable, thus the minimum weight of its columns is 2). It follows that Lemma 6.2.3 applies to $\tilde{\mathbf{H}}^T$ and an equation similar to (6.5) can be written for the rows of $\tilde{\mathbf{H}}$. Thus, using the same arguments as above, we can prove that $s_{k,i} = s_k$, $\forall i$. Summarizing, $s_{k,i} = s_k$, $\forall i$ and $s_{k,i} = s_i$, $\forall k$, i.e. $s_{k,i} = s$, $\forall i, k$. We can finally derive

$$\mathbf{WT}(D) = \frac{2}{s} \sum_{j=0}^{N-1} \mathbf{t}_j(D),$$

which, reversed, leads to (6.4). ■

We remark that there may be different combinations of component codewords leading to the same codeword. In fact, all the combinations that verify (6.4) are valid. The following corollaries permit us to estimate the weight and the number of the component codewords.

Corollary 6.2.1 The component codewords $\mathbf{t}_j$, $j = 0, 1, \ldots N-1$ that must be combined in order to obtain any codeword $\mathbf{T}$ of a code described by a cycle-coverable symbolic matrix satisfy

$$g \leq \tilde{g} \leq 2\mathcal{W}(\mathbf{t}_j) \leq 2\mathcal{W}(\mathbf{T}), \tag{6.6}$$

$\forall j \in [0, \ldots, N-1]$, where $\tilde{g}$ is the girth of $\tilde{\mathbf{H}}$.

Proof. Given $\mathbf{T}$, let us consider $\tilde{\mathbf{H}}$ which, by definition, contains a subset of the columns of $\mathbf{H}$. It can be easily proved that the girth of $\tilde{\mathbf{H}}$ is greater than or equal to the girth of $\mathbf{H}$. Considering that any $\mathbf{t}_j$ is associated to a cycle in $\tilde{\mathbf{H}}$, we have that the minimum weight of a component codeword is $\frac{\tilde{g}}{2}$. Furthermore, any cycle can cover at most all the $\mathcal{W}(\mathbf{T})$ columns of $\tilde{\mathbf{H}}$, if its length is $2\mathcal{W}(\mathbf{T})$. Equation (6.6) is a straightforward consequence of these considerations. ■

Corollary 6.2.2 The component codewords $\mathbf{t}_j$, $j = 0, 1, \ldots N-1$ that must be combined in order to obtain any codeword $\mathbf{T}$ of a code described by a cycle-coverable symbolic matrix satisfy

$$2\sum_{j=0}^{N-1} \mathcal{W}(\mathbf{t}_j) = s \sum_{i \in I_{\mathbf{T}}} w_{i \pmod a}. \tag{6.7}$$

Proof. Equation (6.7) is obtained by applying $\mathcal{W}(\cdot)$ to both sides of (6.4). ■

A straightforward consequence of Corollary 6.2.2 is that low-weight codewords do not necessarily derive from low-weight component codewords, but may be obtained as the combination of few component codewords with relatively high weight, as long as (6.7) is satisfied.

Let us consider the case in which $\mathbf{T}(D)$ is an unavoidable codeword, denoted by $\mathbf{U}(D)$; the codewords of the component code, previously denoted as $\mathbf{t}_j(D)$, will be denoted as $\mathbf{u}_j(D)$.

Corollary 6.2.3 Any unavoidable codeword $\mathbf{U}(D)$ of an SC-LDPC-CC described by a cycle-coverable symbolic matrix can be written as

$$\mathbf{U}(D) = \left[\sum_{j=0}^{N-1} \mathbf{u}_j(D)\right] \mathbf{W}^{-1} \frac{2}{s}, \tag{6.8}$$

where N is the number of unavoidable component codewords $\mathbf{u}_j(D)$, $j = 0, \ldots, N-1$.

Proof. Similar to the proof of Theorem 6.2.1. ■

In other words, the unavoidable codewords are obtained as the combination of unavoidable component codewords which, in their turn, are associated to unavoidable cycles. Depending on the structure of the code, it may happen that low-weight unavoidable codewords must be associated to unavoidable cycles with length larger than that of the avoidable ones. This means that, in particular scenarios, the cycles with the shortest length may not be the most dangerous ones.

The results of Theorem 6.2.1 and Corollaries 6.2.1, 6.2.2 and 6.2.3 can be summarized into a procedure which allows us to find the component codewords,

given a cycle-coverable symbolic parity-check matrix and any of its codewords. The procedure is as follows:

i. Given $\mathbf{H}(D)$ and $\mathbf{T}(D)$, derive $\tilde{\mathbf{H}}$ and enumerate all its cycles, whose lengths are characterized by(6.6).

ii. Combine the codewords corresponding to the cycles, in such a way that (6.7) is verified.

iii. If the chosen codewords guarantee a constant superposition degree, they are component codewords; otherwise, go back to step ii).

According to this method, component codewords can be found starting from any codeword and, in particular, from minimum weight codewords. On the other hand, it is not possible to exploit the component codes to find the minimum weight codewords, since they have to be found beforehand.

In the following the superpositions of the component codewords for some codes we study are provided in tables with N rows and $1+a$ columns. The first column contains the indexes $j = 0, 1, \ldots, N-1$ of the component codewords and the entry at position (i, j), $i = 1, \ldots, a$ contains the exponents of D in the polynomial $t_{j,i}(D)$. The null polynomial is referred to as $\{\}$.

6.2.1 Types of component codes

Let us consider component codes with $a = 3$ and $c = 2$; their symbolic matrix, which has column weight $d_v = 2$, may contain:

1) Two binomials, noted as b'_i and a pair of monomials, noted as m_i. In this case we have

$$\mathbf{H}_1(D) = \begin{bmatrix} b'_1(D) & m_1(D) & 0 \\ 0 & m_2(D) & b'_2(D) \end{bmatrix}.$$

Another valid parity-check matrix for the same code can be computed as

$$\mathbf{H}(D) = \begin{bmatrix} m_1^{-1}(D) & 0 \\ 0 & m_2^{-1}(D) \end{bmatrix} \mathbf{H}_1(D) = \begin{bmatrix} b_1(D) & 1 & 0 \\ 0 & 1 & b_2(D) \end{bmatrix} = \begin{bmatrix} D^a + D^b & 1 & 0 \\ 0 & 1 & D^c + D^d \end{bmatrix}.$$

By (2.6), we derive

$$\mathbf{U}(D) = \begin{bmatrix} b_2(D) & b_1(D)b_2(D) & b_1(D) \end{bmatrix}, \tag{6.9}$$

having, at most, weight 8; the associated weight vector, in fact, is

$$\mathcal{W}(\mathbf{U}(D)) = \left[\begin{array}{ccc} \mathcal{W}(b_2(D)) & \mathcal{W}(b_1(D)b_2(D)) & \mathcal{W}(b_1(D)) \end{array} \right] \leq \left[\begin{array}{ccc} 2 & 4 & 2 \end{array} \right].$$

The weight vector is smaller than or equal to $\left[\begin{array}{ccc} 2 & 4 & 2 \end{array} \right]$ because coincident terms, causing cancellations, may occur in the product between $b_1(D)$ and $b_2(D)$. The unavoidable codeword expressed by (6.9) can be associated to an unavoidable cycle of length 16, corresponding to the following equation of the type (6.3), whose terms have been calculated using (6.1) and (6.2)

$$2|b-a|+1+2|d-c|-1-2|b-a|+1-2|d-c|-1=0.$$

2) one binomial and two pairs of monomials. In this case we have

$$\mathbf{H}(D) = \left[\begin{array}{ccc} b_1(D) & m_1(D) & m_3(D) \\ 0 & m_2(D) & m_4(D) \end{array} \right] = \left[\begin{array}{ccc} D^a + D^b & D^c & D^e \\ 0 & D^d & D^f \end{array} \right].$$

It can be easily verified that this code has an unavoidable codeword of the type

$$\mathbf{U}(D) = \left[\begin{array}{ccc} m_1(D)m_4(D) + m_2(D)m_3(D) & b_1(D)m_4(D) & b_1(D)m_2(D) \end{array} \right]$$

having, at most, weight 6. This unavoidable codeword can be associated to an unavoidable cycle of length 12, corresponding to the following equation of the type (6.3)

$$2|b-a|+[2(d-c)+1]-[2(f-e)+1]-2|b-a|+[2(f-e)+1]-[2(d-c)+1]=0.$$

3) three pairs of monomials. In this case we have

$$\mathbf{H}(D) = \left[\begin{array}{ccc} m_1(D) & m_3(D) & m_5(D) \\ m_2(D) & m_4(D) & m_6(D) \end{array} \right] = \left[\begin{array}{ccc} D^a & D^c & D^e \\ D^b & D^d & D^f \end{array} \right].$$

and we have an unavoidable codeword of the type

$$\begin{aligned} \mathbf{U}(D) = \Big[& m_3(D)m_6(D) + m_4(D)m_5(D) \quad m_1(D)m_6(D) + m_2(D)m_5(D) \\ & m_1(D)m_4(D) + m_2(D)m_3(D) \Big] \end{aligned}$$

having, at most, weight 6. This unavoidable codeword can be associated to an unavoidable cycle of length 12, corresponding to the following equation of the

type (6.3)

$$[2(b-a)+1]-[2(d-c)+1]+[2(f-e)+1]-[2(b-a)+1]+[2(d-c)+1]-[2(f-e)+1] = 0.$$

If collisions in the polynomials forming the unavoidable codewords occur, then the upper bound on the free Hamming distance, given by (2.7), cannot be reached.

This analysis can be easily generalized to the case of larger values of c.

6.3 Analysis of some ensembles of codes

In this section we prove the effectiveness of our approach by focusing on some well known code ensembles. We first consider three ensembles of codes with low rate ($R = \frac{1}{c+1}$) which contain only one type of unavoidable codewords, noted as $\mathbf{U}(D)$ next.

Codes with higher rates ($R = \frac{a-c}{a}, a > c+1$) have more than one unavoidable codeword. Our approach can be applied without loss of generality also to these codes, by analyzing separately all their unavoidable codewords. We provide an example showing how this can be done.

Based on our analysis, a partially heuristic approach to methodically increase the code free Hamming distance can be defined as follows:

1) Given $\mathcal{W}(\mathbf{H}(D))$, choose the polynomial entries of $\mathbf{H}(D)$ in such a way that cancellations do not occur in the unavoidable codewords, that is, the weight of the codewords obtained using (2.6) coincides with the upper bound given by (2.7).

2) Search for low-weight codewords in the code defined by $\mathbf{H}(D)$ using, for example, the tools in [73]. If the code free Hamming distance is smaller than the upper bound, find the component codewords forming the codewords having weight below the upper bound.

3) Analyze the cycles with the same length of those associated to the component codewords found at step 2). If those cycles are avoidable, proceed to step 4), otherwise stop and report failure.

4) Modify the exponents of the polynomials in $\mathbf{H}(D)$ in such a way that all the cycles found in the previous step are removed and estimate the code free Hamming distance as in step 2).

5) Repeat step 3) until the code free Hamming distance achieves the bound given by (2.7), or the weight of all the component codewords is not smaller than that of the unavoidable ones.

According to step 3), this procedure can succeed only when the low-weight codewords are obtained as the superposition of component codewords associated to cycles with length shorter than that of the unavoidable ones.

We remark that the tools in [73] permit to *estimate* the free Hamming distance of a code, since finding its exact value is an NP-hard problem [74]. Still, our method relies on the optimization of the girth profile of the code. Thus, even if the estimate of the free distance is incorrect, the code obtained by means of our procedure has better cycle and distance spectra than the original one. Thus, the codes obtained through our method exhibit desirable properties even when the distance estimate is not accurate.

6.3.1 Low rate codes

Let us consider codes with parity-check matrix column weights equal to or greater than 2. As we will show next, the analysis of these codes can be used as a starting point to study more general cases.

In the following we describe their unavoidable codewords.

Monomial codes

We have already mentioned in Section 2.3.1 that a matrix $\mathbf{H}(D)$ of the following type, characterized by all monomial entries, describes an ensemble of codes called *monomial codes*,

$$\mathbf{H}(D) = \begin{bmatrix} m_{0,0}(D) & \cdots & m_{0,a-1}(D) \\ \vdots & \ldots & \vdots \\ m_{c-1,0}(D) & \cdots & m_{c-1,a-1}(D) \end{bmatrix}.$$

Monomial codes are characterized by an unavoidable codeword with weight vector [50]

$$\mathcal{W}(\mathbf{U}(D)) = \begin{bmatrix} c! & c! & \ldots & c! \end{bmatrix}.$$

In this case, (6.8) is satisfied with all cycles of length 12.

Let us consider monomial codes with $c = 3$; we can study, without loss of generality, the case in which the first row of $\mathbf{H}(D)$ is composed of ones. In fact, any monomial code parity-check matrix can be put in this form through simple linear transformations. This symbolic parity-check matrix and the corresponding unavoidable codeword are as follows

$$\mathbf{H}(D) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ D^a & D^b & D^c & D^d \\ D^e & D^f & D^g & D^h \end{bmatrix},$$

$$\mathbf{U}(D) = \Big[(D^{b+g} + D^{c+f} + D^{b+h} + D^{d+f} + D^{c+h} + D^{d+g}) \quad (D^{c+h} + D^{d+g} + \\ + D^{a+h} + D^{d+e} + D^{a+g} + xD^{c+e}) \quad (D^{b+h} + D^{d+f} + D^{a+h} + D^{d+e} + \\ + D^{a+f} + D^{b+e}) \quad (D^{b+g} + D^{c+f} + D^{a+g} + D^{c+e} + D^{a+f} + D^{b+e})\Big] . \tag{6.10}$$

If no cancellations occur, this unavoidable codeword has weight $\mathcal{W}(\mathbf{U}) = 24$. The $N = 12$ resulting component codewords with total weight 6, associated to cycles of length 12, are reported in Table 6.1.

Table 6.1: Component codewords forming the unavoidable codeword (6.10).

j	$u_{j,0}(D)$	$u_{j,1}(D)$	$u_{j,2}(D)$	$u_{j,3}(D)$
0	$b+h,\ c+h$	$a+h,\ c+h$	$a+h,\ b+h$	$\{\}$
1	$g+b,\ d+g$	$a+g,\ d+g$	$\{\}$	$a+g,\ b+g$
2	$d+f,\ c+f$	$\{\}$	$a+f,\ d+f$	$a+f,\ c+f$
3	$\{\}$	$d+e,\ c+e$	$b+e,\ d+e$	$b+e,\ c+e$
4	$d+f,\ g+f$	$d+e,\ d+g$	$d+e,\ d+f$	$\{\}$
5	$c+f,\ c+h$	$c+e,\ c+h$	$\{\}$	$c+e,\ c+f$
6	$b+g,\ b+h$	$\{\}$	$b+f,\ b+h$	$b+f,\ b+g$
7	$\{\}$	$a+g,\ a+h$	$a+f,\ a+h$	$a+f,\ a+g$
8	$c+f,\ b+g$	$c+e,\ a+g$	$b+e,\ a+f$	$\{\}$
9	$d+f,\ b+h$	$d+e,\ a+h$	$\{\}$	$b+e,\ a+f$
10	$c+h,\ d+g$	$\{\}$	$d+e,\ a+h$	$c+e,\ a+g$
11	$\{\}$	$c+h,\ d+g$	$b+h,\ d+f$	$c+f,\ b+g$

According to (6.8), by multiplying $\sum_{j=0}^{11} \mathbf{u}_j(D)$ by

$$\frac{1}{2}\mathbf{W}^{-1} = \frac{1}{2}\mathbf{diag}\left[\begin{array}{cccc} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}\right],$$

the above expression of $\mathbf{U}(D)$ is verified.

The case above can be easily extended to higher values of $d_v = c$.

A matrix $\mathbf{H}(D)$ characterized by a staircase diagonal of pairs of monomials, and $h_{1,1}(D) = p(D)$, $h_{c-1,a-1}(D) = z(D)$, being $p(D)$ and $z(D)$ polynomials of whichever weight, describes an ensemble of codes called in the following *two-polynomial codes*. Their symbolic matrix is as follows,

$$\mathbf{H}_1(D) = \begin{bmatrix} p'(D) & m_1(D) & & & & \\ & m_2(D) & m_3(D) & & & \\ & & m_4(D) & \ddots & & \\ & & & & m_{2a-5}(D) & \\ & & & & m_{2a-4}(D) & z'(D) \end{bmatrix}. \quad (6.11)$$

We can apply some linear transformations to (6.11), in order to obtain the simpler structure in (6.12). For the sake of brevity, we omit the passages and only report the final form of the symbolic matrix, which is as follows

$$\mathbf{H}(D) = \begin{bmatrix} p(D) & 1 & & & & \\ & 1 & 1 & & & \\ & & 1 & \ddots & & \\ & & & & 1 & \\ & & & & 1 & z(D) \end{bmatrix}. \quad (6.12)$$

Two-polynomial codes with $c = 2$ are characterized by the following parity-check matrix and unavoidable codeword

$$\mathbf{H}(D) = \begin{bmatrix} p(D) & 1 & \\ & 1 & z(D) \end{bmatrix},$$

$$\mathbf{U}(D) = \begin{bmatrix} z(D) & p(D)z(D) & p(D) \end{bmatrix}.$$

As shown in [75], in this case, (6.8) is satisfied by considering cycles of length 16, with the structure described in Section 6.2.1. Generalization is straightforward for whichever value of c.

Diagonal codes

A matrix $\mathbf{H}(D)$ formed by a diagonal block with c polynomials $p_j(D)$ along the main diagonal, followed by a column containing c monomials, shown in 6.13, describes an ensemble of codes called in the following *diagonal codes*,

$$\mathbf{H}_0(D) = \begin{bmatrix} p'_1(D) & & & & & m_1(D) \\ & p'_2(D) & & & & m_2(D) \\ & & p'_3(D) & & & m_3(D) \\ & & & \ddots & & \vdots \\ & & & & p'_c(D) & m_c(D) \end{bmatrix}. \quad (6.13)$$

Also in this case, through simple linear transformations we can obtain a

simpler form of (6.13), that is,

$$\mathbf{H}(D) = \begin{bmatrix} m_1^{-1}(D) & & & \\ & m_2^{-1}(D) & & \\ & & \ddots & \\ & & & m_c^{-1}(D) \end{bmatrix} \mathbf{H}_0(D) = \begin{bmatrix} p_1(D) & & & & 1 \\ & p_2(D) & & & 1 \\ & & \ddots & & \vdots \\ & & & p_c(D) & 1 \end{bmatrix}.$$

For $c = 2$, diagonal codes can be reduced to two-polynomial codes by performing a simple column permutation. For $c = 3$, instead, they exhibit the following symbolic parity-check matrix and unavoidable codeword

$$\mathbf{H}(D) = \begin{bmatrix} p_1(D) & & & 1 \\ & p_2(D) & & 1 \\ & & p_3(D) & 1 \end{bmatrix},$$

$$\mathbf{U}(D) = \begin{bmatrix} p_2(D)p_3(D) & p_1(D)p_3(D) & p_1(D)p_2(D) & p_1(D)p_2(D)p_3(D) \end{bmatrix}.$$

Also in this case, (6.8) is satisfied with cycles of length 16, with the structure described in Section 6.2.1.

Let us consider, for example, diagonal codes characterized by $\mathcal{W}(b_1(D)) = \mathcal{W}(b_2(D)) = \mathcal{W}(b_3(D)) = 2$; we have

$$\mathbf{H}(D) = \begin{bmatrix} b_1(D) & & & 1 \\ & b_2(D) & & 1 \\ & & b_3(D) & 1 \end{bmatrix} = \begin{bmatrix} 1+D^a & & & 1 \\ & 1+D^b & & 1 \\ & & 1+D^c & 1 \end{bmatrix},$$

$$\begin{aligned} \mathbf{U}(D) = \Big[&(1+D^b)(1+D^c) \quad (1+D^a)(1+D^c) \quad (1+D^b)(1+D^a) \\ &(1+D^a)(1+D^b)(1+D^c)\Big]. \end{aligned} \tag{6.14}$$

The superposition is obtained with $N = 6$ component codewords with weight 8, which are reported in Table 6.2 and correspond to cycles of length 16.

Summing the $N = 6$ terms and multiplying by $\mathbf{W}^{-1}$, we find that (6.8) holds with $s = 2$.

Table 6.2: Component codewords forming the unavoidable codeword (6.14).

j	$u_{j,0}(D)$	$u_{j,1}(D)$	$u_{j,2}(D)$	$u_{j,3}(D)$
0	0, b	0, a	{}	0, a, b, $a+b$
1	{}	0, c	0, b	0, b, c, $b+c$
2	0, c	{}	0, a	0, a, c, $a+c$
3	c, $b+c$	c, $a+c$	{}	c, $a+c$, $b+c$, $a+b+c$
4	{}	a, $a+c$	a, $a+b$	a, $a+b$, $a+c$, $a+b+c$
5	b, $b+c$	{}	b, $a+b$	b, $a+b$, $c+b$, $a+b+c$

The cases of $\mathcal{W}(p_i(D)) > 2$, $1 \leq i \leq 3$, are simple extensions of the above case and are not reported.

Generalization to larger values of c is possible by studying even and odd values of c separately.

Examples of code constructions

In the following examples we show that, when the free Hamming distance is smaller than the bound given by (2.7), the weight or the number of the component codewords decreases, as expected. In fact, low-weight codewords can be obtained as the combination of many short cycles, or few long cycles.

When the distance drop is due to cycles at least as long as the unavoidable ones, the girth cannot be improved by picking any other code of the ensemble; this is an intrinsic limitation of the heuristic approach proposed above. On the other hand, it is possible to choose the entries of $\mathbf{H}(D)$ such that the girth profile is improved; this way, the free Hamming distance properties of the code can still be improved as well.

Example 6.3.1 Consider a code with symbolic parity-check matrix

$$\mathbf{H}(D) = \begin{bmatrix} 1+D+D^3 & 1 & \\ & 1 & 1+D^2 \end{bmatrix}.$$

We obtain

$$\mathbf{U}(D) = \begin{bmatrix} (1+D^2) & (1+D+D^2+D^5) & (1+D+D^3) \end{bmatrix}, \tag{6.15}$$

whose support is the union of the supports of $U_i(D)$, $i = 0, 1, 2$, obtained by reversing (2.4), that is,

$$\{0, 6\} \cup \{1, 4, 7, 16\} \cup \{2, 5, 11\}.$$

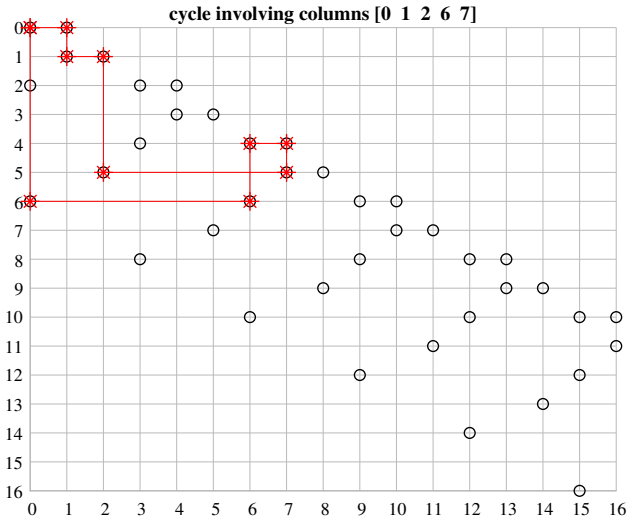


Figure 6.1: Cycle of length 10, associated to $\mathbf{u}_0(D)$, forming (6.15). It involves columns with indexes $\in \{0, 1, 2, 6, 7\}$.

The weight of this unavoidable codeword ($\mathcal{W}(\mathbf{U}) = 9$) is lower than the bound, given by (2.7), that is, 11. Therefore, the free Hamming distance of this code is $d_{\text{free}} \leq 9$. The low-weight codeword can be obtained as the combination of three component codewords associated to three cycles of length 10, 12 and 18, respectively, instead of three cycles of length 16. These short cycles, which are, in fact, the only cycles involving the ones in $\tilde{\mathbf{H}}$, easily lead to $s = 2$ and are shown in Fig.s 6.1,6.2 and 6.3, where the ones of the parity-check matrix $\mathbf{H}$ are represented as circles. The polynomials forming the corresponding codewords are shown in Table 6.3.

Table 6.3: Component codewords forming the unavoidable codeword (6.15).

j	$u_{j,0}(D)$	$u_{j,1}(D)$	$u_{j,2}(D)$
0	0, 2	0, 2	0
1	0, 2	1, 5	1, 3
2	0, 2	0, 1, 2, 5	0, 1, 3

The above example is significant, since it shows that the first condition that must be matched in order to aim for the maximum obtainable free Hamming distance is that the unavoidable codewords have full weight. A code meeting

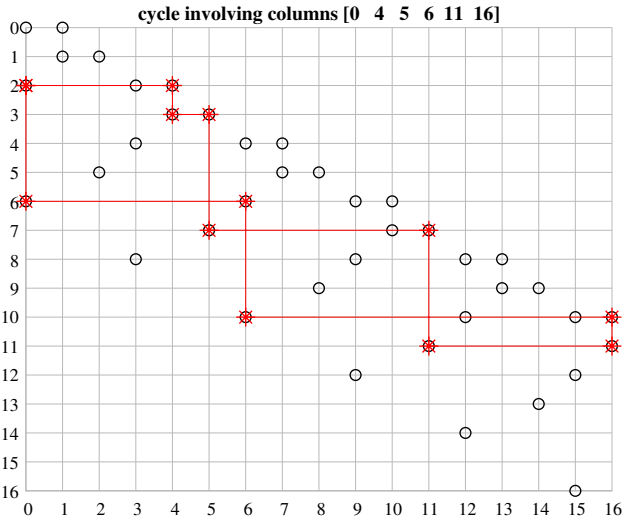


Figure 6.2: Cycle of length 12, associated to $\mathbf{u}_1(D)$, forming (6.15). It involves columns with indexes $\in \{0, 4, 5, 6, 11, 16\}$.

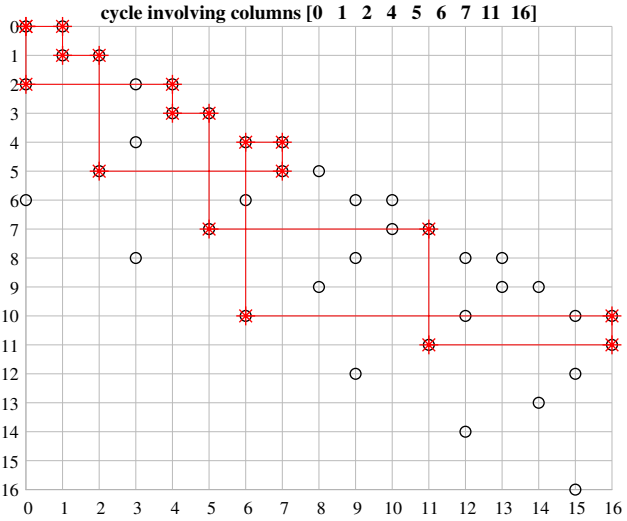


Figure 6.3: Cycle of length 18, associated to $\mathbf{u}_2(D)$, forming (6.15). It involves columns with indexes $\in \{0, 1, 2, 4, 5, 6, 7, 11, 16\}$.

this requirement is described by

$$\mathbf{H}(D) = \begin{bmatrix} 1 + D + D^3 & 1 \\ & 1 \quad 1 + D^4 \end{bmatrix}.$$

The entries of the symbolic parity-check matrix have been chosen so that the cycles associated to the component codewords in Table 6.3, which were responsible of the cancellations occurring in (6.15), do not exist. In this case, the unavoidable codeword is

$$\mathbf{U}(D) = \left[\begin{array}{ccc} (1+D^4) & (1+D+D^3+D^4+D^5+D^7) & (1+D+D^3) \end{array} \right],$$

which, as expected, has full weight $\mathcal{W}(\mathbf{U}) = 11$. It follows that the free Hamming distance of the code is $d_{\text{free}} \leq 11$.

Let us consider the widely investigated ensemble of monomial codes with $d_v = c = 3$. In the following examples we show how some particular short cycles cause the free Hamming distance to drop from 24 to 22 [25, 50].

Example 6.3.2 Let us consider the symbolic matrix,

$$\mathbf{H}(D) = \left[\begin{array}{cccc} 1 & D^{19} & D^{11} & D^{16} \\ D^7 & D^5 & 1 & D^{20} \\ D^{29} & 1 & D^8 & 1 \end{array} \right],$$

which has a cycle of length 6 in the last three columns, leading to the following codeword with weight 22

$$\begin{aligned} \mathbf{U}(D) = \Big[& (D^6+D^{16}+D^{31}+D^{47}) \quad (1+D^{18}+D^{21}+D^{28}+D^{35}+D^{60}) \\ & (D^5+D^{13}+D^{20}+D^{26}+D^{40}+D^{68}) \\ & (1+D^{13}+D^{18}+D^{34}+D^{45}+D^{48}) \Big]. \end{aligned}$$

The chosen superposition, with $s = 2$ and $N = 12$, involves three cycles of length 6, three cycles of length 10, three cycles of length 12 and three cycles of length 16. One cancellation in the first element of $\mathbf{U}(D)$ is responsible for the aforementioned weight reduction.

Similarly, the monomial code with symbolic parity-check matrix

$$\mathbf{H}(D) = \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^4 \\ 1 & D^3 & D^8 & D^{14} \end{array} \right] \tag{6.16}$$

contains the following codeword with weight 22

$$\mathbf{T}(D) = \Big[(D^4 + D^5 + D^{11} + D^{16}) \quad (D + D^2 + D^5 + D^6 + D^{15} + D^{16}) \\ (1 + D + D^3 + D^6 + D^9 + D^{11} + D^{15}) \quad (1 + D^2 + D^3 + D^4 + D^9)\Big],$$

which is due to five cycles of length 8, one cycle of length 12 and five cycles of length 16. In this case, $s = 2$ and $N = 11$.

The ensemble of monomial codes, where cycles of length smaller than 10 cause a drop of the free Hamming distance, are useful to prove the effectiveness of our method to improve the free Hamming distance of a code. Our analysis in fact shows that codewords with such low weights can be avoided by removing the corresponding cycles, thus increasing the free Hamming distance up to the weight of unavoidable codewords. To confirm this fact, we have designed a monomial code with girth 10 having the following symbolic parity-check matrix

$$\mathbf{H}(D) = \begin{bmatrix} D^{10} & D & D^2 & 1 \\ 1 & D^{10} & 1 & D^8 \\ D^5 & 1 & D^{10} & D \end{bmatrix}.$$

The free Hamming distance of this code, estimated through the tools in [73], indeed is $d_{\text{free}} = 24$.

In the following examples, where diagonal codes are considered, we show that, in some cases, low-weight codewords are not necessarily obtained as a combination of many short cycles, but can be due to few relatively long cycles.

Example 6.3.3 Let us consider the symbolic parity-check matrix

$$\mathbf{H}(D) = \begin{bmatrix} 1 + D & & & 1 \\ & 1 + D^2 & & 1 \\ & & 1 + D^4 & 1 \end{bmatrix},$$

which characterizes a code with girth 10. The unavoidable codeword has total weight $\mathcal{W}(\mathbf{U}) = 20$. Nevertheless, the following low-weight codeword can be found as a linear combination of the unavoidable codeword, namely $\mathbf{T}(D) = (1 + D)\mathbf{U}(D)$,

$$\mathbf{T}(D) = \Big[(1 + D + D^2 + D^3 + D^4 + D^5 + D^6 + D^7) \quad (1 + D^2 + D^4 + D^6) \\ (1 + D^4) \quad (1 + D^8)\Big]. \tag{6.17}$$

Equation (6.17) can be obtained as the combination of three component codewords associated to three cycles of length 16, 24 and 28, instead of six

cycles of length 16. The cycles, which are, as usual, the only cycles in $\tilde{\mathbf{H}}$, result in a combination with $s = 2$. The component codewords are shown in Table 6.4.

Table 6.4: Component codewords forming the codeword (6.17).

j	$t_{j,0}(D)$	$t_{j,1}(D)$	$t_{j,2}(D)$	$t_{j,3}(D)$
0	{}	0, 2, 4, 6	0, 4	0, 8
1	0, 1, 2, 3, 4, 5. 6, 7	{}	0, 4	0, 8
2	0, 1, 2, 3, 4, 5. 6, 7	0, 2, 4, 6	{}	0, 8

The above example is important, since it shows that a low-weight codeword can be obtained combining relatively long cycles. In this case it is not possible to increase the code free Hamming distance by avoiding all the cycles of length 16, which are unavoidable; improvements can still be obtained by punctually removing the few harmful long cycles. In order to confirm the validity of this method, we have considered

$$\mathbf{H}(D) = \begin{bmatrix} 1+D & & & 1 \\ & 1+D^5 & & 1 \\ & & 1+D^8 & 1 \end{bmatrix},$$

which belongs to the same ensemble of the code of Example 6.3.3. The entries of $\mathbf{H}(D)$ have been chosen in such a way that the code has girth 16, and the cycles associated to the component codewords in Table 6.4 are removed. In this case the free Hamming distance, estimated through the tools in [73], is $d_{\text{free}} = 18$.

6.3.2 Medium and high rate codes

In Section 6.3.1 we have validated the approach we propose by considering codes with rate equal to $\frac{1}{c+1}$. Nevertheless, our approach is general and holds also for codes having whichever rate of the type $R = \frac{a-c}{a}$, where $a > c + 1$. It follows from (2.6) that any code contains $\binom{a}{c+1} > 1$ unavoidable codewords. Each of the unavoidable codewords can be seen as the superposition of the unavoidable codewords of a number of component codes, as shown in Section 6.3.1, therefore the approach we propose can be iteratively applied to all the unavoidable codewords of a code. In the next example we show how this can be done for a code with rate $\frac{2}{c+2}$, but the extension to higher code rates is straightforward.

Example 6.3.4 Let us consider the monomial code with symbolic parity-check matrix as in (6.16) and let us append one column to $\mathbf{H}(D)$, in such a way that

its girth $g = 10$ is maintained. We obtain a code with $R = \frac{2}{a} = \frac{2}{5}$, described by

$$\mathbf{H}(D) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & D & D^2 & D^4 & D^9 \\ 1 & D^3 & D^8 & D^{14} & D^{18} \end{bmatrix},$$

which has 5 unavoidable codewords, associated to all the sub-matrices of $\mathbf{H}(D)$ with size 3×4. All the unavoidable codewords, not reported here for the sake of brevity, have maximal weight and, hence, $d_{\text{free}} \leq 24$. The component codewords corresponding to these unavoidable codewords can be obtained as shown in the previous examples.

Similarly, more columns can be appended to the symbolic parity-check matrices of codes with rate $R = \frac{1}{a}$, until an arbitrary rate, up to $\frac{a-1}{a}$, is achieved, and the corresponding unavoidable codewords can be analyzed through the same approach.

6.4 Performance assessment

In this section we validate the effectiveness of our method through numerical simulations. In particular, we consider two codes obtained with a classical design approach and apply the procedure described in Section 6.3 to improve its free Hamming distance. Then we assess the BER performance of these codes through Monte Carlo simulations of BPSK modulated transmissions over the AWGN channel. A BP-based decoder performing 100 iterations is used.

The classical code $\mathcal{C}_{p1}$ is a Type-I packing-based SC-LDPC-CC designed following the approach in [58]. It has asymptotic rate $R = \frac{3}{4}$, $m_s = 1$, $v_s = 272$ and $g = 6$, as in [58, Example 5]. The parity-check matrix of this code has the form (2.1), with $\mathbf{H}_0^{(\mathcal{C}_{p1})}$ and $\mathbf{H}_1^{(\mathcal{C}_{p1})}$ as in (6.18) and (6.19), where I^x is a 17×17 circulant permutation matrix obtained by cyclically shifting the rows of an identity matrix by x positions. We have estimated the free Hamming distance of $\mathcal{C}_{p1}$, obtaining $d_{\text{free}} = 8$.

$$\mathbf{H}_0^{(\mathcal{C}_{p1})} = \begin{bmatrix} I^1 & I^9 & I^{13} & I^{15} & I^{16} & I^8 & I^4 & I^2 \\ I^{16} & I^8 & I^4 & I^2 & I^1 & I^9 & I^{13} & I^{15} \end{bmatrix} \tag{6.18}$$

$$\mathbf{H}_1^{(\mathcal{C}_{p1})} = \begin{bmatrix} I^{10} & I^5 & I^{11} & I^{14} & I^7 & I^{12} & I^6 & I^3 \\ I^7 & I^{12} & I^6 & I^3 & I^{10} & I^5 & I^{11} & I^{14} \end{bmatrix} \tag{6.19}$$

In order to improve it, we have modified the exponents in (6.18) and (6.19) following the procedure described in Section 6.3. This way we obtained a code, denoted as $\mathcal{C}_1$, having the same asymptotic rate, syndrome former memory order, syndrome former constraint length and girth as $\mathcal{C}_{p1}$, but with a number of

length-6 cycles in the first two syndrome former matrices reduced by more than 10% . The block $\mathbf{H}_1^{(\mathcal{C}_1)}$ of the syndrome former matrix of the new code is shown in (6.20), whereas $\mathbf{H}_0^{(\mathcal{C}_1)} = \mathbf{H}_0^{(\mathcal{C}_{p1})}$. The estimated free Hamming distance of the new code is $d_{\text{free}} = 12$, which represents a significant improvement with respect to $\mathcal{C}_{p1}$.

$$\mathbf{H}_1^{(\mathcal{C}_1)} = \begin{bmatrix} I^{10} & I^{0} & I^{0} & I^{0} & I^{4} & I^{9} & I^{10} & I^{5} \\ I^{7} & I^{0} & I^{3} & I^{7} & I^{14} & I^{4} & I^{16} & I^{16} \end{bmatrix} \tag{6.20}$$

The same method has been applied to a Type-I packing-based SC-LDPC-CC, denoted as $\mathcal{C}_{p2}$, having asymptotic rate $R = \frac{2}{3}$, $m_s = 1$, $v_s = 156$ and $g = 6$. The estimate of the free Hamming distance of this code is $d_{\text{free}} = 8$. The two blocks forming the syndrome former matrix are shown in (6.21) and (6.22), where I^x represents a circulant permutation matrix with size 13×13.

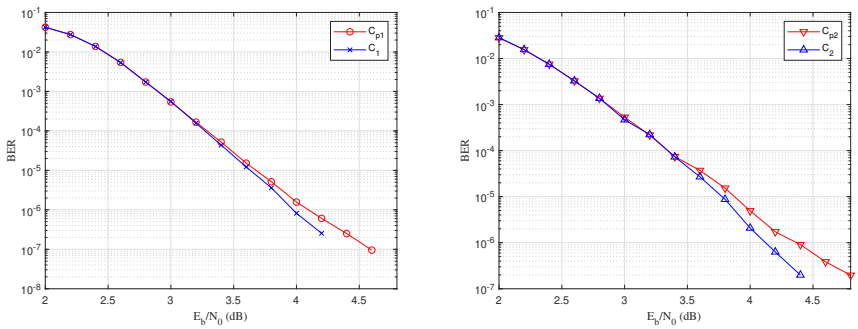
$$\mathbf{H}_0^{(\mathcal{C}_{p2})} = \begin{bmatrix} I^{1} & I^{4} & I^{3} & I^{12} & I^{9} & I^{10} \\ I^{12} & I^{9} & I^{10} & I^{1} & I^{4} & I^{3} \end{bmatrix} \tag{6.21}$$

$$\mathbf{H}_1^{(\mathcal{C}_{p2})} = \begin{bmatrix} I^{8} & I^{6} & I^{11} & I^{5} & I^{7} & I^{2} \\ I^{5} & I^{7} & I^{2} & I^{8} & I^{6} & I^{11} \end{bmatrix} \tag{6.22}$$

As for $\mathcal{C}_{p1}$, we have modified the exponents in (6.21) and (6.22) according to the procedure in Section 6.3. Such an approach has led to a code, noted as $\mathcal{C}_2$, which has the same parameters as $\mathcal{C}_{p2}$, but benefits from a reduction of the number of cycles of length 6 by more than 20% with respect to the original code. Also in this case, $\mathbf{H}_0^{(\mathcal{C}_2)} = \mathbf{H}_0^{(\mathcal{C}_{p2})}$, whereas $\mathbf{H}_0^{(\mathcal{C}_2)}$ is shown in (6.23). The estimated value of the free distance of $\mathcal{C}_{p2}$ is $d_{\text{free}} = 12$, thus resulting again in a significant improvement over the original code.

$$\mathbf{H}_1^{(\mathcal{C}_2)} = \begin{bmatrix} I^{8} & I^{0} & I^{2} & I^{4} & I^{10} & I^{3} \\ I^{5} & I^{9} & I^{6} & I^{5} & I^{9} & I^{6} \end{bmatrix} \tag{6.23}$$

The simulated performance of these codes is shown in Figure 6.4. We note that the codes obtained by means of our procedure exhibit a performance gain over the original ones in the region of high signal-to-noise ratios, due to their higher free Hamming distance. This confirms the effectiveness of our method also from a performance standpoint, especially in the error-floor region, where short cycles and small free Hamming distances usually have a frustrating impact.



(a) (b)

Figure 6.4: BER performance of classical and optimized SC-LDPC-CCs with: (a) $v_s = 272$ and (b) $v_s = 156$ as a function of the SNR per bit, E_b/N_0.

6.5 Summary

In this chapter we have related cycles and codewords of SC-LDPC-CCs. We have noticed that the cycles with the shortest length are not always responsible for the existence of low-weight codewords. Low-weight codewords may indeed arise because of few long cycles, rather than many short ones. The removal of the most dangerous cycles yielded better minimum distance properties and improved error rate performance, with respect to previous solutions.

Chapter 7

Conclusions

In this thesis we have proposed innovative design methods of SC-LDPC-CCs. Particular attention has been devoted to compact codes, in that they are suitable to modern scenarios, characterized by stringent demands on latency and complexity.

After the introduction of some preliminary notions in the initial chapters, in the central chapters we deal with the design of SC-LDPC-CCs with short constraint length, for which we have derived several bounds. We have proposed construction methods for the most common ensembles of SC-LDPC-CCs, which allow significant savings in terms of decoding complexity with respect to previous solutions. Moreover, we have considered a complexity-constrained scenario, where the best trade-off between the maximum number of decoding iterations and the window size is usually unknown. We have proposed a heuristic method based on PEXIT analysis to solve this problem.

In the final chapter of the thesis the relationship between minimum distance and harmful objects in the code Tanner graph is studied. We prove that codewords of the main code can be seen as the superposition of codewords of component codes which, in their turn, are associated to cycles. The removal of these cycles yields benefits in terms of both error rate performance and minimum distance properties.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, pp. 379–423 (Part 1), 623–656 (Part 2), 1948.

[2] H. Gamage, N. Rajatheva, and M. Latva-aho, "Channel coding for enhanced mobile broadband communication in 5G systems", in *Proc. European Conference on Networks and Communications*, June 2017, pp. 1–6.

[3] T. J. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio", *IEEE Communications Magazine*, vol. 56, no. 3, pp. 28–34, March 2018.

[4] R. G. Gallager, *Low-Density Parity-Ceck Codes*, PhD thesis, Dept. Elect. Eng., MIT, Cambridge, MA, USA, Jul. 1963.

[5] A. Jiménez Felström and K. Sh. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix", *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, September 1999.

[6] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes", *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5274–5289, October 2010.

[7] S. Kudekar, T. J. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC", in *Proc. IEEE International Symposium on Information Theory*, Austin, TX, USA, June 2010, pp. 684–688.

[8] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC", *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 803–834, February 2011.

[9] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation", in *Proc. IEEE International Symposium on Information Theory*, Istanbul, Turkey, December 2013, vol. 59, pp. 7761–7813.

[10] G. Durisi, T. Koch, and P. Popovski, "Toward massive, ultrareliable, and low-latency wireless communication with short packets", *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1711–1726, September 2016.

[11] D. J. C. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes", *Electronic Notes in Theoretical Computer Science*, vol. 74, pp. 97 – 104, 2003, MFCSIT 2002, The Second Irish Conference on the Mathematical Foundations of Computer Science and Information Technology.

[12] Changyan Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel", *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1570–1579, June 2002.

[13] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding", *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[14] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes", *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 181–201, January 2010.

[15] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity for Gallager low-density codes", *Problems of Information Transmission*, vol. 11, pp. 23–26, 1975.

[16] R. Tanner, "A recursive approach to low complexity codes", *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, September 1981.

[17] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1", in *Proc. ICC '93 - IEEE International Conference on Communications*, Geneva, Switzerland, May 1993, number 2, pp. 1064–1070.

[18] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes", *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1723–1731, November 1996.

[19] M. Sipser and D. A. Spielman, "Expander codes", *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, November 1996.

[20] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes", *Electronics Letters*, vol. 33, no. 6, pp. 457–458, March 1997.

[21] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, March 1999.

[22] N. Wiberg, H.-A. Loeliger, and R. Kotter, "Codes and iterative decoding on general graphs", in *Proc. IEEE International Symposium on Information Theory*, Whistler, Canada, September 1995, pp. 468–.

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

[24] H. Zhou and N. Goertz, "Cycle analysis of time-variant LDPC convolutional codes", in *Proc. 6th Int. Symp. Turbo Codes Iterative Information Processing*, Hong Kong, September 2010, pp. 48–52.

[25] R. Smarandache and P. O. Vontobel, "Quasi-cyclic LDPC codes: Influence of proto- and tanner-graph structure on minimum hamming distance upper bounds", *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 585–607, Feb. 2012.

[26] R. M. Tanner, *Convolutional Codes from Quasi-cyclic Codes: A Link Between the Theories of Block and Convolutional Codes*, University of California, Santa Cruz, Computer Research Laboratory, 1987.

[27] A. E. Pusane, R. Smarandache, P. O. Vontobel, and D. J. Costello, "Deriving good LDPC convolutional codes from LDPC block codes", *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 835–857, February 2011.

[28] M. B. S. Tavares, K. S. Zigangirov, and G. P. Fettweis, "Tail-biting LDPC convolutional codes based on protographs", in *Proc. IEEE 66th Vehicular Technology Conf*, Baltimore, MD, USA, September 2007, pp. 1047–1051.

[29] M. B. S. Tavares, K. S. Zigangirov, and G. P. Fettweis, "Tail-biting LDPC convolutional codes", in *Proc. IEEE Int. Symp. Information Theory*, Nice, France, June 2007, pp. 2341–2345.

[30] R. Townsend and E. Weldon, "Self-orthogonal quasi-cyclic codes", *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 183–195, April 1967.

[31] M. Karlin, "New binary coding results by circulants", *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 81–92, January 1969.

[32] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices", *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 2966–2984, December 2004.

[33] J. Thorpe, "Low-density parity-check codes constructed from protographs", *IPN Progress Report 42-154*, Aug. 2003.

[34] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "Spatially coupled LDPC codes constructed from protographs", *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 4866–4889, September 2015.

[35] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's "belief propagation" algorithm", *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, February 1998.

[36] J. Pearl, *Reverend Bayes on inference engines: A distributed hierarchical approach*, Cognitive Systems Laboratory, School of Engineering and Applied Science, University of California, Los Angeles, 1982.

[37] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[38] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes", *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

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

[40] G. Liva and M. Chiani, "Protograph LDPC codes design based on exit analysis", in *Proc. IEEE Global Telecommunications Conference (GLOBECOM) 2007*, Washington, DC, USA, November 2007, pp. 3250–3254.

[41] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli, and G. E. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels", *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2303–2320, April 2012.

[42] X. Y. Hu, E. Eleftheriou, D. M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes", in *Proc. IEEE Global Telecommunications Conference (GLOBECOM) 2001*, San Antonio, TX, USA, Nov. 2001, vol. 2, pp. 1036–1036E.

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

[44] M. Lentmaier, M. M. Prenda, and G. P. Fettweis, "Efficient message passing scheduling for terminated LDPC convolutional codes", in *Proc. IEEE International Symposium on Information Theory*, Saint Petersburg, Russia, July 2011, pp. 1826–1830.

[45] S. Lin and D. J. Costello, *Error Control Coding*, Prentice-Hall, Inc., Upper Saddle River, NJ, second edition, 2004.

[46] M. Baldi, M. Bianchi, G. Cancellieri, and F. Chiaraluce, "Progressive differences convolutional low-density parity-check codes", *IEEE Communications Letters*, vol. 16, no. 11, pp. 1848–1851, November 2012.

[47] J. Cho and L. Schmalen, "Construction of protographs for large-girth structured LDPC convolutional codes", in *Proc. ICC '15 - IEEE International Conference on Communications*, London, UK, June 2015, pp. 4412–4417.

[48] M. Baldi, M. Battaglioni, F. Chiaraluce, and G. Cancellieri, "Time-invariant spatially coupled low-density parity-check codes with small constraint length", in *Proc. IEEE Int. Black Sea Conference on Communications and Networking (BlackSeaCom)*, Varna, Bulgaria, June 2016.

[49] A. Tasdighi, A. H. Banihashemi, and M.-R. Sadeghi, "Efficient search of girth-optimal QC-LDPC codes", *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1552–1564, April 2016.

[50] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices", *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1788–1793, August 2004.

[51] I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov, and R. V. Satyukov, "Searching for voltage graph-based LDPC tailbiting codes with large girth", *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2265–2279, April 2012.

[52] M. Battaglioni, A. Tasdighi, G. Cancellieri, F. Chiaraluce, and M. Baldi, "Design and analysis of time-invariant SC-LDPC convolutional codes with

small constraint length", *IEEE Transactions on Communications*, vol. 66, no. 3, pp. 918–931, March 2018.

[53] H. Park, S. Hong, J. No, and D. Shin, "Construction of high-rate regular quasi-cyclic LDPC codes based on cyclic difference families", *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3108–3113, August 2013.

[54] A. Tasdighi and M.-R. Sadeghi, "On advisability of designing short length QC-LDPC codes using perfect difference families", *arXiv preprint arXiv:1701.06218*, 2017.

[55] L. A. Wolsey, *Integer Programming*, John Wiley and Sons, 1998.

[56] A. Tasdighi, A. H. Banihashemi, and M. Sadeghi, "Symmetrical constructions for regular girth-8 QC-LDPC codes", *IEEE Transactions on Communications*, vol. 65, no. 1, pp. 14–22, January 2017.

[57] J. L. Fan, "Array codes as LDPC codes", in *Constrained Coding and Soft Iterative Decoding*, pp. 195–203. Springer, 2001.

[58] M. Zhang, Z. Wang, Q. Huang, and S. Wang, "Time-invariant quasi-cyclic spatially coupled LDPC codes based on packings", *IEEE Transactions on Communications*, vol. 64, no. 12, pp. 4936–4945, 2016.

[59] K. Liu, M. El-Khamy, and J. Lee, "Finite-length algebraic spatially-coupled quasi-cyclic LDPC codes", *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 329–344, February 2016.

[60] M. Gholami and Z. Gholami, "An explicit method to generate some QC LDPC codes with girth 8", *Iranian Journal of Science and Technology, Transactions A: Science*, vol. 40, no. 2, pp. 145, June 2016.

[61] M. E. O'Sullivan, "Algebraic construction of sparse matrices with large girth", *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 718–727, February 2006.

[62] F. Amirzade and M. Sadeghi, "Lower bounds on the lifting degree of QC-LDPC codes by difference matrices", *IEEE Access*, vol. 6, pp. 23688–23700, 2018.

[63] V. Jamali, Y. Karimian, J. Huber, and M. Ahmadian, "An efficient complexity-optimizing LDPC code design for the binary erasure channel", in *Proc. 8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Bremen, Germany, August 2014, pp. 238–242.

[64] I. P. Mulholland, E. Paolini, and M. F. Flanagan, "Design of LDPC code ensembles with fast convergence properties", in *Proc. IEEE Int. Black Sea Conference on Communications and Networking (BlackSeaCom)*, Constanta, Romania, May 2015, pp. 53–57.

[65] I. P. Mulholland, E. Paolini, and M. F. Flanagan, "Design of protograph-based LDPC code ensembles with fast convergence properties", in *Proc. Communications and Coding SCC 2017; 11th Int. ITG Conf. Systems*, Hamburg, Germany, February 2017, pp. 1–6.

[66] T. Koike-Akino, D. S. Millar, K. Kojima, K. Parsons, Y. Miyata, K. Sugihara, and W. Matsumoto, "Iteration-aware LDPC code design for low-power optical communications", *Journal of Lightwave Technology*, vol. 34, no. 2, pp. 573–581, January 2016.

[67] T. J. Richardson, "Error floors of LDPC codes", in *Proc. Forty-first Annual Allerton Conference*, Monticello, IL, September 2003.

[68] Q. Huang, Q. Diao, S. Lin, and K. Abdel-Ghaffar, "Trapping sets of structured LDPC codes", in *Proc. IEEE ISIT 2011*, Saint Petersburg, Russia, August 2011.

[69] D. G. M. Mitchell, A. E. Pusane, and D. J. Costello, "Minimum distance and trapping set analysis of protograph-based LDPC convolutional codes", *IEEE Transactions on Information Theory*, vol. 59, no. 1, pp. 254–281, January 2013.

[70] S. M. Khatami, L. Danjean, D. V. Nguyen, and B. Vasić, "An efficient exhaustive low-weight codeword search for structured LDPC codes", in *Proc. Information Theory and Applications Workshop (ITA)*, San Diego, California, USA, February 2013, pp. 1–10.

[71] H. Zhou and N. Goertz, "Unavoidable cycles in polynomial-based time-invariant LDPC convolutional codes", in *Proc. 17th European Wireless 2011 - Sustainable Wireless Technologies*, Vienna, Austria, April 2011, pp. 1–6.

[72] M. Battaglioni, M. Baldi, and G. Cancellieri, "Connections between low-weight codewords and cycles in spatially coupled LDPC convolutional codes", *IEEE Transactions on Communications*, vol. 66, no. 8, pp. 3268–3280, August 2018.

[73] D. J. C. MacKay, "Source code for approximating the mindist problem of LDPC codes", 2008, http://www.inference.eng.cam.ac.uk/mackay/MINDIST_ECC.html.

[74] X.-Y. Hu, M. P. C. Fossorier, and E. Eleftheriou, "On the computation of the minimum distance of low-density parity-check codes", in *IEEE International Conference on Communications*, Paris, France, Jun. 2004, vol. 2, pp. 767–771.

[75] M. Baldi and G. Cancellieri, "Low-rate LDPC convolutional codes with short constraint length", in *Proc. International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2015)*, Split-Bol, Croatia, September 2015.