LOW COMPLEXITY ITERATIVE ALGORITHMS IN CHANNEL CODING AND COMPRESSED SENSING

by

Ludovic Danjean

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 1 3

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Ludovic Danjean,
entitled Low-Complexity Iterative Algorithms in Channel Coding and Compressed Sensing and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

_____ Date: November 7, 2013

Bane Vasić

_____ Date: November 7, 2013

Michael W. Marcellin

_____ Date: November 7, 2013

Ali Bilgin

_____ Date: November 7, 2013

David Declercq

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

_____ Date: November 7, 2013

Dissertation Director: Bane Vasić

Statement by Author

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Signed: _____

Ludovic Danjean

ACKNOWLEDGEMENTS

When you start to write the acknowledgements section of your doctoral dissertation, I guess it means that you are very close to conclude a big chapter in your life and, for sure, to start another big one. For me at least, it is the biggest chapter I have ever closed so far. When I started this PhD program couple of years ago, a high school friend who were already working for 2 years asked me naively: so when are you done with school? Suddenly I realized that I was one of the last, among my friends, still studying and wondering if it was worth it. Four years later, I can now say it, it was worth it! Not only I have learned so many advanced topics in science, but above all, I grew up while meeting people, while traveling from one side of the Atlantic ocean to another. When I start to look back, sure there were few sacrifices to do, but this journey is ending, and I am grateful to so many people that I will try not to forget any.

I would like to have a first thought to all my French teachers, especially my science teachers in "classes préparatoires", Mrs. Devouge, Mrs. Dekker and Mr. Reboud and Mr. Devie, who encouraged me throughout the years. I would like to also show my deep acknowledgments to the pedagogical team of the French graduate school, ENSEA, for providing me such a great education which enabled me to continue my education toward the PhD degree, first in France and then here in Arizona.

I started this PhD back in France at ETIS, and I must say that I enjoyed the two years spent working there in the ICI team. I would like to thank Professor Inbar Fijalkow and Annick Bertinotti for all the support, help and advice they gave me. I have some special thoughts to all the PhD students I share time with, especially Erbao, Alexis, Mathilde, Leila, Romain, Laurent, Alan, Rmi, Jean-Emmanuel and of course Jean-Christophe, Gaël and Germain.

After two years at ETIS, I continued this dual degree program in Arizona, where I

met new people from other cultures. I would like to acknowledge my labmates, Dung, Shiva, Vida, Mehrdad and Mohammed with whom I had fruitful discussions. I wish them all the best in their future, and I hope that our routes will cross again. While landing in Tucson, I met a fantastic couple to whom I owe a great debt of gratitude, Dr. Penny and Dennis. They have been so kind, helpful, generous and they have facilitated so much our adaptation to the life in the desert.

I would like to address my sincere acknowledgements to the Electrical Computer Engineering (ECE) department staff at the University of Arizona notably Tami, Carroll, Nancy, Christine, Kurt who always do their best to help and provide efficient answers to some unusual questions I may have asked during this two years. I am also extremely grateful to Stella, the so kind custodian who is taking such a good care of the lab.

I also thank the ECE department faculty, notably Dr. Marcellin, Dr. Bilgin, Dr. Lazos, Dr. Djordjevic who agreed on being part of my comprehensive exams committees as well as my dissertation committee. I would like to show my deep appreciation to my sponsor, DARPA under the Knowledge Enhanced Compressive Measurements (KECoM) program through contract #N6601-10-4079.

To conclude with my university-related acknowledgements, I am extremely grateful to my French advisor, Professor David Declercq, who was at first my professor at ENSEA and then convinced me to start a PhD program with him as an advisor. I am so glad he trusted me to start this journey, and I thank him a lot for helping me leading my research. Luckily, he also gave me this big opportunity to work with Professor Bane Vasic, and eventually initiated this dual degree program. I then would like to show my deep appreciation to Dr. Vasic for welcoming in his research family and also for agreeing to be my co-advisor. I will remember for a long time the discussions we had as well as the help he provided me regarding paperworks as well as in my daily life in Tucson.

Finally, I would like to thank the most important persons for me who helped,

encouraged and supported me in every single steps of my education, my mom and my dad, without whom I could not have done everything I did. It is so important to feel that you parents completely understand and support your choices, even if these ones include living very far from home. I want to thank my lovely wife, Juliette, who has been such a great support during all these years especially since our departure from France. I am so grateful she chose to accompany me in this adventure in Arizona. I do not forget my sister, my brother-in-law, my four lovely nieces and my nephew and all my family and family-in-law back in France and Scotland who always encouraged me to pursue this career even 9,000 km far from home (yes I still cannot use miles!).

I wish to anyone to be as lucky as I have been so far. Some will say that you need to provoke or deserve luck, I just like to remember a sentence of my favorite bear advising to look for *the bare necessities*!

A vous tous, un grand merci !

DEDICATION

À Juliette,

Nicole, Hervé, Anne-Lucie, Stephan,

Jeanne, Valentine, Emma, Louison, Rose,

Madeleine, Paulette,

et toute ma famille et belle-famille.

TABLE OF CONTENTS

TABLE OF CONTENTS—*Continued*

LIST OF FIGURES

LIST OF FIGURES—*Continued*

LIST OF TABLES

ABSTRACT

Iterative algorithms are now widely used in all areas of signal processing and digital communications. In modern communication systems, iterative algorithms are notably used for decoding low-density parity-check (LDPC) codes, a popular class of error-correction codes known to have exceptional error-rate performance under iterative decoding. In a more recent field known as compressed sensing, iterative algorithms are used as a method of reconstruction to recover a sparse signal from a linear set of measurements. This work primarily deals with the development of low-complexity iterative algorithms for the two aforementioned fields, namely, the design of low-complexity decoding algorithms for LDPC codes, and the development and analysis of a low complexity reconstruction algorithm for compressed sensing.

In the first part of this dissertation, we focus on the decoding algorithms for LDPC codes. It is now well known that LDPC codes suffer from an error floor phenomenon in spite of their exceptional performance. This phenomenon originates from the failures of traditional iterative decoders, like belief propagation (BP), on certain low-noise configurations. Recently, a novel class of decoders, called *finite alphabet iterative decoders* (FAIDs), were proposed with the capability of surpassing BP in the error floor region at a much lower complexity. We show that numerous FAIDs can be designed, and among them only a few will have the ability of surpassing traditional decoders in the error floor region. In this work, we focus on the problem of the selection of good FAIDs for column-weight-three codes over the binary symmetric channel. Traditional methods for decoder selection use asymptotic techniques such as the density evolution method, but the designed decoders do not guarantee good performance for finite-length codes especially in the error floor region. Instead we propose a methodology to identify FAIDs with good error-rate performance in the error floor. This methodology relies on the knowledge of potentially harmful topolo-

gies that could be present in a code. The selection method uses the concept of *noisy trapping set*. Numerical results are provided to show that FAIDs selected based on our methodology outperform BP in the error floor on a wide range of codes. Moreover first results on column-weight-four codes demonstrate the potential of such decoders on codes which are more used in practice, for example in storage systems.

In the second part of this dissertation, we address the area of iterative reconstruction algorithms for compressed sensing. This field has attracted a lot of attention since Donoho's seminal work due to the promise of sampling a sparse signal with less samples than the Nyquist theorem would suggest. Iterative algorithms have been proposed for compressed sensing in order to tackle the complexity of the optimal reconstruction methods which notably use linear programming. In this work, we modify and analyze a low complexity reconstruction algorithm that we refer to as the *interval-passing algorithm* (IPA) which uses sparse matrices as measurement matrices. Similar to what has been done for decoding algorithms in the area of coding theory, we analyze the failures of the IPA and link them to the stopping sets of the binary representation of the sparse measurement matrices used. The performance of the IPA makes it a good trade-off between the complex ℓ_1-minimization reconstruction and the very simple verification decoding. The measurement process has also a lower complexity as we use sparse measurement matrices. Comparison with another type of message-passing algorithm, called approximate message-passing, show the IPA can have superior performance with lower complexity. We also demonstrate that the IPA can have practical applications especially in spectroscopy.

GENERAL INTRODUCTION

Context and Motivation

Iterative algorithms are nowadays widely used in all areas of signal processing because either they perform at a low complexity, or they provide fast processing on large scale problems, without any significant loss in performance. In most cases iterative methods are even able to provide excellent performance, which explains why iterative algorithms can be found from the first algorithms solving linear system of equations to the most advanced algorithms in signal processing systems that primarily deal with the recovery of a signal that was transmitted, stored, or corrupted somehow.

The aim of this dissertation is to design and analyze low-complexity iterative algorithms for two domains of the signal processing, namely the channel coding, and the more recent field of compressed sensing. In the first part of the dissertation we seek to design low-complexity finite precision iterative decoders. In the second step we address the design and analysis of low-complexity iterative reconstruction algorithms in compressed sensing.

Design of low-complexity finite precision iterative decoders

Since their rediscovery, low-density parity-check (LDPC) codes [1] have been used intensively in channel coding, including in some standards such as the second generation of Digital Video Broadcasting by Satellite, and in the standards of wireless networking like Wifi and Wimax. In spite of their excellent performance under iterative decoding, LDPC codes exhibit an abrupt degradation in the slope of the error rate in the high signal-to-noise ratio (SNR) regime known as the error floor region [44]. The error floor problem has attracted significant interest with emphasis on designing codes and low-complexity decoders to lower error floors in fields which require very low error rates like in storage systems.

Recently we have proposed a new class of low-complexity finite precision iterative decoders, called *finite alphabet iterative decoders* (FAIDs) for LDPC codes over the binary symmetric channel (BSC), where the messages can take a finite number of levels [63]. These decoders exhibit better performance in the error floor region compared to traditional iterative decoders using a very low complexity as only a small number of bits (as small as 3) are required for the message representation. The update functions used at the variable nodes can be easily described by maps which are chosen with the goal of increasing the error correction capability of the code.

Although it has been proven that FAIDs have excellent performance in the error floor, the problem of identifying particularly good FAIDs for a given code is not trivial. This is due to the very large number of FAIDs that can be designed for a particular number of precision bits. The main goal of this dissertation in this topic is to address the problem of the selection of good FAIDs for column-weight-three LDPC codes on the BSC. The first proposed approach relies on the structure of a code we want to select a FAID for, and the second approach aims at selecting a set of potential good FAIDs without being code specific.

Finally, we present the first result of the extension of the design of FAIDs on column-weight-four codes, as these family of codes are present in many practical applications as in storage systems where error floor is a critical issue, and where FAIDs can have their industrial interest.

Design of low-complexity iterative reconstruction algorithms

In compressed sensing, reconstructing a sparse signal from a small set of measurements via compressed sensing [79] has attracted significant attention in the last few years. A k-sparse signal $\mathbf{x} \in \mathbb{R}^n$, i.e. a signal $\mathbf{x}$ with at most k non-zero values, is observed indirectly through a shorter measurement vector $\mathbf{y} \in \mathbb{R}^m$ and obtained from the linear equations $\mathbf{y} = \mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is an $m \times n$ measurement matrix, with $m \ll n$. The task of

compressed sensing is to recover $\mathbf{x}$ from $\mathbf{y}$. The first approach to solve the compressed sensing problem is to find a signal $\mathbf{x}$ with the smallest ℓ_0-norm given $\mathbf{y} = \mathbf{A}\mathbf{x}$. The ℓ_0-norm minimization of compressed sensing is known to be NP-hard. Instead, the ℓ_1-norm minimization solution based on linear programming (LP) was introduced to reconstruct $\mathbf{x}$. The LP technique for the compressed sensing problem, called Basis Pursuit [78, 86], has a remarkable performance, but its high complexity and running time makes it impractical in some applications which require fast reconstruction, or when the dimension of a measurement matrix is large.

To tackle the issue of complexity, message-passing algorithms for compressed sensing have been proposed, originating from channel coding mainly based on belief propagation [99], or iterative thresholding [92].

Recently low-complexity message-passing algorithm was proposed [103], which we refer to as the *interval-passing algorithm* (IPA). This algorithm performs at a very low complexity with very promising reconstruction performance compared to the original and complex Basis Pursuit and another very simple algorithm, known as verification decoding. However, it was introduced primarily for sparse binary measurement matrices.

The goal of this dissertation is to modify the IPA for non-negative real-valued measurement matrices, and analyze its reconstruction possibilities. We also present a practical application of the IPA, namely the spectroscopy and the chemical mixture estimation problem.

Contributions and organization of the dissertation

The main contributions of this dissertation are summarized as follows.

1) We introduce FAIDs and propose selection methods for column-weight-three LDPC codes using two approaches, and we design FAIDs for column-weight-four LDPC codes:

The first method of selection of FAIDs for column-weight-three codes uses large attractors of iterative decoders, which are the codewords of LDPC codes. We describe this method on the Tanner code where we extracted all the minimum weight codewords, and identified three distinct topologies of codewords. After explaining the limitations of current techniques which analyze harmful structures, we detail the selection method of good FAIDs by simulating error-patterns of weight 5 and 6 on the three types of minimal codewords.

In the second approach, the main contribution relies on the methodology to select a group of potential good decoders in the error floor on any column-weight-three code. To process this methodology, new concepts are introduced such as the concepts of noisy trapping sets and noisy critical numbers for the analysis of harmful structures, and the concept of decoder domination to perform the selection of the decoders. Simulation results show indeed that selected FAIDs have excellent performance in the error floor region on a variety of codes.

We also provide the first results on the design of FAIDs for column-weight-four LDPC codes using a heuristic-based method on a given high-rate code. We provide a simple procedure to extract some harmful structures responsible for the error floor in this particular code, and then design a FAID which has a better error correction capability on the extracted harmful topologies. Numerical results are provided to demonstrate that with our method, the designed 3-bit and 4-bit precisions FAID can outperform BP (floating-point) as well as offset min-sum in the error floor.

2) We present and analyze the IPA for compressed sensing:

We modify the original IPA in order to deal with non-negative real-valued sparse measurement matrix, and show that this low-complexity reconstruction algorithm provides a good complexity/performance trade-off between the complex reconstruction of the linear programming methods, and the very simple verification algorithm.

We analyze the reconstruction possibilities of the IPA, by establishing a link between the reconstruction failures of the IPA and the stopping sets of the binary

representation of the measurement matrix. Simulation results on codes for which the stopping sets distribution is known, confirm the influence of such structures on the IPA performance.

We demonstrate that the IPA has superior performance than iterative thresholding reconstruction methods which does not require sparse measurement matrices. The low-complexity of the measurement process for the IPA makes it also very attractive.

We show the IPA can be used in spectroscopy to solve the chemical mixture estimation problem. This requires to design carefully the measurement matrix that will act on the chemical mixture spectrum to obtain an overall measurement matrix which is sparse.

From the two main topics of the thesis, this manuscript is divided into two parts: chapters 1 and 2 are related to channel coding and low-complexity iterative decoders, and chapters 3 and 4 are related to compressed sensing and low-complexity iterative reconstruction algorithms. The outline of the dissertation is detailed as follows.

In chapter 1, we provide a detailed overview regarding channel coding, introducing the notations and the necessary background on iterative decoders and the error floor phenomenon.

In the chapter 2 we introduce the FAIDs, both approaches to select good FAID for column-weight-three codes in the error floor region, as well as the first study on FAID for column-weight-four codes.

Chapter 3 reviews recent work on compressed sensing with a description of the problem setting, the traditional approach using linear programming techniques, and the suggested iterative reconstruction methods.

In the chapter 4 we present the modified version of the IPA, and provide a complete analysis of the reconstruction possibilities of the IPA. We also present a first practical application using the IPA as reconstruction algorithm.

1. INTRODUCTION ON CHANNEL CODING

Ever since its inception in the 1950's, error-correcting codes have played an indispensable role in ensuring high reliability and low power consumption in wireless/wired data transmissions as well as in data storage, and have now become ubiquitous in all modern communication and data storage systems. An error-correction code essentially consists of an encoder and a decoder; information that is to be transmitted is encoded by adding redundancy in a specific manner at the transmitter and decoding is performed at the receiver to correct some or all of the errors contained in the received noisy data. In particular, a specific class of error-correcting codes called *low-density parity-check codes* (LDPC) [1] has not only revolutionized the communications and storage industry, but also sparked a widespread research interest over the past two decades, leading to the so-called field of *modern coding theory*.

Since the first goal of this dissertation is centrally themed around LDPC codes, we begin this chapter by providing a brief overview of the historical background of error-correcting codes especially related to LDPC codes, and provide the necessary preliminaries required for understanding this contribution of this dissertation in this field.

1.1. Historical background

The origin of error-correcting codes dates back to the work of Hamming during his time at Bell Labs in the late 1940's. At that time, Hamming got motivated to develop error-correcting codes when he became increasingly frustrated with relay computers he was working with, which would halt any submitted jobs containing errors. He eventually designed the first type of error-correcting linear block codes, now known as Hamming codes, which could correct a single error, and later published his work

in 1950 [2]. This led to the development of other important early codes such as the triple error-correcting Golay code [3] and the Reed-Muller (RM) codes [4, 5].

Meanwhile, Shannon, who happened to be a colleague of Hamming at Bell Labs, published his landmark paper [6] in 1948 that led to the birth of the field of information theory. The paper essentially laid down the fundamental limits of error-free transmission by introducing the notion of *channel capacity*. Shannon pointed out that every channel has an associated capacity, and he proved using random coding arguments that for transmission rates less than the channel capacity, there exists a code that can achieve arbitrarily low error-rates asymptotically with the length of the code. However, Shannon did not provide any insights into the explicit design of capacity-achieving codes, and thus, the quest for the search of such codes began.

In the ensuing years, much of the research was dedicated towards the design of linear block codes that had good error-correcting capabilities and good minimum distance. Initial code designs were for hard-decision channels and relied heavily on algebraic structure based on finite-field algebra. Codes such as the Bose-Chaudhuri-Hocquenghem (BCH) codes [7,8] and the Reed-Solomon (RS) codes [9] became widely used especially in magnetic recoding applications such as magnetic tape systems. The codes were decoded by an efficient algorithm developed by Berlekamp and Massey [10]. However, algebraic-based codes were still far from achieving the limits of Shannon's capacity in their asymptotic performance, and they lacked the random-like properties originally envisioned by Shannon.

Concurrently, another class of codes called *convolutional* codes were developed by Elias [11], which had a more probabilistic approach to channel coding. Interpreted as discrete-time finite-state systems, these codes had an underlying special structure known as the *trellis* which enabled linear-time encoding and inherently allowed for the capability to use practical soft-decision decoding algorithms. One of the most important decoding algorithms for these codes was developed by Viterbi, famously known as the *Viterbi algorithm* [12], which optimally estimated the most likely sequence of

transmitted bits. This proved to be a major attraction for these codes, and they were capable of providing substantial coding gains even though their asymptotic performance was still far from Shannon's capacity. Another algorithm that was developed in the same probabilistic spirit was the BCJR algorithm by Bahl, Cocke, Jelinek, and Raviv [13] which estimated the *a posteriori* probability of each transmitted bit. At the time, the BCJR algorithm was considered to be purely of theoretical interest as it was regarded too complex for practical implementations. However, the BCJR algorithm along with the class of convolutional codes turned out to be the most crucial steps for paving the way towards the development of capacity-achieving codes.

The breakthrough in the search for capacity-achieving codes was finally reached in 1993 by Berrou, Glavieux, and Thitimajshima with their discovery of *turbo codes* [14]. A key feature of these codes was the use of *iterative decoding*, which involved using the BCJR algorithm to iteratively exchange soft information between two convolutional codes concatenated in parallel. This proved to be pivotal for achieving near-Shannon-limit performance with reasonable decoding complexity. Subsequently, the class of *low-density parity-check* (LDPC) codes, originally invented by Gallager in early 1960's but remained forgotten for nearly thirty years, was rediscovered by MacKay [15] who showed that these codes were also capable of capacity-achieving performance. This led to a renaissance in the field of modern coding theory with LDPC codes becoming one of the most active topics of research.

LDPC codes are essentially linear-block codes whose parity-check matrices have a sparse number of non-zero entries. These codes are conveniently represented as bipartite graphs known as the *Tanner graphs* [16], and the decoding algorithm operates on the Tanner graph of the code. The decoding algorithms used for LDPC codes are based on a central iterative algorithm known as the *belief propagation* (BP). The BP is a message-passing algorithm that involves propagating probabilistic messages along the edges of the graph in order to estimate the *a posteriori* probabilities of the codeword bits thereby lending itself to low complexity implementations. The

notion of using graph-based decoding was eventually extended to other codes such as Turbo codes through the significant contributions of Tanner [16] and much later Wiberg [17]. Under BP decoding, LDPC codes are able to achieve an unprecedentedly good error-rate performance which has made them the overwhelming choice among existing codes for both present and emerging technologies in digital communications and data storage.

Within the past decade, with LDPC codes steadily gaining popularity, there has been an outburst of research related to the design of capacity-achieving LDPC codes as well as in the development and analysis of decoding algorithms. Richardson and Urbanke proposed the key technique of *density evolution* [19] for determining the decoding threshold of a given LDPC code under BP decoding, which is a threshold of noise below which the bit error probability tends to zero asymptotically with the length of the code. Using density evolution, Richardson, Shokrollahi and Urbanke optimized capacity-achieving irregular code ensembles for the best (highest) decoding thresholds that were very close to the Shannon limit. The technique was also subsequently used in the design of reduced-complexity BP decoders by Chen *et al.* [20] and Fossorier [21]*et al.*, and quantized BP decoders by Lee and Thorpe [22]. Another important asymptotic technique that was proposed for analyzing decoding algorithms simpler than BP (such as bit flipping) was the use of expander arguments and expander codes proposed by Sipser and Spielman [23]. Burshtein and Miller [24] applied expander arguments to message-passing to show that they could correct a linear fraction of errors. For finite-length analysis of codes, a novel approach to decoding called linear programming (LP) decoding was proposed by Feldman *et al.* [25], where the decoding problem is transformed to an LP formulation.

The problem of constructing codes with desirable structural properties and good minimum distance while maintaining their capacity-achieving ability also gained significant attention, with many applications facing stringent constraints in terms of storage requirements and implementation complexity. A particularly broad class of

codes called *quasi-cyclic* codes generated great appeal to the industry, where the parity-check matrices of these codes consist of blocks of circulants enabling much more efficient encoding and decoding implementations. Some of the notable works on quasi-cyclic codes include the codes proposed by Fossorier [26], the Protograph codes developed by Thorpe [27] and Divsalar *et al.* [28], the algebraic-based quasi-cyclic codes proposed by Lan *et al.* [29], finite geometry codes by Kou *et al.* [30], and combinatorially constructed codes by Vasic *et al.* [31]. Another class of codes that recently gained limelight are LDPC convolutional codes [32, 33], which are the convolutional counterparts to the conventional linear block code version of LDPC codes, and were recently shown by Lentmaier *et al.* to be capable of having BP thresholds nearly matching the MAP thresholds [34].

LDPC codes have already found their way into various standards such as the DVB-S2 (Digital Video Broadcasting), IEEE 802.3an (Ethernet), and are also being considered for the IEEE 802.16e (Wimax) and IEEE 802.11n (Wifi) standards. However, in spite of their excellent error-rate performance under BP decoding and their obvious advantages, they still have a major weakness that manifests itself in the form of an *error floor*. The error floor is an abrupt degradation in the performance of the code in the high signal-to-noise (SNR) region. This problem mainly arises due to the sub-optimality of BP decoding on finite-length codes with loopy graphs, especially when codes are designed to be asymptotically close to Shannon's limit. It could prove to be a major disadvantage particularly for applications requiring very low target error-rates such as storage devices. Therefore, the error floor problem has been widely regarded as one of the most important problems in coding theory and has attracted significant research interest over the past few years. Addressing the error floor problem is one of the main goals of this dissertation along with taking into account the effects of finite-precision for decoder realizations.

1.2. LDPC codes: fundamentals and notations

In this section, we will provide the necessary preliminaries related to LDPC codes starting from the basics of linear block codes to the general concept of message-passing and graph-based decoding. Along the way, we will introduce the required notations that will be used throughout this dissertation.

1.2.1. Linear block codes

An (N, K) binary linear block code $\mathcal{C}$ [35] is a K-dimensional subspace of $\mathrm{GF}(2)^N$, which is the vector space over the field GF(2) consisting of all possible binary N-tuples. Thus, the code $\mathcal{C}$ contains 2^k N-tuples or *codewords*. Given a message vector of k information bits, this vector is mapped to one of the codewords of length N in the code $\mathcal{C}$. The code $\mathcal{C}$ is said to have a *code rate* of $R = K/N$, which represents the amount of redundancy added by the code.

The support of a codeword $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, denoted by $\mathrm{supp}(\mathbf{x})$, is the set of all positions i such that $x_i \neq 0$. The *hamming weight* (or simply weight) of $\mathbf{x}$ denoted $w(\mathbf{x})$, is the cardinality of $\mathrm{supp}(\mathbf{x})$. The *hamming distance* (or simply distance) between any two codewords is the number of positions for which they differ in value. The minimum distance of a code $\mathcal{C}$, denoted by d_{min} is the smallest possible distance between any pair of codewords in $\mathcal{C}$. A codeword vector that has zero in all its positions is referred to as the *all-zero codeword*. Since a linear block must include the all-zero codeword, the minimum distance is simply the weight of the nonzero codeword that has the smallest weight.

The process of mapping a message vector of k information bits to a codeword of length N in the code $\mathcal{C}$ is known as *encoding*. Encoding is carried out through a $K \times N$ generator matrix $\mathbf{G}$ of the code $\mathcal{C}$, whose rows correspond to the basis vectors of $\mathcal{C}$, i.e., the linearly independent codewords. Given a message vector of k bits, say, $\mathbf{u} = (u_1, u_2, \ldots, u_k)$, a codeword $\mathbf{x} \in \mathcal{C}$ can obtained by performing $\mathbf{x} = \mathbf{uG}$.

A linear block code is also characterized by its parity-check matrix $\mathbf{H}$, which is an $M \times N$ matrix whose rows span the null space of $\mathbf{G}$, i.e., $\mathbf{G}\mathbf{H}^T = 0$. Consequently, every codeword $\mathbf{x}$ is orthogonal to the rows of $\mathbf{H}$ so that $\mathbf{x}\mathbf{H}^T = 0$. Given a vector $\mathbf{x}' \in \mathrm{GF}(2)^N$, the parity-check matrix can be used to verify whether $\mathbf{x}'$ is a codeword belonging to $\mathcal{C}$ or not. Therefore, each row of $\mathbf{H}$ is referred to as a *parity-check constraint* and there are M such parity-check constraints. The value of M is related to the code parameters by $M \geq (N - K)$, where the equality holds if $\mathbf{H}$ is a full-rank matrix.

Once a given message $\mathbf{u}$ is encoded to a codeword $\mathbf{x}$, it is then transmitted over a noisy channel and is received as $\mathbf{r} = (r_1, r_2, \ldots, r_N)$. *Decoding* is then performed to estimate the transmitted codeword $\mathbf{x}$ based on the received vector $\mathbf{r}$. The optimal decision rule used for determining the estimate $\hat{\mathbf{x}}$ is the *maximum a posteriori* (MAP) decoding rule, which essentially chooses a codeword $x \in \mathcal{C}$ that maximizes the a posteriori probability $\Pr(\mathbf{x}|\mathbf{r})$ (the probability that the transmitted codeword is $\mathbf{x}$ given that $\mathbf{r}$ was received). More precisely,

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{C}} \Pr(\mathbf{x}|\mathbf{r}).$$

We shall assume that all codewords are equally likely to be transmitted. Under this assumption, the MAP decoding problem reduces to the *maximum likelihood* (ML) decoding problem which determines the codeword that maximizes the $\Pr(\mathbf{r}|\mathbf{x})$. If $\mathbf{r} \in \mathrm{GF}(2)^N$, then the ML decoding problem is equivalent to finding the nearest neighbor of $\mathbf{r}$ in the vector space $\mathrm{GF}(2)^N$ that is a codeword in $\mathcal{C}$. Therefore the code's error-correcting capability is linked to its minimum distance d_{min} as the code is guaranteed to correct $\lfloor (d_{min} - 1)/2 \rfloor$ errors. In general, ML decoding on a linear-block code is NP-hard as it requires a brute-force search of all codewords in the vector space $\mathrm{GF}(2)^N$. Hence, sub-optimal decoding algorithms that enable efficient implementations are used in practice.

1.2.2. Channel assumptions

Let us assume that a codeword $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathcal{C}$ is transmitted over a noisy channel and is received as a vector $\mathbf{r}$. If the channel is memoryless, then the probability $\Pr(\mathbf{r}|\mathbf{x})$ can be expressed as

$$\Pr(\mathbf{r}|\mathbf{x}) = \prod_{i=1}^{N} \Pr(r_i|x_i).$$

This implies that the effect of the channel on every bit in the transmitted codeword is independent from one another. Hence, during decoding, it suffices to decide the value for each x_i independently based on maximizing $\Pr(r_i|x_i)$. This probability is also referred to as a *likelihood*.

For soft-decoding algorithms, the received vector $\mathbf{r}$ is mapped to a vector of probabilities or log-likelihood ratios before serving as input to the decoder. Let $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ denote the vector that is input to a given decoder. We shall refer to the values y_i in vector $\mathbf{y}$ as *channel values*, to signify the fact they are determined based on the values received from the channel. If $\mathbf{y}$ is a vector of log-likelihoods, then the log-likehood ratio (LLR) corresponding to a bit position i is given by

$$y_i = \log\left(\frac{\Pr(r_i|x_i = 0)}{\Pr(r_i|x_i = 1)}\right).$$

Examples of memoryless channels include the Binary Symmetric channel (BSC), the Binary erasure channel (BEC), and the Additive White Gaussian channel (AWGNC). Examples of channels with memory include partial-response channels and other channels in magnetic recording that cause inter-symbol interference.

For this dissertation, we shall only focus on the BSC. The BSC is a binary-input binary-output memoryless channel and therefore the received vector $\mathbf{r}$ is also binary N-tuple. The channel flips a bit in the transmitted codeword with a *cross-over probability* of α. Fig. 1.1 shows an illustration of the BSC. A transmitted bit 0 is received as 0 with probability $1 - \alpha$ and as a 1 with probability α.

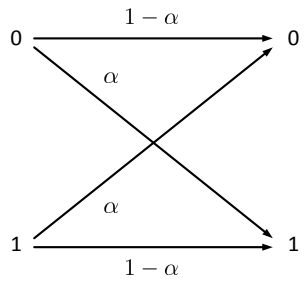


FIGURE 1.1. Illustration of the Binary Symmetric channel

For the BSC, given a cross-over probability α,the LLR is determined from $\mathbf{r}$ as follows

$$y_i = \begin{cases} \log\left(\frac{1-\alpha}{\alpha}\right) & \text{if } r_i = 0 \\ \log\left(\frac{\alpha}{1-\alpha}\right) & \text{if } r_i = 1 \end{cases}$$

Note that by the definition above, y_i being positive implies that the corresponding codeword bit is more likely to be a zero, and y_i being negative implies the negative corresponding codeword bit is more likely to be a one.

1.2.3. Tanner graph representation

LDPC codes are linear block codes that have sparse parity-check matrices. Any (N, K) linear block code $\mathcal{C}$ can be represented by a bipartite graph G, also known as the Tanner graph, which consists of two sets of nodes: the set of variable nodes $V = \{v_1, \cdots, v_N\}$ and the set of check nodes $C = \{c_1, \cdots, c_M\}$. Given the parity-check matrix $\mathbf{H}$ of a code $\mathcal{C}$, the Tanner graph is obtained by assigning a variable node corresponding to each codeword bit and a check node corresponding to each parity-check constraint, and then connecting variable nodes to certain check nodes based on the parity-check matrix. An edge connection between a particular variable node and a particular check node is made if the codeword bit associated with the variable node participates in the parity-check constraint associated with the check node. This is illustrated with the help of an example.

Example 1. *Consider the parity-check matrix of an* $(8, 4)$ *extended Hamming code*

with $d_{min} = 4$, which is given by

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

In order to generate the Tanner graph, first observe that the graph must contain 8 variable nodes corresponding to the 8 columns, and 4 check nodes corresponding to the 4 parity-check constraints. Therefore $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ and $C = \{c_1, c_2, c_3, c_4\}$. By convention, ● is used to depict a variable node, and □ is used to depict a check node in the Tanner graph. Now looking at the first column in $\mathbf{H}$, there is a 1 in the first row and in the fourth row. Therefore, v_1 is connected to check nodes c_1 and c_4. Similarly, all other variable nodes are connected to the appropriate check nodes. Fig. 1.2 shows the resulting Tanner graph of the code.

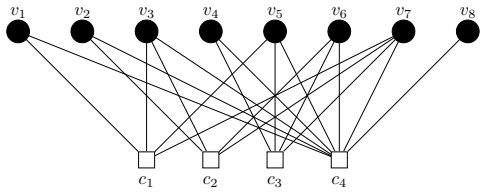


FIGURE 1.2. Tanner graph of the $(8, 4)$ extended Hamming code

The check nodes (variable nodes respectively) connected to a variable node (check node respectively) are referred to as its *neighbors*. The set of neighbors of a node v_i is denoted as $\mathcal{N}(v_i)$, and the set of neighbors of node c_j is denoted by $\mathcal{N}(c_j)$. The degree of a node is the number of its neighbors. Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ be a vector such that x_i denotes the value of the codeword bit associated with the variable node v_i. We shall refer to this value as the *bit value* of node v_i. $\mathbf{x}$ is a codeword if and only if for each check node, the modulo two sum of the bit values of its neighbors is zero.

An LDPC code is to be *regular*, if in its corresponding Tanner graph G, all variable nodes have the same degree d_v, and all check nodes have the same degree d_c. A code is said to be a column-weight-d_v (or d_v-left-regular) code if all variable nodes have the

same degree d_v, which is also referred to as *left degree*. A code is said to be *irregular* if there are variable nodes as well as check nodes that have different degrees. The (8,4) extended Hamming code provided above is an example of an irregular code.

The girth of the Tanner graph g is length of the shortest cycle present in the Tanner graph of the code. A cycle of length g is referred to as a g-cycle. For this dissertation, we will be mostly concerned with column-weight-three LDPC codes. Henceforth, for convenience, Tanner graphs shall be simply referred to as graphs.

1.2.4. Message-passing decoders

Message-passing (MP) decoders are a class of iterative decoders that operate on the graph of the code. The a posteriori probability of a codeword bit associated to each variable node in the graph is calculated by exchanging messages iteratively between the set of variable nodes and the set of check nodes along the edges of the graph. A major attraction with MP decoders is that all the computations are carried out locally at each node, and therefore an efficient decoder implementation can be realized.

Recall that $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ is the input to the MP decoder. Let $\mathcal{Y}$ denote the alphabet of all possible channel values. Let $\mathcal{M}$ denote the alphabet of all possible values that the messages can assume. Let $m^{(k)}(v_i, c_j)$ denote the message passed by a variable node $v_i \in V$ to its neighboring check node $c_j \in C$ in the k^{th} iteration and $m^{(k)}(c_j, v_i)$ denote the vice versa. Let $m^{(k)}(\mathcal{N}(v_i), v_i)$ denote the set of all incoming messages to variable node v_i and $m^{(k)}(\mathcal{N}(v_i)\backslash c_j, v_i)$ denote the set of all incoming messages to variable node v_i except from check node c_j. Let $m^{(k)}(\mathcal{N}(c_j)\backslash v_i, c_j)$ and $m^{(k)}(\mathcal{N}(c_j)\backslash v_i, c_j)$ be defined similarly.

Any MP decoder requires two update functions: Φ_v used for update at the variable nodes, and Φ_c used for check nodes. Both update functions are symmetric functions on the incoming messages, i.e., they remain unchanged by any permutation of its messages. Thus without abuse of notation, if the arguments of a function are written

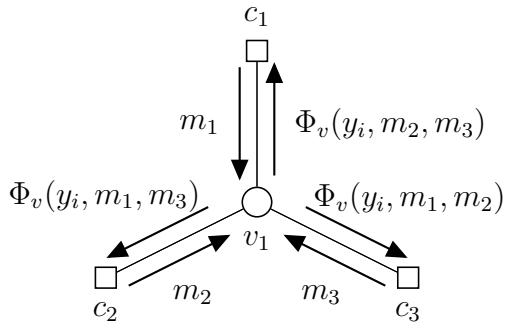


FIGURE 1.3. Update of the messages at a variable node.

as a set, we imply that the order of these arguments is insignificant.

We now describe the MP decoding process. Initially, all the messages are set to zero, i.e., $m^{(0)}(\mathcal{N}(v_i), v_i) = 0 \quad \forall v_i \in V$ and $m^{(0)}(\mathcal{N}(c_j), c_j) = 0 \quad \forall c_j \in C$. Then for every iteration $k > 0$, messages are propagated in the following manner:

$$\begin{aligned} m^{(k)}(v_i, c_j) &= \Phi_v\left(y_i, m^{(k-1)}(\mathcal{N}(c_j)\backslash v_i, c_j)\right) \\ m^{(k)}(c_j, v_i) &= \Phi_c\left(m^{(k)}(\mathcal{N}(c_j)\backslash v_i, c_j)\right) \end{aligned}$$

An important feature of MP decoding that can be noted from the above is that during the update of messages at a particular node, the outgoing message from the node on an edge is determined as a function of all local messages incoming from its neighbors excluding the message incoming on that particular edge. We refer to such messages as *extrinsic incoming messages*. By doing so, the node is to an extent treating all its incoming messages as independent messages. This is the subtle difference between MP decoding and iterative bit-flipping decoding algorithms. Fig. 1.3 and 1.4 describes the update of the messages at a degree-three variable node and check node, respectively.

At the end of each iteration, a symmetric *decision function* Ψ, is used to decide the bit value of each node $v_i \in V$ based on its incoming messages. Let $\hat{\mathbf{x}}^{(k)} = (\hat{x}_1^{(k)}, \hat{x}_2^{(k)}, \ldots, \hat{x}_N^{(k)})$ denote the vector of bit values decided at the end of the k^{th} iter-

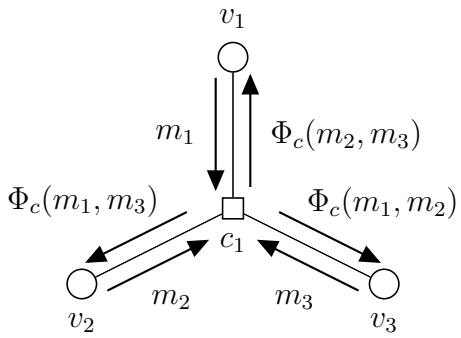


FIGURE 1.4. Update of the messages at a check node.

ation. The bit value $\hat{x}_i^{(k)}$ of each node v_i is determined by

$$\hat{x}_i^{(k)} = \Psi\left(y_i, m^k(\mathcal{N}(v_i), v_i)\right).$$

The estimate $\hat{\mathbf{x}}^{(k)}$ is then verified to check whether it is a codeword or not. This is easily done by passing the decided bit values along the edges and verifying whether every check node is *satisfied* or not. A check node is satisfied if the modulo 2 sum of all the bit values of its neighbors is zero. If all check nodes are satisfied, then $\hat{\mathbf{x}}^{(k)}$ is a codeword and the decoding is terminated.

In conventional MP decoders, the decision function is either a simple majority rule (if messages are binary), or simply the sign of the sum of its arguments (if messages are log-likelihoods). More precisely, for a node v_i with degree d_v

$$\Psi(y_i, m^k(\mathcal{N}(v_i), v_i)) = \begin{cases} 1 & \text{if } \left(y_i + \sum m^{(k)}(\mathcal{N}(v_i), v_i)\right) > 0 \\ 0 & \text{if } \left(y_i + \sum m^{(k)}(\mathcal{N}(v_i), v_i)\right) < 0 \\ \text{sgn}(y_i) & \text{otherwise} \end{cases}$$

where the sgn function outputs a 0 if the sign of the argument is positive, and a 1 if it is negative.

The algorithm runs until either $\hat{\mathbf{x}}^{(k)}$ is a codeword, or until the maximum number of iterations is reached, whichever occurs first. We say that the decoder has *converged* if $\hat{\mathbf{x}}^{(k)}$ is a codeword for some k, else we say that the decoder has *failed*. A *success*

is declared if the decoder converges to the right codeword, and a *miss-correction* is declared if the decoder converges to a wrong codeword.

Note that our general description of MP decoding assumed that the scheduling used is a *flooding* scheduling, which involves updating all variable nodes simultaneously update followed by a simultaneous update of all check nodes. In contrast, a *serial* or layered scheduling could also be carried out for MP [36]. Discussions regarding scheduling are beyond the scope of this dissertation, and throughout we implicitly assume that the flooding scheduling is used for any MP decoder being described.

MP decoders can be broadly classified into two classes: hard decoding algorithms (where the messages assume binary values) such as the Gallager A and B algorithms [1], and soft decoding algorithms (where the messages assume real values) which are conventionally based on the BP algorithm such as the sum-product algorithm and the min-sum algorithm [20]. We now discuss the BP algorithm in greater detail.

1.2.5. Belief propagation: Sum-Product and other low-complexity variants

All conventional MP algorithms used for decoding LDPC codes are based on the BP algorithm [37]. The BP has its roots in the broad class of Bayesian inference problems [38], and it is used to compute marginals of functions on a graphical model. Although exact inference in Bayesian belief networks is hard in general and inference using BP is exact only on loop-free graphs (trees), it provides surprisingly close approximations to exact marginals on loopy graphs.

The problem of decoding on the graph of the code is also a Bayesian inference problem, and therefore the BP algorithm is well-suited for this purpose. As a decoding algorithm for LDPC codes, the BP propagates probabilistic messages along the edges of the Tanner graph in an iterative fashion. BP can be implemented in two domains, namely the probabilistic domain where the messages are probabilities, and the LLR

domain where messages are LLRs. We shall describe the BP algorithm in the LLR, domain which is more commonly used due to it being less sensitive to numerical precision issues. The BP algorithm is also known as the *Sum-Product* algorithm.

In the LLR domain, the messages and channel values are real-valued LLRs, i.e., $\mathcal{M} = \mathbb{R}$ and $\mathcal{Y} = \mathbb{R}$. The function $\Phi_v : \mathbb{R} \times \mathbb{R}^{d_v - 1} \to \mathbb{R}$ is the update function used at a variable node of degree d_v, and $\Phi_c : \mathbb{R}^{d_c - 1} \to \mathbb{R}$ is the update rule used at a check node with degree d_c.

Let $m_1, m_2, \ldots, m_{l-1}$ denote the $l-1$ extrinsic incoming messages of a node with degree l, which are used in the calculation of the outgoing message. The update rules for the BP algorithm are given as follows:

$$\Phi_v(y_i, m_1, \ldots, m_{d_v - 1}) = y_i + \sum_{j=1}^{d_v - 1} m_j \tag{1.1}$$

$$\Phi_c(m_1, \ldots, m_{d_c - 1}) = 2 \tanh^{-1} \left(\prod_{j=1}^{d_c - 1} \tanh \left(\frac{m_j}{2} \right) \right). \tag{1.2}$$

When the magnitudes of the messages reach a large value, the check node update function of the BP algorithm can be approximated to:

$$\Phi_c(m_1, \ldots, m_{d_c - 1}) = \left(\prod_{j=1}^{d_c - 1} \operatorname{sgn}(m_j) \right) \min_{j \in \{1, \ldots, d_c - 1\}} (|m_j|) \tag{1.3}$$

where sgn denotes the standard sign function. The check node update function defined in Eq. (1.3) along with variable node update function defined in Eq. (1.1) together constitute the min-sum decoder. Thus, the min-sum is an approximation of BP decoding.

Clearly the min-sum decoder is much lower in complexity than the BP algorithm due to its simplified check node operation. However, the performance loss arising from using min-sum instead of BP is quite large. Hence, many low-complexity variants have been proposed which attempt to reduce this gap in performance. All of the variants proposed typically deal with simplifying the check node update function while keeping the variable node update function intact, as defined in Eq. (1.1).

One important low-complexity variant of BP that requires mentioning is the *offset min-sum* decoder proposed by Chen *et al.* [20]. For this algorithm, an offset factor γ is introduced into the check node update function. This offset factor serves to reduce the overestimate of the outgoing message produced by a check node using the rule in Eq. (1.3), especially when the magnitudes of the incoming messages are small. Therefore, the offset factor is also often referred to as a *correction factor*. The modified check node update function is as follows:

$$\Phi_c(m_1, \ldots, m_{d_c-1}) = \left(\prod_{j=1}^{d_c-1} \operatorname{sgn}(m_j) \right) \max \left(\min_{j \in \{1,\ldots,d_c-1\}} (|m_j|) - \gamma, 0 \right). \tag{1.4}$$

The offset factor γ could be fixed or could be varied as a function of the SNR. Also the value for γ can be chosen using the density evolution technique for maximizing the decoding threshold as done in [20], or by a brute-force simulation on a given code that checks different values of γ and selects the one giving the best performance.

1.3. Error floor problem

For any typical finite-length LDPC code, the error-rate performance curve plotted as a function of SNR under iterative decoding, consists of two distinct regions: the waterfall region, and the error floor region. In the waterfall region (which is the low SNR region), the error-rate drops significantly with increase in SNR making the curve look like a waterfall. On the other hand, in the error floor region (which is the high SNR region), the decrease in the error-rate dramatically slows down and the curve tends to flatten out turning into an error floor. Fig. 1.5 illustrates the two regions on a typical FER performance curve of an LDPC code plotted as a function of the cross-over probability α of the BSC.

The error floor problem arises due to the suboptimality of BP decoding on loopy graphs. It can be troublesome for applications requiring very low target error-rates, and especially when high-rate codes are employed, which have relatively dense graphs.

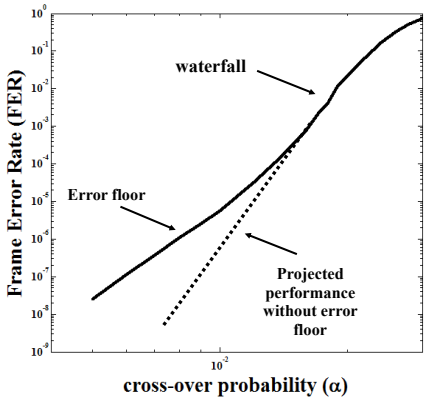


FIGURE 1.5. Typical performance of BP decoding on an LDPC code over the BSC

Although asymptotic techniques such as density evolution provide an accurate prediction of the FER performance in the early waterfall region, the point at which the error floor starts as well as its slope are greatly dependent on the particular structure of a code, and hence cannot be predicted by density evolution. Therefore, finite-length analysis techniques are required for the study of error floors. Besides, for codes with moderate to large lengths, the ability to predict error floors becomes even more critical as the error floor may be unreachable by Monte-Carlo simulations.

1.3.1. Prior work on error floor estimation

The first crucial observation on error floors was made by MacKay and Postol [39], who found that the minimum distance of an LDPC code did not necessarily play a dominant role in the error floor of a code. In their investigations with a Margulis construction of an LDPC code for the AWGNC, they observed that it showed a high error floor in spite of its good minimum distance, and they attributed it to the presence of *near-codewords* in the graph. Another notion proposed in the spirit of finite-length analysis was the notion of *pseudocodewords* originally introduced by Wiberg [17], and later further developed by Frey *et al.* [40] and Forney *et al.* [41]. Pseudocodewords are essentially outputs of the decoder that are not necessarily codewords and stem from

the different possible binary configurations of the computation tree. These works further motivated investigations into finite-length analysis of codes under iterative decoding with attempts to characterize the error floors for different channels.

For the BEC, the error floor could be well-characterized through the work of Di *et al.* [42], who introduced the notion of *stopping sets* which are purely combinatorial objects. Later Orlitsky *et al.* [43] provided asymptotic results on the stopping set distribution for different code ensembles.

However, a huge stride towards the understanding of the error floor problem in a general setting, was made through the pioneering work of Richardson [44], who showed that the problem is at least partially combinatorial in nature. Richardson introduced the notion of *trapping sets*, which are certain problematic loopy structures present in the graph of the code that cause the decoder to fail for certain low-noise configurations, and are a function of both the code and the decoding algorithm in operation. Using this notion, he proposed a semi-analytical method for estimating the error-rates of BP decoding on the AWGNC in the error floor by identifying the dominant trapping sets. Another approach based on statistical physics was also proposed for the AWGNC by Stepanov *et al.* [45] through the notion of *instantons*, which are certain low-noise configurations that swayed the decoding trajectory away from the correct codeword. Later, Chilappagari *et al.* [46] in line with Richardson's work, presented results on the error floor estimation for the BSC under Gallager B decoding. Another notion called *absorbing sets*, was proposed by Dolecek *et al.* [47], which are a special type of trapping sets that are purely combinatorial and stable under the bit-flipping algorithm, and this notion enabled them to perform a detailed analysis on array-based LDPC codes.

Many other works also subsequently emerged that further developed on the notion of pseudocodewords. Vontobel and Koetter introduced the concept of *graph covers* to explain failures of iterative decoding [49], and showed that pseudocodewords arising from graph covers are identical to pseudocodewords of the linear programming

(LP) decoding [25] which constituted the vertices of the *fundamental polytope*. Later, Kelly and Sridhara [50] used graph covers to derive bounds on the minimum pseudocodeword weight in terms of girth and the minimum left-degree of the graph. The pseudocodeword analysis was also extended to the class LDPC-convolutional codes by Smarandache *et al.* [51]. However, pseudocodewords are purely topological in nature and do not take a particular decoding algorithm into account. Therefore, the notion of trapping sets is required for studying failures of a particular iterative decoder.

It is evident from previous discussions that the main utility behind the notion of trapping sets is in enabling the error floor estimation of a given code under a particular decoding algorithm. If the relevant topological structures corresponding to different trapping sets could be identified a priori and enumerated on the graph of a given code without the need for simulation, a reasonable estimate of the error floor could be obtained provided that each trapping set's contribution towards the error floor is known or determined.

Although it was shown by McGregor and Milenkovic [53] that performing an exhaustive search of all trapping sets in the graph of a given code is NP-hard, several good practical approaches have been proposed. Milenkovic *et al.* [54] examined the asymptotic trapping set distributions for both regular and irregular LDPC code ensembles. Cole *et al.* [55] proposed a method to estimate the error floor based on importance sampling, which is similar to Richardson's approach in [44]. Wang *et al.* proposed an efficient algorithm to exhaustively enumerate both trapping sets and stopping sets for codes with relatively short to moderate block lengths ($N \approx 500$). Abu-Surra *et al.* [56] proposed an efficient improved impulse method that could enumerate trapping sets by augmenting the Tanner graph with auxiliary variable nodes, and treating the problem as if it were searching for low-weight codewords. Although the method does not guarantee enumeration of all trapping sets, it is reasonably reliable and is applicable to any arbitrary graph. A method that uses the *branch-and-bound* approach to enumerate stopping sets in a given code was proposed by

Rosnes and Ytrehus [57]. Karimi and Bannihasemi [58] recently proposed an algorithm which could efficiently enumerate the dominant trapping sets present in any arbitrary graph by recursively expanding the cycles present in the graph. More recently, Zhang and Siegel [59] proposed an efficient technique that expanded on the branch-and-bound approach by transforming the bounding step to an LP formulation, and which allowed a complete enumeration of trapping sets up to a certain size with reasonable computation time.

In spite of the fact that all the above works are useful for enumerating trapping sets in a given graph which aid in estimating the error floor, none of them directly address the issue of how to utilize the knowledge of trapping sets for constructing better codes or improving the decoding algorithms. For instance, it is highly non-trivial to determine which particular subgraphs (that correspond to certain trapping sets) should be avoided during the construction of codes in order to improve the error floor performance. Vasić *et al.* [60] proposed the *trapping set ontology*, which is a hierarchy of trapping sets that exploits the topological relations, and can be utilized for code construction or decoder design. We shall use this in our dissertation in the decoder design.

For the remainder of this section, we shall elaborate on some of the relevant notions used to characterize decoder failures, and describe the trapping set ontology.

1.3.2. Characterization of failures of iterative decoders

We begin by defining the notion of *trapping sets* and their related terminologies as originally provided by Richardson in [44]. Note that for purpose of exposition, we shall assume that the all-zero codeword was transmitted. This is a valid assumption as the MP decoders considered and the channel (BSC) are symmetric [19]. Recall that $\mathbf{y}$ is the decoder input, and $\hat{\mathbf{x}}^{(k)}$ is the estimate on the codeword bits after k iterations. Let I denote the maximum number of iterations allowed for decoding.

Definition 1. *A variable node v_i is said to be eventually correct if there exists a positive integer l such that for all $l \leq k \leq I$, $\hat{x}_i^{(k)} = 0$*

Definition 2. *For a decoder input $\mathbf{y}$, a trapping set $\mathbf{T}(\mathbf{y})$ is a non-empty set of variable nodes that are not eventually corrected by the decoder.*

The above definition is quite general as it includes all possible decoder failures such as stable fixed points as well as failures resulting from the oscillatory behavior of the decoder output among different states constituting a basin of attraction. Note that if $\mathbf{T}(\mathbf{y}) = \emptyset$, then the decoding is successful on input $\mathbf{y}$.

A trapping set (TS) $\mathbf{T}(\mathbf{y})$ is said to be an (a, b) TS, if it has a variable nodes, and b odd-degree check nodes in the subgraph induced by $\mathbf{T}(\mathbf{y})$. For convenience, we shall use the notation $\mathcal{T}(a, b)$ to denote the topological structure associated with an (a, b) TS, which is simply a graph consisting of a variable nodes and b odd-degree check nodes. Note that two trapping sets can share the same parameters (a, b) but have different underlying topologies.

If the degree of a check node is at most two in the graph, then the TS is said to be an *elementary* TS. It has been observed by many researchers that the dominant trapping sets in the error floor are typically elementary trapping sets [44, 52]. Therefore, we shall place more focus on elementary trapping sets during our discussions. Fig. 1.6 shows an example of a subgraph corresponding to an elementary $(5, 3)$ TS. Note that ■ is used to denote an odd-degree check node, and will be used throughout this dissertation for any future graphical illustrations.

Note that non-zero codewords (under the all-zero codeword assumption) are $(a, 0)$ trapping sets whose corresponding graphs have only even-degree check nodes. These are precisely the only trapping sets under ML decoding. However, for any sub-optimal iterative decoder, there will be other (a, b) TSs.

Stopping sets [42] are a sub-class of trapping sets that were introduced to characterize failures on the BEC. Note that for the BEC, a transmitted bit is either received

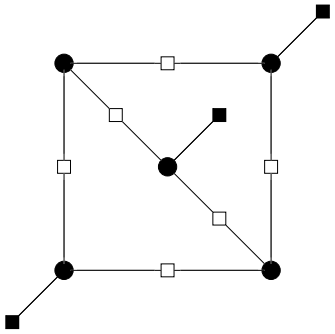


FIGURE 1.6. Subgraph corresponding to a $\mathcal{T}(5,3)$

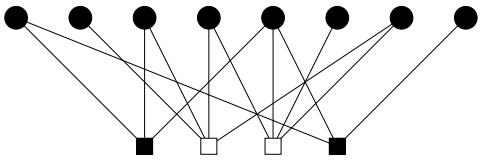


FIGURE 1.7. Subgraph corresponding to a $(8,2)$ stopping set

correctly or is erased. Let S denote a subset of variable nodes and let $\mathcal{N}(S)$ denote the set of neighbors $\{\mathcal{N}(v_i) : \forall v_i \in S\}$. The definition is given below.

Definition 3. *A stopping set is a subset of variable nodes S in the graph G, such that every neighbor in $\mathcal{N}(S)$ is connected to subset S at least twice.*

The above definition implies that stopping sets are (a,b) TSs whose corresponding graphs do not contain any degree-one check node. For the BEC, iterative decoding fails if all a variable nodes of an (a,b) stopping set are erased, regardless of its neighborhood in the original graph G and the number of iterations allowed. Therefore, stopping sets can be treated as purely combinatorial objects, and the error floor can be completely characterized for the BEC under iterative decoding. Fig. 1.7 shows an example of an $(8,2)$ stopping set whose corresponding graph contains two degree-3 check nodes.

An *absorbing set* [47] is a special type of (a,b) TS that is stable under the bit-flipping decoding. For a given subset S of variable nodes, let $\mathcal{E}(S)$ (resp. $\mathcal{O}(S)$) denote the set of neighboring checks of S that are even-degree (resp. odd-degree).

Definition 4. *An absorbing set is a subset of variable nodes S in the graph G, such that each variable node in S has strictly greater neighbors in $\mathcal{E}(S)$ than in $\mathcal{O}(S)$. In addition, if all remaining variable nodes in $V \setminus S$ also have strictly greater neighbors in $\mathcal{E}(S)$ than in $\mathcal{O}(S)$, then it is said to be a fully absorbing set.*

Another closely related type of trapping set is a *fixed set* that was introduced by Chilappagari and Vasić [48] to characterize failures of Gallager A/B decoding on the BSC. It is defined as follows.

Definition 5. *For a received vector $\mathbf{r}$ from the BSC and decoder input $\mathbf{y}$, a trapping set $\mathbf{T}(\mathbf{y})$ is said to be a fixed set, if $\mathbf{T}(\mathbf{y}) = \text{supp}(\mathbf{y})$.*

The necessary and sufficient conditions for a set of variable nodes to form a fixed set for the Gallager A/B algorithm is provided through the following theorem [48].

Theorem 1. *Let $\mathcal{C}$ be an LDPC code with d_v-left-regular graph G. Let $\mathbf{T}$ be a subset of a variable nodes with induced subgraph $\mathcal{T}$. Let the checks in $\mathcal{T}$ be partitioned into two disjoint subsets; $\mathcal{O}$ consisting of checks with odd degree and $\mathcal{E}$ consisting of checks with even degree. Then $\mathcal{T}$ is a fixed set for the Gallager A/B algorithm iff : (a) Every variable node in $\mathcal{I}$ has at least $\lceil \frac{d_v}{2} \rceil$ neighbors in $\mathcal{E}$ and (b) No $\lfloor \frac{d_v}{2} \rfloor$ of $\mathcal{O}$ share a neighbor outside $\mathcal{I}$.*

Note that the definition of absorbing set satisfies the above theorem and therefore is always a fixed set; however the opposite is not necessarily true. For column-weight-three codes though, they are the same. For this dissertation, we shall mainly utilize the notion of trapping sets to characterize decoder failures.The notion of fixed set will be used only mainly for discussing the trapping set ontology which will be introduced shortly.

Given an (a, b) TS, it is also important to determine a measure of relative *harmfulness* of the TS which is based on its underlying topological structure. Again for

the BSC under the Gallager B decoding, Chilappagari and Vasic [46] introduced the notion of *critical number* as a measure of harmfulness. Given the subgraph corresponding to a TS is $\mathcal{T}(a,b)$, let $\mathbf{y}_\mathcal{T}$ denote the vector of channel values received by the variable nodes in $\mathcal{T}(a,b)$. Let $\mathbf{T}(\mathbf{y}_\mathcal{T})$ denote the set of nodes that failed to converge to the right values.

Definition 6. *The critical number n_c of a trapping set whose subgraph is $\mathcal{T}(a,b)$, is the minimum number of errors introduced in the subgraph $\mathcal{T}(a,b)$ that causes the Gallager-B decoder. More precisely,*

$$n_c = \min_{\mathbf{T}(\mathbf{y}_\mathcal{T})\neq\emptyset} |\mathrm{supp}(\mathbf{y}_\mathcal{T})|.$$

Note that the dominant trapping sets are the subgraphs with the least critical number. Although this notion is limited to Gallager-B decoding, later on, we will generalize this notion to include other decoding algorithms as well.

Henceforth, for convenience, whenever we refer to a TS, we shall implicitly refer to its underlying topological structure, which is a subgraph induced by the variable nodes belonging to the TS.

1.3.3. Trapping set ontology

The trapping set ontology (TSO) [60] is a database of trapping sets that is organized as a hierarchy based on their topological relations. This topological relationship between trapping sets is specified in the form of a *parent-child* relationship. A trapping set $\mathcal{T}_1$ is said to be a *parent* of a trapping set $\mathcal{T}_2$ if $\mathcal{T}_2$ contains $\mathcal{T}_1$, and $\mathcal{T}_2$ is then considered to be a *child* of $\mathcal{T}_1$. The main objective for using the TSO is to make the identification of relevant trapping sets independent of a given code, as well as to serve as a guide for code construction or decoder design in addition to enabling efficient enumeration techniques.

The development of TSO was primarily motivated by the work of Chilappagari *et al.* in [61], who observed that the trapping sets found for various iterative decoders over different channels are closely related. For instance on a given code, many trapping sets found for the BP decoding over the AWGN were either the same as the trapping sets of Gallager B over BSC, or bigger subgraphs containing them. This implies that there exists a topological interrelation among trapping sets and in a broader sense, a topological interrelation among error patterns that cause decoding failures for various algorithms on different channels. Relying on this fact, the TSO is generated based on the notion of fixed sets for Gallager B, with the purpose of capturing these topological relations in order to provide a hierarchy.

Since the necessary and sufficient conditions for a fixed set of Gallager B are clearly defined through Theorem 1, this notion is used to generate the TSO. Also it is well-known that, in general, trapping sets are subgraphs formed by cycles or union of cycles [44]. Therefore, assuming that any code considered has at least girth g and left degree d_v, the TSO is generated as follows. Starting with a g-cycle, variable nodes of d_v degree are added to the g-cycle in all possible ways thereby expanding the g-cycle to a chosen maximum size, while enforcing the constraint that the resulting subgraph after each addition of a variable node, is elementary and satisfies Theorem 1. Once all different expanded subgraphs are generated, a hierarchy is then established based on their parent-child relationship. The procedure is then repeated starting with a $(g+2)$-cycle, and so on.

Note that although the methodology is restricted by using Theorem 1 for fixed sets, it can be easily generalized for generating other types of subgraphs, provided that the constraints or desired properties of the subgraph are clearly specified. Moreover, as previously pointed out, since failures of other iterative decoders over channels other than BSC are also topologically linked to failures of Gallager B decoding on the BSC, we regard the subgraphs in the TSO as a good starting point for consideration in decoder design or code construction.

2. FINITE ALPHABET ITERATIVE DECODERS: PRESENTATION, SELECTION AND DESIGN

The goal of this chapter is to introduce a new class of finite precision iterative decoders on the BSC, which we refer to as *Finite Alphabet Iterative Decoders* (FAIDs), and which are able to surpass the BP in the error floor region at a very low complexity. We then propose two selection methods based on two approaches for column-weight-three LDPC codes. The first approach is code dependent, and deals with the FAID selection on minimal codewords of a given code which are seen as larger attractors than the sole trapping sets. The second approach provides a methodology to select candidate FAIDs that are potentially good in the error floor without considering any specific code. This approach relies on the introduction of three essential concepts, namely the *noisy trapping sets*, the *noisy critical number vector* and the *decoder domination*. Finally, this chapter addresses also the design of FAID for column-weight-four codes, providing good performing 3-bit FAID derived from the extraction of harmful structures on a given code.

2.1. Preliminaries

Let G denote the Tanner graph of an (N, K) binary LDPC code $\mathcal{C}$ of rate $R = K/N$, which consists of the set of variable nodes $V = \{v_1, \cdots, v_N\}$ and the set of check nodes $C = \{c_1, \cdots, c_M\}$. The degree of a node in G is the number of its neighbors in G. A code $\mathcal{C}$ is said to have a regular column-weight d_v if all variable nodes in V have the same degree d_v. The set of neighbors of a node v_i is denoted as $\mathcal{N}(v_i)$, and the set of neighbors of node c_j is denoted by $\mathcal{N}(c_j)$. The girth of G is the length of shortest cycle present in G.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ denote a codeword of $\mathcal{C}$ that is transmitted over the BSC,

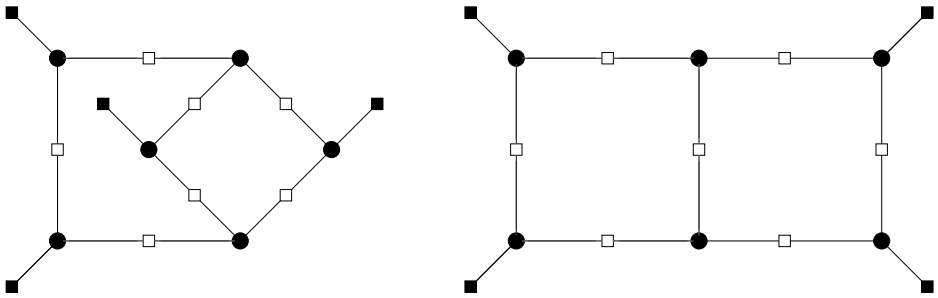
where x_i denotes the value of the bit associated with variable node v_i, and let the vector received from the BSC be $\mathbf{r} = \{r_1, r_2, \ldots, r_N\}$. Let $\mathbf{e} = (e_1, e_2, \ldots, e_N)$ denote the *error pattern* introduced by the BSC such that $\mathbf{r} = \mathbf{x} \oplus \mathbf{e}$, and $\oplus$ is the modulo-two sum. The support of an error vector $\mathbf{e} = (e_1, e_2, \ldots, e_N)$, denoted by $\text{supp}(\mathbf{e})$, is defined as the set of all positions i such that $e_i \neq 0$. Let $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ be the input to the decoder, where each y_i is calculated based on the received value r_i. We shall also refer to the values y_i as *channel values*. During the analysis of decoders, we shall assume that the all-zero codeword was transmitted. This is a valid assumption since the decoders we consider are symmetric [19].

We remind that a trapping set (TS) denoted by $\mathbf{T}(\mathbf{y})$ (as originally defined in [44]) is a non-empty set of variable nodes that are not eventually corrected for a given decoder input $\mathbf{y}$. If $\mathbf{T}(\mathbf{y})$ is empty, then the decoding is successful. Note that $\mathbf{T}(\mathbf{y})$ will depend on the number of decoding iterations. A common notation used to denote a TS is (a, b), where $a = |\mathbf{T}(\mathbf{y})|$, and b is the number of odd-degree check nodes in the subgraph induced by $\mathbf{T}(\mathbf{y})$.

Let $\mathcal{T}(a, b)$ denote the topology associated with a (a, b) TS, whose induced subgraph contains a variable nodes and b odd-degree check nodes. This standard notation is however not sufficient to describe in details the topologies which are the supports of the TSs. In particular, for large a there could be several different topologies with the same values for a and b. To circumvent this problem, we propose to extend the notation of TS by adding to the first two parameters, an additional topological information which allows to distinguish between different structures with the same a and b. We propose the following notation:

$$\mathcal{T}(a, b)\ n_{c_3} - n_{c_4} - n_{c_5} - n_{c_6} - n_{c_7} - n_{c_8} - \ldots - n_{c_a}$$

where n_{c_k} represents the number of cycles containing k variable nodes. This new notation will be used only when necessary. For example the Fig. 2.1 shows the two different topologies for a $(6, 4)$ TS (considering TSs for girth-eight and column-weight-



(a) $\mathcal{T}(6,4)$ $0-1-2-0$. (b) $\mathcal{T}(6,4)$ $0-2-0-1$.

FIGURE 2.1. The two different structures for a $(6,4)$ TS.

three codes) which differs only in the cycles contained in these two structures. In this figure, ● denotes a variable node, □ denotes a degree-two check node and ■ denotes a degree-one check node.

A TS is said to be elementary if $\mathcal{T}$ contains only degree-one or/and degree-two check nodes. Throughout this chapter, we restrict our focus to elementary TSs, since they are known to be dominant in the error floor [44, 46]. Also, whenever we refer to a TS, we will implicitly refer to its topological structure $\mathcal{T}$.

2.2. Finite alphabet iterative decoders

We now introduce a new type of finite precision decoders which we refer to as FAIDs [63, 64]. An N_s-level FAID denoted by D is defined as a 4-tuple given by $\mathrm{D} = (\mathcal{M}, \mathcal{Y}, \Phi_v, \Phi_c)$. The finite alphabet $\mathcal{M}$ defined as
$\mathcal{M} = \{-L_s, \ldots, -L_2, -L_1, 0, L_1, L_2, \ldots, L_s\}$, where $L_i \in \mathbb{R}^+$ and $L_i > L_j$ for any $i > j$, consists of $N_s = 2s + 1$ levels for which the message values are confined to. The sign of a message $x \in \mathcal{M}$ can be interpreted as the estimate of the bit associated with the variable node for which x is being passed to or from (positive for zero and negative for one) and the magnitude $|x|$ as a measure of how reliable this value is.

The set $\mathcal{Y}$ denotes the set of all possible channel values. For FAIDs over the BSC, $\mathcal{Y}$ is defined as $\mathcal{Y} = \{\pm \mathrm{C}\}$, where $\mathrm{C} \in \mathbb{R}^+$ and the value $y_i \in \mathcal{Y}$ for node v_i

is determined by $y_i = (-1)^{r_i}\mathrm{C}$, i.e., we use the mapping $0 \to \mathrm{C}$ and $1 \to -\mathrm{C}$. Note that for the BP and min-sum algorithms (where the messages are log-likelihoods), the decoder input $\mathbf{y}$ is a real-valued vector ($\mathcal{Y} = \mathbb{R}$). Let $m_1, \cdots, m_{l-1}$ denote the extrinsic incoming messages to a node with degree l.

2.2.1. Definitions of the update functions Φ_v and Φ_c

The function $\Phi_c : \mathcal{M}^{d_c-1} \to \mathcal{M}$ used for update at a check node with degree d_c is defined as

$$\Phi_c(m_1, \ldots, m_{d_c-1}) = \left(\prod_{j=1}^{d_c-1} \mathrm{sgn}(m_j) \right) \min_{j \in \{1,\ldots,d_c-1\}} (|m_j|). \tag{2.1}$$

Note that this is the same function used in the min-sum decoder, hence the novelty in the proposed decoders lies in the definition of the variable node update function Φ_v.

The function $\Phi_v : \mathcal{Y} \times \mathcal{M}^{d_v-1} \to \mathcal{M}$ is the update function used at a variable node with degree d_v, and is defined in closed form as

$$\Phi_v(y_i, m_1, m_2, \cdots, m_{d_v-1}) = Q\left(\sum_{j=1}^{d_v-1} m_j + \omega_i \cdot y_i \right) \tag{2.2}$$

where the function $Q(.)$ is defined below based on a threshold set $\mathscr{T} = \{T_i : 1 \leq i \leq s+1\}$ with $T_i \in \mathbb{R}^+$ and $T_i > T_j$ if $i > j$, and $T_{s+1} = \infty$.

$$Q(x) = \begin{cases} \mathrm{sgn}(x)L_i, & \text{if } \ T_i \leq |x| < T_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

The weight ω_i is computed using a symmetric function $\Omega : \mathcal{M}^{d_v-1} \to \{0, 1\}$. Based on this definition, the function Φ_v can be classified as a linear-threshold (LT) function or a non-linear-threshold (NLT) function. If $\Omega = 1$ (or constant), i.e., if the value of ω_i is always 1 (or constant) for all possible inputs of Ω, then Φ_v is an LT function and a FAID with such a Φ_v is classified as an LT FAID. Else, Φ_v is an NLT function and a FAID with such a Φ_v is an NLT FAID.

Note that for an LT FAID, Φ_v takes a linear combination of its arguments and then applies the function Q to determine its output. Therefore, Φ_v will always output the same value for any possible set of incoming messages for a given y_i, if their sum remains the same. For example, for a node with $d_v = 3$, $\Phi_v(-\mathrm{C}, m_1, m_2) = \Phi_v(-\mathrm{C}, m_3, m_4)$ when $m_1 + m_2 = m_3 + m_4$. This is also a typical property present in existing quantized decoders such as quantized BP and min-sum.

On the other hand, for an NLT FAID, Φ_v takes a non-linear combination of its arguments (due to Ω) before applying the function Q on the result. Therefore, Φ_v can output different values even for distinct sets of incoming messages that have the same sum. For instance, consider a map Φ_v for a node with $d_v = 3$ such that $\Phi_v(-\mathrm{C}, -L_3, L_3) = 0$ and $\Phi_v(-\mathrm{C}, -L_2, L_2) = -L_1$. In this case, the two distinct sets of incoming messages are $\{-L_3, L_3\}$ and $\{-L_2, L_2\}$, and the sums are zero for both the sets. However, Φ_v still gives different outputs for each of the sets namely, 0 and $-L_1$ respectively. Hence these decoders are different from existing quantized message-passing decoders. Note that the function Φ_v satisfies the following two properties.

Property 1 (Property of symmetry)**.** $\Phi_v(y_i, m_1, ..., m_{d_v-1}) = -\Phi_v(-y_i, -m_1, ..., -m_{d_v-1})$.

Property 2 (Property of monotonicity)**.** $\Phi_v(y_i, m_1, \ldots, m_{d_v-1}) \geq \Phi_v(y_i, m'_1, \ldots, m'_{d_v-1})$ $\forall m_j \geq m'_j$, $i \in \{1, \ldots, d_v - 1\}$.

Property 2 ensures that for a given channel value, the output is always non-decreasing with increase in the values of incoming messages. For instance, a map Φ_v where $\Phi_v(-\mathrm{C}, L_1, L_2) = L_2$ and $\Phi_v(-\mathrm{C}, L_2, L_2) = L_1$ is forbidden, since $\Phi_v(-\mathrm{C}, L_2, L_2) \geq \Phi_v(-\mathrm{C}, L_1, L_2)$. This is a typical property also present in existing message-passing decoders.

Note that Property 2 is enforced only with respect to the arguments involving the incoming messages, and not the channel value. In other words, Φ_v does not need to satisfy the constraint $\Phi_v(C, m_1, \ldots, m_{d_v-1}) \geq \Phi_v(-C, m'_1, \ldots, m'_{d_v-1})$. However we

found that FAIDs whose Φ_v is not monotonic on the channel value argument exhibit very poor DE thresholds and therefore poor error-rate performance.

The decision rule used in FAIDs at the end of each iteration, to determine the bit value corresponding to each node v_i, is simply the sign of the sum of all incoming messages plus the channel value y_i (positive implies zero and negative implies one).

It is evident from the definition that Φ_v can be uniquely described either by assigning real values to the elements of $\mathcal{M}$, $\mathscr{T}$ and $\mathcal{Y}$, and defining Ω, or by providing a set of constraints which the assigned values can take. As examples, we provide the closed-form description of Φ_v for a 5-level NLT FAID and a 7-level LT FAID defined for column-weight-three codes.

Example 2 (5-level NLT FAID)**.** *The constraints on the values assigned to elements of* $\mathcal{M}$ *and* $\mathscr{T}$ *that describe this map are:* $\mathrm{C} = L_1$, $L_2 = 3L_1$, $T_1 = L_1$, $T_2 = L_2$, *and the function* Ω *is given by* $\omega_i = \Omega(m_1, m_2) = 1 - \big(\mathrm{U}(m_1) \oplus \mathrm{U}(m_2)\big) \cdot \delta(|m_1| + |m_2| - 2L_2)$ *where* U *is the unit step function and* δ *is the Kronecker delta function.*

Example 3 (7-level LT FAID)**.** *The constraints on the values assigned to elements of* $\mathcal{M}$ *and* $\mathscr{T}$ *that describe this map are:* $L_1 < \mathrm{C} < 2L_1$, $L_2 = 2L_1$, $L_3 = 2L_2 + \mathrm{C}$, *and* $T_1 = L_1$, $T_2 = L_2$, *and* $T_3 = L_3 - C$, *where* $\Omega = 1$.

2.2.2. Describing the maps of Φ_v as arrays

We can alternatively define $\mathcal{M}$ to be $\mathcal{M} = \{M_1, M_2, \cdots, M_{N_s}\}$ where $M_1 = -L_s$, $M_2 = -L_{s-1}, \cdots$, $M_s = -L_1$, $M_{s+1} = 0$, $M_{s+2} = L_2, \cdots$, $M_{N_s} = L_s$. Then, Φ_v can be defined using d_{v-1}-dimensional arrays or look-up tables (LUTs) rather than as closed-form functions, which enables simple implementations and also may be more convenient for decoder selection. For column-weight-three codes, the map specifying Φ_v is a simple two-dimensional array defined by $[l_{i,j}]_{1 \le i \le N_s, 1 \le j \le N_s}$, where $l_{i,j} \in \mathcal{M}$, such that $\Phi_v(-\mathrm{C}, M_i, M_j) = l_{i,j}$ for any $M_i, M_j \in \mathcal{M}$. The values for $\Phi_v(\mathrm{C}, M_i, M_j)$ can be deduced from the symmetry of Φ_v. Table 2.1 shows an example of a Φ_v defined

TABLE 2.1. LUT for Φ_v of a 7-level FAID with $y_i = -\mathrm{C}$ (NLT FAID).

$\boldsymbol{m_1/m_2}$	$\boldsymbol{-L_3}$	$\boldsymbol{-L_2}$	$\boldsymbol{-L_1}$	$\boldsymbol{0}$	$\boldsymbol{+L_1}$	$\boldsymbol{+L_2}$	$\boldsymbol{+L_3}$
$\boldsymbol{-L_3}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$
$\boldsymbol{-L_2}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	$+L_1$
$\boldsymbol{-L_1}$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	$-L_1$	$+L_1$
$\boldsymbol{0}$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	0	0	$+L_1$
$\boldsymbol{+L_1}$	$-L_3$	$-L_2$	$-L_1$	0	0	$+L_1$	$+L_2$
$\boldsymbol{+L_2}$	$-L_3$	$-L_1$	$-L_1$	0	$+L_1$	$+L_1$	$+L_3$
$\boldsymbol{+L_3}$	$-L_1$	$+L_1$	$+L_1$	$+L_1$	$+L_2$	$+L_3$	$+L_3$

TABLE 2.2. LUT for Φ_v of a 5-level FAID with $y_i = -\mathrm{C}$ (NLT FAID).

$\boldsymbol{m_1/m_2}$	$\boldsymbol{-L_2}$	$\boldsymbol{-L_1}$	$\boldsymbol{0}$	$\boldsymbol{+L_1}$	$\boldsymbol{+L_2}$
$\boldsymbol{-L_2}$	$-L_2$	$-L_2$	$-L_2$	$-L_2$	0
$\boldsymbol{-L_1}$	$-L_2$	$-L_2$	$-L_1$	$-L_1$	$+L_1$
$\boldsymbol{0}$	$-L_2$	$-L_1$	$-L_1$	0	$+L_1$
$\boldsymbol{+L_1}$	$-L_2$	$-L_1$	0	$+L_1$	$+L_2$
$\boldsymbol{+L_2}$	0	$+L_1$	$+L_1$	$+L_2$	$+L_2$

as an array for a 7-level FAID when $y_i = -\mathrm{C}$, that have been identified as a NLT FAID.

Fig. 2.2 shows the frame error-rate (FER) performances of the 5-level NLT and 7-level LT FAIDs defined in the two examples, and the 7-level NLT FAID defined by the Table 2.1, along with the floating-point BP and min-sum decoders on the well-known $(155, 64)$ Tanner code [68]. All decoders were run for a maximum of 100 iterations. From the plot, we see that all the FAIDs significantly outperform the floating-point BP and min-sum on the code. We will provide in the sequel of this chapter the methodology used to identify good FAIDs. We also provide in the Table 2.2 and 2.3 the arrays corresponding to the Example 2 and 3, respectively.

TABLE 2.3. LUT for Φ_v of a 7-level FAID with $y_i = -\mathrm{C}$ (LT FAID).

$\boldsymbol{m_1/m_2}$	$\boldsymbol{-L_3}$	$\boldsymbol{-L_2}$	$\boldsymbol{-L_1}$	$\boldsymbol{0}$	$\boldsymbol{+L_1}$	$\boldsymbol{+L_2}$	$\boldsymbol{+L_3}$
$\boldsymbol{-L_3}$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	0
$\boldsymbol{-L_2}$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	$+L_1$
$\boldsymbol{-L_1}$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	0	$+L_2$
0	$-L_3$	$-L_2$	$-L_2$	$-L_1$	0	$+L_1$	$+L_2$
$\boldsymbol{L_1}$	$-L_3$	$-L_2$	$-L_1$	0	0	$+L_1$	$+L_2$
$\boldsymbol{L_2}$	$-L_3$	$-L_1$	0	$+L_1$	$+L_1$	$+L_2$	$+L_3$
$\boldsymbol{L_3}$	0	$+L_1$	$+L_2$	$+L_2$	$+L_2$	$+L_3$	$+L_3$

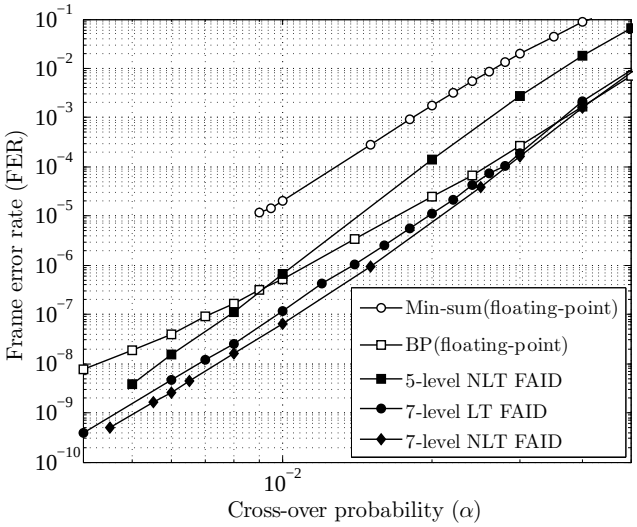


FIGURE 2.2. Performance comparisons between the floating-point decoders: BP and min-sum , and the 3-bit precision decoders: 5-level NLT, 7-level LT, and 7-level NLT FAIDs on the $(155, 64)$ Tanner code.

A natural question that arises at this point is how many FAIDs exist for a given N_s. This can be easily enumerated by establishing a connection between FAIDs and symmetric plane partitions.

2.2.3. Symmetric plane partition representation of Φ_v

A symmetric plane partition π is an array of nonnegative integers $(\pi_{i,j})_{i\geq 1, j\geq 1}$ such that $\pi_{i,j} \geq \pi_{i+1,j}$, $\pi_{i,j} \geq \pi_{i,j+1}$ $\forall i, j \geq 1$, and $\pi_{i,j} = \pi_{j,i}$. If $\pi_{i,j} = 0$ $\forall i > r$, $\forall j > s$, and $\pi_{i,j} \leq t$ $\forall i, j$, then the plane partition is said to be *contained* in a box with side lengths (r, s, t). The value $\pi_{i,j}$ is represented as a box of height $\pi_{i,j}$ positioned at (i, j) coordinate on a horizontal plane.

Due to the properties 1 and 2 of Φ_v, there exists a bijection between the array $[l_{i,j}]_{1\leq i\leq N_s, 1\leq j\leq N_s}$ and a symmetric plane partition contained in a $(N_s \times N_s \times N_s - 1)$ box, where each $\pi_{i,j}$ is determined based on $l_{i,j}$. Fig. 2.3 shows the visualization of a plane partition corresponding to Φ_v of the 7-level FAID defined in Table 2.1.

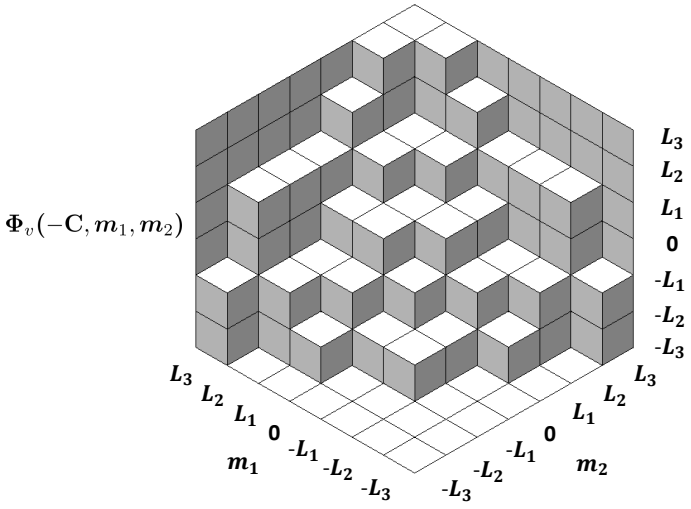


FIGURE 2.3. A visualization of the plane partition as stacked boxes for the 7-level FAID whose Φ_v is described in Table 2.1.

Kuperberg in [65] gave an elegant formula for the enumeration of symmetric plane partitions contained in a box, and we can directly utilize this for the enumeration of N_s-level FAIDs as well.

Theorem 2 (Number of N_s-level FAID)**.** *The total number $K_A(N_s)$ of N_s-level FAIDs is given by*

$$K_A(N_s) = \frac{H_2(3N_s)H_1(N_s)H_2(N_s-1)}{H_2(2N_s+1)H_1(2N_s-1)}$$

where $H_k(n) = (n-k)!\,(n-2k)!\,(n-3k)!\ldots$ is the staggered hyperfactorial function.

Proof. The proof of the theorem follows from the bijection between the map Φ_v of a FAID and a symmetric boxed plane partition. □

Using this formula, the total number of FAIDs for $N_s = 5$ and $N_s = 7$ levels are 28,314 and 530,803,988 respectively.

2.3. Selection of finite alphabet iterative decoders

It is clear from the previous section that identifying particularly good FAIDs from the set of all possible FAIDs is highly non-trivial since the number of such FAIDs is very large. Moreover not all the FAIDs are performing well, as an example we can think of the classic min-sum decoder which obviously belong to the FAID class and which does not have good performance in the error floor region.

As a first attempt of selection, one can think to rely on the computation of the density evolution (DE) to determine the decoding threshold (α^*) of each FAIDs on a family of LDPC codes over the BSC. In Fig. 2.4 we present the performance comparison on the Tanner code between the floating-point BP, the 5-level FAID (Table 2.2), the 7-level FAID (Table 2.3), and the 5-level and 7-level FAIDs which have the best DE thresholds. These last ones are not the best decoders, in fact they are presenting a pretty high error-floor. The DE threshold have been computed on the class of ($d_v = 3$,$d_c = 5$) LDPC codes, class which includes the Tanner code. The thresholds of the different FAIDs are given as follows: 5-level FAID of the Table 2.2: $\alpha^* = 0.09781$, 7-level FAID of Table 2.3: $\alpha^* = 0.010155$, best threshold 5-level FAID: $\alpha^* = 0.10319$ and best threshold 7-level FAID: $\alpha^* = 0.010349$. We can then conclude, that the selection of FAIDs only based on the DE threshold is not efficient, hence the need of other techniques, presented in the sequel of this section.

We now describe a general approach that can be used to identify a subset of candidate N_s-level FAIDs, one or several of which are potentially good for any column-weight-three code. Our main aim behind this approach is to restrict the choice of FAIDs to a possibly small subset containing good candidates. Given a particular code, it would then be feasible to identify the best performing FAID from this subset by using brute-force simulation or emulation or some other technique on the code. Moreover, since the performance of a FAID on a given code depends on its structure, the goal of identifying several candidate FAIDs is more realistic than identifying a

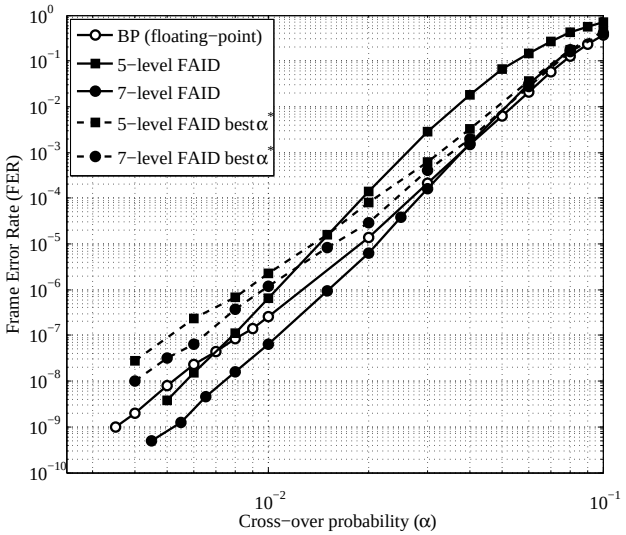


FIGURE 2.4. Performance comparisons between the floating-point BP, the 5-level FAID from the Table 2.2, the 7-level FAID from the Table 2.3, and 5-level and 7-level FAID exhibiting the best DE thresholds α^* on the Tanner code.

single good FAID, and allows for devising a selection method that is not code-specific. Another important objective of our approach is to ensure that any FAID belonging to this subset is capable of surpassing BP in the error floor not just on a single code but on several codes.

The approach we use relies on the knowledge of potentially harmful subgraphs that could be TSs for traditional iterative decoders when present in a given code. The candidate FAIDs are chosen by analyzing their behavior on each of these subgraphs with errors introduced in them. We will first introduce some important notions that form the basis of our approach, then we present a first approach of selection based on Monte-Carlo simulations on TSs of large sizes, and then present the proposed methodology for FAID selection for column-weight-three codes.

2.3.1. Critical number and isolation assumption

The notion of *critical number* associated with a TS of type $\mathcal{T}(a, b)$ was originally introduced for Gallager-A/B algorithms on the BSC [66]. It is computed by analyzing

the Gallager-A/B decoding on errors contained in the topology $\mathcal{T}$ that is present in a code, assuming that all nodes outside the topology are initially correct. It provides a measure of how harmful a TS is, and hence, this notion is not only useful for predicting the error floor performance [46] but also for determining the harmful subgraphs that should be avoided in the code design.

In order to be able to extend the notion of critical number for FAIDs, we introduce the notion of *isolation assumption* [67] which is used to analyze the decoder on a potential $\mathcal{T}(a, b)$ TS. Under this assumption, the neighborhood of the TS is such that the messages flowing into the TS from its neighborhood are not in any way influenced by the messages flowing out of the TS. Therefore, the messages flowing into the TS can be computed while completely disregarding the neighborhood [67, Theorem 1]. We now precisely define this notion.

Let $\mathscr{T}_i^k(G)$ denote the computation tree corresponding to an iterative decoder on G enumerated for k iterations with node $v_i \in V$ as its root. A node $w \in \mathscr{T}_i^k(G)$ is a *descendant* of a node $u \in \mathscr{T}_i^k(G)$ if there exists a path starting from node w to root v_i that traverses through node u.

Definition 7 (Isolation assumption)**.** *Let H be a subgraph of G induced by $P \subseteq V$ with check node set $W \subseteq C$. The computation tree $\mathscr{T}_i^k(G)$ with the root $v_i \in P$ is said to be isolated if for any node $u \notin P \cup W$ in $\mathscr{T}_i^k(G)$, u does not have any descendant belonging to $P \cup W$. If $\mathscr{T}_i^k(G)$ is isolated $\forall v_i \in P$, then H is said to satisfy the isolation assumption in G for k iterations.*

The *critical number* can now be defined in the framework of FAIDs.

Definition 8. *The critical number of a FAID* D *on a $\mathcal{T}(a, b)$ is the minimum number of variable nodes in $\mathcal{T}$ that have to be initially in error for* D *to fail under the isolation assumption.*

We set the critical number to ∞ if D corrects all possible error patterns on H.

The critical number can now be used as possible parameter for decoder selection where a decoder is chosen to maximize the critical number on a given TS or set of TSs. In principle, one could consider a database of potential TSs that are generated either through analytical or empirical evaluations of traditional decoders such as BP and min-sum on several different codes, and then select a FAID based on its critical numbers on all these TSs. The critical number is then supposed to be more representative than the TS itself to measure its impact in the error floor region. For example, a $(7, 3)$ TS with critical number 4 is supposed to have a larger contribution to the error floor than a $(5, 3)$ TS with critical number 5.

However, the isolation assumption of a TS typically does not hold in an actual code for more than few iterations and hence the critical number may not reflect the true error-correction capability of the FAID on a code containing the TS. This is especially true for TSs of small sizes where the influence of the neighborhood can play a greater role. Therefore, unless a very large database of TSs is considered or unless TSs with large sizes are considered such that isolation assumption holds for many more iterations , the strategy will remain ineffective. This motivates the need for a new framework that considers to an extent the influence of the neighborhood.

Before presenting the main tool that will be useful in general in the selection of good FAIDs, the next section provides a first approach based on large size TSs in order to identify the best FAID on a given code based on Monte Carlo simulations. These large TSs are taken to be the codewords of the given code known to be attractor of iterative decoding. We present the developed methodology throughout the example of the Tanner code, but it can be applied to any code, provided the distribution of codewords is known.

2.3.2. Investigation of the (155,64) Tanner code

As preliminaries, we briefly review the construction and the details regarding the Tanner code, before presenting in the next few sections a methodology to identify the best FAID on this particular code.

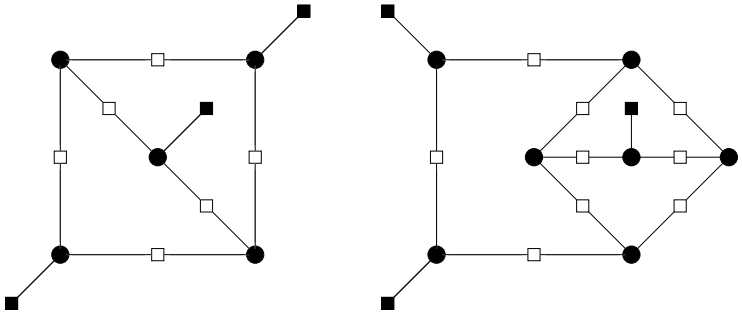
The (155,64) Tanner code has been constructed in [68] and we remind here the principle of its construction. The parity-check matrix of the Tanner code is made of $d_v \times d_c$ block of permutation matrices. Each permutation matrix is an $p \times p$ matrix obtained by circularly shifting the identity matrix of size p. For this code the chosen parameters are $m = 31$, $d_v = 3$ and $d_c = 5$. We denote here I_j the matrix obtained by circularly shifting the identity matrix j times. The LDPC matrix obtained in [68] is given by :

$$H_{[155,64]} = \begin{bmatrix} I_1 & I_2 & I_4 & I_8 & I_{16} \\ I_5 & I_{10} & I_{20} & I_9 & I_{18} \\ I_{25} & I_{19} & I_7 & I_{14} & I_{28} \end{bmatrix}_{(93\times155)}$$

We refer to [68] for a complete explanation on the design of the parity-check matrix. The girth of the Tanner graph associated with the matrix is 8, and the resulting code has a minimal distance $d_{min} = 20$ which makes this code pretty attractive given its relative short length. The Tanner code belongs to the class of quasi-cyclic LDPC code, hence its the minimum distance is upper-bounded by $(d_v + 1)!$ [26]. In the case of the column-weight-three codes, the largest possible minimum distance is then 24.

2.3.2.1. Trapping sets distributions

We now provide the details of the topologies present in the Tanner code which are supposed to be dominant in the error floor region. This topologies are in fact structures with small number of bits involved in it. Using expansion of the neighboring tree from each variable node, it is quite easy to derive an algorithm which detects and counts the small closed topologies, and therefore TSs, in a graph. Different algorithms have been suggested to detect TSs [56, 69]. We have reported on Table 2.4



(a) $\mathcal{T}(5,3)$ (b) $\mathcal{T}(7,3)$

FIGURE 2.5. Two TSs present in the Tanner code

the distribution of TSs up to $a = 7$ bits which are present in the Tanner Code. Two examples of these TSs are drawn on Fig. 2.5(a) and Fig. 2.5(b).

The $\mathcal{T}(5,3)$ is the corner point of the weakness of iterative decoders on the Tanner code. This *very small* TS makes several iterative decoders fail when the bits in error are located on the 5 bits which compose the TS which is very far from the ML decoding performance which can correct up to 9 errors.

2.3.2.2. Minimal codeword structures

The knowledge of the dominant TSs could be sufficient to predict the behavior of usual decoders in the error floor region [44, 46]. However, as demonstrated in the next section, when the iterative decoder is more general, which is the case of the FAID decoders, looking at the TS alone is not sufficient. Naturally one can think to consider bigger structures which are also attractor points of the decoders. Instead of considering larger and larger TSs, we propose to study the behavior of the decoders on

TABLE 2.4. Trapping Set spectrum of the Tanner Code.

(155,64,20) Tanner code		
$\mathcal{T}(5,3)$ $0-3-0-0-0-0$	$\rightarrow$	**155**
$\mathcal{T}(6,4)$ $0-1-2-0-0-0$	$\rightarrow$	**930**
$\mathcal{T}(7,3)$ $0-3-2-0-2-0$	$\rightarrow$	**930**
$\mathcal{T}(7,5)$ $0-1-1-0-1-0$	$\rightarrow$	**11160**
$\mathcal{T}(7,5)$ $0-1-0-2-0-0$	$\rightarrow$	**2790**

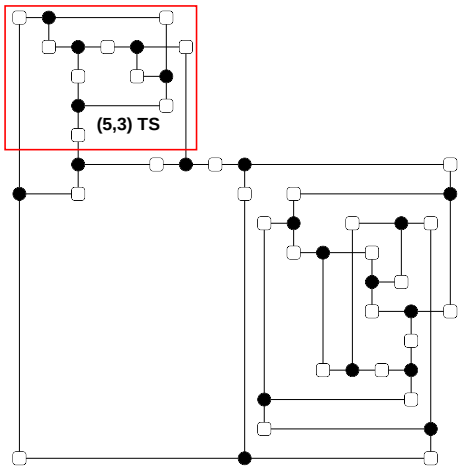
the closed structures formed by the codewords of the Tanner code. The main reason is that the multiplicity of TS with constant $b > 1$ becomes rapidly cumbersome with increasing a. Looking at error events located inside a codeword should give a lot of information about iterative decoding convergence points, although it has not been proven yet that at least one TS is necessarily nested in a codeword.

The Hamming distance spectrum of the Tanner code is given on Table 2.5. This spectrum has been obtained with the impulse algorithm presented in [70]. We will focus on the minimum codewords of weight $d_{min} = 20$ for the sake of simplicity of the analysis. Note that these minimal codewords are actually $\mathcal{T}(20, 0)$.

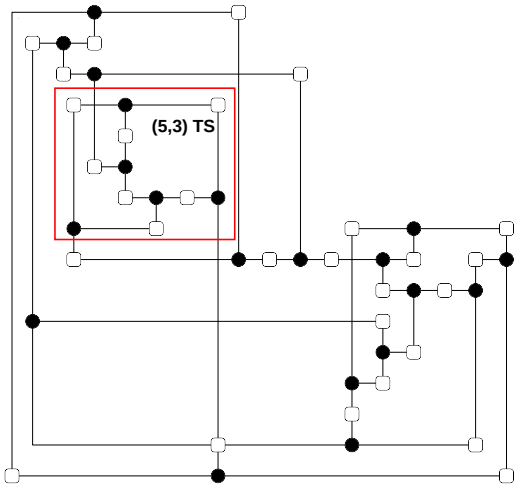
By analysis of the topologies of these codewords, we have identified that there are only 3 types of structures for the minimal codewords, which we will denote Type-I, Type-II and Type-III. This means that each and every codeword of weight 20 belongs to one of the automorphism group of the subgraph induced by one of the 3 types of codewords. This is especially interesting since we can restrict the study of the decoders on 3 subgraphs instead of 1023 subgraphs. Another observation is that only 2 out of the 3 types contain the minimal TS, *i.e.* the $\mathcal{T}(5, 3)$. Type-I codewords contain only degree-two check nodes, and they are composed of two $\mathcal{T}(10, 2)$, Type-II codewords contain one degree-four check node, and the minimal TS in the Type-III codewords contains is the $\mathcal{T}(6, 4)$. We have drawn on the Fig. 2.6(a), 2.6(b) and 2.6(c) the structure of Type-I, Type-II and Type-III codewords, respectively. We emphasized the $\mathcal{T}(5, 3)$ on the Fig. 2.6(a) and 2.6(b).

TABLE 2.5. Codeword distribution of the Tanner Code.

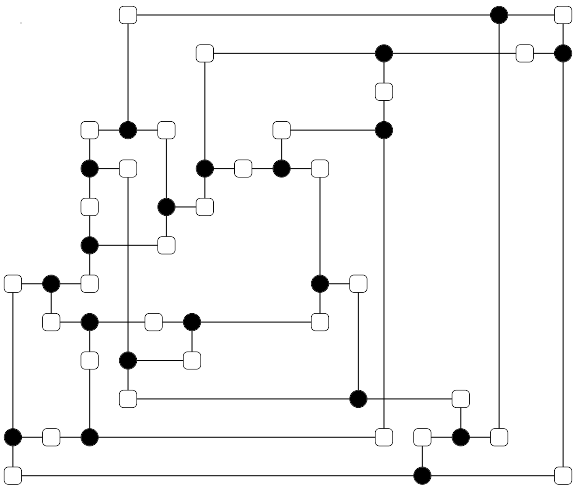
(155,64,20) Tanner code		
weight 20	$\rightarrow$	**1023**
weight 22	$\rightarrow$	**6200**
weight 24	$\rightarrow$	**43865**
weight 26	$\rightarrow$	**259918**



(a) Type-I



(b) Type-II



(c) Type-III

FIGURE 2.6. Topological structures of the minimal codewords of the Tanner code

2.3.2.3. Limitations of predicting decoder behavior based on minimal trapping sets

We have computed and indicated on the Table 2.8 the critical numbers of 4 different FAIDs, and for all TSs present in the Tanner code up to 8 bits. The 4 FAIDs considered are the 5-level FAID of the Table 2.2 denoted D_1, the 5-level FAIDs of the Table 2.6 and 2.7 denoted D_2 and D_3, respectively, and the 7-level FAID of the Table 2.3, denoted D_4.

The $\mathcal{T}(8,2)$ (displayed later in this chapter in the Fig. 2.9(b)) appears to be the most difficult TS to cope with for all decoders. Note that all 4 FAIDs have infinite critical number on the two smallest TSs, namely the $\mathcal{T}(5,3)$ and the $\mathcal{T}(6,4)$, which seems to indicate that it is possible to derive finite precision decoders, even with very few quantization bits, which are not trapped by the TSs of traditional decoders (Gallager-B, min-sum).

These critical numbers are however not predictive at all when the isolation assumption is not satisfied. We have verified some error-correction properties with extensive Monte Carlo simulations on the whole Tanner code. Although the decoder D_3 has the exact same statistics as D_2 and even better statistics than D_1 in terms of critical numbers, D_3 fails on 110 five-error patterns when we simulate the rule on the whole Tanner code, while D_1 and D_2 correct all five-errors patterns in less than 100 iterations. Another contradiction is that the critical number for D_2 on $\mathcal{T}(8,2)$ is 5, which means that there are five-error patterns such that decoder D_2 fails in an isolated way, but D_2 successfully corrects the five errors when the $\mathcal{T}(8,2)$ is simu-

TABLE 2.6. LUT for Φ_v of the 5-level FAID D_2 with $y_i = -\mathrm{C}$.

$\boldsymbol{m_1 \backslash m_2}$	$\boldsymbol{-L_2}$	$\boldsymbol{-L_1}$	$\boldsymbol{0}$	$\boldsymbol{+L_1}$	$\boldsymbol{+L_2}$
$\boldsymbol{-L_2}$	$-L_2$	$-L_2$	$-L_2$	$-L_2$	0
$\boldsymbol{-L_1}$	$-L_2$	$-L_2$	$-L_1$	$-L_1$	$+L_1$
$\boldsymbol{0}$	$-L_2$	$-L_1$	$-L_1$	0	$+L_2$
$\boldsymbol{+L_1}$	$-L_2$	$-L_1$	0	$+L_1$	$+L_2$
$\boldsymbol{+L_2}$	0	$+L_1$	$+L_2$	$+L_2$	$+L_2$

TABLE 2.7. LUT for Φ_v of the 5-level FAID D_3 with $y_i = -C$.

$\boldsymbol{m_1 \backslash m_2}$	$\boldsymbol{-L_2}$	$\boldsymbol{-L_1}$	$\boldsymbol{0}$	$\boldsymbol{+L_1}$	$\boldsymbol{+L_2}$
$\boldsymbol{-L_2}$	$-L_2$	$-L_2$	$-L_2$	$-L_2$	0
$\boldsymbol{-L_1}$	$-L_2$	$-L_1$	$-L_1$	$-L_1$	$+L_2$
$\boldsymbol{0}$	$-L_2$	$-L_1$	$-L_1$	0	$+L_2$
$\boldsymbol{+L_1}$	$-L_2$	$-L_1$	0	$+L_2$	$+L_2$
$\boldsymbol{+L_2}$	0	$+L_2$	$+L_2$	$+L_2$	$+L_2$

TABLE 2.8. Critical numbers on the TSs of the Tanner code for the selected decoders. The DE threshold of each rule is also given.

Trapping Set Label	D_1	D_2	D_3	D_4
$\mathcal{T}(5,3)\ 0-3-0-0-0-0$	∞	∞	∞	∞
$\mathcal{T}(6,4)\ 0-1-2-0-0-0$	∞	∞	∞	∞
$\mathcal{T}(7,3)\ 0-3-2-0-2-0$	7	∞	∞	6
$\mathcal{T}(7,5)\ 0-1-1-0-1-0$	∞	∞	∞	∞
$\mathcal{T}(7,5)\ 0-1-0-2-0-0$	∞	∞	∞	∞
$\mathcal{T}(8,2)\ 0-3-4-2-4-2$	6	5	5	6
$\mathcal{T}(8,4)\ 0-3-0-2-0-2$	∞	∞	∞	6
$\mathcal{T}(8,4)\ 0-1-3-1-1-1$	∞	∞	∞	7
$\mathcal{T}(8,4)\ 0-1-2-2-2-0$	∞	∞	∞	7
$\mathcal{T}(8,6)\ 0-1-0-1-0-1$	∞	∞	∞	∞
$\mathcal{T}(8,6)\ 0-1-0-0-2-0$	∞	∞	∞	∞
Decoding threshold α^*	0.09781	0.09778	0.09777	0.10155

lated in the whole Tanner code. As we can see, contradictions in the analysis of the isolated critical numbers are in both positive and negative directions, which makes them difficult to use in the goal of choosing a good decoder for a particular code.

Fig. 2.7 presents the frame error rate (FER) performance of the FAIDs D_1 and D_3 on the Tanner code. All curves have been plotted with a maximum of 100 iterations, and at least 300 frame errors have been recorded for each simulated cross-over probability. The difference between the two rules is not large, although the curves start to split apart in the error floor region due to the fact that D_1 corrects all five-error patterns while D_1 does not. As a catastrophic counter-example of using only isolated critical numbers for designing FAIDs, we have also plotted the performance of a FAID which have *all its isolated critical numbers* equal to ∞ (labeled as 'Bad Rule' in Fig. 2.7). The DE for this rule is only $\alpha^* = 0.07778$, and then is a lot worse than the thresholds of the FAIDs considered.

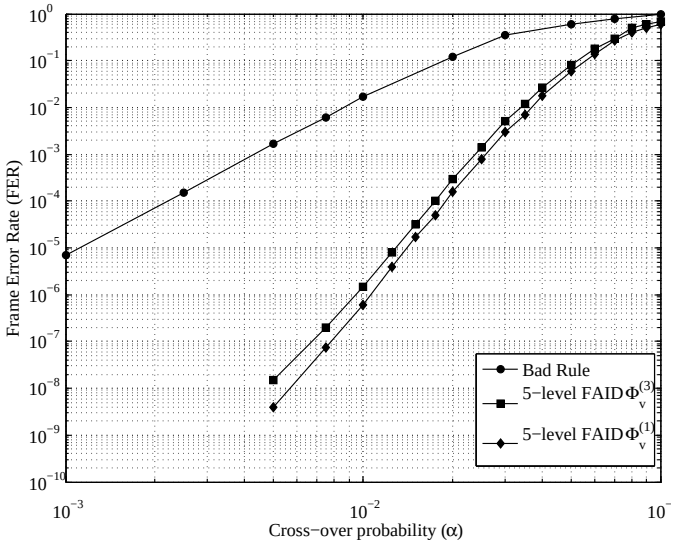


FIGURE 2.7. Performance comparisons of different FAIDs on the Tanner Code

Of course, brute force simulations on the whole code would give the desired ordering between rules, but at the price of a too large computational burden. The problem of finding the best decoding rules for a specific code cannot be solved with the knowledge of critical numbers alone, and remains an open issue. In the following, we propose an approach to partially solve this issue which is still based on Monte Carlo simulations, but on larger structures than the smallest TSs.

2.3.2.4. Selection of FAIDs by Monte-Carlo simulations

We propose to make the selection of the best FAID with respect to the ordering between FAIDs, by simulations of n_e-errors patterns on the subgraphs induced by the minimal codewords of weight 20. Although we do not claim that simulation results on these subgraphs are strictly predictive, looking at the codeword structures makes sense with respect to the isolation assumption described in the preceding section. Indeed a codeword is a particular $\mathcal{T}(a, 0)$, and then is connected to the rest of the Tanner graph only by even-degree check nodes. More importantly, no edge connects the codeword to the rest of the graph via odd-degree check nodes. From our observations on TSs,

TABLE 2.9. Low-weight error-event correction on the codewords.

	5-errors patterns			6-errors patterns		
	D_1	D_2	D_3	D_1	D_2	D_3
type-I	3	2	> 10	172	138	> 500
type-II	$0^{(7)}$	$0^{(8)}$	$0^{(9)}$	$0^{(16)}$	$0^{(21)}$	> 21
type-III	$0^{(4)}$	$0^{(4)}$	$0^{(4)}$	$0^{(4)}$	$0^{(4)}$	$0^{(5)}$

the isolation assumption is more often *"violated"* when the external paths go through an odd-degree check node than when they go only through even-degree check nodes of the TS. It seems that codewords are *almost isolated*, at least during a number of iterations more important than smaller types of TSs (non-codewords).

We have simulated all 5-errors patterns and all 6-errors patterns on the three types of codewords, for a large number of FAIDs. We report on Table 2.9 the results for the 5-level FAIDs of Tables 2.2, 2.6 and 2.7. The numbers in Table 2.9 indicate the number of error patterns which are *not* corrected by the decoders, and in the case all error events are corrected, we indicate in brackets the maximum number of iterations needed to correct all events.

Although it has not been developed in this work, we remark that these numbers can be seen as the *strength* of the critical numbers. For instance we know from Table 2.8 that the FAIDs D_2 and D_3 have a critical number 5 on the $\mathcal{T}(8, 2)$, however D_2 might fail on many 5-error events into the $\mathcal{T}(8, 2)$, whereas D_3 might fail on only one 5-error event. This could give more information in order to compare the error-correction capability between two FAIDs. Another parameters could also consider the convergence speed when the critical numbers are infinite. However we did not used these idea for the design and selection of FAIDs.

As a first observation, we can see that the three types of codewords have completely different behaviors. Type-I codewords seem to be the most problematic ones, and Type-III codewords the easiest to decode. Remember that Type-III codewords do not contain $\mathcal{T}(5, 3)$, which could explain why they have the best behaviors with

iterative decoding. This is a very interesting differentiation of codewords which have although the same Hamming weight, and therefore cannot be differentiated with Maximum Likelihood Decoding.

Regarding the ordering of the different FAIDs, these statistics are in better accordance with the simulations on the whole Tanner code than the critical numbers of Table 2.8. It is readily seen on these statistics that rule D_3 is worse than rules D_1 and D_2. Since we verified that rule D_3 does not correct all five-error patterns on the Tanner code while D_1 and D_2 do, we can see that the ordering of rules made with simulations on the codewords is somewhat more predictive than the isolated critical numbers. A more important result is that we performed those statistics for all possible 5-level decoders (we remind that there are 28,314 possible FAIDs with $N_s = 5$), and rules D_1 and D_2 have the best overall statistics of all decoders. Since this approach of simulating error patterns on codewords appears to be predictive, we conjecture that we have found the best 5-level FAIDs for the (155,64,20) Tanner code, that is decoders D_1 and D_2 for which the FER curves are very close. As we already reported in Fig. 2.2 where we remind that the D_1 is the 5-level FAID, and D_4 being the 7-level FAID, both FAIDs D_1 and D_4 surpass the BP decoder in the error floor region, as expected with our conjectures from the minimal codewords in the Tanner code. We presented this approach using $N_s = 5$ levels, the extension to higher number of levels is straightforward.

To conclude this preliminary part on the selection we should note that the selection of the best FAIDs on the Tanner code was a possible task to perform due to its short length, and then its easiness to compute the contained TSs and the minimal codewords. For longer codes, this methodology could not be tractable due to the known complexity of searching for minimal codewords. We also notice that it is still an open issue to ensure that there is only a finite number of isomorphic topologies for minimal-weight codewords for any codes. Indeed, for the Tanner code we were able to list all codewords of minimal-weight and report the three different minimal-weight

codewords, but it seems way more difficult for other codes, hence we need another tool to select FAIDs that rely only on the knowledge of dominant TSs. This other tool needs the definition of other concepts presented in the next section and which constitutes the main contribution of this dissertation in this topic.

2.3.3. Noisy trapping sets and noisy critical number vector

Consider the analysis of a FAID D on $\mathcal{T}(a,b)$ with errors introduced in $\mathcal{T}$ (by setting variable nodes in $\mathcal{T}$ to be initially in error). In order to at least partially capture the influence of an arbitrary (unknown) neighborhood of $\mathcal{T}$, we do the following. We first carry out message-passing on $\mathcal{T}$ under the isolation assumption for only two iterations (which is satisfied since we focus on girth-eight graphs). Under this assumption, the message passed by a degree-one check in the first iteration is $\mu = \Phi_v(C, 0, 0)$, and in the second iteration is $\Phi_v(C, \mu, \mu)$ [67, Theorem 1]. From the third iteration (which is the point when the isolation assumption may not hold), we use the notion of initialization vector defined below. Let N_I denote the maximum number of iterations allowed for message-passing under a particular D on $\mathcal{T}(a,b)$.

Definition 9. *An initialization vector on $\mathcal{T}(a,b)$ is defined as a vector $\mathbf{\Theta} = (\theta_1, ..., \theta_b)$ where $\theta_i \in \mathcal{M}$, such that during the message-passing of a FAID on $\mathcal{T}$, θ_i is the message passed by the i^{th} degree-one check node from the third iteration up to N_I iterations. $\mathcal{T}(a,b)$ is said to be initialized by such a vector and is referred to as a noisy trapping set.*

Note that $\mathcal{T}(a,b)$ is initialized by vector $\mathbf{\Theta}$ only after two iterations of message-passing under the isolation assumption, and it is carried out only through the degree-one check nodes. Also note that the initialization vector is not iteration dependent.

As a first example, Fig. 2.8 depicts how a FAID is analyzed for a three-error pattern on a $\mathcal{T}(6,2)$ initialized by a vector $\mathbf{\Theta} = (\theta_1, \theta_2)$. In this figure, ● denotes a variable node initially in error (v_1, v_2, and v_4) and ○ denotes a node initially correct

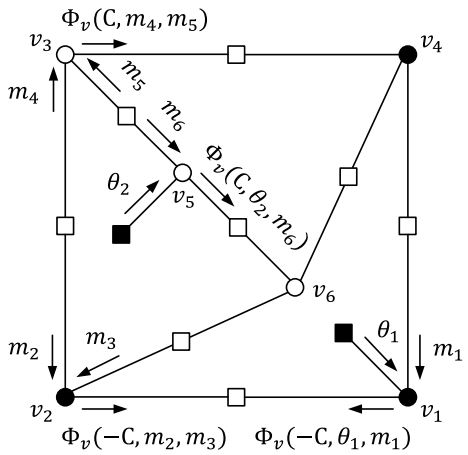
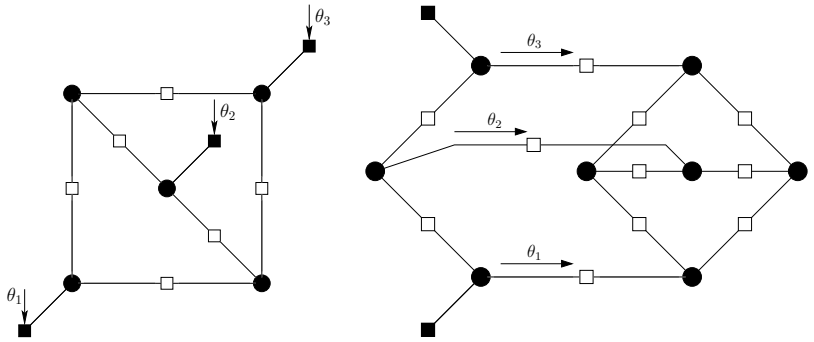


FIGURE 2.8. An example of a noisy $\mathcal{T}(6,2)$ initialized by a vector $\mathbf{\Theta}$.

(v_3, v_5, and v_6). We remind that a □ denotes a degree-two check node and a ■ denotes a degree-one check node. Initially all messages are set to zero. Then the messages are iteratively updated using the maps Φ_v and Φ_c by treating the subgraph $\mathcal{T}$ in an isolated manner as if it were the Tanner graph of a code but with the exception that a degree-one check node sends to its neighbor $\mu = \Phi_v(C, 0, 0)$ in the first iteration, $\Phi_v(C, \mu, \mu)$ in the second iteration, and then θ_1 (or θ_2) for the remaining iterations. The message update on a single edge in the third iteration from a variable node is shown for each of the nodes v_1, v_2, v_3 , and v_5 (v_4 and v_6 are similar to v_2 and v_3 respectively). Note that the messages $m_1, m_2, \ldots, m_6$ denote the extrinsic incoming messages to these nodes.

As a second example Fig. 2.9(a) illustrates the initialization of a $\mathcal{T}(5,3)$ with an initialization vector $\mathbf{\Theta} = (\theta_1, \theta_2, \theta_3)$. In order to demonstrate how the initialization can mimic the influence of the neighborhood, consider the Fig. 2.9(b) where a $\mathcal{T}(8,2)$ is formed by "adding" three variable nodes to $\mathcal{T}(5,3)$. The initialization vector of the $\mathcal{T}(5,3)$ simulates the 3 possible messages that could enter $\mathcal{T}(5,3)$ due to the influence of its neighborhood defined by $\mathcal{T}(8,2)$. Tables 2.10(a) and 2.10(b) show the observations of the evolution of messages θ_1, θ_2, and θ_3 for the first 10 decoding iterations when the 5-level FAID of Table 2.2 was employed on the Tanner code.



(a) (b)

FIGURE 2.9. (a) Noisy $\mathcal{T}(5,3)$. (b) $\mathcal{T}(5,3)$ + 3 variable nodes = $\mathcal{T}(8,2)$.

TABLE 2.10. Evolution of the messages entering a $\mathcal{T}(5,3)$ part of a $\mathcal{T}(8,2)$ in the Tanner code.

(a) Case 1

Iter.	θ_1	θ_2	θ_3
1	$+L_1$	$+L_1$	$+L_1$
2	$+L_2$	$+L_2$	$+L_2$
3	$+L_2$	$+L_2$	$+L_2$
4	$+L_1$	$+L_1$	$+L_1$
5	0	0	0
6	$+L_1$	$+L_1$	$+L_1$
7	0	0	0
8	$+L_1$	$+L_1$	$+L_1$
9	$+L_1$	$+L_1$	$+L_1$
10	$+L_1$	$+L_1$	$+L_1$

(b) Case 2

Iter.	θ_1	θ_2	θ_3
1	$+L_1$	$+L_1$	$+L_1$
2	$+L_2$	$+L_2$	$+L_1$
3	$+L_2$	$+L_1$	$+L_2$
4	$+L_2$	$-L_1$	$+L_1$
5	0	$+L_1$	0
6	$+L_2$	0	$+L_2$
7	$+L_2$	$+L_1$	$+L_1$
8	$+L_1$	0	$+L_1$
9	$+L_1$	$+L_1$	$+L_1$
10	$+L_1$	0	$-L_1$

Table 2.10(a) corresponds to the case where all 5 errors are introduced on $\mathcal{T}(5,3)$ contained in $\mathcal{T}(8,2)$. Table 2.10(b) corresponds to the case of adding an extra 6^{th} error in the neighborhood of $\mathcal{T}(5,3)$. From the Tables, we can see that not only does the isolation assumption not hold as messages do not remain $+L_2$ after iteration 3, but at certain iterations even messages such as 0 and $-L_1$ begin to enter the $\mathcal{T}(5,3)$ due to the neighborhood.

We can now examine whether an error pattern is corrected by the FAID within N_I iterations under a given initialization vector on the $\mathcal{T}(a,b)$. Our main intuition for defining the notion of noisy trapping sets is as follows. Let us consider a code whose graph G contains a subgraph H that is isomorphic to the topology $\mathcal{T}(a,b)$. Assume that a particular FAID is being used for decoding an error pattern where some (or all)

of the variable nodes in H are initially in error and the nodes outside H are initially correct. For the first two decoding iterations, we know that the messages passed into the nodes in H will be the same as messages passed under the isolation assumption due to G satisfying it for two iterations. However from the third iteration, different possible messages belonging to $\mathcal{M}$ will be passed into the nodes of H from outside of H depending on its neighborhood. The initialization vector can be considered as a possible snapshot of the messages entering H through its degree-one check nodes in some arbitrary iteration after the second one, and different initializations represent the different possible influences that the neighborhood of H can have. Therefore, analyzing the FAID under different initializations on a given $\mathcal{T}$ can provide a good indication of its error correction capability on a code whose graph contains H.

Although the initialization vector should ideally be iteration-dependent and include all messages passed to all check nodes of $\mathcal{T}(a,b)$ from outside of $\mathcal{T}(a,b)$, this would make analyzing a FAID on $\mathcal{T}(a,b)$ computationally intractable. We therefore only include constant values that are passed to degree-one check nodes into the initialization vector. Let $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_{N_\Theta}\}$ denote the set of all possible initialization vectors on $\mathcal{T}(a,b)$, where $N_\Theta = |\mathcal{M}|^b$. We now define the notion of *noisy critical number vector* for FAIDs.

Definition 10. *The noisy critical number vector (NCNV) of a FAID* D *on* $\mathcal{T}(a,b)$ *is defined as* $\mathbf{u}_{\mathrm{D}}(\mathcal{T}(a,b), N_I) = (\zeta_1, \zeta_2, \ldots, \zeta_{N_\Theta})$ *where* ζ_i *is the minimum number of variable nodes that have to be initially in error for* D *to fail after* N_I *iterations under initialization vector* $\boldsymbol{\Theta}_i \in \mathcal{M}^b$.

The NCNV of a FAID can now be computed for different TSs (note: for ensuring that all NCNVs have the same dimension, only TSs with same b are considered). However, in order to be used as a parameter for decoder selection, we need to be able to compare the NCNVs of different FAIDs in an effective way. We achieve this through the notion of *decoder domination*.

2.3.4. Decoder domination

Let $\mathcal{F} = \{\mathrm{D}_1, \ldots, \mathrm{D}_{N_\mathcal{F}}\}$ denote the set of N_s-level FAIDs considered for possible decoder selection and let $\Lambda = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{N_\Lambda}\}$ denote the set of TSs used for selecting FAIDs from $\mathcal{F}$. An NCNV of FAID $\mathrm{D}_k \in \mathcal{F}$ computed on every $\mathcal{T}_j \in \Lambda$. Suppose we want to compare these NCNVs with the NCNVs of another FAID $\mathrm{D}_l \in \mathcal{F}$. To facilitate this comparison conveniently, an array $\mathscr{N}_{\mathrm{D}_k}$ of size $N_\Theta \times N_\Lambda$ can be defined whose j^{th} column corresponds to the NCNV computed in $\mathcal{T}_j$. In other words, the entry $\mathscr{N}_{\mathrm{D}_k}(i,j)$ is the noisy critical number of D_k computed on $\mathcal{T}_j$ under initialization vector $\mathbf{\Theta}_i$.

We now compare the two arrays $\mathscr{N}_{\mathrm{D}_k}$ and $\mathscr{N}_{\mathrm{D}_l}$ using the notion of *domination multiplicity*. The domination multiplicity of D_k on D_l is given by

$$\tilde{n}(\mathrm{D}_k, \mathrm{D}_l) = \sum_{i=1}^{N_\Theta} \left(\prod_{j=1}^{N_\Lambda} \mathbb{1}\left(\mathscr{N}_{\mathrm{D}_k}(i,j) \geq \mathscr{N}_{\mathrm{D}_l}(i,j)\right) \right) \mathcal{W}_{\mathrm{D}_k,i} \tag{2.3}$$

where $\mathbb{1}$ is the indicator function that outputs a one when the condition in its argument is true and zero otherwise, and $\mathcal{W}_{\mathrm{D}_k,i}$ is a non-negative weight corresponding to a vector $\mathbf{\Theta}_i$ that depends on D_k. Plainly speaking the domination multiplicity of D_k on D_l is computed by identifying a vector $\mathbf{\Theta}_i$ for which $\mathbf{u}_{\mathrm{D_k}}(\mathcal{T}_j, N_I) \geq \mathbf{u}_{\mathrm{D_l}}(\mathcal{T}_j, N_I)$ $\forall \mathcal{T}_j \in \Lambda$, and then taking a weighted sum of all such vectors.

The weight $\mathcal{W}_{\mathrm{D}_k,i}$ is used to signify the fact that certain initialization vectors may be more important than others in terms of a FAID having a higher critical number, since for a particular FAID, certain vectors may be more likely to occur than others during the actual message-passing on a graph G. In other words, the weight $\mathcal{W}_{\mathrm{D}_k,i}$ gives a measure of how likely the vector $\mathbf{\Theta}_i$ is to occur during decoding by FAID D_k. We capture this measure by choosing these weights in the following manner. Let $\Phi_v^{(k)}$ denote the map of FAID D_k. For a component of the vector $\mathbf{\Theta}_i$, say θ_r, we obtain a multiplicity y counting the number of unordered pairs of messages $(m_1, m_2) \in \mathcal{M}^2$ for which $\Phi_v^{(k)}(C, m_1, m_2) = \theta_r$. The weight $\mathcal{W}_{\mathrm{D}_k,i}$ is then the product of all multiplicities

obtained for each component. For ease of understanding, let us consider the case of $b = 2$. Then $mathcalW_{\mathrm{D}_k,i}$ for a vector $\boldsymbol{\Theta}_i = \{\theta_1, \theta_2\}$ is precisely given by

$$\mathcal{W}_{\mathrm{D}_k,i} = \left|\left\{(m_1, m_2), (m_3, m_4) \in \mathcal{M}^2 \ : \Phi_v^{(k)}(C, m_1, m_2) = \theta_1, \Phi_v^{(k)}(C, m_3, m_4) = \theta_2\right\}\right|$$

where (m_1, m_2) and (m_3, m_4) are unordered pairs of messages. If a vector $\boldsymbol{\Theta}_i$ has a higher associated weight than a vector $\boldsymbol{\Theta}_p$, we regard $\boldsymbol{\Theta}_i$ to be more likely to occur since it has a greater number of possible entries in Φ_v that output its corresponding components. We found this choice of weights to be appropriate for the decoder selection.

If $\tilde{n}(\mathrm{D}_k, \mathrm{D}_l) \geq \tilde{n}(\mathrm{D}_l, \mathrm{D}_k)$, then D_k is said to dominate D_l with *domination strength* $\tilde{n}(\mathrm{D}_k, \mathrm{D}_l) - \tilde{n}(\mathrm{D}_l, \mathrm{D}_k)$. A high domination strength implies that there is a much larger number of vectors (or vectors that have higher weights) for which D_k has higher critical numbers than D_l. For convenience, we shall use the symbol $\rhd$ to denote domination, i.e., $(\mathrm{D}_k \rhd \mathrm{D}_i) = 1$ implies that D_k "dominates" D_i.

2.3.5. Choice of trapping sets for decoder selection

Since our approach for identifying good FAIDs relies on comparing the NCNVs of FAIDs computed on different TSs, the choice of TSs to be included in the set Λ plays an important role in the effectiveness of the decoder selection. We utilize the TSO [60] of column-weight-three codes in order to choose the TSs. The TSO was introduced earlier in Chapter 1 as a hierarchy of trapping sets based on their topological relations in the form of a predecessor-successor relation.

As an illustration, Fig. 2.10 provides a partial list of graphs belonging to the TSO of girth-eight column-weight-three codes which originates from an 8-cycle as a $\mathcal{T}(4, 4)$, and moves right towards larger graphs by adding extra nodes. The predecessor-successor relationship between the graphs is highlighted through the arrows directed from the predecessors towards successors. Note that $(a, b)\{i\}$ is used to distinguish between multiple graphs in the TSO having the same (a, b).

In order to select graphs from the TSO to be included in set Λ, we use the following criteria: 1) we select graphs from the TSO which together cover as many of the harmful predecessors in the TSO as possible but also with each having some successors in the TSO not considered, and 2) the graphs selected must ensure reasonable complexity in the computation of the NCNVs since larger graphs will require more computations. The main intuition for criterion 1 is that the successors are at least as harmful or more than their predecessors since they contain their predecessors. On the other hand, selecting only the successors with least values of b such as the $\mathcal{T}(7,1)$ may render the selection method ineffective, since even the good FAIDs may not have a good error correction capability on them. Further, the harmfulness of predecessors can be known based on failures of conventional decoders. For instance, the $\mathcal{T}(5,3)$ has been found to be more harmful than the $\mathcal{T}(6,4)\{1\}$ for Gallager-B [46] as well as BP decoding [62] so successors of T $\mathcal{T}(5,3)$ should be chosen. A more elaborate discussion on the harmfulness of TSs can be found in [62]. Also an additional criterion that can be used is to take into account graphs that are more commonly found in practical high-rate codes. Note that for practicality of comparing NCNVs computed on different TSs, we select graphs that have the same values of b so that the NCNVs all have the same dimension. Fixing the value of b in the choice of TSs does not diminish the effectiveness of the decoder selection, since they will contain predecessors in the TSO that can have different values of b.

2.3.6. Methodology for selection: a general approach

We now present a methodology for identifying good N_s-level FAIDs can now be devised based on the notions of decoder domination and the NCNVs. We remind the reader that the main goal of our approach is to be able to identify a small subset of candidate N_s-level FAIDs $\mathcal{F}_c$ from the set of N_s-level FAIDs $\mathcal{F}$, where each candidate FAID in $\mathcal{F}_c$ is potentially good on several codes. Ideally, if a candidate FAID could

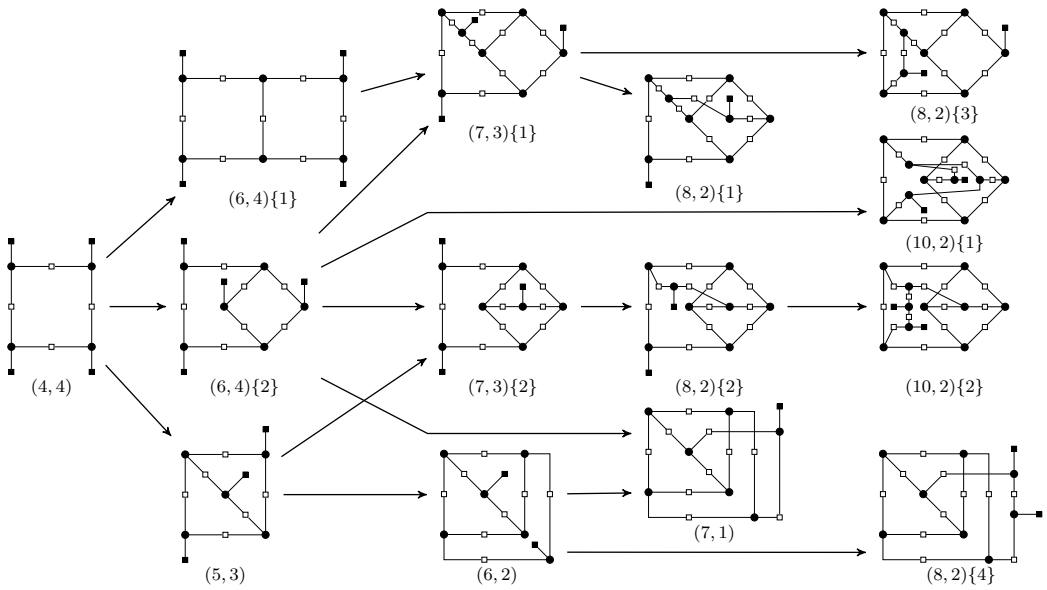


FIGURE 2.10. Some graphs belonging to the TSO of girth-eight column-weight-three codes with their relations

be selected solely based on how it dominates all the other FAIDs in $\mathcal{F}$, then one could possibly obtain an ordering of the FAIDs in $\mathcal{F}$ in terms of their dominance and conclude as to which ones are more likely to be good on a given code. Unfortunately, we have found that such an ordering does not exist since there can be many FAIDs that dominate a particularly good FAID (known a priori to be good) and yet perform poorly on certain codes.

Therefore, we propose an approach for selection that utilizes pre-determined small sets of "good" FAIDs and "bad" FAIDs denoted by $\mathcal{F}_g$ and $\mathcal{F}_b$ respectively. Let a FAID in $\mathcal{F}_g$ be denoted by $\mathcal{G}_i$ and a FAID in $\mathcal{F}_b$ be denoted by $\mathcal{B}_i$ (i being the index number). A "good" FAID $\mathcal{G}_i \in \mathcal{F}_g$ is known a priori to have good FER performance (surpassing BP) on several codes, whereas a "bad" FAID $\mathcal{B}_i \in \mathcal{F}_b$ may perform well on one code but not on several codes. We regard FAIDs in $\mathcal{F}_b$ to be "bad" since our goal is to identify FAIDs that perform well on several codes. We shall first present the general methodology used to select FAIDs, and then later we provide the specific details used to generate the list of candidate FAIDs.

The selection procedure begins with computing the NCNVs on each of the TSs in Λ for all the FAIDs contained in the sets $\mathcal{F}$, $\mathcal{F}_g$, and $\mathcal{F}_b$. The value of N_I used to compute the NCNVs should generally be chosen to be small (5 to 10 iterations). We then evaluate whether a particular FAID $\mathrm{D}_k \in \mathcal{F}$ dominates or is dominated by the FAIDs in the sets $\mathcal{F}_g$ and $\mathcal{F}_b$. By using the sets $\mathcal{F}_g$ and $\mathcal{F}_b$ to compare with, we are inherently trying to select FAIDs whose NCNVs have characteristics similar to the NCNVs of FAIDs in $\mathcal{F}_g$ but dissimilar to the NCNVs of the FAIDs in $\mathcal{F}_b$.

Therefore, we define a cost function, $\mathscr{C}_{\tilde{n}}$, that is based on domination strengths, and whose value determines whether the FAID D_k should be accepted for inclusion into $\mathcal{F}_c$.

$$\begin{aligned}\mathscr{C}_{\tilde{n}}(\mathrm{D}_k) = 2 \sum_{\substack{\forall\ \mathcal{G}_i \in \mathcal{F}_g, \\ \mathrm{D}_k \rhd \mathcal{G}_i}} \big(\tilde{n}(\mathrm{D}_k, \mathcal{G}_i) - \tilde{n}(\mathcal{G}_i, \mathrm{D}_k)\big) + \sum_{\substack{\forall\ \mathcal{B}_i \in \mathcal{F}_b, \\ \mathrm{D}_k \rhd \mathrm{D}_j}} \big(\tilde{n}(\mathrm{D}_k, \mathcal{B}_i) - \tilde{n}(\mathcal{B}_i, \mathrm{D}_k)\big) \\ - \sum_{\substack{\forall\ \mathcal{G}_j \in \mathcal{F}_g, \\ \mathcal{G}_j \rhd \mathrm{D}_k}} \big(\tilde{n}(\mathcal{G}_j, \mathrm{D}_k) - \tilde{n}(\mathrm{D}_k, \mathcal{G}_j)\big) - 2 \sum_{\substack{\forall\ \mathcal{B}_j \in \mathcal{F}_b, \\ \mathrm{D}_j \rhd \mathrm{D}_k}} \big(\tilde{n}(\mathcal{B}_j, \mathrm{D}_k) - \tilde{n}(\mathrm{D}_k, \mathcal{B}_j)\big)\end{aligned} \tag{2.4}$$

One crucial aspect that we have observed is that a FAID should dominate most (or all) FAIDs in $\mathcal{F}_g$, but also not be dominated by most (or all) FAIDs in $\mathcal{F}_b$ for it to be considered potentially good. Hence, the factors of 2 are assigned to the first and the last sum terms in $\mathscr{C}_{\tilde{n}}$ to reflect this. Other factors can be chosen as long as this aspect is reflected in $\mathscr{C}_{\tilde{n}}$. The value of the cost function $\mathscr{C}_{\tilde{n}}(\mathrm{D}_k)$ is then compared to a threshold τ. If $\mathscr{C}_{\tilde{n}}(\mathrm{D}_k) \geq \tau$, then the FAID D_k is selected as a candidate to be included in $\mathcal{F}_c$, else it is rejected. The selection methodology is summarized in the algorithm 1.

It is important to note that the cost function provides an ordering on the selected FAIDs, and therefore τ only controls the cardinality of the final set $\mathcal{F}_c$. If a large $\mathcal{F}_c$ is desired, then a smaller τ is chosen, and vice versa if a smaller $\mathcal{F}_c$ is desired. Also note that the approach we have presented here is slightly different from the one proposed in [71]. With the approach presented here, we have found that the outcome of the selection procedure is greatly improved and we were able to obtain much better sets

Algorithm 1: FAID selection algorithm

1. Choose the parameter b, and select the (a, b) TSs to be included in Λ from the TSO. Choose the parameters N_I and τ.
2. Specify the sets $\mathcal{F}$, $\mathcal{F}_g$, and $\mathcal{F}_b$. Initialize $j = 1$.
3. Compute the NCNVs $\mathbf{u}_{\mathcal{G}_i}(\mathcal{T}_j, N_I)$ $\forall \mathcal{G}_i \in \mathcal{F}_g$ and $\mathbf{u}_{\mathcal{B}_i}(\mathcal{T}_j, N_I)$ $\forall \mathcal{B}_i \in \mathcal{F}_b$ on TS $\mathcal{T}_j \in \Lambda$. Increment j by 1 and repeat step until $j = N_\Lambda$. Initialize $k = 0$.
4. Compute the NCNV $\mathbf{u}_{\mathrm{D}_k}(\mathcal{T}_j, N_I)$ of FAID $\mathrm{D}_k \in \mathcal{F}$ on $\mathcal{T}_j \in \Lambda$ $\forall j \in \{1, \ldots, N_\Lambda\}$.
5. Compute the value of cost function $\mathscr{C}_{\tilde{n}}(\mathrm{D}_k)$. If $\mathscr{C}_{\tilde{n}}(\mathrm{D}_k)$, accept D_k into the set $\mathcal{F}_c$. Else, reject it.
6. Increment k by 1. If $k < N_\mathcal{F}$, go to Step 4. Else STOP.

of candidate FAIDs $\mathcal{F}_c$ (in terms of their error floor performance).

Using our methodology, we were able to derive a set of good candidate 7-level FAIDs (which are 3-bit precision decoders) for column-weight-three codes. The specific parameters used in the selection algorithm to derive the set of candidate 7-level FAIDs $\mathcal{F}_c$ are provided below:

- In Step 1, we chose $N_I = 6$. Regarding the TSs, we chose $b = 2$ to ensure reasonable complexity and four $(a, 2)$ TSs from the TSO to be included in Λ based on the criteria described previously. They are $\mathcal{T}(6, 2)$, $\mathcal{T}(8, 2)\{4\}$, $\mathcal{T}(10, 2)\{1\}$, and $\mathcal{T}(10, 2)\{2\}$ whose graphs are shown in Fig. 2.10. Note that they are either successors of $\mathcal{T}(5, 3)$ or $\mathcal{T}(6, 4)\{2\}$. Additionally, we also verified on the selected TSs, that the FAIDs chosen in $\mathcal{F}_g$ and $\mathcal{F}_b$ were distinguishable in terms of their NCNVs, further validating them to be good choices. We remark that this choice of TSs may not be the only one for the selection algorithm to be effective. But for this choice, we were able to derive good candidate 7-level FAIDs.

- In Step 2, we used 12 FAIDs for the set $\mathcal{F}_g$ and 18 FAIDs for the set $\mathcal{F}_b$. The

list of FAIDs used in those sets are provided in [72]. The sets were determined based on simulations on different column-weight-three test codes with different rates and lengths, one of them being the $(155, 64)$ Tanner code. The FAIDs included in $\mathcal{F}_g$ are the ones that were found to surpass BP on all the test codes, whereas the FAIDs in $\mathcal{F}_b$ were found to surpass BP only on the $(155, 64)$ Tanner code and not other codes. For the set $\mathcal{F}$, instead of considering all possible 7-level FAIDs of which there is a large number, we considered a reduced set by performing density evolution and removing all the FAIDs with very poor DE thresholds.

- In Step 3, we chose τ such that the final set $\mathcal{F}_c$ contained 31 candidate FAIDs. Note that τ can be varied to obtain a larger or smaller set $\mathcal{F}_c$. The list of candidate 7-level FAIDs is reported in [72].

On a variety of codes of different rates and lengths tested, one or several 7-level FAIDs belonging to $\mathcal{F}_c$ outperformed the BP (floating-point) in the error floor. Moreover, the loss in the waterfall compared to BP was found to be very reasonable. The numerical results to support this statement are provided in the next section. Another interesting remark related to our selection procedure that we have found is that, although the density evolution (DE) was not used as part of the selection method to ensure good threshold values, the candidate FAIDs that we obtained in set $\mathcal{F}_c$ were all found to have fairly good DE thresholds.

2.4. Numerical results

Earlier in Section 2.2, we demonstrated the capability of 5-level and 7-level FAIDs to outperform BP in the error floor on the $(155, 64)$ Tanner code. We now provide additional numerical results on the BSC to further illustrate the efficacy of FAIDs on column-weight-three codes of higher practical interest and validate our approach

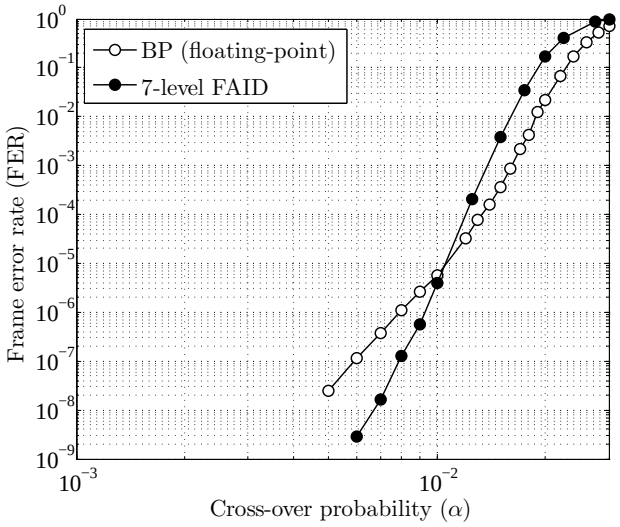


FIGURE 2.11. Performance comparisons between the BP (floating-point), the 7-level FAID defined by Table 2.1 on the $(2388, 1793)$ structured code.

for decoder selection. The three codes used for the simulations were chosen to cover a broad variety of LDPC codes in terms of rate, length, and structure. They are: 1) an $R = 0.751$ $(2388, 1793)$ structured code based on latin squares, 2) an $R = 0.5$ $(504, 252)$ code, and 3) an $R = 0.833$ $(5184, 4322)$ quasi-cyclic code.

The $(2388, 1793)$ structured code with girth-eight was designed using the method of Nguyen *et. al* [62], which is based on latin squares and avoids certain harmful TSs in the code design. The $R = 0.5$ $(504, 252)$ code with girth-eight was designed using the progressive edge-growth (PEG) method of [73] while ensuring that it contains no $(5, 3)$ TS (see [60] for the topology). The $(5184, 4322)$ quasi-cyclic code is a high-rate girth-eight code with a minimum distance of 12.

Figures 2.11, 2.13, and 2.12 show the frame error-rate (FER) performance comparisons versus the cross-over probability α between the particularly good 7-level (3-bit precision) FAIDs we identified and the BP (floating-point). Table 2.1 defines the FAID used on the $(2388, 1793)$ and the $(5184, 4322)$ codes, while Table 2.3 defines the FAID used on the $(504, 252)$ PEG-based code. Note that both are 7-level NLT

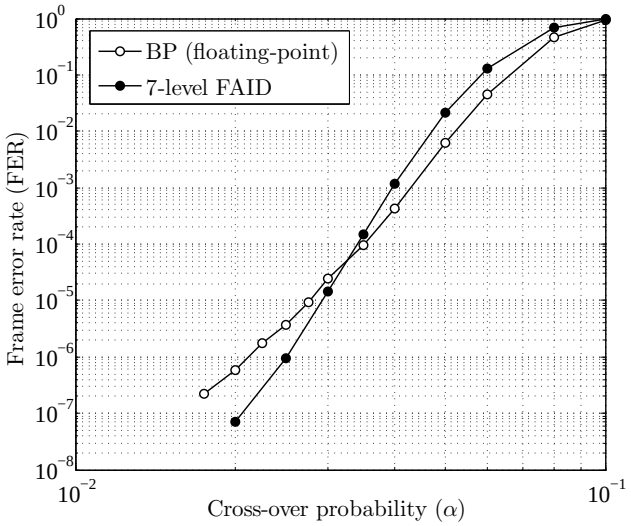


FIGURE 2.12. Performance comparisons between the BP (floating-point) and the 7-level FAID defined by Table 2.3 on the $(504, 252)$ PEG code.

FAIDs and all decoders were run for a maximum of 100 iterations.

In all three codes, the 7-level FAIDs begin to surpass the BP at an FER$\simeq 10^{-5}$. Also notice the difference in the better slopes of the error floor for the 7-level FAIDs which can be attributed to their improved error correction capability. For instance, all FAIDs used on the $(155, 64)$ Tanner code in Fig. 2.2 guarantee a correction of 5 errors, whereas BP fails to correct several 5-error patterns. It must also be noted that the good 7-level FAIDs identified using our approach outperformed BP on several other tested codes as well. Therefore, the methodology is applicable to any column-weight-three code and provides to an extent "universally" good FAIDs, as they are all capable of surpassing BP on not just few but several codes.

2.5. First results on column-weight-four LDPC codes

To conclude this chapter, we provide a *practical* section in order to identify harmful structures, and design FAIDs for a given column-weight-four code. This section does not contain any theoretical work, but just relies on experiments, observations and

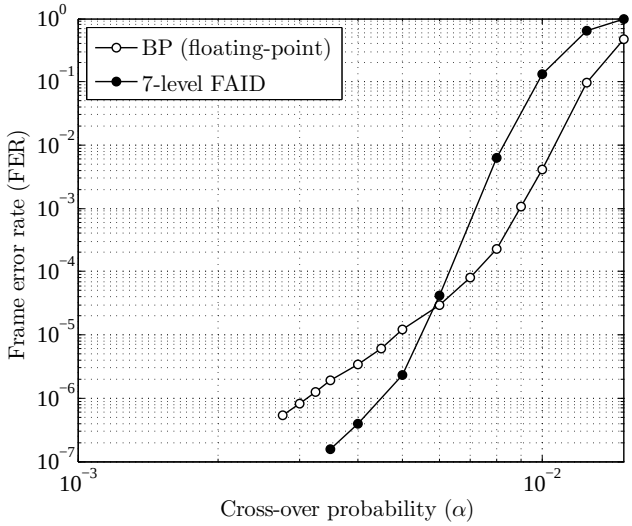


FIGURE 2.13. Performance comparisons between the BP (floating-point) and the 7-level FAID defined by Table 2.1 on the $(5184, 4322)$ quasi-cyclic code.

intuition leading nevertheless to promising results.

All the work performed previously considered column-weight-three girth-eight LDPC codes where the work on the harmful structures responsible for the error floor phenomenon of classical decoders was known, notably using the TSO. However for practical applications (notably in storage systems), we need to address the case of a high-rate column-weight-four girth-six LDPC codes. In this section we propose a practical case of study of designing a FAID for such a given code on the BSC. The code we take as a practical test code is a $(4995, 4458)$ quasi-cyclic (QC) with column-weight four, girth-six and high-rate.

The FAIDs aim at improving the performance of an LDPC code in the error floor region. For column-weight-four and girth-six LDPC codes, apart from the study of absorbing sets [47], the low-weight error patterns are not really known. Considering the length and the rate of the code, the exhaustive extraction of these structures cannot be done quickly and efficiently with the known algorithms. We choose another approach which consists in computing the performance of the BP/min-sum decoders

on the BSC until the mid-SNR region, and then from the error patterns obtained, we derive some low-weight error patterns that are responsible for the error floor on the high SNR region. Due to the QC property of the code, the knowledge of a particular harmful attractor provides multiple ones. Finally the designed FAID should be able to correct some of the extracted structures, or at least improve the number of bits that can be corrected on them. Finding the minimum TSs of a given code is not an obvious issue, especially when this code is a high-rate code with a high minimum distance. The existing algorithms suggest to focus on the Tanner graph and to derive some attractors with cycle enumerators in order to predict the error floor. Here we just want to identify some of the smallest TSs in order to design decoders with better performance on these TSs compared with the classical decoders (min-sum/BP).

On the QC-LDPC code of length 4995, it is not possible (by simulation in a reasonable amount of time) to obtain error patterns of weight less than 15. We simulate the performance of both min-sum and BP decoders on the BSC starting at a fixed α. We then collect the error patterns with a maximum weight of 40. The idea is next to claim that if a binary vector is an error event, it is because the bits initially in error in the Tanner graph are located in the neighborhood of an attractor from which he cannot escape. With this obvious statement we develop an iterative method to identify those attractors. From the error patterns collected, we run again the simulations on the BSC, recording for each bit their frequency of decision in error during the decoding (we used a maximum of 100 iterations). From this records we can extract the indices of the variable nodes often decided in error (practically we use a threshold on the percentage of iterations spent by a variable decided in error). We then run again the decoding with this new vector as an input vector and we keep recording the indices of the variable nodes decided in error by eventually increasing the percentage. Once the method converges to a fix point, we obtain one attractor (with a low-weight) derived from the original error event.

Remark: This procedure does not necessarily converge to a TS, as it can also

Algorithm 2: Harmful attractor procedure

1. Simulate the frame error rate at $\alpha = \alpha_0$ in the mid-SNR region.
2. Take an error pattern $\mathbf{E}_0$ of weight w_0 (which is not already an attractor, ie. which evolves during the decoding).
3. Initialize $i = 0$, and the threshold percentage τ=50%
4. Simulate again $\mathbf{E}_i$ at $\alpha_{i+1} = \frac{w_i}{N}$
5. If $\mathbf{E}_i$ is still an error pattern :
   (a) Record the frequency $\mathbf{f}^{(i)} = \{f_1^{(i)}, f_2^{(i)}, ..., f_N^{(i)}\}$ of bits $\{b_1, ...b_N\}$ decided in error during the decoding
   (b) $\mathbf{E}_{i+1} = \{b_k | f_k^{(i)} \geq \tau\}_{(k=1...N)}$
   (c) $i = i + 1$ Redo step 4. until $\mathbf{E}_{i+1} = \mathbf{E}_i$

converge eventually to the original codeword sent: in this case no harmful attractor is found. Note that we are not claiming that with this method, each error-pattern will output a low-weight harmful topology, but if this error pattern is "close enough" to an harmful structure, this procedure should capture the effect of it.

In the following we expose the method, and then two examples are explained to show the extraction of two harmful subgraphs containing 8 bits.

2.5.1. Scheme of the proposed algorithm

We consider a decoder $\mathcal{D}$ on the BSC with a transition probability α on a LDPC code of length N. The algorithm proceeds as described by the algorithm 2. The idea of the algorithm is the following one : if a bit is decided in error more than half of the decoding time in error, it means that it is involved in some attractors that make the decoding fail. We point out that the selection of the original error event is very important. We need to select an error event that evolves during the decoding, which means that the decided output vector changes from one iteration to another.

A weakness of this algorithm concerns the time of convergence in the decoding. With a high length code, some vectors can be error events in 100 iterations, but they can be corrected in 150 iterations. We want here to select "real" error events, for which the decoder is trapped in an attractor no matter the number of iterations done. Practically the threshold percentage is iteration dependent. We obtained the best results with a low starting percentage and finishing at a high percentage.

To illustrate this algorithm we propose two examples. The first one with a constant percentage and the second one with an adaptive one.

2.5.2. Examples of the extraction of a minimal trapping set

2.5.2.1. Example 1 : Fix percentage - τ=50%

This first example shows that from an error event of weight 18, we derive a harmful subgraph whose induced subgraph contains 8 variable nodes while keeping the threshold at 50%. The method starts with the original error event of weight 18 decoded with the BP decoder. At the end of the 100 decoding iterations we keep the bits decided in error more than 50% of the decoding time - there are 9 with an average number of iterations decided in error of 79. We feed these bits to the decoder as a new error event, and we record the bits decided in error more than 50% of the time. At the third iteration our algorithm has converged, the 8 bits decided more than 50% of the time in error are the same as the ones in error at the input of the decoder with an average number of iteration decided in error of 97. In three iterations of this procedure we obtained the harmful subgraph shown in Fig 2.14(a). Moreover we checked that when the 8 bits of this harmful subgraph are initially in error, the decoding fails.

TABLE 2.11. Steps of the algorithm - Example 2

Iteration	Threshold τ	Error event weight	Number of bits with an error decision frequency $\geq \tau$	Average number of iterations for the bits with an error decision frequency $\geq \tau$
1	30%	16	9	56
2	30%	9	10	58
3	30%	10	11	77
4	30%	11	11	85
5	55%	11	9	92
6	70%	9	8	94

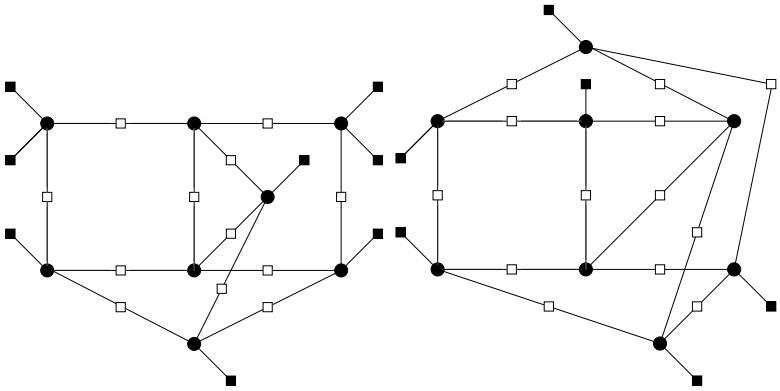
2.5.2.2. Example 2 : Adaptative percentage - 30% $\leq \tau \leq$ 70%

This second example starts from an error event obtained of weight 16. The algorithm is slightly modified because the values of the percentage evolve with the iterations. Indeed we start with τ=30% to end up to τ=70%. We have to consider dynamic τ because we witnessed that the error event is quite far from the bad attractor we want to reach. By starting with a lower τ, we can "follow" better the evolution of the errors, and get closer to bad attractors.

During the first four iterations we then keep τ=30%. Once the output error event is the same as the input one, we increase τ to 55% and then to 70% until we obtain a low-weight error pattern, in this case with 8 bits. The evolution of the procedure for this example is summarized in the Table 2.11. The obtained subgraph is depicted in the Fig. 2.14(b). It is pretty easy to see that the absence of two odd-degree check nodes on one variable node, leads to a highly harmful structure, indeed the BP cannot correct all the 6-error event in it.

2.5.2.3. Subgraphs extracted

Similarly to the two previous examples, we extracted the topological structures for the BP and/or the min-sum decoder. We used the two decoders, as we witnessed that some error events for the BP decoder were not harmful when the min-sum decoder was used. We extracted a small database of subgraphs for which the classical decoders fail for 8, 7, 6 and even 5 errors. We present in the Table 2.13 the number of bits that



(a) $\mathcal{T}(8,4,2)$ (b) $\mathcal{T}(8,6,0)$

FIGURE 2.14. Extracted subgraph: (a) from Example 1, (b) from Example 2

a given decoder can correct on the whole code using the new notation. Although it is not a standard notation, we propose to denote the harmful subgraph for column-weight-four code TSs using the following definition.

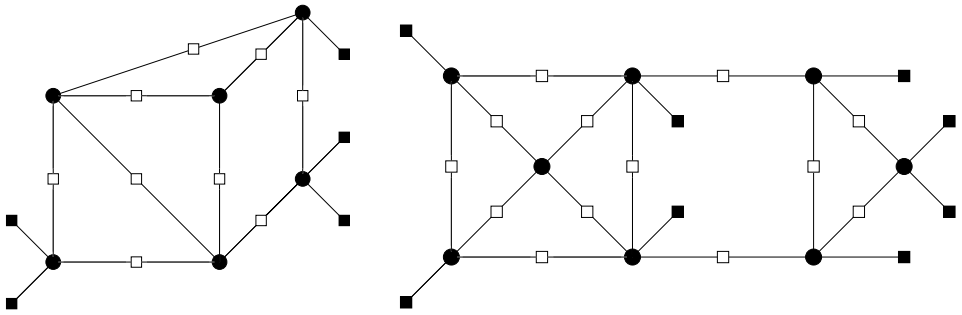
Definition 11. *Let* $\mathbf{T}(\mathbf{y})$ *be an harmful subgraph for column-weight-four code. This subgraph is denoted* $\mathcal{T}(a,b,c)$*, where* $a = |\mathbf{T}(\mathbf{y})|$ *and whose induced subgraph has* b *variable nodes connected to one degree-one check node and* c *variable nodes connected to two degree-one check nodes.*

Remark : For the column-weight-three LDPC code, the former notation still stands with a $(a,b,0)$ TS.

Fig. 2.15(a) presents one of the most harmful structures that we found in the QC-LDPC code, hence 5 errors cannot be corrected for the BP decoder. The Fig. 2.15(b) presents another structure on which the BP cannot correct more than 6 errors.

2.5.2.4. Proposed decoder on the BSC

We now present the decoders that have been designed for this particular QC high-rate code. The update function at the check nodes is still taken to be the update rule of the min-sum decoder presented in the beginning of the chapter in Eq. (2.1).



(a) $\mathcal{T}(6,2,2)$ TS (b) $\mathcal{T}(8,6,1)\{2\}$

FIGURE 2.15. Two harmful TSs extracted from the given code.

TABLE 2.12. Array map for a 5-level FAID on a column-weight-four code.

m_1	m_2	m_3	$\Phi_v(-C, m_1, m_2, m_3)$
$-L_2$	$-L_2$	$-L_2$	$-L_2$
$-L_2$	$-L_2$	$-L_2$	$-L_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$+L_2$	$+L_2$	$+L_2$	$+L_2$

On the variable node we can still define a FAID map Φ_v as for the column-weight-three LDPC code with an array. With $d_v = 4$ this array for $y_i = -C$ for a 5-level FAID is now a look-up table given by the Table 2.12 where m_1, m_2, and m_3 are the incoming messages. For $N_s = 7$ levels, this array contains 84 rows, which means we need 84 outputs to design our rule, for $N_s = 15$, this value increases to 680. The number of possible decoders is then very high as soon as we increase N_s. We can certainly not generate all the possible decoders, and then we choose the best one by looking at its characteristics. We then propose to design a FAID using a non-linear equation at the variable nodes, and afterwards we derive the corresponding look-up tables. This equation has been derived in order to enable the design of many possible decoders. Its general shape has been governed by the FAIDs for column-weight-three codes whose output tends to flatten out the high level inputs. The update rule for

the designed FAID with m_1, m_2, m_3 as incoming messages, is given by

$$\Phi_v(-C, m_1, m_2, m_3) = sign(S) \times \left\lfloor \max\left(\left(\sqrt{|S|} - \rho_{off}\right), \frac{N_s - 1}{2}\right) + 0.5 \right\rfloor \quad (2.5)$$

$$\text{where : } \begin{cases} S & = & sign(m_1)m_1^2 + sign(m_2)m_2^2 + sign(m_3)m_3^2 + sign(C)C^2 \\ C & \in & \mathbb{R} \\ \rho_{off} & \in & [0, 1] \end{cases}$$

This decoder is a multi-level decoder, as the output of the variable node will belong to the alphabet due to the saturation value at $\frac{Ns-1}{2}$ and the floor function. Note that we assign integer value to the alphabet level, such that $\tilde{\mathcal{M}} = \{-\frac{Ns-1}{2}, ..., 0, ..., +\frac{Ns-1}{2}\}$. The idea of the decoder is to compute a quadratic sum taking into account the signs of the messages, and then flatten out the results with the square root. We add the offset parameter ρ_{off} to be able to design more decoders, and to improve its performance. The different value for C is providing different FAIDs, we select the FAIDs with high DE thresholds as potential candidates, and then simulate them on the given code.

The 3-bit decoder presented in the next section (on 7 levels) has the parameters $|C| = 2.5$, $\rho_{off} = 0.2$. The 4-bit decoder presented in the next section (on 15 levels) has the parameters $|C| = 3.5$, $\rho_{off} = 0.5$. These two decoders are the best FAIDs we were able to design, although it is likely that better FAIDs for column-weight-four code can certainly be designed.

2.5.2.5. Performance of the decoder

The performance curves are shown in Fig. 2.16. We can observe the improvement in the performance with our decoder especially with 15-level FAID (performing on 4 bits of precision). Indeed this error rate curve of the designed FAID is stuck to the one of the offset min-sum decoder and slightly better in the beginning of the error floor. In order to illustrate the improvement of the performance of the designed FAIDs in the error floor, we compare their error correction capability on the structures

TABLE 2.13. Maximum number of correctable bits per FAID and per TS

	Maximum Number of bits correctable			
Trapping sets	Min-sum	BP	3-bit FAID	4-bit FAID
$\mathcal{T}(6,2,2)$	5	5	**6**	**6**
$\mathcal{T}(6,4,1)\{1\}$	5	5	**6**	**6**
$\mathcal{T}(6,4,1)\{2\}$	5	5	5	5
$\mathcal{T}(7,2,2)$	6	6	6	6
$\mathcal{T}(7,4,2)$	6	6	6	6
$\mathcal{T}(8,4,2)$	7	6	**7**	**7**
$\mathcal{T}(8,6,0)$	6	6	6	6
$\mathcal{T}(8,6,1)\{1\}$	7	7	7	7
$\mathcal{T}(8,6,1)\{2\}$	6	6	**7**	**7**
$\mathcal{T}(9,2,3)$	7	7	7	7
$\mathcal{T}(9,6,2)\{1\}$	8	8	8	8
$\mathcal{T}(9,6,2)\{2\}$	8	7	**8**	**8**
$\mathcal{T}(10,4,2)$	7	7	**8**	**8**

extracted in the previous section with the min-sum and BP decoder. These results are summarized in the Table 2.13. We can see that the designed finite precision decoders correct at least the same number of errors, and even more in some cases. The TS of the Fig. 2.15(a) is actually a TS only for the min-sum and BP decoders, as the designed FAIDs can correct the 6 bits inside. As a comparison, we also report the error-rate performance in Fig. 2.17 on a $(2212, 1899)$ LDPC designed such that no small absorbing sets are present in it [74]. Interestingly, our designed FAIDs surpass also BP decoder in the error floor, although these decoders were designed to correct failures on this code. This support the idea that FAIDs with good performance in the error floor can be designed for any LDPC code. However the selection and design of such decoders based on the failures of a given code or a family of codes is obviously not trivial.

2.6. Conclusion

In this chapter we have introduced a new class of low-complexity iterative decoders, the FAIDs. After a complete description of their structures we have shown their outstanding performance in the error floor region, as we have shown that FAIDs on 7-level, hence using 3-bit of precision, were able to surpass the traditional floating-

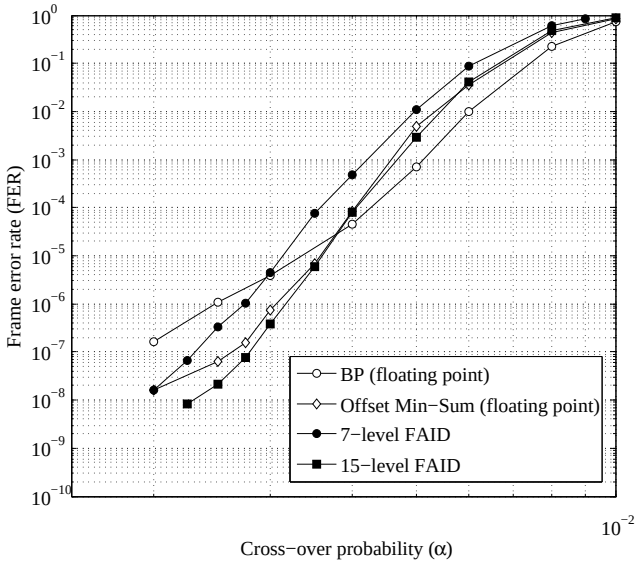


FIGURE 2.16. Performance comparisons between the floating point decoders: BP and offset min-sum, and the 7-level and 15-level FAIDs design for the $(4995, 4458)$ code.

point BP decoding. Given the high number of possible FAIDs which can be designed, we tackle the issue of the selection of FAIDs. At first we studied deeply the famous Tanner code to show that based on the minimal codeword we could select the best 5-level and 7-level FAIDs on this particular code. For more complicated codes, where the codewords cannot be listed easily, we propose a methodology based on the concept of *noisy critical number vector* and *decoder domination* to extract the best FAIDs on column-weight-three codes. We also provided an embryo of methodology for girth-six column-weight-four LDPC code where harmful structures are not completely known. Based on the observations of the min-sum and BP decoder during the decoding process, we were able to extract some harmful structures of the given code. We then designed finite precision FAIDs whose error correction capabilities were better than the min-sum and/or BP. The simulation results confirmed that a FAID using 3-bit can also surpass the BP for column-weight-four LDPC codes.

Future work include to find an efficient and systematic methodology to design FAIDs for column-weight-four codes possibly by deriving them from the identified

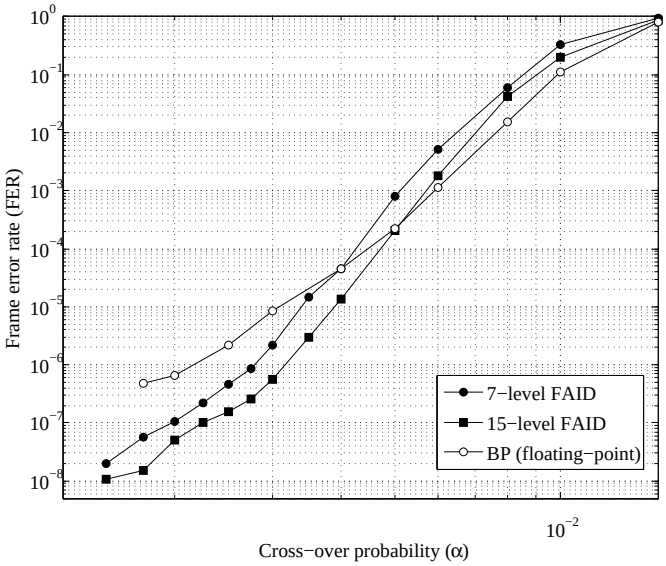


FIGURE 2.17. Performance comparisons between the floating point BP and the 7-level and 15-level FAIDs design for the $(2212, 1899)$ code.

good FAIDs for column-weight-three codes, although it is now obvious that good FAIDs for column-weight-three codes will engender good FAIDs for column-weight-four codes. The next step is obviously to tackle the more common channel, the AWGNC, in which the issue of the quantization of the channel values will be the core of the analysis.

3. COMPRESSED SENSING: INTRODUCTION AND RECONSTRUCTION METHODS

In this chapter we present an overview of the work done in the last few years in the emerging field of compressed sensing, also called compressive sensing. We remind the original problem setting, and the main relevant reconstruction methods proposed recently. For the sake of clarity of the manuscript, we only refer to the main algorithms among the large amount of methods published in the literature since the past five years regarding compressed sensing.[1]

3.1. Introduction

The compressed sensing problem as it has been suggested by Candès, Romberg and Tao [77, 78] and Donoho [79] aims at detecting a compressed or compressible signal with many less measurements than the Nyquist-Shannon sampling theorem would suggest. In the case of an analogous signal, the sampling theorem demonstrates that the number of samples that needed to be taken to recover completely the original signal needs to be at least twice its bandwidth. However in the case of a sparse signal (potentially in a certain basis), the compressed sensing provides a different approach which reduces drastically the number of samples that need to be acquired and then stored for a signal. This property of reducing the number of samples taken in a sparse signal explains the massive interest around this recent field of the signal processing.

The applications of the compressed sensing are numerous, as the assumption of sparsity can make sense in many topics of the signal processing. To name a few, let us remind that the wavelet decomposition [80] generally provides a sparse or

[1] An important database of papers on the compressed sensing topic can be found in the Digital Signal Processing group website at Rice University [75] and some up-to-date articles and papers as well as some interesting discussions are available in [76].

pseudo-sparse approximation of a given image. Other domains of the signal processing are concerned by compressed sensing, like the radar [81], imaging [82], the digital communications [83] or even the biomedical [84] as long as a sparse representation of a signal exists in some basis. Applications of compressed sensing can also be found in genetics, as it is stated in [85], in the case we want to study n genes doing some analysis on m patients. Commonly only a few genes will be active for each patient, hence the sparsity. Many other examples can be found in the literature. We continue this chapter by introducing rigorously the compressed sensing setting.

3.2. Problem setting and original solver methods

The original compressed sensing problem is to recover a high-dimensional vector from a lower dimension set of linear equations. We then consider a signal $\mathbf{s} \in \mathbb{R}^n$, and we suppose that a certain basis $\mathbf{\Psi}$ provides a k-sparse representation of $\mathbf{s}$ such that $\mathbf{s} = \mathbf{\Psi x}$ with $\|\mathbf{x}\|_0 = k$. This vector $\mathbf{s}$ is not measured directly, but instead m projections of $\mathbf{s}$ are taken where $m < n$. The projection vector is denoted $\mathbf{y}$ such that

$$\mathbf{y} = \mathbf{\Phi s} \tag{3.1}$$

with $\mathbf{y}$ a $m \times 1$ column vector and $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$ being the measurement matrix, also called the sampling matrix. Since $m < n$ the recovery of the signal $\mathbf{x}$ from $\mathbf{y}$ is in general not possible; however the assumption of a sparse signal makes the reconstruction possible and practical.

As the sparse representation of $\mathbf{s}$ in the basis $\mathbf{\Psi}$ is $\mathbf{x}$, the recovery of $\mathbf{s}$ from $\mathbf{y}$ can be formulated as the ℓ_0-norm minimization problem on $\mathbf{x}$:

$$\tilde{\mathbf{x}} = argmin\|\mathbf{x}\|_0 \text{ s.t. } \mathbf{y} = \Phi\mathbf{s} = \mathbf{\Phi\Psi x}. \tag{3.2}$$

This problem is often depicted as the Fig. 3.1 where $\mathbf{\Psi}$ is the basis matrix providing a sparse approximation of $\mathbf{s}$, and $\mathbf{\Phi}$ being the measurement matrix. In other

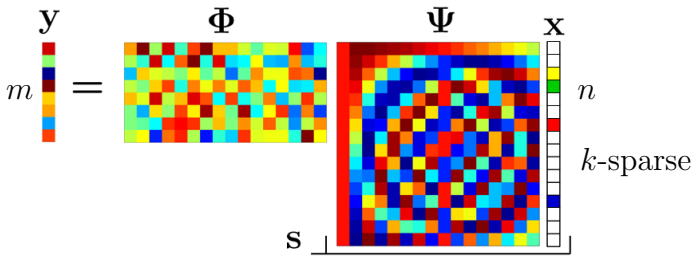


FIGURE 3.1. Traditional vision of the compressed sensing

words we look for the sparsest vector $\mathbf{x}$ such that $\mathbf{y} = \mathbf{\Phi\Psi s}$. This problem can be also viewed as finding a k-term approximation to $\mathbf{y}$ from the columns of the basis $\mathbf{\Phi\Psi}$, which is called the *holographic basis* because of the complex pattern in which it encodes the sparse signal coefficients [79]. This problem is sometimes called *recovery via combinatorial optimization* and is known to be NP-hard as the decoder needs to perform a combinatorial enumeration of all the $\binom{n}{k}$ possible sparse subspaces. Hence it cannot be used for practical applications.

Practically, a much easier problem yields an equivalent solution by solving for the ℓ_1-sparsest coefficients $\mathbf{x}$ that agree with the measurements $\mathbf{y}$. This reconstruction is known as the ℓ_1-norm minimization, or the *recovery via convex optimization*

$$\tilde{\mathbf{x}} = argmin\|\mathbf{x}\|_1 \text{ s.t. } \mathbf{y} = \mathbf{\Phi s} = \mathbf{\Phi\Psi x}. \tag{3.3}$$

This optimization is more tractable and can be solved by a linear programming technique called *Basis Pursuit (BasP)* [86]. The complexity of this technique is polynomial in n, more precisely $\mathcal{O}(n^3)$.

In the sequel we will only consider the reconstruction of the sparse vector $\mathbf{x}$ from the observation vector $\mathbf{y}$ with the measurement matrix $\mathbf{A}$ such that we are trying to solve

$$\mathbf{y} = \mathbf{Ax}. \tag{3.4}$$

In other words we either assume the basis $\mathbf{\Psi}$ being taken as the identity matrix and $\mathbf{A} = \mathbf{\Phi}$, and hence our signal is sparse in the basis of the study, or that $\mathbf{\Phi\Psi} = \mathbf{A}$.

This little modification in the problem will simplify the notations for the rest of the manuscript.

In [78] it has been shown that the ℓ_1-norm minimization technique performs the exact reconstruction of the vector $\mathbf{x}$ for a sufficient sparsity and a properly chosen measurement matrix $\mathbf{A}$. The conditions on the measurement matrix were expressed in terms of the *Restricted Isometry Property* (RIP).

We say that $\mathbf{\Phi}$ has RIP of order k if for every k-sparse vector $\mathbf{v}$, and for constant $0 \leq \epsilon \leq 1$

$$(1-\epsilon)\left\|\mathbf{v_k}\right\|_2^2 \leq \left\|\mathbf{A_k v_k}\right\|_2^2 \leq (1+\epsilon)\left\|\mathbf{v_k}\right\|_2^2 \tag{3.5}$$

where $\mathbf{v_k}$ is the mutilated vector $\mathbf{v}$ constituted of the k non-zero elements of $\mathbf{v}$, and $\mathbf{A_k}$ corresponding to the matrix formed by the extracted columns of $\mathbf{A}$ which corresponds to the k indices of the non-zero elements of $\mathbf{v}$. In other words the RIP indicates that the matrix $\mathbf{A}$ does not distort too much the ℓ_2-norm of any k-sparse vector.

The RIP gives the following interesting property: for a k-sparse vector $\mathbf{x}$ we can take the number of measurements m on the order of $k\log(n/k)$. It is also important to notice that the RIP provides an insurance of the output of the BasP to be the same output as the ℓ_0 minimization, however it can be different from the original vector $\mathbf{x}$.

3.3. Iterative reconstruction methods

3.3.1. Introduction

Soon after the seminal works of Cands, Tao, Romberg and Donoho, many studies have been led in order to tackle the complexity of the ℓ_1-minimization reconstruction technique. The main goal was to propose a variation of the LP solver providing good reconstruction performance with a lower complexity. Among all the proposed methods, we cite here some famous greedy algorithms like the Matching Pursuit [87], the Orthogonal Matching Pursuit (OMP) [88, 89], or Stagewise OMP [90]. The

main idea of these greedy algorithms consists in the iterative approximation of the coefficients of the sparse signal to recover. They are known to be very fast and easy to implement, and guarantee the performance to be close to the original ℓ_1-minimization. Many other algorithms exist in the literature, the chapter 8 of [91] presents an overview of the greedy algorithms used in compressed sensing.

Other methods for recovering sparse signal from an undetermined system of equations have been suggested equivalently to the message-passing decoder in channel coding. The first work mentioning this type of reconstruction methods is in [92]. In this chapter we give an overview of the iterative reconstruction methods starting by exploring the link between message-passing algorithms in compressed sensing and in channel coding. Then we present the current knowledge in the use of LDPC matrices as measurement matrix. Finally we explore some of the iterative reconstruction methods.

3.3.2. Motivation for message-passing reconstruction methods

The success of sparse codes in channel coding such as the Low-Density Parity-Check (LDPC) codes [1,15] has strongly suggested that the use of sparse compressed matrices needed to be explored in the compressed sensing case.

As we already mentioned, in a communication system, a codeword x containing n bits is transmitted through a noisy channel. The noise added by the channel to the original information can be modeled as an error vector $\mathbf{e}$ added to the codeword $\mathbf{x}$. In the case of the BSC this error vector is binary, in the case of an AWGNC, the samples of $\mathbf{e}$ are drawn for a Gaussian distribution. The output vector is defined as $\mathbf{y} = \mathbf{x} + \mathbf{e}$, and the addition is proceeded accordingly to the channel (binary addition for the BSC, real addition for the AWGNC). Although this has been already presented, let us briefly remind that an LDPC code is defined by its parity-check matrix $\mathbf{H}$ of size $m \times n$ with $n > m$. The particularity of $\mathbf{H}$ is its sparsity, as the number of non-zero

elements in $\mathbf{H}$ decreases in $1/n$. The sparsity of $\mathbf{H}$ and then its equivalent bipartite Tanner graph [16] plays a key role in the decoding process. Indeed the decoding will be simplified by running a low-complexity message-passing algorithm (such as the belief propagation, or the FAID presented in the chapter 1 to decode $\mathbf{y}$ and then recover $\mathbf{x}$.

One possible link between compressed sensing and LDPC codes stands in the property of $\mathbf{H}$. Indeed we remind that, from the construction of $\mathbf{x}$ we have $\mathbf{Hx} = \mathbf{0}$. If we consider a sparse noisy vector $\mathbf{e}$, the output of the channel can be transformed in $\tilde{\mathbf{y}} = \mathbf{Hy} = \mathbf{H}(\mathbf{x} + \mathbf{e}) = \mathbf{He}$. Then $\mathbf{H}$ can be viewed as a measurement matrix, and by applying the reconstruction method of the compressed sensing, we can recover $\mathbf{e}$ and then $\mathbf{x}$.

In the following we consider $\mathbf{H}$ as equivalently a parity-check matrix, or a measurement matrix.

3.3.3. LDPC measurement matrices

The relation between the theory of LDPC codes and compressed sensing was explored in the work of Dimakis and Vontobel [93], who studied the relation between two linear programs: the BasP of compressed sensing which can be restated as a Linear Program (LP) and the so-called LP-decoder [94] of a related LDPC code. (The LP decoder is a sub-optimal decoder whose performance is governed by the parity-check matrix used to define the LP constraints.) It was shown in [93] that a binary matrix $\mathbf{H}$ which is a good parity-check matrix for LP decoding of the corresponding LDPC code is also a good measurement matrix for BasP in the respective compressed sensing problem over a binary alphabet. The contraposition gives that "bad" measurement matrices in compressed sensing will have "bad" performance results in the channel coding side. Besides it is shown that the basis pursuit using the matrices constructed by Gallager [1] forms the best known sparse measurement matrices that maximize the recovery of

sparse signals using these sparse measurement matrices. We already mentioned that a *sufficient* condition to certify that a given measurement matrix is "good" is that this matrix follows the RIP. However the RIP is an incomplete characterization of the LP relaxation of "good" measurement matrices [95]. A *necessary and sufficient condition* for a measurement matrix to be "good" requires that the matrix $\mathbf{H}$ respects the *nullspace* condition. The attribute "good" means here that the ℓ_0-minimization and the ℓ_1-minimization (BasP) outputs the same estimate $\hat{\mathbf{x}}$. Dimakis *et al.* [96] presented a necessary and sufficient condition for a good measurement matrix for k-sparse signals. While Restricted Isometry Property (RIP) provides a sufficient condition for a good measurement matrix for compressed sensing, Null Space Property (NSP) which is defined below, provides the necessary and sufficient condition for a good compressed sensing measurement matrix. Let V be the set of columns of a measurement matrix $\mathbf{H}$ and the null-space $\text{NSpace}(\mathbf{H}) = \{\omega \in \mathbb{R}^n : \mathbf{H}\omega = \mathbf{0}\}$. The measurement matrix H has the Null Space Property (NSP) if for $k \in \mathbb{Z}_{\geq 0}$, $c \in \mathbb{R}_{\geq 0}$ and $I' = V \backslash I$.

$$c\| \omega_I \|_1 \leq \| \omega_{I'} \|_1 \ \ \forall\, \omega \in \text{NSpace}(\mathbf{H}),\ \forall I \subseteq V,\ |I| \leq k$$

Before giving the main result representing the connection between channel coding and compressed sensing, a summary of LP decoding and some related definitions are explained.

Let $\mathcal{C}$ be a binary linear code and let $\mathbf{H}$ be a parity-check matrix of $\mathcal{C}$ with the set of columns V and the set of rows C. For every $j \in C$, let h_j be the j^{th} row of H and let

$$\text{Null}_{\mathcal{C}}(j) = \{x \in \{0,1\}^n | \sum_{i \in V} h_{ji}.x_i = 0 \mod 2\}.$$

Then, the *fundamental polytope* P of $\mathbf{H}$ is defined as $P(\mathbf{H}) = \bigcap_{j \in C} \text{conv}(\text{Null}_{\mathcal{C}}(j))$, where $\text{conv}(\text{Null}_{\mathcal{C}}(j))$ is the convex hull of $\text{Null}_{\mathcal{C}}(j)$. The conic hull of $\mathbf{H}$ is called

fundamental cone and is denoted by $\mathcal{K}(\mathbf{H})$. Hence,

$$\mathcal{K}(\mathbf{H}) = \text{conic}(P(\mathbf{H})) = \bigcap_{j \in C} \text{conic}(\text{Null}_{\mathcal{C}}(j)).$$

The LP decoding [94] tries to solve the following optimization problem.

$$\text{minimize} \quad \sum_{i=1}^{n} \lambda_i . f_i \quad \text{s.t.} \quad f \in P(\mathbf{H})$$

where $\lambda_i = \log \frac{\Pr(y_i|0)}{\Pr(y_i|1)}$ and $\mathbf{y} = (y_1, y_2, ..., y_n)$ is the received word. In analyzing the performance of LP decoder, it is assumed that the all-zero codeword is transmitted. So, the LP decoder will succeed on decoding when the all-zero codeword has the lowest value on the cost function $\sum_{i=1}^{n} \lambda_i . f_i$ compared to the all non-zero vectors in $\mathcal{K}(\mathbf{H})$. With these preliminaries, the fundamental connection between channel coding and compressed sensing is provided as follows.

Let $\mathbf{H}$ be a zero-one measurement matrix, then

$$\omega \in \text{NSpace}(H) \Rightarrow |\omega| \in \mathcal{K}(\mathbf{H}).$$

This result was also extended to a measurement matrix $\mathbf{H}$ with entries in $\mathbb{C}$ such that $|H_{ij}| \in \{0, 1\}$ for all $(i, j) \in V \times C$ (*Lemma 12* [96]). Also, based on definitions of *pseudo-weight* on BSC, BEC and AWGNC, they provided some results for performance guarantee of compressed sensing LP-decoder. The minimum binary symmetric channel (BSC) pseudo-weight $w_p^{BSC,min}(\mathbf{H})$ is defined in the Definition 9 of [93], and brings the following theorems:

Theorem 3. *If* $\mathbf{H}$ *is a parity-check matrix of a LDPC code* $\mathcal{C}$ *and* $w_p^{BSC,min}(\mathbf{H}) > 2K$ *then linear programming decoding will successfully decode any output from the BSC with at most* K *errors.*

Theorem 4. *If* $\mathbf{H}$ *is a 0-1 measurement matrix that satisfies* $w_p^{BSC,min}(\mathbf{H}) > 2K$ *and* $\mathbf{x}$ *is* K*-sparse then BasP and the* ℓ_0*-minimization will output the same estimate* $\hat{\mathbf{x}}$ *given* $\mathbf{y} = \mathbf{H}\mathbf{x}$.

Moreover, using expander graphs, they obtained threshold results (strong and weak bounds [96]) for compressed sensing LP-decoder and provided a discussion on construction of good measurement matrices with large girth.

3.3.4. Message-passing reconstruction methods

To tackle the issue of the complexity of LP solver, and similarly to channel coding, low-complexity algorithms have been introduced. These algorithms primarily used the graphical representation of the measurement matrix to exchange information iteratively to recover the original vector $\mathbf{x}$. As for the Tanner graph of LDPC codes, the columns of the measurement matrix are associated with the variable nodes corresponding to the vector $\mathbf{x}$, and the rows of the measurement matrix correspond to the summation nodes, also called measurement nodes, or even check nodes to make the bridge with the graphical representation of an LDPC matrix.

3.3.4.1. Message-passing and approximate message-passing

A first message-passing algorithm was introduced by Donoho *et al.* in [92] for noise free measurements. The variables in message-passing algorithms are associated with edges in the bipartite graph representation of the matrix $\mathbf{A}$. Messages are updated according to the rules

$$\begin{aligned} x_{i\to a}^{(l+1)} &= \eta_l\Big(\sum_{b\in N(i)\setminus a} A_{b,i} z_{b\to i}^{(l)}\Big) \\ z_{a\to i}^{(l)} &= y_a - \sum_{j\in N(a)\setminus i} A_{a,j} x_{j\to a}^{(l)} \end{aligned} \tag{3.6}$$

where a and b are measurement nodes, i and j are variable nodes, η_l is a sequence of threshold functions (applied componentwise), $\mathbf{x}^{(l)} \in \mathbb{R}^n$ is the current estimate of the solution $\mathbf{x}$, $\mathbf{z}^{(l)} \in \mathbb{R}^m$ is the current residual and $N(v)$ is the set of neighbor nodes of the node v in the Tanner graph.

Donoho's *et al.* [92] iterative thresholding algorithm, called the *approximate message-passing algorithm* (AMPA) follows from the message-passing algorithm given in (3.6). The authors of [92] argue that the right side of the equation for update of messages $x_{i\to a}^{(l)}$ in (3.6) does not depend strongly on the index a (specially if the matrix A is dense in which case only one out of n terms is excluded). They also argue that the right-hand side of the equation for updating $z_{a\to i}^{(l)}$ does not depend strongly on i. If these two dependencies are neglected, the messages are associated to graph vertices (x to variable nodes and z to summation nodes), and the algorithm has a flavor of a bit flipping algorithm [23]. In contrast to message-passing which in general has to update nm messages, the bit flipping algorithms update only n variable nodes and m summation nodes.

The first algorithm [92] starts from an initial guess $\mathbf{x}^{(0)} = \mathbf{0}$ and $\mathbf{z}^{(0)} = \mathbf{y}$, and iteratively proceeds by calculating

$$\begin{aligned} \mathbf{x}^{(l+1)} &= \eta_l(\mathbf{A}^T\mathbf{z}^{(l)} + \mathbf{x}^{(l)}) \\ \mathbf{z}^{(l)} &= \mathbf{y} - \mathbf{A}\mathbf{x}^{(l)} + \frac{1}{\delta}\mathbf{z}^{(l-1)}\langle \eta'_{l-1}(\mathbf{A}^T\mathbf{z}^{(l-1)} + \mathbf{x}^{(l-1)})\rangle, \end{aligned}$$

where η_l is a sequence of threshold functions (applied componentwise), $\mathbf{x}^{(l)} \in \mathbb{R}^n$ is the current estimate of the solution $\mathbf{x}$, $\mathbf{A}^T$ denotes the transpose of $\mathbf{A}$ and $\eta'(\mathbf{u}) = \partial\eta(\mathbf{u})/\partial\mathbf{u}$. The bracket $\langle\rangle$ operator applied on a vector $\mathbf{v} = (v_i)_{1\le i\le n}$ gives $\langle\mathbf{v}\rangle = (1/n)\sum_{1\le i\le n} v_i$. The role of the additional term in the update of $\mathbf{z}^{(l)}$ is to cancel the correlation between the present vector estimates and their past values. A typical thresholding function η is the *soft thresholding* given by

$$\eta(\mathbf{x};\lambda) = \mathrm{sgn}(\mathbf{x})(|x| - \lambda)_+$$

where the subscript $(u)_+ = u\mathbb{I}(u \ge (0))$, and $\mathbb{I}$ is the indicator function equal to one when the Boolean expression in his argument is true, and zero otherwise. An

algorithm by Tropp and Wright [98] uses slightly modified update equations

$$\begin{aligned} \mathbf{x}^{(l+1)} &= \eta_l((1/c)\mathbf{A}^T\mathbf{z}^{(l)} + \mathbf{x}^{(l)}) \\ \mathbf{z}^{(l)} &= \mathbf{y} - \mathbf{A}\mathbf{x}^{(l)}, \end{aligned}$$

where the constant c is chosen to help the convergence.

Pham *et al.* [99] introduced two low-complexity algorithms, list decoding and multiple-basis belief propagation. The basic idea behind using BP in these algorithms is to identify the columns of the measurement matrix $\mathbf{A}$ that maximizes the correlation with $\mathbf{y}$. The measurement matrix $\mathbf{A}$ is constructed in the following manner. First, an LDPC code $\mathcal{C}$ of length n is considered. Then, all codewords of $\mathcal{C}$ are converted to their Binary Phase Shift Keying (BPSK) images and are normalized by $\frac{1}{\sqrt{n}}$. The normalized BPSK images of codewords are used as the columns of $\mathbf{A}$. In list-based BP algorithm, a list of T vectors based on $\mathbf{y}$, a biasing value b and sparsity parameter k is constructed. A binary parity-check matrix $\mathbf{H}$ of $\mathcal{C}$ with the row-weight d_c is chosen uniformly from the ensemble of binary matrices of the same size and with the row-weight d_c. Running the BP algorithms gives a list of binary words $\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_T}$. The repeated words or those which do not satisfy $\mathbf{Hw_i} = 0$ are deleted. The outputs are at most k words whose (BPSK) images have the largest correlation with $\mathbf{y}$.

In case of a non-binary vector $\mathbf{x}$, they proposed another algorithm, multiple basis belief propagation (MBBP) which uses more than one parity-check matrix to find the columns of $\mathbf{A}$ with the largest correlation with $\mathbf{y}$. The MBBP algorithm begins with initializing the sparsity parameter k, the measurement vector $\mathbf{y}$ and γ parity-check matrices $\mathbf{H_1}, \mathbf{H_2}, ..., \mathbf{H}_\gamma$. The interference is modeled as a zero-mean Gaussian variable with the variance $\sigma^2 = \max(|\mathbf{y}|)\frac{k-1}{k}$. Running BP algorithm for the received vector $\mathbf{y}$ and using the parity-check matrix $\mathbf{H_i}$ $(i = 1, 2, ..., \gamma)$ outputs a word $\mathbf{w_i}$. The words $\mathbf{w_i}$ which do not satisfy $\mathbf{H_i w_i} = 0$ are deleted and the codeword whose BPSK image has the largest correlation with $\mathbf{y}$ is set to be output.

Combination of the list decoding with Orthogonal Matching Pursuit (OMP), (*Al-*

gorithm 2: BP-OMP Algorithm in [99]), has an asymptotic optimal performance with computational complexity $O(kT \log n)$. Also, the simulation results show that both algorithms, list decoding and MBBP, have good performance for recovery of sparse signals.

In the rest of this chapter we present 2 low-complexity algorithms, starting with the Sparse Matching Pursuit.

3.3.4.2. Sparse matching pursuit

In [100] the authors suggest an algorithm called *Sparse Matching Pursuit* for compressed sensing with sparse measurement matrices. This algorithm can be described as a message-passing algorithm in the context of a K sparse signal. We keep here the scheme $y = Ax$.

Let us remind that the rows of $\mathbf{H}$, labeled $m = 1, \ldots, M$, are the check nodes of the graph representing the values of $\mathbf{y_m}$, and the columns of $\mathbf{H}$, labeled $n = 1, \ldots, N$, are the variable nodes representing the values of $\mathbf{x_n}$. At the l^{th} iteration the message from a variable node n to a check node m is denoted $\mu_{n\to m}^{(l)}$, and the message from a check node m to a variable node n is denoted $\mu_{m\to n}^{(l)}$. These two messages exist only on the edges of the Tanner graph, in other words if $A_{m,n} = 1$. $\mathcal{N}(n)$ (resp. $\mathcal{N}(m)$ denotes the neighbors of the variable node n (resp. check node m). The algorithm of the Sparse Matching Pursuit is given by the Algorithm 3. This algorithm requires a number of measurement M to be on the order of $Klog(N/K)$. The performances are the same asymptotically as the BasP.

Finally, we conclude this introductive chapter on compressed sensing by presenting in detail the most simple algorithm that can be imagined to recover a sparse high-dimensional vector from a set of linear measurements.

Algorithm 3: Sparse Matching Pursuit Reconstruction Algorithm

Initialization : $\mu_{m\to n}^{(0)} = y_m$
for *iteration* $l = 1, \ldots, L$ **do**
  for $n = 1, \ldots, N$ **do**
    for $\forall m \in \mathcal{N}(n)$ **do**
      $\mu_{n\to m}^{(l)} = \underset{m' \in \mathcal{N}(n)\backslash m}{median} \left(\mu_{m'\to n}^{(l-1)}\right)$;
    end
  end
  for $m = 1, \ldots, M$ **do**
    for $\forall n \in \mathcal{N}(m)$ **do**
      $\mu_{m\to n}^{(l)} = y_m - \sum_{n' \in \mathcal{N}(m)\backslash n} \left(\mu_{n'\to m}^{(l)}\right)$;
    end
  end
end

3.3.4.3. Verification decoding

In order to reconstruct strictly sparse signals, Zhang and Pfister [101] used two decoding algorithms, LM1 and LM2, based on verification algorithm. The reconstruction method is an iterative algorithm in which the messages correspond to the vertices (variable and measurement nodes in the factor graph) not to the edges. Hence, the algorithm can be considered as a bit flipping algorithm as introduced in [97]. The decoder is analyzed using density evolution to find the average fraction of verified messages. Then, for each decoder and for regular LDPC code ensembles and strictly sparse signals with $k_n = \lfloor \delta n \rfloor$, they provided a threshold δ^* such that for all $\delta < \delta^*$, iterative decoding can recover the original signal with high probability as $n \to \infty$. Using stopping set analysis, they analyzed the performance of regular LDPC codes in the high-rate regime and found thresholds below than given by density evolution [102]. Then, they generalized their results to the reconstruction of strictly sparse signals in compressed sensing using both uniform reconstruction and non-uniform (randomized) reconstruction. For uniform reconstruction, stopping set analysis and for non-

uniform reconstruction, density evolution were used to evaluate the performance of these reconstruction compressed sensing systems. They also showed that randomized reconstruction compressed sensing has linear-time reconstruction for strictly sparse signals.

In this algorithm each variable node can have two states; one *unverified state* (no value has yet been estimated), and one *verified state* (the variable node has been estimated). Note that when a variable has been verified, it cannot be changed anymore. This algorithm is summarized in the following steps and is described in the Algorithm 2:

- **Step** 1. Variable nodes which are the neighbors of zero-value measurement nodes are verified as 0.
- **Step** 2. Variable nodes connected to measurement nodes with degree one (only one edge connected) are verified as the value of the measurement node.
- **Step** 3. A single variable node connected to two measurement nodes with the same measurement value is verified to the common value of the measurement nodes.
- **Step** 4. Subtract the values of the verified variable nodes from the neighboring measurement nodes and then remove all verified variable nodes and edges connected to them.

Steps 1 to 4 are repeated until reconstruction succeeds or no more progress can be done. The Fig. 3.2(a) 3.2(a) and 3.2(c) sketch the Steps 1, 2 and 3. Note that the verification decoding was originally introduced for the q-ary Symmetric Channel [101] . The justification of Step 3 is based on the observation that, over large alphabets, the probability that two independent random numbers are equal (or equivalently two independent measurements) is quite small. This leads to consider that any two common measured values (during decoding) are generated by the same set of non-zero

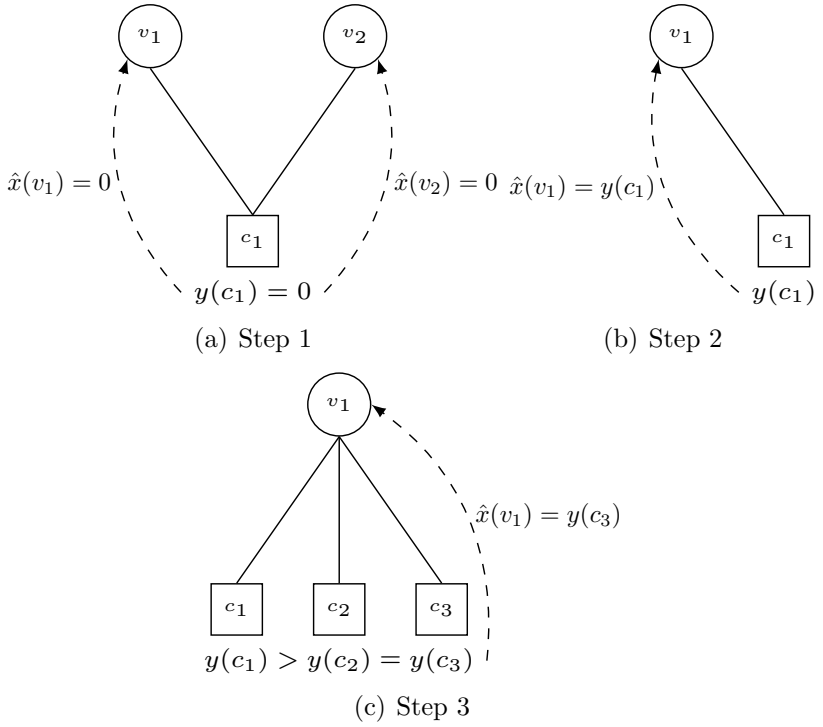


FIGURE 3.2. Steps of the verification decoding algorithm

data. This observation holds for large alphabets and then for the real numbers too. Note that in the Algorithm 2, we added one condition to the Step 3 (Fig. 3.2(b)) to avoid that at one iteration, the value on one measurement node becomes negative. In the Algorithm 2, L is the maximum number of iterations for the decoding, [*] denotes the unverified state and δ is a precision threshold to measurement the correct recovery of our signal (practically $\delta \sim 10^{-3}$).

It is pretty easy to see that for a sparse signal $\mathbf{x}$, and a sparse measurement matrix $\mathbf{A}$, the first step will decrease the number of edges that need to be considered and then the complexity will be drastically reduced from one iteration to another. This very low-complexity reconstruction algorithm will provide a benchmark in terms of recovery performance, and we will then be able to estimate the performance of the low-complexity algorithm presented in the next chapter.

Algorithm 4: Verification Decoding

Input : $\mathbf{y}$ and $\mathbf{A}$ such that $\mathbf{y} = \mathbf{Ax}$, L, δ.
Output: $\hat{\mathbf{x}}$ the estimate of $\mathbf{x}$.
$l = 0$;
$\hat{x} = [*, \ldots, *]$;
while $\mathbb{E}\left[|\hat{\mathbf{x}} - \mathbf{x}|^2\right] > \delta$ *or* $l < L$ **do**
1 **foreach** c_j *such that* $y(c_j) = 0$ **do**
$\hat{x}\left(\mathcal{N}(c_j)\right) = 0$;
end
2 **foreach** c_j *such that* $CheckDegree(c_j) = 1$ **do**
$\hat{x}\left(\mathcal{N}(c_j)\right) = y(c_j)$;
$y\left(\mathcal{N}(v_i)\right) = y\left(\mathcal{N}(v_i)\right) - y(c_j)$ with $v_i = \mathcal{N}(c_j)$;
end
3 **foreach** (c_j, c_k) *such that* $|\mathcal{N}(c_j) \cap \mathcal{N}(c_k)| = 1$ *and* $y(c_j) = y(c_k)$ **do**
$v_i = \mathcal{N}(c_j) \cap \mathcal{N}(c_k)$;
if $y\left(\mathcal{N}(v_i)\right) - y(c_j) \geq 0$ **then**
$\hat{x}(v_i) = y(c_j)$;
$y\left(\mathcal{N}(v_i)\right) = y(\mathcal{N}(v_i)) - y(c_j)$;
end
end
4 **foreach** v_i *such that* $\hat{x}(v_i)$ *is verified* **do**
$A\left(\mathcal{N}(v_i), v_i\right) = 0$;
end
$l = l + 1$;
end

4. INTERVAL-PASSING ALGORITHM: DESCRIPTION, ANALYSIS AND APPLICATIONS

After the review done in the previous chapter on the compressed sensing and the presentation of some reconstruction algorithms we present in this chapter the low-complexity message-passing algorithm developed for the reconstruction of sparse signal, called interval-passing algorithm. This algorithm uses sparse non-negative measurement matrices and has the advantage to perform at a very low complexity, and its performance is very decent compared to some other complex reconstruction methods.

We first provide a complete description of this algorithm with some simulation results and then we provide an complete analysis of it. We also present the link between the reconstruction failures of this algorithm and some particular structures of the graphical representation of the measurement matrix. Finally we present an application of compressed sensing in which this this algorithm can be used.

4.1. Interval-passing algorithm

Chandar *et al.* [103] introduced a simple message passing algorithm for reconstructing non-negative signals using sparse binary measurement matrices. We modified this algorithm in order to deal with non-negative real-valued measurement matrices and refer to it as interval-passing algorithm (IPA) [104–106]. From [103], the complexity of the algorithm is $\mathcal{O}(n(\log(\frac{n}{k}))^2 \log(k))$ which is a good trade-off between the polynomial complexity of the LP reconstruction, and the linear complexity of the simple verification decoding [102] presented in the last chapter.

4.1.1. Description of the algorithm

The IPA is an iterative algorithm, and thus messages are associated with the graphical representation of the measurement matrix to perform the reconstruction. Let $V = \{v_1, v_2, ..., v_n\}$ and $C = \{c_1, c_2, ..., c_m\}$ be respectively the sets of variable nodes and measurement nodes[1] in the graphical representation of the measurement matrix $\mathbf{A} = \{a_{j,i}\}$ for $1 \leq j \leq m$ and $1 \leq i \leq n$. The graphical representation of $\mathbf{A}$ is actually the Tanner graph [16] of the binary image[2] of $\mathbf{A}$, whose edges are labeled by the real values corresponding to the non-zero position in $\mathbf{A}$. The graphical representation of $\mathbf{A}$ has the flavor of a non-binary LDPC code Tanner graph [107, 108].

The messages passing through the edges are intervals $[\mu, M]$ corresponding to the lower and upper bounds of the estimated variable node. At each iteration l, the message update from the variable v_i to the measurement node c_j is given by:

$$\mu_{v_i \to c_j}^{(l)} = \max_{c'_j \in \mathcal{N}(v_i)} \left(\mu_{c'_j \to v_i}^{(l-1)} \right) \times a_{j,i} \tag{4.1}$$

$$M_{v_i \to c_j}^{(l)} = \min_{c'_j \in \mathcal{N}(v_i)} \left(M_{c'_j \to v_i}^{(l-1)} \right) \times a_{j,i} \tag{4.2}$$

and the messages from node c_j to node v_i are updated as:

$$\mu_{c_j \to v_i}^{(l)} = \max \left\{ 0, \frac{y_j - \sum_{v'_i \in \mathcal{N}(c_j) \setminus \{v_i\}} M_{v'_i \to c_j}^{(l)}}{a_{j,i}} \right\} \tag{4.3}$$

$$M_{c_j \to v_i}^{(l)} = \frac{y_j - \sum_{v'_i \in \mathcal{N}(c_j) \setminus \{v_i\}} \mu_{v'_i \to c_j}^{(l)}}{a_{j,i}} \tag{4.4}$$

where $\mathcal{N}(v_i)$ (resp. $\mathcal{N}(c_j)$) is the set of measurement (resp. variable) nodes which are the neighbors of v_i (resp. c_j) in the Tanner graph of $\mathbf{A}$. Updating messages from a variable (resp. measurement) node to a measurement (resp. variable) node is shown in Fig. 4.1 and 4.2, respectively.

[1] Analogous to check nodes in LDPC codes.

[2] A matrix $\mathbf{H} = \{h_{j,i}\}$ is said to be the binary image of a matrix $\mathbf{A} = \{a_{j,i}\}$ if $h_{j,i} = 1$ if $a_{j,i} \neq 0$ and $h_{j,i} = 0$ if $a_{j,i} = 0$

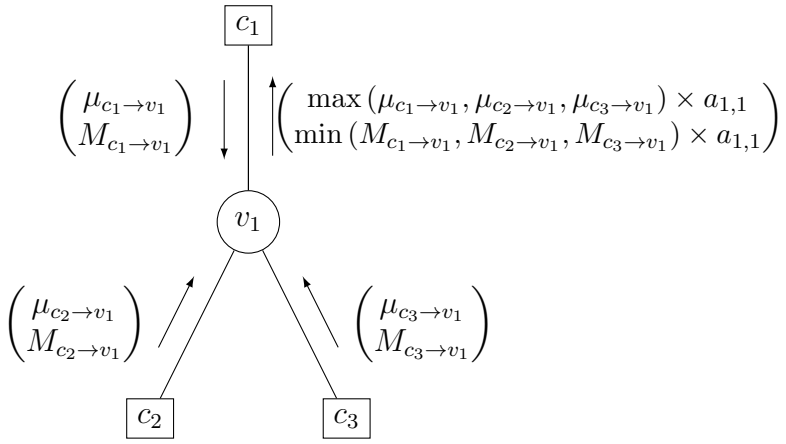


FIGURE 4.1. IPA: Updating messages from the variable node v_1 to the measurement node c_1.

The IPA is formally given in the Algorithm 5, where L represents the maximum number of reconstruction iterations. So far we consider the reconstruction in a noise free environment. The reconstruction process stops when the maximum number of iterations is reached, or the lower bound and the upper bound of the interval from variable nodes to measurement nodes has converged to a common value for every variable node. This common value is set as the estimate of each connected variable node value. When the lower bound and he upper bound did not converge, we arbitrarily set the estimate value to the lower bound.

4.1.2. Simulation results

4.1.2.1. Performance comparisons

We now compare the IPA performance using the non-negative real valued measurement matrix with the complex LP reconstruction algorithm, and with simple verification decoding. We used the LDPC matrix design from an array of permutation matrices from [62] to design our measurement matrices as quasi-cyclic (QC) LDPC matrices. We designed a $(2, 3)$-regular LDPC matrix with $m = 159$ and $n = 265$ and

Algorithm 5: Interval-Passing Algorithm

Input : $\mathbf{y}$ and $\mathbf{A}$ such that $\mathbf{y} = \mathbf{Ax}$, L.
Output : $\hat{\mathbf{x}}$ the estimate of $\mathbf{x}$.
Initialization: $\forall c_j \in C, \forall v_i \in \mathcal{N}(c_j)$, $\mu^{(0)}_{c_j \to v_i} = 0$ and $M^{(0)}_{c_j \to v_i} = y(c_j)/a_{j,i}$;
for $l = 1$ *to* L **do**

1 **foreach** $v_i \in V$ **do**
foreach $c_j \in \mathcal{N}(v_i)$ **do**

$$\mu^{(l)}_{v_i \to c_j} = \max_{c'_j \in \mathcal{N}(v_i)} (\mu^{(l-1)}_{c'_j \to v_i}) \times a_{j,i};$$

$$M^{(l)}_{v_i \to c_j} = \min_{c'_j \in \mathcal{N}(v_i)} (M^{(l-1)}_{c'_j \to v_i}) \times a_{j,i};$$

end
end

2 **foreach** $c_j \in C$ **do**
foreach $v_i \in \mathcal{N}(c_j)$ **do**

$$\mu^{(l)}_{c_j \to v_i} = \frac{1}{a_{j,i}} \left(y(c_j) - \sum_{\substack{v'_i \in \mathcal{N}(c_j) \\ v'_i \neq v_i}} M^{(l)}_{v'_i \to c_j} \right);$$

if $\mu^{(l)}_{c_j \to v_i} < 0$ **then**
$\mu^{(l)}_{c_j \to v_i} = 0$;
end

$$M^{(l)}_{c_j \to v_i} = \frac{1}{a_{j,i}} \left(y(c_j) - \sum_{\substack{v'_i \in \mathcal{N}(c_j) \\ v'_i \neq v_i}} \mu^{(l)}_{v'_i \to c_j} \right);$$

end
end

3 **for** $v_i \in V$ **do**
if $(l > 1$ & $\mu^{(l)}_{v_i \to \mathcal{N}(v_i)} = M^{(l)}_{\mathcal{N}(v_i) \to v_i}) \parallel l = L$ **then**
$\hat{x}(v_i) = \mu^{(l)}_{v_i \to \mathcal{N}(v_i)}$;
end
end
end

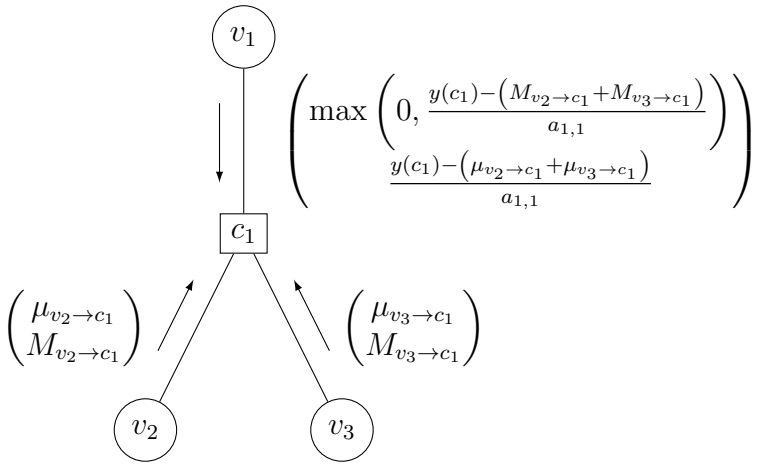


FIGURE 4.2. IPA: Updating messages from the measurement node c_1 to the variable node v_1.

then substituted the non-zero elements in the LDPC matrix by a random number drawn from the uniform distribution in $(0, 1)$ to obtain the matrix $\mathbf{A}$.

For each sparsity k, at least 75 random k-sparse signals $\mathbf{x}$ are generated and 50 reconstruction iterations are performed. The proportion of correct reconstruction results is summarized in the plot of the Fig. 4.3 for the IPA, the verification decoding algorithm, and the LP reconstruction. A random k-sparse vector is said to be correctly recovered if each of its n samples is correctly estimated as close as 10^{-6}. We can see then that the IPA is a good trade-off between performance and complexity.

4.1.2.2. Influence of the rate

In this section we present the reconstruction performance of the IPA by using different measurement matrices that are still QC-LDPC matrices of lengths 120, 1800 with column weight $d_v = 4$. We still used the method from [62] to design different measurement matrices. For each length we design 4 measurement matrices which have different rates, $R = 0.2$, $R = 0.33$, $R = 0.5$, $R = 0.67$. The results in terms of proportion of correct reconstruction using noise-free measurements for each measure-

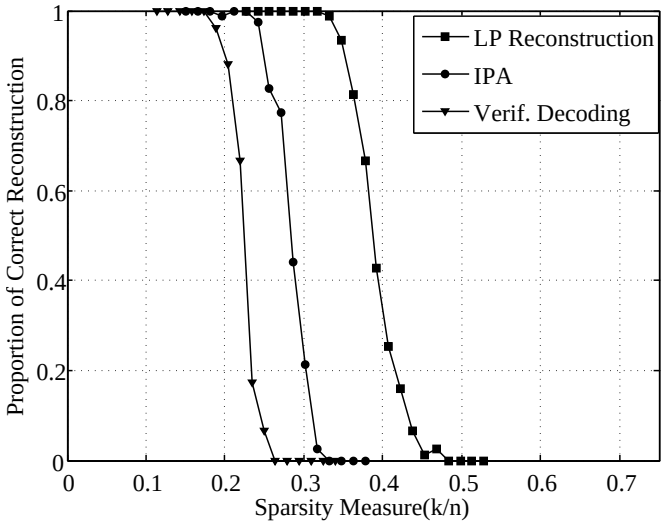
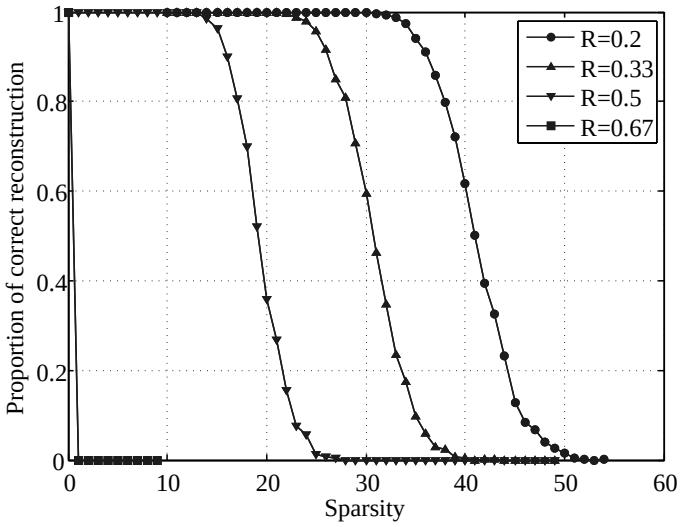


FIGURE 4.3. Simulation results using the designed measurement matrix $\mathbf{A}$.

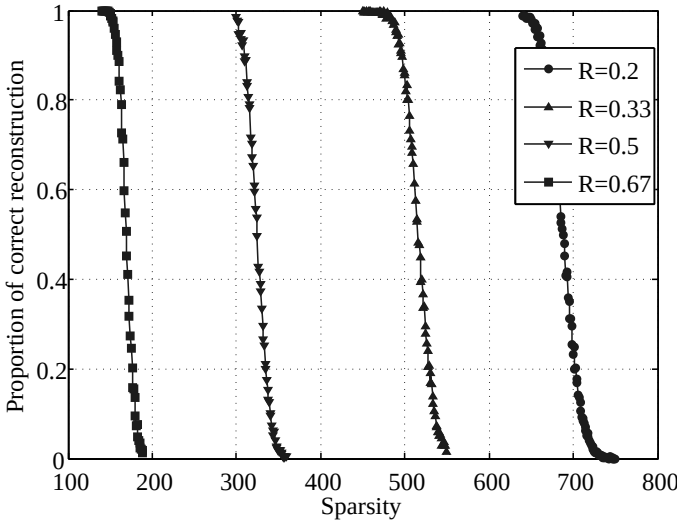
ment matrix are shown in Fig. 4.4. We observe, as expected, that the reconstruction performance decreases when the rate of the code increases. Indeed from the channel coding side, we know that increasing the rate decreases the redundancy present in the code. From the compressed sensing point of view, increasing the rate of the measurement matrix leads to obtain less measurements, which means less linear combinations of the input which intuitively leads to a harder problem to solve. Moreover, the results show that the measurement matrix of length $n = 120$ and rate $R = 0.67$ cannot be a good candidate to reconstruct data, as in a noise-free environment it cannot even recover a 1-sparse vector.

4.1.2.3. Performance comparisons with the approximate message-passing algorithm

In the previous chapter we have presented the AMPA [92]. In this section we compare the reconstruction performance of the IPA and the AMPA. As mentioned in [92], the AMPA is suitable when the entries $a_{i,j}$ of the measurement matrix $\mathbf{A}$ are independent and identically distributed. The measurements need then to be acquired randomly to be able to use the AMPA. It has been shown that the reconstruction possibilities



(a) $n = 120$



(b) $n = 1800$

FIGURE 4.4. Reconstruction of noise free measurements with QC-LDPC measurement matrices of lengths $n = 120$ and $n = 1800$ and column weight $d_v = 4$ for different rates.

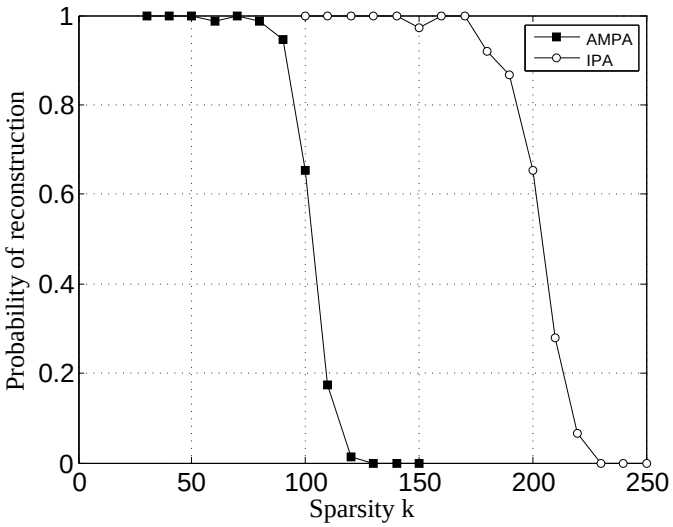


FIGURE 4.5. Reconstruction results using the AMPA and the IPA with parameters $n = 1008$, $m = 504$.

can be established through the phase-transition diagram which basically provide the bounds on k, n, m so that the AMPA is able to recover k-sparse signals.

As in the previous section, we perform 75 reconstructions for each sparsity k and display the proportion of correct reconstruction as a function of k. A vector is said to be correctly recovered if each x_j $j = 1, \ldots, n$ is recovered with absolute value no greater than 10^{-6} for the IPA reconstruction and no greater than 10^{-3} for the AMPA reconstruction. The AMPA does not allow, in general, for perfect recovery (in the sense of the IPA) which explains why we relax the criterion to 10^{-3}.

We first report the reconstruction results in the noise free case using $n = 1008$ and a fixed $m = 504$ in Fig. 4.5. This gives a compression ratio of 2. The measurement matrix for the IPA was chosen to be the one from the Gallager codes of rate half and column-weight-three in [109]. One can see that the exact recovery can be achieved up to a sparsity of 175 for the IPA, whereas the AMPA is only accurate enough for a sparsity of 80, even with the relaxed criterion for correctness.

As a second example we report the comparison of the AMPA and the IPA for a

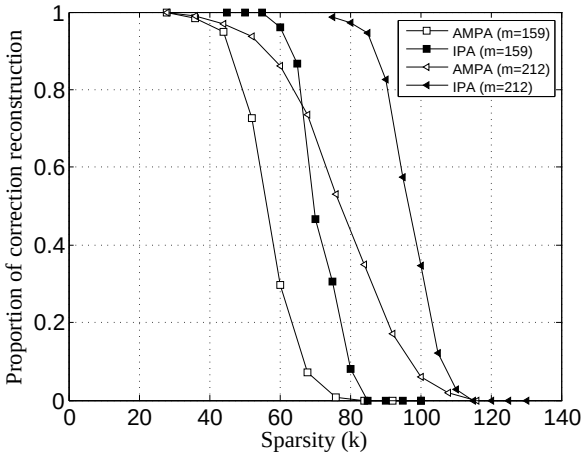


FIGURE 4.6. Reconstruction results using AMPA and IPA with parameters $n = 265$ for different values of m.

fixed signal length $n = 265$ and two different values for m. The random matrices for the AMPA were designed as previously. The deterministic matrices for IPA are described in what follows. These measurement matrices $\mathbf{M}$ originate from the code construction technique of structured LDPC matrices [62] where each matrix $\mathbf{M}$ is a block matrix of circulants. We use binary measurement matrices for the sake of simplicity. The use of non-negative measurement matrices give similar reconstruction results. For the first example, let $p = 53$ and let:

$$\mathbf{M_1} = \begin{pmatrix} \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} \\ \mathbf{I_0} & \mathbf{I_{14}} & \mathbf{I_1} & \mathbf{I_{36}} & \mathbf{I_{37}} \\ \mathbf{I_0} & \mathbf{I_3} & \mathbf{I_4} & \mathbf{I_{38}} & \mathbf{I_{42}} \end{pmatrix} \tag{4.5}$$

where $\mathbf{I}_j$ denotes the $p \times p$ identity matrix circularly shifted by a factor j. This measurement matrix $\mathbf{M_1}$ is of size $m \times n = 3p \times 5p = 159 \times 265$. Similarly, the second matrix $\mathbf{M_2}$ we use here is of size $m \times n = 4p \times 5p = 212 \times 265$ and is given by:

$$\mathbf{M_2} = \begin{pmatrix} \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} \\ \mathbf{I_0} & \mathbf{I_{34}} & \mathbf{I_{39}} & \mathbf{I_{49}} & \mathbf{I_{48}} \\ \mathbf{I_0} & \mathbf{I_{25}} & \mathbf{I_9} & \mathbf{I_{27}} & \mathbf{I_{51}} \\ \mathbf{I_0} & \mathbf{I_{28}} & \mathbf{I_4} & \mathbf{I_7} & \mathbf{I_{15}} \end{pmatrix}. \tag{4.6}$$

We can see in Fig. 4.6 that the reconstruction performance for the IPA is always superior to the performance for the AMPA. For instance, the AMPA exhibits failures

whenever $k \geq 30$ for both choices of the number of measurement taken. On the other hand, the IPA does not exhibit failures until $k \geq 55$ and $m = 159$, and $k \geq 75$ for $m = 212$. The improvement in performance appears to be slightly less that the example of Fig. 4.5. The failures of the IPA (studied in the next chapter) as well as the phase transition diagram of the AMPA could also explain this difference.

4.2. Reconstruction analysis

In this section, we study the recovery of the IPA on non-negative real-valued signals. We present the analysis on binary measurement matrices for the sake of clarity, but the extension of these results to non-negative real-valued measurement matrices is straightforward. First, we give a theorem given in [104] which proves the failure of the IPA on stopping sets. A stopping set is defined as follows.

Definition 12 ([110]). *A stopping set T is a subset of the set of variable nodes V such that all neighbors of T are connected to T at least twice.*

The cardinality of a stopping set is called the size of the stopping set.

Theorem 5 ([104]). *Let $\mathbf{A}_{\mathbf{m}\times\mathbf{n}}$ be a binary measurement matrix. The IPA fails on the recovery of a signal $\mathbf{x}$ if the non-zero entries contain a stopping set in $\mathbf{A}$.*

Proof. We prove that if all variable nodes in T have non-zero values, the IPA cannot recover them. In other words, we prove that the bounds of the intervals passing through the edges of the graphical representation of $\mathbf{A}$ never converge, i.e. we show that $\forall v \in T$ such that $x(v) > 0$, then $\mu_{v\to c}^{(l)} < x(v) < M_{v\to c}^{(l)}$, $\forall l \geq 0, \forall c \in \mathcal{N}(v)$.
Suppose $\forall v \in T$, $x(v) > 0$, from the definition of a stopping set $\forall c \in \mathcal{N}(v)$, $y(c) > x(v)$. Then, at the initialization ($l = 0$) we have:

$$\mu_{c\to v}^{(0)} = 0 < x(v) < y(c) = M_{c\to v}^{(0)} \tag{4.7}$$

At the first iteration we have:

$$\mu_{v\to c}^{(1)} = \max_{c\in\mathcal{N}(v)} \left(\mu_{c\to v}^{(0)}\right) = 0 < x(v) \tag{4.8}$$

$$M_{v\to c}^{(1)} = \min_{c\in\mathcal{N}(v)} \left(M_{c\to v}^{(0)}\right) > x(v) \tag{4.9}$$

Now consider the update at the measurement node. For the lower bound of the interval we have:

$$\begin{aligned} \mu_{c\to v}^{(1)} &= \max\left(0, y(c) - \sum_{v'\in\mathcal{N}(s)\setminus\{v\}} M_{v'\to c}^{(1)}\right) && (4.10)\\ &< \left(y(c) - \sum_{v'\in\mathcal{N}(s)\setminus\{v\}} x(v')\right) && (4.11)\\ &= x(v) && (4.12) \end{aligned}$$

The last equation stands from the initialization $x(v) < y(c) = M_{v\to c}^{(1)}$. Similarly, for the upper bound at the measurement node we have:

$$\begin{aligned} M_{c\to v}^{(1)} &= y(c) - \sum_{v'\in\mathcal{N}(s)\setminus\{v\}} \mu_{v'\to c}^{(1)} && (4.13)\\ &> y(c) - \sum_{v'\in\mathcal{N}(s)\setminus\{v\}} x(v') && (4.14)\\ &= x(v) && (4.15) \end{aligned}$$

The last equation results from the initialization $x(v) > y(c) = \mu_{v\to c}^{(1)}$.

Thus we obtain $\mu_{c\to v}^{(1)} < x(v) < M_{c\to v}^{(1)}$. For $l > 1$ the messages from variable nodes to check nodes are simply the intersection of intervals from the check nodes at the previous iteration, and then we still have $\mu_{v\to c}^{(l)} < x(v) < M_{v\to c}^{(l)}$. The proof is completed by induction for every $l > 0$. □

Theorem 5 also indicates that the IPA fails on reconstruction of a signal $\mathbf{x}$ whose non-zero values form the smallest stopping set in the measurement matrix $\mathbf{A}$. However, as we explain in the following example, the smallest stopping set is not the smallest configuration on which the IPA fails.

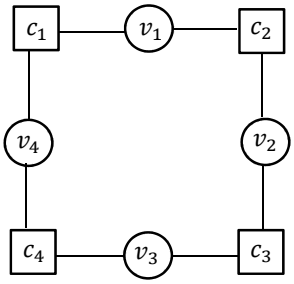


FIGURE 4.7. A stopping set of size 4.

Example 4. *Consider a stopping set of size four as given in Fig. 4.7. According to Theorem 5, the IPA cannot recover a signal* $\mathbf{x}$ *with non-zero values on* $\{v_1, v_2, v_3, v_4\}$. *The algorithm also fails on a 2-sparse signal* $\mathbf{x}$ *whose non-zero values are* $\{v_1, v_3\}$ *or* $\{v_2, v_4\}$, *which implies that the variable nodes forming one of the smallest stopping set are not necessarily the smallest configuration on which the IPA fails. However, the IPA can recover a 2-sparse signal with non-zero values on* $\{v_1, v_2\}$ *or* $\{v_1, v_4\}$.

Example 4 shows the main difference between the iterative decoding on the BEC and the signal recovery of the iterative IPA in compressed sensing. The iterative decoder over the BEC fails if and only if the erasures contain a stopping set, while the IPA fails even if the non-zero values do not involve a stopping set. The following results identify recoverable signals whose non-zero values are subsets of stopping sets. First, we show that every zero-value variable node is recoverable by the IPA. We say that a node is zero if its value is equal to zero.

Lemma 1. *The IPA can recover all zero variable nodes.*

Proof. Suppose v is a variable node with value 0 and $\{c_1, c_2, ..., c_{d_v}\}$ is the set of the d_v measurement neighboring nodes of v with measurement values $\{\alpha_1, \alpha_2, ..., \alpha_{d_v}\}$. At each iteration of the IPA, the message which is sent from c_j $(j = 1, ..., d_v)$ to v is either $[0, 0]$ or $[0, \beta_j]$ where $0 < \beta_j \leq \alpha_j$. If v receives at least one $[0, 0]$ from one of its neighbors, the value of v is recovered as 0. If all messages from every c_j to v

are $[0, \beta_j]$, the decision rule of the algorithm leads to recover the value of v to the maximum value of lower bounds of the intervals $[0, \beta_j]$, which is 0. □

Since all zero variable nodes are recovered by the IPA, it is enough to study the recovery of non-zero variable nodes. We note that reconstruction of all zero variable nodes does not necessarily result that the IPA can be used to estimate the support of the signal, since it is possible that the IPA fails to recovery of a non-zero variable node and recovers the value of the non-zero variable node to zero.

Definition 13. *A set of variable nodes S is called a minimal stopping set, if S forms a stopping set and it does not contain a smaller stopping set.*

It is clear that the smallest stopping set in a measurement matrix $\mathbf{A}$ is a minimal stopping set while a minimal stopping set is not necessarily the smallest stopping set. The size of the smallest stopping set is called the *stopping distance* [111] and plays a significant role in iterative decoding of LDPC codes over the BEC.

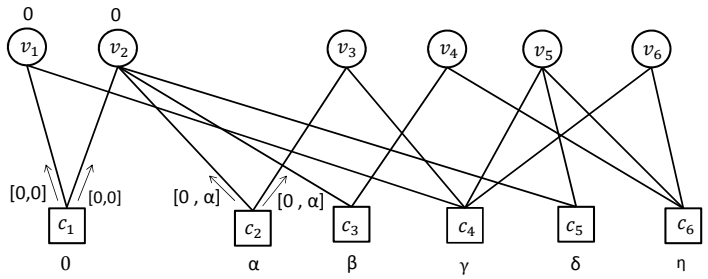
Theorem 6. *Let $\mathbf{A_{m\times n}}$ be a binary measurement matrix and $V_S = \{v_1, v_2, ..., v_k\}$ be a subset of variable nodes forming a minimal stopping set. Let $\mathbb{R}^n_{\geq 0}$ be a set of non-negative real vectors in $\mathbb{R}^n$ and $\mathbf{x} = [x_1, x_2, ..., x_n]^t \in \mathbb{R}^n_{\geq 0}$ be a signal with at most $k-2$ non-zero values, i.e. $\| \mathbf{x} \|_0 \leq k-2$, such that the set of non-zero variables is a subset of V_S. Then, the IPA can recover $\mathbf{x}$ if there exists at least one zero measurement node among the neighbors of V_S.*

Proof. Let $M = \{c_1, c_2, ..., c_q\}$, $1 \leq q \leq m$ be the set of measurement nodes connected to V_S. Suppose that c_1 is the only zero measurement node. Since V_S forms a stopping set, there exist at least two zero variable nodes, say v_1, v_2, connecting to c_1 and there exist non-zero value measurement nodes connected to v_1, v_2. Moreover, V_S is a minimal stopping set, so there exists at least one measurement node, $c_2 \in M \backslash \{c_1\}$ with only one neighbor in $V_S \backslash \{v_1, v_2\}$, namely v_3. Otherwise, $V_S \backslash \{v_1, v_2\}$ will be a smaller stopping set, which is a contradiction. Suppose c_2 has the value α. At the

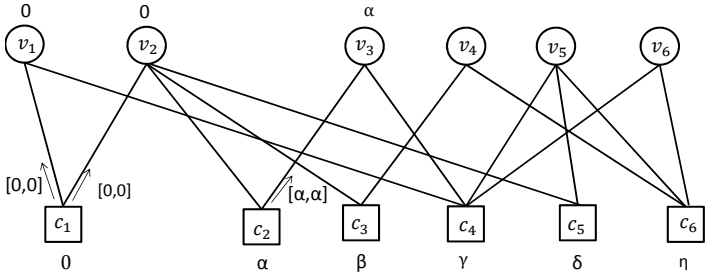
first iteration, c_1 sends $[0,0]$ to v_1 and v_2 and v_1 and v_2 send $[0,0]$ to their neighbors. Based on the above explanation, c_2 is a neighbor of v_1 or v_2 or both of them with only one neighbor in $V_S \setminus \{v_1, v_2\}$. At the second iteration, c_2 will send $[\alpha, \alpha]$ to the only variable node v_3. So, the value of the variable node v_3 is recovered to α. Since the value of this variable node is recovered, we can consider this variable node to have value zero by subtracting α from the value of all measurement nodes which are the neighbors of v_3. Now, we have three variable nodes v_1, v_2, v_3 whose values have been determined. Again, with the same discussion, there exists a measurement node $c_3 \in M \setminus \{c_1, c_2\}$ with only one neighbor in $V_S \setminus \{v_1, v_2, v_3\}$ which can be recovered in the next iteration. Continuing the same process will recover all variable nodes. □

Example 5. *Fig. 4.8 illustrates the previous theorem considering the recovery of a signal with 4 non-zero variable nodes in a minimal stopping set of size 6 in which there exists one zero measurement node c_1 and two zero variable nodes v_1 and v_2. Let $\alpha, \beta, \gamma, \delta, \eta$ be the non-zero values of c_2, c_3, c_4, c_5 and c_6, respectively. Note that c_2 is one the measurement nodes with exactly one neighbor among $\{v_3, v_4, v_5, v_6\}$. At initialization, the zero measurement node c_1 sends $[0,0]$ to v_1 and v_2. And c_2 sends $[0, \alpha]$ to v_2 and v_3. At the first iteration, v_2 sends $[0,0]$ to c_2, which causes c_2 to send $[\alpha, \alpha]$ to v_3. Thus, the value of v_3 is recovered as α. Another measurement node with only one neighbor in $\{v_4, v_5, v_6\}$ is c_3. Again this measurement node sends $[\beta, \beta]$ to v_4 and so the value of v_4 is recovered as β. The same process results in the recovery of all variable nodes.*

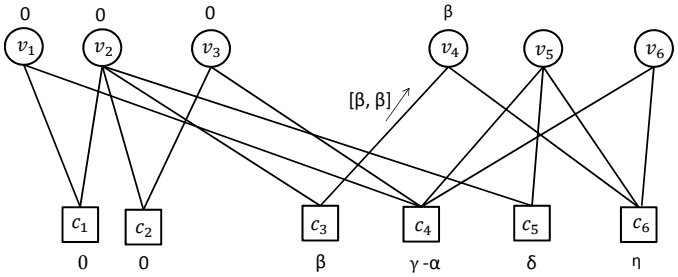
In Theorem 6 we proved that the existence of at least one zero measurement node is enough to reconstruct a signal $\mathbf{x}$ whose non-zero values are a subset of a minimal stopping set. As we will show in the following Lemma and Corollary, in regular measurement matrices we can give an upper bound on the number of variable nodes forming a subset of a minimal stopping set S such that there exists at least



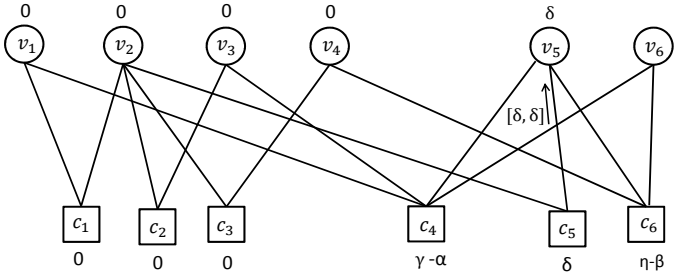
(a) First messages from measurement nodes to variable nodes



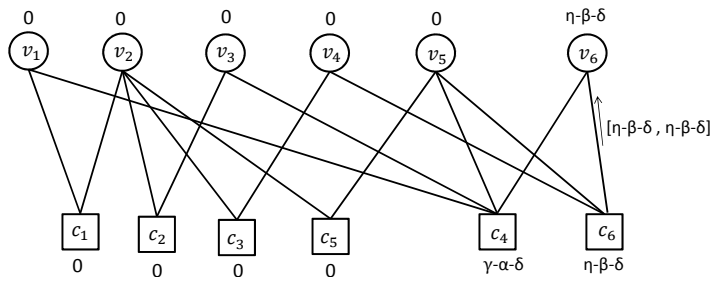
(b) Messages from measurement nodes to variable nodes in the first iteration (recovery of v_3)



(c) Message to v_4 in the second iteration (recovery of v_4)



(d) Message to v_5 in the third iteration (recovery of v_5)



(e) Message to v_6 in the fourth iteration (recovery of v_6)

FIGURE 4.8. Signal reconstruction with non-zero elements in a minimal stopping set of size 6 with the IPA when there exists one zero measurement node.

one measurement node among the neighbors of S with no connection to this subset of variable nodes. This result shows that in a minimal stopping set, if the number of non-zero values is bounded by a fixed number, the IPA can recover the signal $\mathbf{x}$.

Lemma 2. *Let $\mathbf{A_{m\times n}}$ be a binary (d_v, d_c)-regular measurement matrix and let $g = 2r$ be the girth of the Tanner graph corresponding to $\mathbf{A}$. Then, every subset N of variable nodes such that*

$$|N| < \frac{\sum_{i=0}^{r-1}(d_v-1)^{\lceil \frac{i}{2} \rceil}(d_c-1)^{\lfloor \frac{i}{2} \rfloor}}{d_v} \tag{4.16}$$

contains a measurement node which does not have any neighbor in N.

Proof. The proof is obtained by using lower bounds on the number of measurement nodes given in [112]. According to this bound,

$$m \geq \sum_{i=0}^{r-1}(d_v-1)^{\lceil \frac{i}{2} \rceil}(d_c-1)^{\lfloor \frac{i}{2} \rfloor}. \tag{4.17}$$

Now, suppose N is a subset of variable nodes. Since the matrix is regular, the maximum number of measurement nodes that can be connected to the variable nodes in N is at most $|N|d_v$. If $|N|d_v < \sum_{i=0}^{r-1}(d_v-1)^{\lceil \frac{i}{2} \rceil}(d_c-1)^{\lfloor \frac{i}{2} \rfloor}$ which results $|N| < \frac{\sum_{i=0}^{r-1}(d_v-1)^{\lceil \frac{i}{2} \rceil}(d_c-1)^{\lfloor \frac{i}{2} \rfloor}}{d_v}$, there exists at least one measurement node which does not have any connection to N. □

Corollary 1. *Suppose* $\mathbf{A_{m\times n}}$ *is a binary* (d_v, d_c)*-regular measurement matrix with girth* $g = 2r$*. Let* N *be a subset of* k *variable nodes that forms a minimal stopping set. If* $|N| < \frac{\sum_{i=0}^{r-1}(d_v-1)^{\lceil \frac{i}{2} \rceil}(d_c-1)^{\lfloor \frac{i}{2} \rfloor}}{d_v}$*, then the IPA can recover a signal* $\mathbf{x}$ *with non-zero values in* N*. In the case that the girth is 6,* $|N|$ *is bounded by* $|N| \leq \frac{(d_v-1)d_c}{d_v} = \frac{(d_v-1)k}{m}$*. If the girth is 8,* $|N|$ *is bounded by* $|N| \leq \frac{(d_v-1)d_c+(d_v-1)(d_c-1)}{d_v}$*.*

The following theorem gives a sufficient condition on exact recovery of a signal whose support is a subset of a minimal stopping set and all neighboring measurement nodes are non-zero.

Theorem 7. *Let* $\mathbf{A_{m\times n}}$ *be a binary measurement matrix and* $V_S = \{v_1, v_2, ..., v_k\}$ *be a subset of variable nodes forming a minimal stopping set. Let* $\mathbf{x} = [x_1, x_2, ..., x_n]^t \in \mathbb{R}^n_{\geq 0}$ *be a signal with at most* $k-1$ *non-zero values, i.e.* $\| \mathbf{x} \|_0 \leq k-1$ *such that the set of non-zero variables is a subset of* V_S*. Suppose all measurement nodes have non-zero values. Then, the IPA can recover* $\mathbf{x}$ *if*

1. There exists at least one measurement node c_j *such that the variable nodes* $\{v_1, v_2, ..., v_p\}$ *which are connected to* c_j *have non-zero values and do not share a measurement node other than* c_j *and*

2. The measurement nodes $\{c_1, c_2, ..., c_l\}$ *connected to* $\{v_1, v_2, ..., v_p\}$ *do not have non-zero neighboring variable nodes excluding* $\{v_1, v_2, ..., v_p\}$*.*

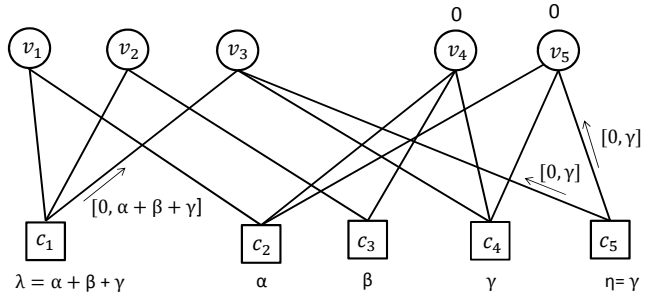
Proof. Suppose $\{v_1, v_2, ..., v_p\}$ have non-zero values $\{\alpha_1, \alpha_2, ..., \alpha_p\}$. Since $\{c_1, c_2, ..., c_l\}$ are connected to zero variable nodes except for $\{v_1, v_2, ..., v_p\}$ and $\{c_1, c_2, ..., c_l\}$ are not shared by more than one variable node in $\{v_1, v_2, ..., v_p\}$, the value of every measurement node in $\{c_1, c_2, ..., c_l\}$ lies in $\{\alpha_1, \alpha_2, ..., \alpha_p\}$. There exists a measurement node in $\{c_1, c_2, ..., c_l\}$ that has only one neighbor in $V_S \backslash \{v_1, v_2, ..., v_p\}$. Otherwise, $V_S \backslash \{v_1, v_2, ..., v_p\}$ will be a smaller stopping set. Without loss of generality, suppose c_1 is a measurement node with this property which is connected to v_1 with the value α_1. In the first iteration, c_j sends $\sum_{i=1}^{p} \alpha_i$ to its neighbors in $\{v_1, v_2, ..., v_p\}$ and c_1

sends $[0, \alpha_1]$ to its neighbors. In the second iteration, c_j sends $[\alpha_1,\ sum_{i=1}^{p}\alpha_i]$ to v_1 and c_1 and other neighbors of v_1 send intervals with the upper bound α_1 to v_1 which results that the message $[\alpha_1, \alpha_1]$ is sent from v_1 to its neighbors. Thus, the value of v_1 is recovered as α_1 and all measurement nodes which are the neighbors of v_1 are satisfied. So, c_1 is satisfied and its value can be considered as zero. The recovery of the other variable nodes is followed by Theorem 6 which implies that the existence of at least one zero measurement node is enough to recover all the variable nodes in a configuration forming a stopping set. □

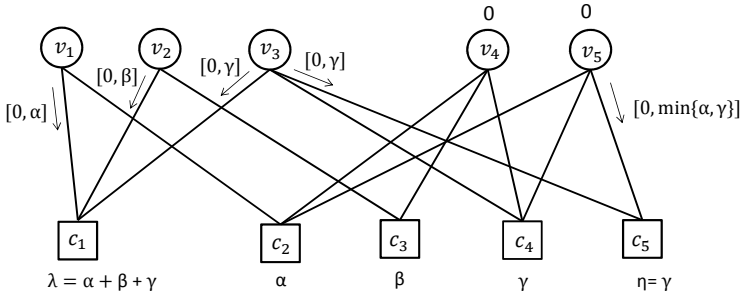
The following example shows how the IPA can recover a vector $\mathbf{x}$ under the conditions of the Theorem 7.

Example 6. *Fig. 4.9 depicts the recovery of the three non-zero variable nodes in* $\{v_1, v_2, v_3\}$ *a minimal stopping set of size 5 in which all measurement nodes are non-zero. First, note that the measurement node* c_1 *and the variable nodes* v_1, v_2, v_3 *satisfy the two conditions of Theorem 7. Thus, if the non-zero variable nodes* v_1, v_2, v_3 *have values* α, β, γ, *then* c_1 *has value* $\lambda = \alpha + \beta + \gamma$ *and other measurement nodes* c_2, c_3, c_4, c_5 *have the values* α, β, γ *and* γ *respectively. For simplicity, we just show how the value of the variable node* v_3 *is recovered. At initialization,* c_1 *sends* $[0, \alpha+\beta+\gamma]$ *to* v_3 *and* c_5 *sends* $[0, \gamma]$ *to* v_3. *In the first iteration,* c_1 *receives* $[0, \alpha]$ *and* $[0, \beta]$ *from* v_1 *and* v_2, *respectively and* c_5 *receives* $[0, \min(\alpha, \gamma)]$ *from* v_5. *Then,* c_1 *sends* $[\gamma, \alpha + \beta + \gamma]$ *to* v_3 *and* c_5 *sends* $[0, \gamma]$ *or* $[\gamma - \min(\alpha, \gamma), \gamma] = [\gamma - \alpha, \gamma]$ *to* v_3. *In the second iteration,* v_3 *sends* $[\gamma, \gamma]$ *to its neighbors which makes* c_5 *is satisfied and can be considered as a zero measurement node. Now, there exists a zero measurement node in this minimal stopping set. Theorem 6 results in the recovery of other variable nodes.*

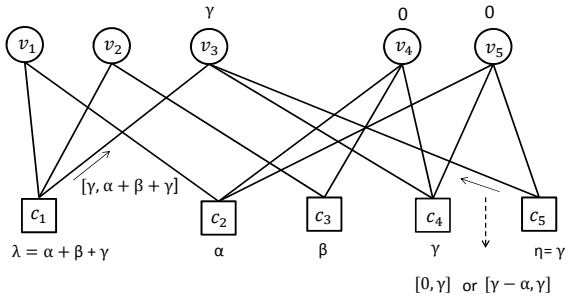
Theorems 6 and 7 give sufficient conditions on the recovery of signals whose non-zero values form a subset of a minimal stopping set. To show how small stopping sets affect the performance of the IPA, we provide simulation results of the performance of the IPA in the next section.



(a) First messages from measurement nodes to variable nodes



(b) Messages from variable nodes in the first iteration



(c) Messages from check nodes in the second iteration

FIGURE 4.9. Signal Reconstruction with non-zero elements in the support of a stopping set of size 5 with IPA and with no zero measurement node.

4.3. Numerical results

4.3.1. Preliminaries

In this section we provide simulation results of the reconstruction performance of the IPA by establishing the link with the analysis done previously related to stopping sets. From Theorem 5, we know that if the support of the signal $\mathbf{x}$ is, or contains, a stopping set, the IPA cannot recover it. The stopping sets in the measurement matrix $\mathbf{A}$ are then responsible for most of the failures of the reconstruction via the IPA.

Orlitsky *et al.* derived a formula to obtain the average distribution $E(s)$ for any size s of stopping sets for a (d_v, d_c)-regular LDPC code of size n [43]:

$$E(s) = \binom{n}{s} \frac{\text{coef}\left(\left((1+x)^{d_c} - xd_c\right)^{\frac{d_v}{d_c}n}, x^{sd_v}\right)}{\binom{nd_c}{sd_v}} \tag{4.18}$$

where $\text{coef}\left(p(x), x^i\right)$ denotes the coefficient of x^i in the polynomial $p(x)$. The average distribution of stopping sets can theoretically be computed using the previous formula, however it has to be done by numerical methods due to its large complexity. To be able to practically find stopping sets, algorithms have been proposed as in [57] and [113] in the channel coding context. In their recent work, Rosnes *et al.* [57, 114] provide the stopping set distribution of various LDPC codes based on their algorithm to find small stopping sets. In [114] they focused more specifically on the LDPC codes from the IEEE 802.16e standard [115], referred to as the Wimax codes. These codes are circulant-based LDPC codes, and the IEEE standard provides the design of codes for 19 different lengths. Also, one model matrix to design codes with rates 1/2, and 5/6 is provided, and two model matrices are provided for codes with rates 2/3 and 3/4 (denoted by A and B). In the sequel we will use the matrix of the Wimax standard as measurement matrix for two different length, $n = 768$, and $n = 2304$. We remind in the Tables 4.1(a) and 4.1(b) the stopping set spectrum for the codes that can be designed for each length in which we adopted the notation of Rosnes *et al.* [114].

TABLE 4.1. Stopping distance s_{min} of the IEEE802.16 standard LDPC codes with different rates R and length n. The number of stopping sets of weight s_{min} is denoted $N_{s_{min}}$ (from [57] and [114]).

(a) $n = 768$

R	s_{min}	$N_{s_{min}}$	$N_{s_{min}+1}$	$N_{s_{min}+2}$	$N_{s_{min}+3}$	$N_{s_{min}+4}$
1/2	14	32	32	0	32	32
2/3 A	8	64	0	0	160	160
2/3 B	12	64	128	384	1120	3352
3/4 A	9	32	128	576	2192	9696
3/4 B	4	16	0	0	32	216
5/6	6	96	672	5376	36512	280128

(b) $n = 2304$

R	s_{min}	$N_{s_{min}}$	$N_{s_{min}+1}$	$N_{s_{min}+2}$	$N_{s_{min}+3}$	$N_{s_{min}+4}$
1/2	28	96	96	288	288	624
2/3 A	15	96	0	96	480	768
2/3 B	15	96	0	0	0	0
3/4 A	12	48	0	0	0	0
3/4 B	12	16	96	0	672	1824
5/6	9	192	288	1920	8616	43584

4.3.2. Simulation results on the Wimax codes

In order to see the influence of stooping set on the performance of the IPA, we have generated all the codes of length $n = 768$ and $n = 2304$ according to the IEEE standard. We used these six codes for each length as measurement matrices and we simulate the recovery performance via the IPA. The simulation results are shown in Fig. 4.10(a) and 4.10(b) where the proportion of correct reconstruction of sparse signals is plotted versus the sparsity measure k/n. For each sparsity k and for each matrix, 500 k-sparse signals are generated, and a maximum of 50 iterations of the IPA for the reconstruction are done.

These results emphasize the connection between the stopping set distribution and the performance of the IPA from the theorems of the previous section. For instance, it is clear when comparing the numbers of Table 4.1(a) and the plot of Fig. 4.10(a), or between Table 4.1(b) and the Fig. 4.10(b) that the stopping set distribution is

responsible for most of the failures of the IPA. For example in the case of $n = 768$, at a constant rate (e.g. 2/3), the measurement matrix with higher stopping distance has a slightly better performance. However, we can see that even if the matrix A with rate 3/4 has a better stopping distance than the rate 5/6 matrix, it performs better. This observation comes from the stopping set distribution as matrix A with rate 3/4 has only 16 stopping sets of weight 4, and the next ones have weight 7 (and there are a few of them) whereas for the rate 5/6, the number of stopping sets of weight 6 or 7 are very numerous. Then, although the stopping set distribution gives an insight of the performance of the IPA on a given measurement matrix, it is not obvious to foresee this performance because it depends on the stopping distance and on the number of stopping sets of each weight. These observations hold for the case of $n = 2304$.

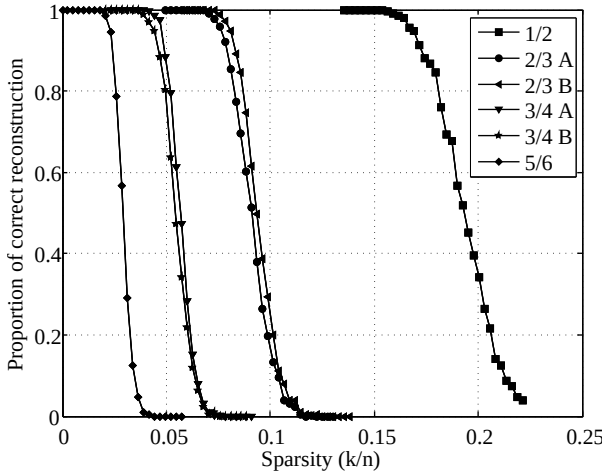
4.4. Applications of compressed sensing

We conclude this chapter by giving a first insight on the applications of compressed sensing which could use the IPA as a reconstruction algorithm. We detail one application related to spectroscopy and give the essence of another related to imaging.

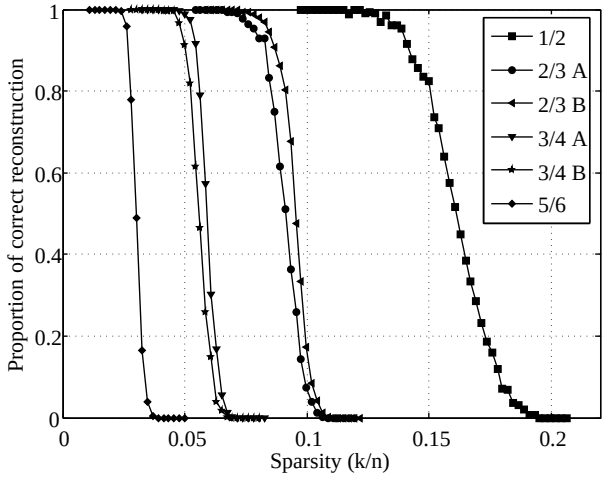
4.4.1. Interval-passing algorithm for chemical mixture estimation

4.4.1.1. Preliminaries

The knowledge of the presence and/or concentration of a particular chemical in a solution or on a surface is crucial in many applications such as detection of explosive devices in defense and security systems [116, 117]. In this section we deal with the mixture estimation problem in Raman spectroscopy, where a spectrum is obtained for a given chemical using a laser to energize its molecules. Once energized, the molecules release a scattered light with a unique spectrum [118]. Specifically, a small fraction



(a) $n = 768$



(b) $n = 2304$

FIGURE 4.10. IPA performance on the IEEE 802.16e LDPC codes for the different available rates and the chosen length.

of the scattered photons have a wavelength different from that of the incoming laser light. Counting the number of photons at each wavelength provides a spectrum for a particular chemical. The spectra of different chemicals are collected in a database, called a chemical library. Traditional spectroscopy-based mixture estimation methods sample the entire spectra of a chemical mixture and use traditional estimation theory to perform the mixture estimation [119]. Another approach has been introduced recently using an adaptive spectrometer together with sequential hypothesis testing [120]. This approach provides a basis for using compressive sensing for chemical mixture estimation.

Compressive sensing aims at recovering sparse signals from a set of linear projections. In the formulation considered here, a vector having entries corresponding to the fraction of each chemical present in a mixture can be seen as a sparse signal to be recovered and the chemical library as a transformation of the chemical mixture vector into the spectrum of the mixture. Practically, a limited number of chemicals are likely to be present among all the chemicals from the library, hence the sparseness of the mixture vector.

In this section we propose the use of an appropriate measurement (projection) matrix for use in the adaptive spectrometer of [120] which allows the recovery of the chemical mixture vector from compressive observations of the spectrum of the mixture. A *global* measurement matrix is chosen to be a sparse non-negative matrix obtained from low-density parity-check (LDPC) code designs. In spectroscopy the number of potential chemicals to be detected may be large, hence the dimension of the chemical mixture vector is often large. For this reason, the usual ℓ_1-norm solution using linear programming (LP) may not be suitable in practice due to its cubic complexity. We use the IPA as reconstruction algorithm as it uses sparse measurement matrices and it is then suited for the chemical mixture problem as it is designed for non-negative sparse vectors. Together with our sparse measurement matrices obtained via LDPC matrix design, the IPA allows perfect recovery of very

high dimensional mixture vectors in the noise free case.

4.4.1.2. The mixture estimation problem

Raman spectroscopy [118] relies on the uniqueness of the proportion of photons scattered inelastically from an energized molecule. A photon is said to be scattered inelastically if its wavelength is different from the wavelength of the incoming energizing laser light. Counting the number of photons with this property at each wavelength for molecules composing a chemical provides a Raman spectrum which can be used as a fingerprint for the identification of the given chemical.

Our Raman spectra estimation approach is supervised and based on the sensor of [120] which acquires compressive measurements. Unlike unsupervised approaches where spectral estimation must be accomplished from measurements without any prior information, our supervised approach assumes complete knowledge of each possible chemical spectrum. In the supervised approach the prior information about the chemicals is helpful for the reconstruction, but this knowledge needs to be acquired beforehand and the considered chemicals need to be carefully chosen. In the unsupervised approach no prior knowledge is required but the measurements process can be tedious and complex in some real time applications.

The mixture estimation problem aims to estimate the proportion of each chemical present in a mixture. As mentioned above, the spectrum of each possible chemical present is assumed. For a given chemical ξ_j, the spectrum is represented by the vector $\mathbf{S}_j = [s_{1,j}, s_{2,j}, ..., s_{n_s,j}]^T$ where $s_{i,j}$ represents the number of photons scattered inelastically for chemical ξ_j in the ith spectral channel, $i = 1, \ldots, n_s$. Organizing these vectors as a matrix of size $n_s \times n_c$ results in a chemical library $\mathbf{L} = [\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k, \ldots, \mathbf{S}_{n_s}]$,

$$\mathbf{L} = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n_c} \\ \vdots & \vdots & \vdots & \vdots \\ s_{n_s,1} & s_{n_s,2} & \cdots & s_{n_s,n_c} \end{pmatrix}.$$

A mixture of chemicals is modeled by a vector $\mathbf{x} = [x_1, ..., x_{n_c}]^T$, in which x_j is the fraction of chemical ξ_j present in the mixture. No chemical outside the library can be present and we assume that only $k \ll n_c$ chemicals are present, which implies $\|\mathbf{x}\|_1 = 1$ and $\|\mathbf{x}\|_0 = k$. The resulting chemical mixture spectrum is modeled by the $n_s \times 1$ vector

$$\mathbf{z} = \mathbf{L}\mathbf{x}. \tag{4.19}$$

Although $\mathbf{x}$ is sparse, $\mathbf{L}$ and $\mathbf{z}$ are not. Indeed a spectrum may not have a single non-zero spectral component.

From the uniqueness of the spectrum for a given chemical, we show in what follows that compressive spectrometer measurements can be obtained from an appropriate choice of measurement matrix. The chemical mixture vector can then be recovered from these measurements using the IPA.

4.4.1.3. Compressive Raman spectroscopy

We now present the proposed compressive sensing scheme. Instead of measuring the vector $\mathbf{z}$ directly, the compressive spectrometer of [120] measures linear combinations of the spectral channels from the mixture spectrum via a measurement matrix $\mathbf{A}$ of size $m \times n_s$. The measurement vector (output of the spectrometer) is denoted by the vector $\mathbf{y}$

$$\mathbf{y} = \mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{L}\mathbf{x} \tag{4.20}$$

of size $m \times 1$. When $m < n_s$, the system is compressive compared to measuring $\mathbf{z}$ directly. In the problem defined above, from the observation vector $\mathbf{y}$, we want to recover $\mathbf{x}$ directly without necessarily recovering $\mathbf{z}$ first. In other words, we formulate the problem as $\mathbf{y} = \mathbf{M}\mathbf{x}$, with $\mathbf{M} = \mathbf{A}\mathbf{L}$. Given that the chemical library $\mathbf{L}$ is fixed and known, by choosing the elements of the matrix $\mathbf{A}$ properly, we can make $\mathbf{M}$ a non-negative sparse matrix which is a necessary condition for use of the IPA for reconstruction. The matrix $\mathbf{A}$ then governs the way that measurements (of the

mixture spectrum) are taken by the spectrometer, while $\mathbf{M}$ governs the recovery of $\mathbf{x}$. If $m > n_c$, Eq. 4.20 is typically over-determined with respect to $\mathbf{x}$, and least square techniques are easily employed. The more interesting case is when $m < n_c$ and the system is undetermined (compressive) with respect to $\mathbf{x}$. This is the case treated in what follows.

As mentioned previously, the IPA has very low computational complexity, but requires that the measurement matrix be sparse. With respect to the mixture vector $\mathbf{x}$, the measurement matrix is $\mathbf{M} = \mathbf{AL}$. The design of a matrix $\mathbf{A}$ to obtain a sparse matrix $\mathbf{M}$ is not a trivial problem. We propose instead to make a choice for $\mathbf{M}$, and then design $\mathbf{A}$ such that the product $\mathbf{AL}$ produces the chosen sparse matrix $\mathbf{M}$. The main idea is to ignore all but n_c spectral channels. The n_c rows in $\mathbf{L}$ corresponding to the n_c selected channels then form a square matrix (denoted $\mathbf{L_s}$). If the columns of $\mathbf{L}$ are independent (which is the case in our example as different chemicals having unique spectra are used), the rank of the matrix $\mathbf{L_s}$ is n_c, meaning that the matrix $\mathbf{L_s}$ is invertible. We then let

$$\mathbf{ML_s^{-1}} = \mathbf{A_s} = \left[\mathbf{A}_{\varphi_1} \dots \mathbf{A}_{\varphi_j} \dots \mathbf{A}_{\varphi_{n_c}} \right]$$

and define $\mathbf{A}$ from the columns of $\mathbf{A_s}$ as:

$$\mathbf{A} = \left[\mathbf{0} \dots \mathbf{0} \mathbf{A}_{\varphi_1} \mathbf{0} \dots \mathbf{0} \mathbf{A}_{\varphi_j} \mathbf{0} \dots \mathbf{0} \mathbf{A}_{\varphi_{n_c}} \mathbf{0} \dots \mathbf{0} \right]$$

where (φ_j) $j = 1, \dots, n_c$ is the sequence of spectral channels which are not disregarded. It is straightforward to verify that $\mathbf{AL} = \mathbf{A_s L_s} = \mathbf{ML_s^{-1} L_s} = \mathbf{M}$ as desired.

In the next subsection we discuss briefly the IPA reconstruction performance, providing insight on the properties that $\mathbf{M}$ should satisfy to obtain the best achievable performance.

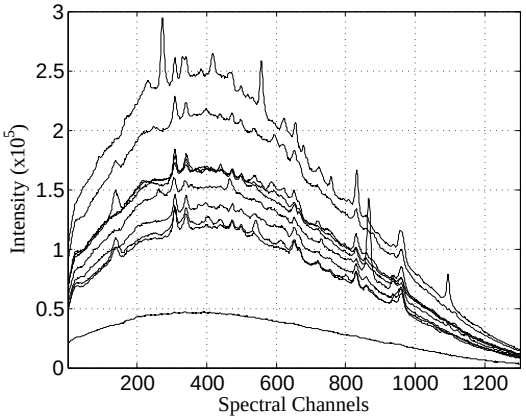


FIGURE 4.11. Raman spectra of 10 pharmaceutical chemicals.

4.4.2. Simulation results and discussion

We present here simulation results for the mixture estimation problem using the IPA for a chemical library from real measurements of different pharmaceutical chemicals. Fig. 4.11 presents an example of 10 spectra from the chemical library. The library is composed of $n_c = 265$ different chemicals each having $n_s = 1300$ spectral samples. Note that in this example $n_s > n_c$, but no restrictions exist on n_s nor n_c so long as $m < n_c$. Each Raman spectra in the library is unique, although they may be very similar (peaks or plateaus around same spectral channels). The sequence of spectral channels φ_j used to construct $\mathbf{A}$ can be selected at random, or in a deterministic fashion. The spectral channels have been selected here by taking the first n_c peaks common to the majority of chemicals. Experiments indicate that reconstruction performance is insensitive to the spectral channels chosen.

We use the measurement matrices generated by Eq. 4.5 and 4.6 which yield overall measurement matrices of size 159×265 and 212×265. Consequently the reconstruction performance are given by Fig. 4.6. Fig. 4.6 then show the proportion of correct reconstructions between the original chemical mixtures and the reconstructed ones in a noise free environment. Fig. 4.6 also demonstrates that chemical mixtures with a large number of chemicals can be recovered exactly. Specifically, the recovery of

mixtures of up to 55 chemicals can be obtained with $m = 159$ measurements from the spectrometer, which represents a compression ratio of 1300:159 $\gtrsim$ 8:1. For $m = 212$, it is about 75 chemicals that can be recovered, which represents a compression ratio of 1300:212 $\gtrsim$ 6:1.

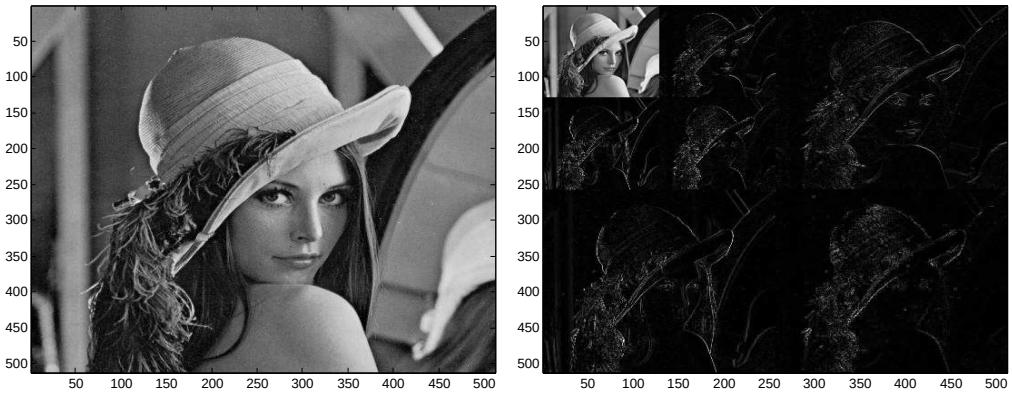
4.4.3. Interval-passing for imaging

Finally we expose here the general motivation which enable the use of compressed sensing for imaging as well as some preliminary results.

4.4.3.1. A motivating example

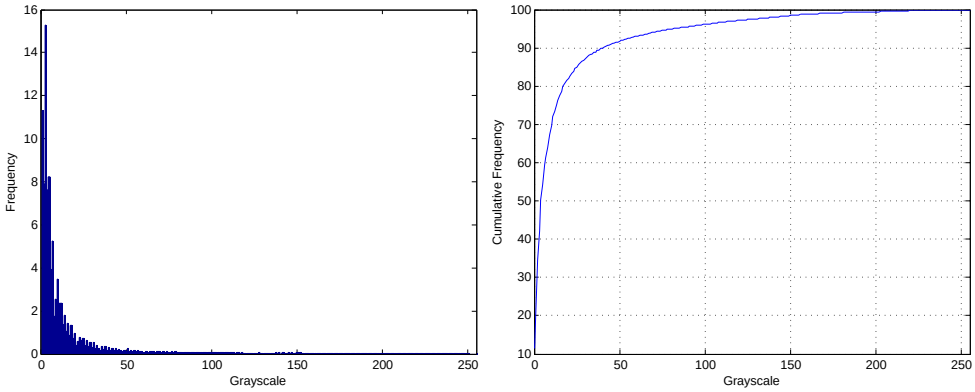
We consider the famous 512×512 Lena image in grayscale and performed the second level wavelet transform using the Haar wavelets. We depicts in Fig. 4.12 the original and the transformed images. It is then obvious that a lot of coefficients in the wavelet decomposition are close to zero (corresponding to black). In fact we have plotted in Fig. 4.13 the frequency of appearance of each grayscale level as well as the cumulative frequencies. These figures corroborate the first conjecture as 80% of the wavelet decomposition entries have a grayscale below 18. By forcing (or considering) these 80% to 0, the real sparsity will become roughly 0.2 as only 20% of the values will be non-zero (and non-negative). This real new sparsity will then enable the use of a measurement matrix to compress the data.

We can then study the influence of setting the low valued components of the wavelet transform to 0 on the reconstructed image. We have set all the entries of the wavelet transform below a grayscale of 18 to 0. This then provides a sparsity measure of 0.21 for the wavelet transform (21% of the entries are non-zero). At the reconstruction side, we have displayed in Fig. 4.14 the reconstructed image. With our naked eye the quality loss is negligible.



(a) (b)

FIGURE 4.12. (a) Lena image. (b) Level-2 wavelet decomposition using Haar wavelets



(a) (b)

FIGURE 4.13. (a) Frequencies and (b) cumulative frequencies of the wavelet decomposition of Lena image

4.4.3.2. On the compressed sensing scheme

Once the wavelet transform has been modified to obtain an averaged sparsity of 0.21, we can start the measuring process. However this sparsity is not uniform on the whole image. As it is not possible to measure the wavelet transform directly (vector of size 1×262144), the measurement process has to be done through smaller length vectors. We could measure row-wise, however due the approximation coefficients, the first 512 rows have a large sparsity measure (in average 0.35 which is more the the average sparsity of the whole image). In order to keep the same measurement

FIGURE 4.14. Reconstructed image with 80% of the wavelet entries set to 0

matrix for each row, we want to reorganize this matrix into rows of a certain size but with the same sparsity. We chose to keep the size of 512 for convenience, we then reshape the approximation coefficients (top-right image), into a 512×32 array. We proceed exactly the same for the second level of horizontal, vertical, and diagonal coefficients. Each first level of coefficient matrix is reorganized into a 512×128 array. All these arrays are concatenate to produce a $512 \times (4 \times 32 + 3 \times 128) = 512 \times 512$ image. We display the new wavelet decomposition in Fig. 4.15. We can see that the sparsity (row-wise) is now roughly the same for each row. This is confirmed with Fig. 4.16 where we depicted the row-wise sparsity for the wavelet decomposition before and after the reshaping. After the reshaping no row has a sparsity greater than 0.3. If we then apply a measurement matrix on the 512 rows and if we ensure that our reconstruction algorithm can recover vectors with sparsity lower or equal than 0.3 we can ensure that the use of our sparse measurement matrix and interval-passing algorithm can be appropriate for this scheme.

4.4.3.3. On the use of the interval-passing algorithm

Up to this point we have designed a sequence of 512 signals of length 512 with a maximum sparsity of 0.3. From our previous work we have designed a sparse measurement matrix which enables to measure/store each row and then reconstruct

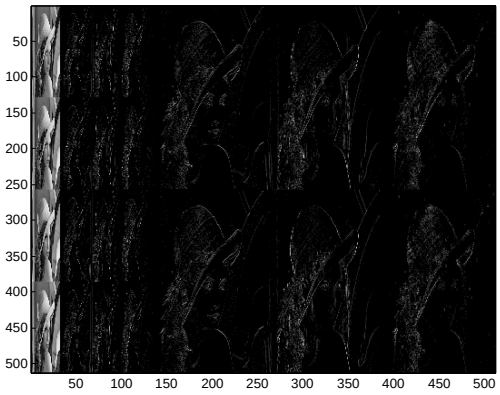
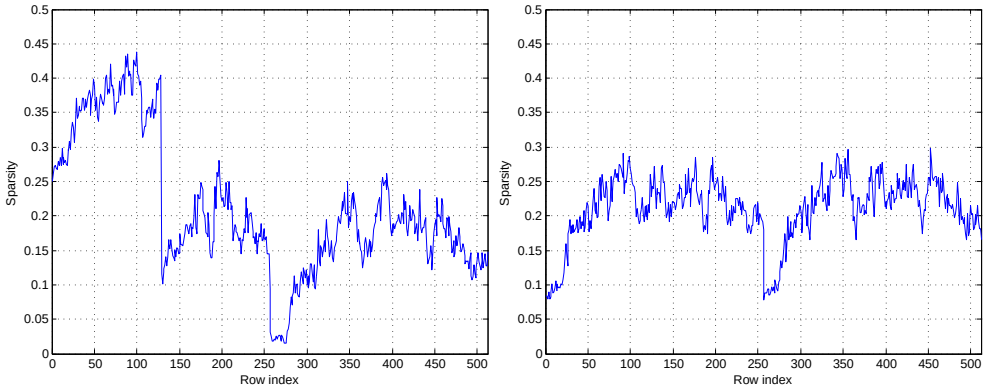


FIGURE 4.15. Reshaped wavelet decomposition



(a) Before reshaping. (b) After reshaping

FIGURE 4.16. Sparsity of the row-wise wavelet decomposition.

with the interval-passing algorithm. The designed measurement matrix corresponds to a (4,5) LDPC matrix of size 412×515. We disregard the last 3 columns to obtain a 412×512 measurement matrix. The designed measurement matrix is given by:

$$\mathbf{M_2} = \begin{pmatrix} \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} & \mathbf{I_0} \\ \mathbf{I_0} & \mathbf{I_{50}} & \mathbf{I_{43}} & \mathbf{I_{29}} & \mathbf{I_{42}} \\ \mathbf{I_0} & \mathbf{I_{62}} & \mathbf{I_{11}} & \mathbf{I_{28}} & \mathbf{I_{56}} \\ \mathbf{I_0} & \mathbf{I_{84}} & \mathbf{I_{47}} & \mathbf{I_{27}} & \mathbf{I_{52}} \end{pmatrix}. \tag{4.21}$$

where $\mathbf{I}_k$ denotes the circulant matrix of order k and of size 103×103 in this case. The reconstruction performance results in the noise free case is reported in Fig. 4.17. We can see that as every signal with a sparsity up to 0.33 can be exactly recovered, our 512 rows of the wavelet decomposition will be also recovered perfectly.

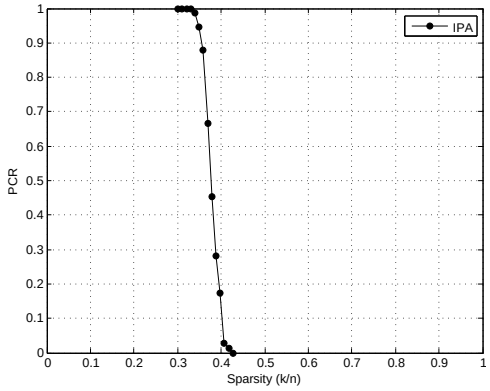


FIGURE 4.17. IPA performance for the measurement matrix designed for the imaging problem

4.5. Conclusion and discussion

In this chapter we have introduced the IPA, a low-complexity iterative reconstruction algorithm for compressed sensing in which sparse non-negative matrices are considered as measurement matrices. We showed that the IPA has good reconstruction results in a noise free case. We also demonstrated that there exists a close link between the failures of the IPA and some topological structures of its equivalent graphical representation, known as stopping sets in the case of LDPC decoding. We have also presented two applications of compressed sensing where the IPA can be used. First, in spectroscopy we showed that by an appropriate design of the measurement matrix on the chemical mixture spectrum, we can produce an artificial overall measurement matrix with is sparse, which allow the use of the IPA as a reconstruction matrix. Finally we showed that from the wavelet transform of an image which provides a sparse approximation, we can apply a sparse measurement matrix on a reshaped image and use the IPA to recover it.

The first perspective for this work is to continue to compare the performance of the IPA with other iterative reconstruction algorithms. The second and major perspective for this work is to adapt the IPA in a noisy case. However this adaptation raises some questions. Indeed the IPA deals with non-negative measurements, the noise added

could flip the sign of some measurements. The decision will also be an issue, as the intervals are likely not to converge to a single value. However the first experiments tend to show that IPA can recover at least the support of the sparse signal.

For instance we took a 10-sparse vector $\mathbf{x}$ of length $n = 265$ with non-zero values between 0 and 255. We then use a sparse matrix $\mathbf{A}$ of size $m \times n$ with $m = 159$. We then measure $\mathbf{x}$ via $\mathbf{A}$, but with addition of a Gaussian noise, such that $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{z}$, where $\mathbf{z}$ is drawn for the Gaussian distribution with variance σ^2. The SNR being defined here as $E_b/N_0 = 1/2\sigma^2$. Fig. 4.18 depicts the first results concerning the reconstruction of IPA in a noisy setting where we applied a noise of 2dB on the measurements. This figure shows in x-axis the indices of the sparse signal, and in y-axis its original value (denoted with a $\times$), as well as the interval which is computed for each location of the signal. Moreover, we introduced some prior information in the setting, as we assume to know, while performing the reconstruction, the minimum value for the non-zero values in x, this removes the false alarms. We then notice that although the approximation of the intervals is not quite accurate, it is pretty promising as it allows to detect, at least, the support of the original vector $\mathbf{x}$. We believe that under certain conditions on the signals (prior knowledge, particular structure of the original vector...) the adaptation of the IPA to noisy measurements is possible.

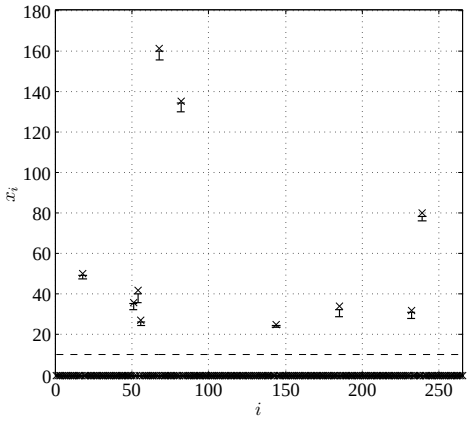


FIGURE 4.18. First results on the IPA reconstruction in a noisy case.

CONCLUSION AND PERSPECTIVES

The presence of iterative algorithms in many digital systems demonstrates their extraordinary efficiency in systems used to communicate, entertain and work. With the increasing number of users of digital technologies, the need in fast and large-scale iterative methods will definitely keep growing.

In digital communications, traditional decoders, such as BP used for the decoding of LDPC codes present an abrupt degradation of the error-rate performance at a high SNR. Recently a class of finite precision iterative decoders, called finite alphabet iterative decoders (FAIDs) was proposed to improve the performance in this particular region using a very low complexity. In addition, it has been shown that FAIDs can surpass BP using only 3 bits of precision on column-weight-three codes, and we have presented in this dissertation a 3-bit and 4-bit FAID surpassing the BP in the error floor region for a column-weight-four code. As the number of possible FAIDs is very large for a given number of precision bits, we have proposed two methodologies to select potential good FAIDs for column-weight-three codes. The first one relying on the particular knowledge of a code, specifically the knowledge of its minimal codewords, and the second one is more general and aims at selecting good FAIDs for any column-weight-three code based on the analysis of potential harmful structures in a code.

The work presented in this dissertation on this topic leads to different perspectives. The complete extension on FAIDs for column-weight-four codes is crucial for the practical applications, and can be addressed by two different ways. The first one is to extend the work done for column-weight-three codes, which is, to have a better understanding of the harmful topologies for this type of codes and possibly design a trapping set ontology, and from there design FAIDs whose performance can

surpass performance of BP decoding in the error floor. However this technique will be limited by the number of potential FAIDs that can be designed, necessarily more numerous than in the column-weight-three case. The second approach can rely on the derivation of FAIDs for column-weight-four codes from a combination of FAIDs for column-weight-three codes. This would definitely lead to a design at a lower complexity, but the combination of FAIDs for column-weight-three codes is not trivial to realize, while ensuring a good error correction capability. Nevertheless we know, thanks to this work, that FAIDs for column-weight-four codes have also the capability of surpassing BP decoding in the error floor. The second perspective of this work concerns the modification of the channel. We have considered the BSC, but designing good FAIDs on AWGNC is a challenge, as the channel values are not binary anymore. The major issue for this will certainly be the quantization of the real values received from the channel to a discrete alphabet. Adapting FAIDs on AWGNC using column-weight-four codes will definitely promote the FAIDs as serious candidate decoders in the storage industry for hard-drive, or flash memories systems. Finally it is likely that more number of alphabet levels will have to be considered to improve even more the performance for either codes with higher column-weight or on different communication channel.

In the second part of the dissertation we have shown the potential of iterative algorithms in compressed sensing, an emerging field in signal processing, in which the sampling of a signal can be done below the Nyquist sampling rate if the signal is sparse in a certain basis. We presented in this dissertation a modified version of a low-complexity iterative reconstruction algorithm called interval-passing algorithm (IPA) whose reconstruction performance is a good trade-off between other low-complexity methods and complex reconstruction algorithms. This algorithm uses non-negative measurement matrices, and the analysis of the failures of the IPA establishes a link between the failures of iterative decoder on the binary erasure channel and the graphical

representation of the measurement matrix. We have also shown that the IPA compete with algorithm necessitating random measurements (random measurement matrices) like the approximate message-passing algorithm.

Considering the very large amount of papers related to compressed sensing, the perspective of this field is certainly to adapt known reconstruction methods to real world applications. In this dissertation, we have presented some first results in that direction, namely in spectroscopy and imaging. Further work in this topic also includes the analysis of the reconstruction possibilities of the IPA under noisy measurements which will help to design measurement matrices to improve the performance. Another related perspective for this work relies on the computation of the mutual information between the original signal and its measurements, and find a connexion between the maximization of this mutual information and the failures of the IPA with the goal of designing measurement matrices maximizing the reconstruction performance.

REFERENCES

[1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.

[2] R. W. Hamming, "Error detecting and error correcting codes," *Bell Sys. Tech. Journal*, vol. 29, pp. 147–160, Apr., 1950.

[3] M. J. E. Golay, "Notes on digital coding," *Proc. IEEE*, 37, pp. 657, 1949.

[4] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *IRE Trans. on Electronic Computers*, vol. 3, pp. 6–12, Sep. 1954.

[5] Irving S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Trans. of the IRE Professional Group on Inform. Theory*, vol 4, pp. 38–49, Sep. 1954.

[6] C.E. Shannon, "A Mathematical Theory of Communication," *Bell Sys. Tech. Journal*, vol. 27, pp. 379–423, 623–656, Jul., Oct., 1948.

[7] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, pp. 147–156, Sep. 1959.

[8] R. C. Bose, D. K. Ray-Chaudhuri, "On A Class of Error Correcting Binary Group Codes," *Information and Control*, vol. 3, pp. 68–79, Mar. 1960.

[9] I. S. Reed, G. Solomon, "Polynomial Codes over Certain Finite Fields," *J. Soc. Indust. Appl. Math.*, vol. 8, pp. 300–304, Jun. 1960.

[10] E. R. Berlekamp, "Factoring Polynomials Over Finite Fields," *Bell Syst. Tech. Journal*, vol. 46, pp. 1853–1859, May 1967.

[11] P. Elias, "Coding for noisy channels," *IRE Conv. Rep.*, Pt. 4, pp. 37–47, 1955.

[12] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 4, pp. 260–269, Apr. 1967.

[13] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 3, pp. 284–287, Mar. 1974.

[14] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," *Proc. IEEE Int. Conf. on Commun.*, Geneva, Switzerland, pp. 1064–1070, May 1993.

[15] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," *Proc. 5th IMA Conf. Cryptography and Coding*, Cirencester, UK, no. 1025, pp. 100–111, Dec. 1995.

[16] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[17] N. Wiberg, " Codes and Decoding on General Graphs," *PhD Dissertation*, 1996.

[18] F. R. Kschischang, B. J. Frey and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[19] T. Richardson and R. Urbanke, "The capacity of LDPC codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, pp. 599–618, Feb. 2001.

[20] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[21] M. Fossorier, M. Mihaljevic, H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[22] J. Lee, J. Thorpe, "Memory-efficient decoding of LDPC codes," *Proc. IEEE Int. Symp. Inform. Theory*, Adelaide, Australia, pp. 459–463, Sep. 2005.

[23] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, pp. 1710–1722, Nov. 1996.

[24] D. Burshtein and G. Miller, "Expander graph arguments for message passing algorithms," *IEEE Trans. Inf. Theory*, vol. 47, pp. 782–790, Feb. 2001.

[25] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes ," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 954–972, Mar. 2005.

[26] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, pp. 1788–1793, Aug. 2004.

[27] J. Thorpe, "Low density parity-check (LDPC)codes constructed from protographs," *IPN progress report 42*, no. 154, Aug. 15, 2003.

[28] D. Divsalar, S. Dolinar, C. R. Jones, K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Select. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.

[29] L. Lan, L. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach," *IEEE Trans. Inf. Theory*, vol.53, pp. 2429–2458, Jul. 2007.

[30] Kou Y. Kou, S. Lin and M. C. Fossorier, "Low-Density Parity-Check Codes Based on Finite Geometries: A Rediscovery and New Results," *IEEE Trans. on Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[31] B. Vasic, O. Milenkovic, "Combinatorial construction of low-density parity-check codes for iterative decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1156–1176, Jun. 2004.

[32] R. M. Tanner, "Convolutional codes from quasi-cyclic codes: A link between the theories of block and convolutional codes," *Univ. Calif. Santa Cruz, Tech. Rep.*, 1987.

[33] A. Feltström and K. S. Zigangirov, "Periodic time-varying convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, no. 5 pp. 2181–2190, Sept. 1999.

[34] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative Decoding Threshold Analysis for LDPC Convolutional Codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, 5274–5289, Oct. 2010.

[35] S. Lin and D. J. Costello, "*Error Control Coding: Fundamentals and Applications*," second edition, Prentice Hall: Englewood Cliffs, NJ, 2005.

[36] E. Sharon, S. Litsyn and J. Goldberger, "Efficient serial message-passing schedule for LDPC decoding," *IEEE Trans. on Inf. Theory*, vol. 53, no. 11, pp. 4076–4091, Nov. 2007.

[37] J. Pearl, *Probablisitic Reasoning in Intelligent Systems*, San Francisco, CA, USA: Kaufmann, 1988.

[38] B. J. Frey, *Graphical models for machine learning and digital communication*, Cambridge, MA, USA: MIT Press, 1998.

[39] D. MacKay and M. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes,", *Electron. Notes in Theoretical Comput. Sci.*, vol. 74, pp. 97–104, 2003.

[40] B. J. Frey, R. Koetter, and A. Vardy, "Signal-space characterization of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 47, no.2, pp. 766–781, Feb. 2001.

[41] G. D. Forney, Jr., R. Koetter, F. R. Kschischang, and A. Reznick, "On the effective weights of pseudocodewords for codes defined on graphs with cycles," *Codes, systems and graphical models*, pp. 101–112, Springer, 2001.

[42] C. Di, D. Proietti. I. E. Teletar, T. J. Richardson, and R. L. Urbanke, "Finite length analysis of low-density parity-check codes on the binary erasure channels," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1576–1579, Jun. 2002.

[43] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensemble," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 929–953, Mar. 2005.

[44] T. Richardson, "Error floors of LDPC codes," in *Proc. Allerton Conf. on Commun., Control and Computing*, vol. 41, no. 3, pp. 1426–1435, Monticello, Illinois, USA, Oct. 2003.

[45] M.G. Stepanov, V. Chernyak, M. Chertkov, B. Vasić, "Diagnosis of weakness in error correction codes: a physics approach," *Phys. Rev. Lett.*, vol. 95, no. 22, Nov. 2005.

[46] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasić, "Error floors of LDPC codes on the binary symmetric channels," in *Proc. IEEE Int. Conf. on Commun.*, Istanbul, Turkey, pp. 1089–1094, Jun. 2006.

[47] L. Dolecek, Z. Zhang, M. J. Wainwright, V. Anantharam, and B. Nikolic, "Analysis of Absorbing Sets for Array-Based LDPC Codes," in *Proc. IEEE Int. Conf. on Commun.*, Glasgow UK, pp. 6261–6268, Jun., 2007.

[48] S. K. Chilappagari and B. Vasić, "Error-correction capability of column-weight-three LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2055–2061, May 2009.

[49] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC code," *Corr arXiv:cs/0512078v1*, Dec. 2005.

[50] C. A. Kelley and D. Sridhara, "Pseudocodewords of Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4013–4038, Nov. 2007.

[51] R. Smarandache, A. Pusane, P. Vontobel, and D.J. Costello, "Pseudocodeword performance analysis for LDPC convolutional codes," *IEEE Trans. Inform. Theory*, vol. 55, no. 6, pp. 2577–2598, Jun. 2009.

[52] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. IEEE Int. Conf. on Wireless Networks, Commun. and Mobile Computing*, Hawaii, USA, pp. 630–635, Jun. 13-16, 2005.

[53] A. McGregor and O. Milenkovic, "On the hardness of approximating stopping and trapping sets in LDPC codes," *IEEE Trans. Inform.Theory*, vol. 56, no. 4, pp. 1640–1650, Apr. 2010.

[54] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 39–55, Jan. 2007.

[55] C. Cole, S. Wilson, E. Hall, and T. Giallorenzi, "A general method for finding low error rates of LDPC codes," *Corr arXiv:cs/0605051*, May 2006.

[56] S. Abu-Surra, D. Declercq, D. Divsalar, and W. Ryan, "Trapping set enumerators for specific LDPC codes," in *Proc. Inform. Theory and Applicat. Workshop*, San Diego, CA, USA, pp. 1–5, Jan. 31- Feb. 5, 2010.

[57] E. Rosnes and O. Ytrehus, "An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices," *IEEE Trans. Inf. Theory*, vol. 55, no. 9, pp. 4167–4178, Sept. 2009.

[58] M. K. Dehkordi and A. H. Banihashemi, "An efficient algorithm for finding dominant trapping sets of LDPC codes," in *Proc. 6th Int. Symp. Turbo Codes Iterat. Inf. Process.*, pp. 444–448, Brest, France, Sep. 6-10, 2010.

[59] X. Zhang and P. H. Siegel, "Efficient algorithms to find all small error-prone substructures in LDPC Codes," in *Proc. IEEE Global Telecommun. Conf.*, pp. 1–6, Houston, TX, USA, Dec. 5–9, 2011.

[60] B. Vasić, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," in *Proc. Allerton Conf. on Commun., Control and Computing*, Monticello, IL, USA, pp. 1–7, Sept. 2009.

[61] S. K. Chilappagari, M. Chertkov, M. G. Stepanov, and B. Vasić, "Instanton-based techniques for analysis and reduction of error floors of LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 855–865, Aug. 2009.

[62] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasic, "On the construction of structured LDPC Codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.

[63] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasic, "Finite alphabet iterative decoders for LDPC codes surpassing floating-point iterative decoders," *IET Electron. Lett.*, vol. 47, no. 16, , pp. 919–921, Aug. 2011.

[64] S. K. Planjery, S. K. Chilappagari, B. Vasic, D. Declercq, and L. Danjean, "Iterative decoding beyond belief propagation," in *Proc. Inform. Theory and App. Workshop*, San Diego, CA, USA, pp. 41–43, Jan. 2010.

[65] G. Kuperberg, "Symmetries of plane partitions and the permanent-determinant method," *J. Comb. Theory A*, vol. 68, pp. 115–151, 1994.

[66] S. K. Chilappagari and B. Vasić, "Error-correction capability of column-weight-three LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2055–2061, May 2009.

[67] S. K. Planjery, D. Declercq, S. K. Chilappagari, and B. Vasić, "Multilevel decoders surpassing belief propagation on the binary symmetric channel," in *Proc. IEEE Int. Symp. on Inf. Theory*, Austin, TX, USA, pp. 769–773, Jun. 2010.

[68] R. M. Tanner, "A [155,64,20] sparse graph (LDPC) code," in *Recent results session, IEEE Int. Symp on Inf. Theory*, Sorrento, Italy, Jun. 2000.

[69] M. Karimi and A. Banihashemi, "Efficient algorithm for finding dominant trapping sets of LDPC codes," *IEEE Trans. Inf. Theory*, vol. 58, no.11, pp 6942–6958, Nov. 2012.

[70] D. Declercq and M. Fossorier, "Improved impulse method to evaluate the low weight profile of sparse binary linear codes," in *Proc. of IEEE Int. Symp. on Inf. Theory*, Toronto, Canada, pp. 1963-1967, Jul. 2008.

[71] L. Danjean, D. Declercq, S. K. Planjery, and B. Vasic, "On the selection of finite alphabet iterative decoders for LDPC codes on the BSC," in *Proc. IEEE Inform. Theory Workshop*, Paraty, Brazil, pp. 345–349, Oct. 2011.

[72] S. K. Planjery, D. Declercq, L. Danjean , B. Vasic, "Finite alphabet iterative decoders, part I: decoding beyond belief propagation on the binary symmetric channel," *IEEE Trans. Commun.* (to appear), 2013.

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

[74] J. Wang, L. Dolecek, and R. Wesel, "Controlling LDPC absorbing sets via the null space of the cycle consistency matrix," in *Proc. Int. Conf. on Commun.*, Kyoto, Japan, Jun. 59 2011, pp. 1-6.

[75] "Compressed sensing resources," Rice University. [Online]. Available: http://dsp.rice.edu/cs

[76] "Nuit blanche blog." [Online]. Available: http://nuit-blanche.blogspot.com

[77] E. R. J. Candes and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[78] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[79] D. Donoho, "Compressed sensing," *IEEE Trans.Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Feb. 2006.

[80] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, 3rd ed, Academic Press, 2008.

[81] Z. Lei and Q. Chunting, "Application of compressed sensing theory to radar signal processing," in *Proc 3rd IEEE Int. Conf. on Comput. Sci. and Inform. Technology*, vol. 6, Chengdu, China, pp. 315–318, Jul. 2010.

[82] J. Romberg, "Imaging via compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 14–20, Mar. 2008.

[83] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressed channel sensing: A new approach to estimating sparse multipath channels," *Proc. of the IEEE*, vol. 98, no. 6, pp. 1058–1076, Jun. 2010.

[84] J. Provost and F. Lesage, "The application of compressed sensing for photo-acoustic tomography," *IEEE Trans. Med. Imag.*, vol. 28, no. 4, pp. 585–594, Apr. 2009.

[85] G. Kutyniok, "Compressed sensing: Theory and applications," *CoRR*, vol. abs/1203.3815, 2012.

[86] S. S. Chen, D. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[87] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, pp. 3397–3415, 1993.

[88] J. Tropp and A. Gilbert, "Signal Recovery from partial information via orthogonal matching pursuit," *IEEE Trans. Inform. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[89] ——, "Signal recovery from random measurements information via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[90] D. L. Donoho, Y. Tsaig., I. Drori., and J.-L. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," Tech. Rep., 2006.

[91] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[92] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, no. 45, pp. 18 914–18 919, Sep. 2009.

[93] A. G. Dimakis and P. Vontobel, "LP decoding meets LP decoding: a connection between channel coding and compressed sensing," in *Proc. 47th Allerton Conf. on Commun., Control, and Computing*, Monticello, Illinois, USA, pp. 8–15, Sep. 30–Oct. 2 2009.

[94] J. Feldman, M. Wainwright, and D. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 954–972, Mar. 2005.

[95] J. D. Blanchard, C. Cartis, and J. Tanner, "Compressed sensing: How sharp is the restricted isometry property," *SIAM Review*, 2010.

[96] A. Dimakis, R. Smarandache, and P. O. Vontobel, "LDPC codes for compressed sensing," *IEEE Trans. Inf. Theory.*, vol. 58, no. 5, pp.30933114, May. 2012.

[97] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.

[98] J. Tropp and S. Wright, "Computational methods for sparse solution of linear inverse problems," in *Proc. of the IEEE*, vol. 98, no. 6, pp. 948–958, Jun. 2010.

[99] H. V. Pham, W. Dai, and O. Milenkovic, "Sublinear compressive sensing reconstruction via belief propagation decoding," in *Proc. IEEE Int. Symp. Inform. Theory*, Seoul, Korea, pp. 674–678, Jun. 28–Jul. 3 2009.

[100] R. Berinde, P. Indyk, and M. Ruzic, "Practical near-optimal sparse recovery in the L1 norm," in *Proc. 46th Allerton Conf. on Commun., Control, and Computing*, Urbana-Champaign, IL, USA, pp. 198–205, Sep. 23–26 2008.

[101] F. Zhang and H. Pfister, "Compressed sensing and linear codes over real numbers," in *Inf. Theory and Applicat. Workshop*, San Diego, CA, USA, pp. 558–561, Feb. 2008.

[102] F. Zhang and H. D. Pfister, "Verification decoding of high-rate LDPC codes with applications in compressed sensing," *IEEE Trans. Inf. Theory*, vol. 58, no. 8, pp. 5042,5058, Aug. 2012.

[103] V. Chandar, D. Shah, and G. Wornell, "A simple message-passing algorithm for compressed sensing," in *Proc. IEEE Int. Symp. Inform. Theory*, Austin, TX, USA, Jun. 13–18 2010, pp. 1968–1972.

[104] A. R. Krishnan, S. Sankararaman, and B. Vasic, "Graph-based iterative reconstruction of sparse signals for compressed sensing," in *Proc. Int. Conf. on Telecommun. in Modern Satellite, Cable and Broadcasting Services*, Nis, Serbia, pp. 133–137, Oct. 5–8 2011.

[105] L. Danjean, V. Ravanmehr, D. Declercq, and B. Vasic, "Iterative reconstruction algorithms in compressed sensing," in *Proc. 19th Telecommun. Forum*, Belgrade, Serbia, pp. 537–541, Nov. 22–24 2011.

[106] V. Ravanmehr, L. Danjean, D. Declercq, and B. Vasic, "On iterative compressed sensing reconstruction of sparse non-negative vectors," in *Proc. Int. Symp. on Appl. Sci. in Biomedical and Commun. Technologies*, Barcelona, Spain, Oct. 26–29 2011.

[107] M. C. Davey and D. J. C. MacKay, "Low-density parity check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998.

[108] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes Over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[109] D. J. MacKay, "Encyclopedia of sparse graph codes." [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[110] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory,* vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[111] M. Schwartz and A. Vardy, "On the stopping distance and the stopping redundancy of codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 922–932, Mar. 2006.

[112] S. Hoory, "The size of bipartite graphs with a given girth," *J. Combinatorial Theory Series B*, vol. 86, no. 2, pp. 215–220, Nov. 2002.

[113] G. Richter, "Finding small stopping sets in the tanner graphs of LDPC codes," in *Proc. Int. Symp. on Turbo Codes Related Topics and Int. ITG-Conf. on Source and Channel Coding*, Munich, Germany, pp. 1–5, Apr. 3–7 2006.

[114] E. Rosnes, O. Ytrehus, M. Ambroze, and M. Tomlinson, "Addendum to: An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices," *IEEE Trans. Inf. Theory,*, vol. 58, no. 1, pp. 164–171, Jan. 2012.

[115] "IEEE standard for local and metropolitan area networks part 16: Air interface for fixed and mobile broadband wireless access systems amendment 2: Physical and medium access control layers for combined fixed and mobile operation in licensed bands and corrigendum 1," *IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004)*, Feb. 2006.

[116] H.-B. Liu, H. Zhong, N. Karpowicz, Y. Chen, and X.-C. Zhang, "Terahertz spectroscopy and imaging for defense and security applications," *Proc. of the IEEE*, vol. 95, no. 8, pp. 1514–1527, Aug. 2007.

[117] N. Butt, M. Nilsson, A. Jakobsson, M. Nordberg, A. Pettersson, S. Wallin, and H. O andstmark, "Classification of Raman Spectra to Detect Hidden Explosives," *IEEE Geosci. and Remote Sensing Lett.*, vol. 8, no. 3, pp. 517–521, May 2011.

[118] D. J. Gardiner, *Practical Raman Spectroscopy*, P. R. Graves, Ed. Springer-Verlag, Apr. 1989.

[119] R. Kopelman and W. Tan, "Near-field optical microscopy, spectroscopy, and chemical sensors," *Appl. Spectroscopy Reviews*, vol. 29, no. 1, pp. 39–66, Aug. 1994.

[120] D. V. Dinakarababu, D. R. Golish, and M. E. Gehm, "Adaptive feature specific spectroscopy for rapid chemical identification," *Opt. Express*, vol. 19, no. 5, pp. 4595–4610, Feb. 2011.