# LOW-DELAY LOW-COMPLEXITY ERROR-CORRECTING CODES ON SPARSE GRAPHS

THÈSE N$^{O}$ 2681 (2002)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES TECHNIQUES

PAR

Xiao-Yu HU

Master of Engineering, East China Institute of Technology, Chine
et de nationalité chinoise

Lausanne, EPFL
2003

*Für Ying*

*Die Liebe ist langmütig, die Liebe ist gütig. Sie ereifert sich nicht, sie prahlt nicht, sie bläht sich nicht auf. Sie handelt nicht ungehörig, sucht nicht ihren Vorteil, lässt sich nicht zum Zorne reizen, trägt das Böse nicht nach. Sie freut sich nicht über das Unrecht, sondern freut sich an der Wahrheit. Sie erträgt alles, glaubt alles, hofft alles, hält allem stand. Die Liebe hört niemals auf.*

*1. Korinther 13,4–8*

# Acknowledgments

First of all, I thank Dr. Evangelos S. Eleftheriou for providing such an excellent research environment, serving as a co-advisor and providing tremendous technical guidance and instructions. Without his contribution, this thesis would not have been possible. I also thank Prof. Martin Vetterli for acting as my thesis advisor and for encouraging me to pursue my own ideas, which will influence my entire career. I thank Prof. Rüdiger Urbanke for acting as a co-advisor and for many enlightening discussions that promoted theoretical insight. I am also grateful to Prof. J. Hagenauer, Prof. M. P. C. Fossorier and Prof. B. Rimoldi for acting as referees of this thesis. A special thank goes to Prof. B. Rimoldi, who sparked my interest in information and coding theory through his wonderful lectures which I attended, in the initial stage of this thesis work.

The information and coding theory community at the IBM Zurich Research Laboratory is a fascinating one to be a member of. I learned quite a lot from many smart people and thank them all for sharing their ideas and experience with me. A special thank goes to my friend Dieter-Michael Arnold, with whom I shared an office for three years. Apart from our many discussions, he introduced me to the beauty of factor graphs and the sum-product algorithm, and shared with me his substantial knowledge of computer skills, ranging from Matlab, Mathematica, to LaTex. My thanks also go to Ajay Dholakia and Thomas Mittelholzer, with whom advances on efficient implementations of iterative decoding were made possible, to Sedat Ölcer and Giovanni Cherubini for the exciting time we spent together on the xDSL project, and to Roy Cideciyan and Haris Pozidis for educating me with their expertise on magnetic recording channels and the "Millipede" project.

Special thanks go to Werner Bux and Pierre Chevillat, for their continuous support and assistance. I am also thankful to people with whom I could discuss and solve work-permit problems, in particular to Anouschka Van Loon. Charlotte Bolliger has earned special acknowledgment for proofreading this thesis with endurance and sharing with me her vast knowledge of LaTex.

Needless to say, my family has always stood by me. I thank my parents for their endless care and support throughout all the stages of my education. I am deeply grateful to my wife Ying Liu, for all her love, patience, support, and encouragement.

Rüschlikon, Zürich

September, 2002
Xiaoyu Hu

# Abstract

This dissertation presents a systematic exposition on finite-block-length coding theory and practice. We begin with the task of identifying the maximum achievable rates over noisy, finite-block-length constrained channels, referred to as $(\epsilon, n)$-capacity $C_\epsilon^n$, with $\epsilon$ denoting target block-error probability and $n$ the block length. We characterize how the optimum codes look like over finite-block-length constrained channels.

Constructing good, short-block-length error-correcting codes defined on sparse graphs is the focus of the thesis. We propose a new, general method for constructing Tanner graphs having a large girth by progressively establishing edges or connections between symbol and check nodes in an edge-by-edge manner, called progressive edge-growth (PEG) construction. Lower bounds on the girth of PEG Tanner graphs and on the minimum distance of the resulting low-density parity-check (LDPC) codes are derived in terms of parameters of the graphs. The PEG construction attains essentially the same girth as Gallager's explicit construction for regular graphs, both of which meet or exceed an extension of the Erdös–Sachs bound. The PEG construction proves to be powerful for generating good, short-block-length binary LDPC codes. Furthermore, we show that the binary interpretation of GF$(2^b)$ codes on the cycle Tanner graph TG(2, $d_c$), if $b$ grows sufficiently large, can be used over the binary-input additive white Gaussian noise (AWGN) channel as "good code for optimum decoding" and "good code for iterative decoding".

Codes on sparse graphs are often decoded iteratively by a sum-product algorithm (SPA) with low complexity. We investigate efficient digital implementations of the SPA for decoding binary LDPC codes from both the architectural and algorithmic point of view, and describe new reduced-complexity derivatives thereof. The unified treatment of decoding techniques for LDPC codes provides flexibility in selecting the appropriate design point in high-speed applications from a performance, latency, and computational complexity perspective.

**Keywords:** Shannon capacity, finite block length capacity, error-correcting codes, low-density parity-check codes, sum-product algorithm, density evolution, codes on graphs

# Kurzfassung

Die vorliegende Dissertation ist eine systematische Abhandlung von Theory und Praxis der Kodierung bei endlicher Blocklänge. Zuerst wird die maximale erreichbare Übertragungsrate bei verrauschten Übertragungskanälen unter der Bedingung, dass die Blocklänge endlich ist, definiert. Wir bezeichnen diese Kapazität als $(\epsilon, n)$-Kapazität $C_\epsilon^n$, wobei $\epsilon$ die Blockfehlerrate und $n$ die Blocklänge bezeichnen. Dies erlaubt es uns, optimale Kodes für Kanäle mit endlicher Blocklänge zu charakterisieren.

Im Mittelpunkt dieser Arbeit steht die Konstruktion von guten, fehlerkorrigierenden Kodes mit kurzer Blocklänge, die von dünn besetzten Graphen stammen. Wir schlagen eine neue, allgemeingültige Methode zur Konstruktion von Tannergraphen mit grosser Taillenweite vor, indem sukzessive Kanten oder Verbindungen zwischen den Symbol- und Prüfbitknoten erstellt werden. Diese Methode bezeichnen wir demzufolge als fortschreitendes Kanten-Wachstum-Verfahren oder kurz FKW-Verfahren. Untere Schranken auf die Taillenweite des FKW-Tanner-Graphen und auf die Minimaldistanz der resultierenden 'low-density parity-check' (LDPC) Kodes werden in Abhängigkeit der Parameter des Graphen hergeleitet. Das FKW-Verfahren erreicht die gleiche Taillenweite wie die explizite Konstruktion von Gallager für reguläre Graphen. Beide Verfahren erreichen mindestens oder übertreffen in manchen Fällen sogar die Erdös-Sachs Schranke. Das FKW-Verfahren erlaubt die Konstruktion von guten binäre! ! n LDPC-Kodes mit kurzer Blocklänge. Weiter zeigen wir, dass die binäre Interpretation von $GF(2^b)$ Kodes basierend auf dem Tanner-Graphen TG(2, $d_c$) für den binären Eingangskanal mit additivem weissen Gausschen Rauschen gute Kodes sowohl bei optimaler als auch bei iterativer Dekodierung sind - vorausgesetzt $b$ wächst genügend schnell.

Kodes, die von dünn besetzten Graphen stammen, können mit geringer Rechenkomplexität iterativ mit dem Summe-Produkt Algorithmus (SPA) dekodiert werden. Wir untersuchen effiziente digitale Implementationen des SPA zur Dekodierung von binären LDPC-Kodes unter dem Gesichtspunkt sowohl der Schaltungsarchitektur als auch der Algorithmik und beschreiben daraus resultiernde neue Varianten mit geringerer Rechenkomplexität. Die umfassende Behandlung von solchen Dekodiertechniken für LDPC-Kodes ermöglicht es, dass bei Hochgeschwindigkeits-Anwendungen der Betriebspunkt vom Standpunkt der Leistung, Verzögerung und Rechenkomplexität flexibel ausgewählt werden kann.

**Stichworte:** Kanalkapazität von Shannon, Kanalkapazität bei endlicher Blocklänge, fehlerkorrigierende Kodes, Kodes mit schwachbesetzter Prüfbit-Matrix, Summe-Produkt Algorithmus, Dichteentwicklung, graphenbasierte Kodes

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **AWGN**: | Additive white Gaussian noise |
| **BCJR**: | Bahl-Cocke-Jelinek-Raviv |
| **BEC**: | Binary erasure channel |
| **BP**: | Belief propagation |
| **BSC**: | Binary symmetric channel |
| **CBP**: | Cluster belief propagation |
| **CDMA**: | Code division multiple access |
| **DMC**: | Discrete-time memoryless channel |
| **DSL**: | Digital subscriber lines |
| **FFT**: | Fast Fourier transform |
| **GF**: | Galois field |
| **LDPC**: | Low-density parity-check |
| **LLR**: | Log-likelihood ratio |
| **LPF**: | Low-pass filter |
| **MAP**: | Maximum a-posteriori probability |
| **ML**: | Maximum likelihood |
| **MRF**: | Markov random field |
| **MTR**: | Maximum-transition-run |
| **PEG**: | Progressive edge-growth |
| **PLF**: | Piecewise linear function |
| **PR**: | Partial response |
| **PWFB**: | Parallel windowed forward-backward |
| **SNR**: | Signal-to-noise ratio |
| **SPA**: | Sum-product algorithm |
| **TG**: | Tanner graph |
| **TSB**: | Tangential sphere bound |

# Chapter 1

# Introduction

## 1.1 Motivation

A major goal of coding theory is to find a class of error-correcting codes, together with the corresponding decoding algorithms, that realize the promise of Shannon's noisy channel coding theorem in a practical way. In the over 50 years since Shannon determined the capacity of ergodic channels [1, 2], the construction of capacity-approaching coding schemes has been the supreme goal of coding research, as evidenced by a large number of significant contributions. Only quite recently have practical codes and decoding algorithms that can closely approach the channel capacity of some classical memoryless channels become available. It is a remarkable fact that all known practical, capacity-approaching coding schemes are now understood to be codes defined on graphs, particularly sparse graphs, together with the associated iterative decoding algorithm — the sum-product algorithm.

It remains a puzzling fact (maybe for ever) that the main ideas pertaining to codes on graphs and to the sum-product algorithm were, in essence, invented 40 years ago by Gallager [3, 4] but were subsequently ignored by the coding community. Only a limited number of researchers, for instance, Zyablov and Pinsker, and Margulis continued to study Gallager's low-density parity-check (LDPC) codes. In 1981, Tanner [5] formally introduced the graphical model notation for describing codes, and proved the optimality of the sum-product algorithm in cycle-free graphs.

The subject of codes on graphs and iterative decoding enjoyed a burst of intense interest with the discovery of turbo codes [6] and the rediscovery of LDPC codes [7, 8] in the past few years. In particular, it has been shown by Richardson, Shokrollahi and Urbanke [9, 10] that an ensemble of LDPC codes of length one million achieves a bit-error probability of

$10^{-6}$, less than 0.13 dB away from capacity, with feasible complexity. It seems unlikely that further improvements could have any practical significance in situations that involve end-to-end communication over memoryless channels, and where very long block lengths are tolerable or affordable.

On the other hand, many real-time applications actually demand relatively short block lengths which are less than one or several thousands, either due to a strict end-to-end delay constraint or due to other implementation considerations, for instance, memory and buffer constraints. As opposed to the classic Shannon theory focusing on block lengths going to infinity, there are still many open issues concerning finite-block-length coding theory and practice. We summarize some but not all below:

- The Shannon theory assumes infinite coding latency, but in practice any application of interest is more or less constrained with a fixed coding latency that is tolerable or affordable. For instance, in magnetic recoding channels, a coding block is often limited to 4096 bits; in wireless communications, the coding block length ranges from several hundred to a few thousand bits, depending on the type of services; and in wireline communications such as xDSL technologies, the required end-to-end coding latency leads to not more than ten thousand bits. In addition, error-free is too costly for all these applications, and each application has its own target error probability. Therefore, it is tempting to ask the following questions: What is the maximum achievable rate under a fixed block length constraint together with a target error probability, and fundamentally, what do the optimum coding scheme look like? Moreover, can codes on sparse graphs together with iterative decoding closely approach the optimum coding scheme at relatively short block lengths?

- Constructions of sparse-graph codes in the existing literature focused on random graphs, among which bad graphs are assumed to be easily picked out. Unfortunately this assumption is true only for large graphs and a good construction method for short-block-length sparse-graph codes is still lacking. Can one define an expurgated random construction that effectively precludes bad graphs with short cycles and small minimum distance? How does the expurgated random ensemble perform asymptotically?

- Currently there are increasing activities to try to realize the sum-product decoding algorithm in many applications, ranging from wireless services, deep-space communications, xDSL technologies, to magnetic recoding channels. In order to enable extremely high-speed applications, can one find an efficient implementation of iterative decoding from both the algorithmical and the architectural standpoint?

These are the issues that will be addressed in this thesis.

## 1.2 Historical Development

### 1.2.1 Channel Capacity

The pioneering work on channel characterization in terms of channel capacity and random coding was done by Shannon [1, 2]. Additional contributions were subsequently made by Gilbert [11], Elias [12], Gallager [13], Wyner [14], Shannon, Gallager and Berlekamp [15], Forney [16] and Viterbi [17].

The Noisy Channel Coding Theorem proved by Shannon states:

**Theorem 1.1** *All rates below the "information" channel capacity $C$ are achievable, where the information channel capacity of a memoryless channel is defined as*

$$C = \max_{p(x)} I(X;Y), \tag{1.1}$$

*the maximum is taken over all possible input distributions $p(x)$ on the mutual information of the channel input $X$ and output $Y$. Specifically, for every rate $R < C$, there exists a sequence of $(n,\ 2^{nR})$ codes, where $n$ is the block length and $2^{nR}$ is the number of codewords in this code, with maximum probability of block error $\epsilon^{(n)} \to 0$ exponentially as $n \to \infty$. Conversely, any sequence of $(n,\ 2^{nR})$ codes with $\epsilon^{(n)} \to 0$ must have $R \leq C$.*

Although the theorem shows that there exist good codes with exponentially small error probability for long block lengths, it does not provide a way of constructing the best codes. If one uses the scheme suggested by the random-coding proof [18] and generates a code at random with the appropriate distribution, the code constructed is likely to be good for long block lengths. However, without some structure in the code, it is very difficult to decode. Hence the theorem does not provide a practical coding scheme. Ever since Shannon's original paper on information theory, researchers have tried to develop structured codes that are easy to encode and decode. So far, many codes with interesting and useful structures have been developed, but only quite recently, with the discovery of turbo codes and rediscovery of LDPC codes, have practical codes and decoding algorithms become available that can closely approach the channel capacity of some classical memoryless channels.

### 1.2.2 Classic Error-Correcting Codes

The pioneering work on coding and coded waveforms for digital communications was done by Hamming [19], Golay [20], Muller [21] and Reed [22]. During the period 1960–1970, there were a number of significant contributions to the development of coding theory and decoding algorithms. In particular, we cite the papers by Reed and Solomon [23] on Reed–Solomon (RS) codes, which are now ubiquitously used, the papers by Hocquenghem [24] and Bose and Ray-Chaudhuri [25, 26] on BCH codes, and the Ph.D dissertation of Forney [27] on concatenated codes. These works were followed by the papers of Goppa [28, 29] on the construction of a new class of linear cyclic codes called Goppa codes, and the paper of Justesen [30] on a constructive technique for asymptotically good codes. The first decoding algorithm for binary BCH codes was developed by Peterson [31]. A number of refinements and generalizations to RS codes were investigated by Chien [32], Forney [33], Massey [34], and Berlekamp [35].

In parallel to these developments on block codes are the developments in convolutional codes, which were invented by Elias [12]. Wozencraft and Rieffen [36] described a sequential decoding algorithm for convolutional codes. This algorithm was later refined by Fano [37], and is now called the Fano algorithm. Subsequently, the stack algorithm was devised by Zigangirov [38] and Jelinek [39], and the Viterbi algorithm was devised by Viterbi [40]. In decoding convolutional codes with small constraint lengths, the Viterbi algorithm is the most popular one because of its optimality and modest complexity.

### 1.2.3 Codes on Sparse Graphs

Codes on sparse graphs is a hot topic of great current interest. The main ideas pertaining to codes on sparse graphs and to sum-product decoding algorithm were, in essence, invented 40 years ago by Gallager [3] but were subsequently neglected by the coding community. Zyablov and Pinsker [41] and Margulis [42] studied Gallager's LDPC codes in earlier times. In 1981, Tanner [5] wrote a landmark paper that formally introduced the graphical model notation for describing codes, proved the optimality of the sum-product algorithm in cycle-free graphs, and founded the topic of algebraic methods for constructing graphs suitable for sum-product decoding.

Recent excitement about codes on sparse graphs and sum-product decoding was sparked by the excellent simulation performance exhibited by the turbo codes of Berrou, Glavieux and Thitimajshima [6], MacKay and Neal's [7] near-capacity results on Gallager codes, and the linear-complexity expander graph codes of Sipser and Spielman [43]. Many researchers quickly recognized a unifying theme in the iterative decoding algorithms, and papers showing the connections between iterative decoding of codes on sparse graphs and algorithms in the artificial intelligence [44,45] and systems theory communities were published by Wiberg [46], Loeliger and Koetter [47], McEliece, MacKay and Cheng [48], Kschichang and Frey [49], and Aji and McEliece [50].

The asymptotic analysis of codes on sparse graphs has attracted a lot of attention lately. It has been shown by Luby, Mitzenmacher, Shokrollahi and Spielman [51–53] that an ensemble of randomly constructed codes defined on irregular sparse graphs asymptotically achieves the channel capacity of the binary erasure channel. An important general discovery that arose from this work was the superiority of irregular Tanner graphs. Inspired by their analytical techniques, Richardson, Urbanke and Shokrollahi [9,10] have been able to design long irregular LDPC codes that for all practical purposes achieve the Shannon limit of binary additive white Gaussian noise (AWGN) channels using the density evolution approach.

More recently, using an improved density evolution algorithm, Chung, Forney, Richardson, and Urbanke [54] have designed an ensemble of rate-1/2 codes having a threshold within 0.0045 dB of the Shannon limit, and a performance within 0.036 dB of the Shannon limit at a bit-error rate of $10^{-6}$ with a block length of $10^{7}$.

## 1.3 Results of the Thesis

The main subject of this thesis is *low-delay low-complexity error-correcting codes on sparse graphs*, in particular on sparse Tanner graphs or bipartite graphs, with an emphasis on *code construction and decoding algorithm*. The term low-delay has two meanings: first the block length of the error-correcting codes is confined to a relatively small number (depending on the specific application of interest) so that the waiting time for encoding a block of data and for collecting a block of received signals for the purpose of decoding is strictly fixed; secondly, the decoding delay of one block is minimized by two levels of parallelism in the iterative decoding procedure: the node level and the edge level. The node-level parallelism is widely recognized, whereas the edge-level parallelism explored in this thesis is original.

Special focus is on the following topics:

- The capacity of finite-block-length constrained channels is investigated. For the AWGN channel, the maximum achievable (transmit) rate under the finite-block-length constraint, referred to as $(\epsilon, n)$-capacity $C_{\epsilon}^{n}$, with $\epsilon$ denoting the target block-error probability and $n$ the block length, is identified.

- We investigate a family of time-varying block codes based on a probabilistic construction that closely approaches the finite-block-length constrained capacity and provably achieves the Shannon limit asymptotically over the AWGN channel. We present an improved construction of a probabilistic code with *correlated* codewords, enhancing its asymptotic Euclidean distance by introducing a specific amount of correlation between codewords. Analytical results show that, if the correlation coefficients are uniformly chosen to be $-1/(2^{nR}-1)$, where $2^{nR}$ denotes the number of codewords, the corresponding probabilistic code is asymptotically the best code in the sense of maximizing the expected Euclidean distance, known as the sphere-packing code.

- We propose a general method for constructing Tanner graphs having a large girth by progressively establishing edges or connections between symbol and check nodes in an edge-by-edge manner, called the progressive edge-growth (PEG) algorithm. The algorithm is simple, flexible, yet powerful to generate good LDPC codes.

- Lower bounds on the girth of PEG Tanner graphs and on the minimum distance of the resulting LDPC codes are derived in terms of parameters of the graphs. The PEG construction attains essentially the same girth as Gallager's explicit construction for regular graphs, both of which meet or exceed an extension of the Erdös–Sachs bound. An asymptotic analysis of a relaxed version of the PEG construction is performed.

- We describe an empirical approach using a variant of the "downhill simplex"search algorithm to design irregular PEG graphs for small codes with fewer than a thousand bits, complementing the density evolution approach originally devised for larger, randomly constructed codes. The encoding of LDPC codes based on the PEG principle is also being investigated. We show how to exploit the PEG construction to obtain LDPC codes that allow linear time encoding.

- We investigate regular and irregular LDPC codes using PEG Tanner graphs but allowing symbol nodes to take values over GF($2^b$), $b > 1$. It is demonstrated that one can consistently improve the performance of binary interpretation of sparse-graph GF($2^b$) codes over the binary AWGN channel under iterative decoding, with larger field order $b$, while gradually decreasing the average column weight.

- Analytical and simulation results show that the binary interpretation of GF($2^b$) codes on the cycle Tanner graph TG($2$, $d_c$), if $b$ reaches sufficiently large values, can be used over the binary-input AWGN channel as "good code for optimum decoding" and "good code for iterative decoding"

- Efficient implementations of the sum-product algorithm (SPA) for decoding LDPC codes using log-likelihood ratios (LLR) as messages between symbol nodes and parity-check nodes are presented.

- Various reduced-complexity derivatives of the LLR-SPA are proposed. Both serial and parallel architectures for the check-node update are investigated, leading to trellis and tree topologies, respectively. Density evolution is utilized as an analyzing tool to verify the effectiveness of approximate LLR-SPAs. The unified treatment of decoding techniques for LDPC codes provides flexibility in selecting the appropriate design point in high-speed applications in terms of performance, latency and computational complexity.

- We propose a new message-passing schedule, characterized by a parallel windowed forward-backward (PWFB) schedule over a trellis, for joint iterative decoding of LDPC codes and partial response (PR) channels. It is demonstrated by simulation that with an appropriate selection of the window size this schedule attains low latency and essentially the same performance as the ordinary forward-backward schedule (over the PR channel) with the same computational complexity in the context of turbo equalization of LDPC-coded partial response system.

## 1.4 Outline of the Thesis

In Chapter 2 we start with the definition of the channel capacity involving the block length and target block-error probability. Using Shannon's spherical random-coding bound and sphere-packing bound, we provide numerical examples of finite-block-length capacity of the AWGN channel. We propose two constructions of probabilistic codes, one with independent codewords and the other with correlated codewords. This chapter provides some knowledge of and insight into "good codes for optimum decoding" for finite-block-length constrained channels.

One of the main results of this thesis is the PEG construction of Tanner graphs that have a large girth and respectable minimum distance in their resulting LDPC codes. The PEG algorithm and its features are the main topic of Chapter 3. We formally present the description of the PEG algorithm and derive a lower bound on the graph girth. Following Tanner's approach, we are able to provide a lower bound on the minimum distance of the resulting LDPC codes, thanks to the lower bound of the girth. The asymptotic analysis of the PEG construction is treated as well. In this chapter, we also consider practical issues such as how to design a good irregular degree sequence for the PEG construction, how to incorporate the linear-time encoding property into the PEG construction, and how to extend it to GF($2^b$)

codes. Finally we offer extensive simulation results.

The binary interpretation of GF($2^b$) codes on the cycle Tanner graph TG(2, $d_c$) is the main subject of Chapter 4. We derive an expression for the average Hamming-weight spectrum of sparse-graph GF($2^b$) codes and analyze their corresponding "cluster belief propagation" decoding algorithm in terms of the Kikuchi approximation in the context of statistical physics. Constructions of TG(2, $d_c$) on the basis of Moore and Ramanujan graphs are also treated.

Chapter 5 deals with the topic of low-delay and low-complexity digital implementation of LLR-SPA for decoding binary LDPC codes. We investigate LLR-SPA from both the architectural and the algorithmic point of view. Trellis and tree topologies and their corresponding core operations are derived and simplified. This chapter also consists of three approximates of LLR-SPA, focusing on a maximum level of simplicity at the expense of accuracy and correctness of SPA. Density evolution is utilized as an analyzing tool to verify the effectiveness of approximate SPAs.

Chapter 6 deals with the turbo equalization for an LDPC-coded partial-response channel from the message-passing scheduling perspective. Within the framework of factor graph model and the associated SPA algorithm, (almost) all existing turbo equalization algorithms boil down to various message-passing schedules.

Finally, Chapter 7 contains the conclusions and an outlook on possible further research activities in the area of this thesis.

# Chapter 2

# Finite-Block-Length-Constrained Channel Capacity

## 2.1 Introduction

Ideas presented in the special issue on codes and graphs and iterative algorithms [55] allow to approach the Shannon limit of the AWGN channel to within hundredth of a decibel at the expense of very long block lengths. However, in most applications where the system delay is strictly limited, approaching the Shannon limit becomes problematic. To construct good block codes, the major parameters of interest are the probability of block (word) decoding error $p_w$, the code block length $n$, and the rate $R$. The Shannon Noisy Channel Coding Theorem [1] states that, if $R$ is less than the Shannon limit $C$, no matter how close they are, surely there exist codes for which the word error probability $p_w$ becomes small exponentially with increasing $n$. There are, of course, prices to be paid for increasing the block length, one of which is *coding latency.* At the transmitter side, the first information bit in a block of incoming data stream must generally be delayed by $n$ samples (or symbols) before a codeword can be formed, and at the receiver it is the same case that decoding also requires a complete codeword. The block length $n$ is thus referred to as *coding latency.* Note that the coding latency differs from the processing delay in that it is inherent in a coding scheme and can not be reduced by increasing the processing capability.

The Shannon limit assumes infinite coding latency, but in practice any application of interest is often constrained with a fixed coding latency that is tolerable or affordable, and a target error probablity $p_w$. It is tempting to ask the following questions: what is the maximum achievable rate under a finite-block-length constraint and the target-error-probability

$p_w$ constraint, and subsequently, what is the optimum coding scheme in such a case? These two fundamental issues will be addressed in this chapter [1].

In Section 2.2, we introduce a modified definition of channel capacity, referred to as $(\epsilon, n)$-capacity $C_\epsilon^n$, in the sense that the (block) error probability of a block code with block length $n$ is required to be no larger than $\epsilon$. It follows immediately from the definition that the Shannon limit $C$ turns out to be an asymptotic version of $(\epsilon, n)$-capacity, namely $C = \lim_{\epsilon \to 0} \lim_{n \to \infty} C_\epsilon^n$. Utilizing the upper and lower bounds on the block-error probability of a code as a function of its block length, we derive a universal approach to calculate the respective lower and upper bounds on the $(\epsilon, n)$-capacity of the AWGN channel. Numerical results agree with the old folk theorem that random codes are always good for large block lengths [1, 56], and more importantly, we observe that even for moderate to relatively small block lengths, the average performance of the ensemble of spherical random codes is remarkably close to that of a "best" sphere-packing code, despite the fact that a sphere-packing code does not necessarily exist at all. This observation strongly suggests a new philosophy to construct good codes, i.e., the use of probabilistic method to "mimic" the ensemble of spherical random codes, rather than the traditional idea of searching for a *deterministic* "best" code.

Section 2.3 deals with a particular construction of a probabilistic code over AWGN channels. Codewords are generated by i.i.d. Gaussian random variables. As such, the codebook is inherently *time-varying* from block to block, namely each time that an information block is transmitted, the old codebook is discarded and a new codebook is generated. As the Shannon limit is an asymptotic version of the $(\epsilon, n)$-capacity, optimum codes in the sense of achieving the $(\epsilon, n)$-capacity should at least be capable of achieving it asymptotically. We show that, if the block length $n$ tends to infinity, the probabilistic code is indeed capable of achieving the Shannon limit by means of a suboptimum decoding procedure—typical set decoding. This section is basically a revisit of the Shannon's noisy channel coding theorem.

The optimum detector (minimum probability of block-error) for the probabilistic code is investigated in Section 2.4. The error probability performance of the optimum detector is analyzed and approximately formulated. Further, we argue that the performance of the probabilistic code with independent codewords approaches closely the average performance of the ensemble of spherical random codes for moderate to large block lengths, implying that the probabilistic code closely approaches the $(\epsilon, n)$-capacity. This reasoning depends primarily on the empirical observation that the random-coding bound is nearly indistinguishable from the sphere-packing bound for a wide range of combinations of block lengths and rates.

[1]This work has been presented in part at the 2001 Canadian Workshop on Information Theory, Vancouver, BC. Also submitted to IEEE Trans. Information Theory.

Much better insight can be obtained by establishing a direct relationship between the probabilistic code and the sphere-packing code. In Section 2.5 we investigate the (normalized) Euclidean distance properties of the probabilistic code with independent codewords over the AWGN channel. Using a linear transformation, we present a new construction of the probabilistic code with correlated codewords, improving its asymptotic (normalized) Euclidean distance by the introduction of correlation between codewords. Analytical results show that if the correlation coefficients are chosen uniformly to be $-1/(M-1)$, where $M$ is the number of codewords, the corresponding probabilistic code is asymptotically (in the sense of block length) the best code maximizing the expected Euclidean distance between codewords, incidently being the sphere-packing code.

## 2.2 The $(\epsilon, n)$-Capacity

The definition of channel capacity involving coding latency $n$ and a target block-error probability $\epsilon$ is the following.

**Definition 2.1** *Given a discrete-time memoryless noise channel (DMC) with input space $\mathcal{X}$. Let $\mathcal{C}^n$ be the set of codes of block length $n$, and each component of the codewords be selected from the space $\mathcal{X}$. Consider any code $c^n \in \mathcal{C}^n$, whose block-error probability $p_w(c^n)$ over the DMC is no greater than $\epsilon$, then the maximum rates of the set of $c^n$ is called the $(\epsilon, n)$-capacity $C_\epsilon^n$ of the channel, namely*

$$C_\epsilon^n := \max_{c^n \in \mathcal{C}^n} \{R(c^n) : p_w(c^n) \leq \epsilon\},$$

*where $R(c^n)$ is the rate of code $c^n$.*

The Shannon limit $C$ is defined as the maximum rate that is $(\epsilon, n)$-achievable for all $0 < \epsilon < 1$ and for all positive integers $n$. It follows immediately from the definition that

$$C = \lim_{\epsilon \to 0} \lim_{n \to \infty} C_\epsilon^n. \tag{2.1}$$

For any noisy channel, it can be easily checked that

$$\lim_{\epsilon \to 0} C_\epsilon^n = 0, \quad \text{for any finite } n, \tag{2.2}$$

which justifies the necessity of the inclusion of target error probability in the $(\epsilon, n)$-capacity.

Closely related to the $(\epsilon, n)$-capacity is the so-called $\epsilon$-capacity $C_\epsilon$ where the block length goes to infinity, namely

$$C_\epsilon := \lim_{n \to \infty} C_\epsilon^n .$$

If $\epsilon$ denotes the target *bit* error probability, the $\epsilon$-capacity $C_\epsilon$ evaluates to $\frac{C}{1-h(\epsilon)}$, where $h(\cdot)$ is the binary entropy function, i.e. $h(\epsilon) = -\epsilon \log \epsilon - (1-\epsilon)\log(1-\epsilon)$. This can be understood as follows: First we start with a source producing data at a rate of $C_\epsilon$ bits/second with probability 1/2 of the data being 0 and probability of 1/2 of the data being 1, then we can compress that data source into a source with a smaller rate by allowing some errors in the reproduction. If we require the bit-error probability to be $\epsilon$ or less, then from rate-distortion theory [57] we can compress to the rate $C_\epsilon(1 - h(\epsilon))$. This is the best we can do in compression and these compressed bits can be reliably communicated over a channel with capacity $C = C_\epsilon(1-h(\epsilon))$. It follows that $C_\epsilon = \frac{C}{1-h(\epsilon)}$.

Back to the case where $\epsilon$ denotes *block* error probability, we recall Eq. (8.110) in [18],

$$R \leq \epsilon R + \frac{1}{n} + C$$

Now letting $n \to \infty$, we see that

$$\begin{aligned} R &\leq \frac{C}{1-\epsilon} \\ &= C(1+\epsilon+\epsilon^2+\epsilon^3+\cdots)\ , \end{aligned}$$

which is a weak upper bound on $C_\epsilon$ because Eq. (8.110) leads to the weak converse to the channel coding theorem. The strong converse to the channel coding theorem states that, the Shannon capacity is a very clear dividing point — at rates below $C$, $\epsilon \to 0$ exponentially, and at rates above $C$, $\epsilon \to 1$ exponentially with block length $n$. Therefore, for any finite (non-vanishing) $\epsilon$, $C_\epsilon$ is essentially the same as $C$.

For the sake of simplicity, we concentrate on a discrete-time memoryless AWGN channel (as shown in Fig. 2.1), which often serves as a basic modeling tool for many other kinds of non-ideal channels. Before going on, we cite the well-known Shannon channel capacity formula — the supremum of all rates $R$ for which there exists at least one code with vanishing error probability, that is

$$C = \max_{p_X} I(X;Y), \tag{2.3}$$

where $p_X$ denotes the probability distribution of the real-valued channel input $X$ and $Y$ is the real-valued channel output. This formula holds for any ergodic memoryless channels [58]. Specifically, the Shannon limit of a Gaussian channel with average power constraint $P$ and

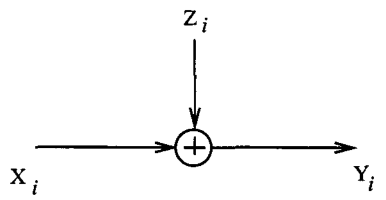


**Figure 2.1:** The discrete-time additive white Gaussian noise channel.

noise variance $\sigma^2$ is [2]

$$\begin{aligned} C &= \max_{EX^2 \le P} I(X;Y) \\ &= \frac{1}{2}\log_2\left(1+\frac{P}{\sigma^2}\right) \end{aligned} \tag{2.4}$$

and the maximum is attained only when $X \sim \mathcal{N}(0,P)$, where $\mathcal{N}(0,P)$ is a zero-mean Gaussian distribution of variance $P$.

Evaluating the $(\epsilon, n)$-capacity is not as simple as calculating the Shannon limit, but we can nevertheless employ known analytical results to obtain an upper-bound as well as a lower-bound on the $(\epsilon, n)$-capacity. A classic lower-bound on the error probability for codes of a specific block length is the sphere-packing bound developed by Shannon [59]. This bound has been recently employed as an important tool to evaluate the "imperfectness" of turbo codes [60] and has also been treated in [61]. The problem posed by Shannon is to estimate as well as possible, the probability of error for a "best" code of length $n$ containing $M$ codewords, each of power $P$ and perturbed by Gaussian noise of variance $\sigma^2$. We denote this minimum or optimum probability of error by $p_w^{\mathrm{opt}}(M, n, \sqrt{P/\sigma^2})$. The sphere-packing bound, $Q_{sp}$, is equal to the probability that the output sequence $Y$ of an AWGN channel will not be confined to the cone with a solid half-angle $\theta$ centralized with respect to the transmitted codeword, which can be expressed in the form

$$\begin{aligned} p_w^{\mathrm{opt}}(M,n,\sqrt{P/\sigma^2}) \ge Q_{sp}(\theta) = \int_\theta^\pi \frac{(n-1)(\sin\phi)^{n-2}}{2^{n/2}\sqrt{\pi}\,\Gamma(\frac{n+1}{2})} \cdot \\ \int_0^\infty r^{n-1} e^{-(r^2+nA^2-2r\sqrt{n}A\cos\phi)/2} dr\, d\phi, \end{aligned} \tag{2.5}$$

where $A$ is the square root of the signal-to-noise ratio (SNR), i.e., $\sqrt{P/\sigma^2}$, $\Gamma(p)$ is the Gamma function $\int_0^\infty t^{p-1}e^{-t}dt$, and $\theta$ is the root of the following equation:

$$\int_0^\theta \frac{n-1}{n} \frac{\Gamma(\frac{n}{2}+1)}{\Gamma(\frac{n+1}{2})\sqrt{\pi}} (\sin\phi)^{n-2} d\phi = 2^{-nR}. \tag{2.6}$$

[2]Throughout the thesis we only deal with real-valued channels; however the results can easily be extended to complex channels, i.e., bandpass signals represented in the equivalent complex baseband. In that case, the factor of 1/2 in Eq. (2.4) does not appear.

For moderate to large $n$, Eq. (2.5) can be approximated with great accuracy by

$$Q_{sp}(\theta) \approx \frac{[G(\theta)\sin\theta e^{-(A^2-AG(\theta)\cos\theta)/2}]^n}{\sqrt{n\pi}\sqrt{1+G^2(\theta)}\sin\theta[AG(\theta)\sin^2\theta-\cos\theta]}, \tag{2.7}$$

where $G(\theta) = (1/2)[A\cos\theta + \sqrt{A^2\cos^2\theta+4}]$, and Eq. (2.6) becomes, asymptotically,

$$\frac{\Gamma(\frac{n}{2}+1)(\sin\theta)^{n-1}}{n\Gamma(\frac{n+1}{2})\sqrt{\pi}\cos\theta} = 2^{-nR}. \tag{2.8}$$

The sphere-packing lower bound on word error probability would be reached with equality only if the code were a *perfect* code for the channel, i.e., if equal-size non-intersecting cones could be drawn around every codeword to completely fill the $n$-dimensional space. Such a partitioning is clearly possible only for $n$=1 or 2, if $M > 2$ [59]. It is very plausible intuitively that any realistic code would have a higher error probability than a sphere-packing code. Recognizing the monotonically increasing error probability with more codewords, the rates specified by the sphere-packing bound can naturally be utilized to *upper-bound* the $(\epsilon, n)$-capacity. Formally, we obtain

$$C_\epsilon^n \leq \max\{R : p_w(2^{nR}, n, \sqrt{P/\sigma^2}) = Q_{sp}(\theta) \leq \epsilon\}. \tag{2.9}$$

Shannon also computed an upper bound on word error probability by a spherical "random coding" method [59]. The random coding bound gives an expression for the ensemble average word error probability, averaged over the ensemble of all possible spherical codes, where each codeword is selected independently and completely at random, subject to an equal energy constraint. As $n$ grows large enough, an asymptotic formula of the random coding bound turns out to be the sphere-packing bound multiplied by a factor essentially independent of $n$, that is

$$\begin{aligned} & p_w^{\text{opt}}(M, n, \sqrt{P/\sigma^2}) \leq Q_{rc}(\theta) \\ = \ & Q_{sp}(\theta) + 2^{nR}\int_0^\theta \frac{\Gamma(\frac{n}{2}+1)(\sin\phi)^{n-1}}{n\Gamma(\frac{n+1}{2})\pi^{1/2}\cos\phi}\sqrt{\frac{n}{\pi}} \\ & \cdot\frac{[G(\theta)\sin\phi\exp(-\frac{P}{2\sigma^2}+\frac{1}{2}\sqrt{\frac{P}{\sigma^2}}G(\theta)\cos\phi)]^n}{\sqrt{1+G^2(\theta)\sin^2\phi}}d\phi \\ \approx \ & Q_{sp}(\theta)\left(1+\frac{AG(\theta)\sin^2\theta-\cos\theta}{2\cos\theta-AG(\theta)\sin^2\theta}\right). \end{aligned} \tag{2.10}$$

As the average error probability over the ensemble of spherical random codes satisfies Eq. (2.10), it is clear that at least one code in the ensemble must have a sufficiently small error probability, i.e. at least one code of block length $n$ meets the target error probability $\epsilon$ with a certain rate, which in turn, gives rise to a *lower bound* on the $(\epsilon, n)$-capacity. That is

$$C_\epsilon^n \geq \max\{R : p_w(2^{nR}, n, \sqrt{P/\sigma^2}) = Q_{rc}(\theta) \leq \epsilon\}. \tag{2.11}$$

It is worth emphasizing that, in the case of moderate to large $n$, the multiplying factor in Eq. (2.10) is just a little over unity; the sphere-packing and the random-coding bounds are close together, thereby yielding a sharp estimate of the $(\epsilon, n)$-capacity.

The significance of the definition of the $(\epsilon, n)$-capacity can be seen from the following numerical example. Consider an AWGN channel on which we wish to transmit information with rate 1 bit per sample, for which the minimum SNR specified by the Shannon limit is 3.0 (about 4.7712 dB). By applying the sphere-packing bound, Fig. 2.2 shows that, for the same code rate, the minimum threshold for reliable communication (in the sense of achieving a target error probability) is significantly higher than the Shannon limit, provided that the code block length is constrained to a relatively small size. For some real-time applications where large delay is not tolerable, the Shannon limit does not convey much useful information, but the $(\epsilon, n)$-capacity reveals the ultimate limit instead. It is also suggested that, even if a code operates far from the Shannon limit it might perform nearly as well as the best code possible

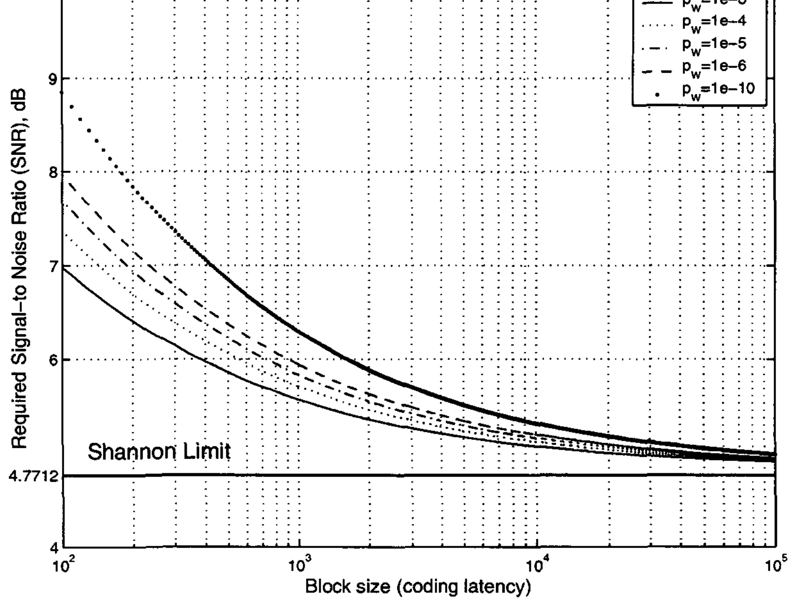


**Figure 2.2:** The minimum required signal-to-noise ratio (SNR) of the Shannon sphere-packing bound for codes with varying block length $n$ and rate 1 bit per sample, operating over a continuous-input AWGN channel at $p_w = 10^{-3}$, $10^{-4}$, $10^{-5}$, $10^{-6}$, and $10^{-10}$, respectively.

of the same length. The same message has been delivered in [60,61]

A quantitative overview of the $(\epsilon, n)$-capacity (upper bound) versus the Shannon limit is exhibited in Fig. 2.3 where an AWGN channel at SNR of 3.0 is considered. It should not be surprising that, if the code block length is less than $10^4$, only the rates significantly lower than the Shannon limit are achievable. For example, codes of block length 100 have a penalty of 0.3 bits per sample with $p_w = 10^{-3}$, and even a penalty of over 0.5 bits per sample with $p_w = 10^{-10}$, as compared with the Shannon limit. The Shannon limit can be approached within 0.05 bits per sample only for block lengths 100,000 and greater. Again, it is evident that, not the Shannon limit, but the $(\epsilon, n)$-capacity should be employed in evaluating a practical coding scheme with finite block lengths.

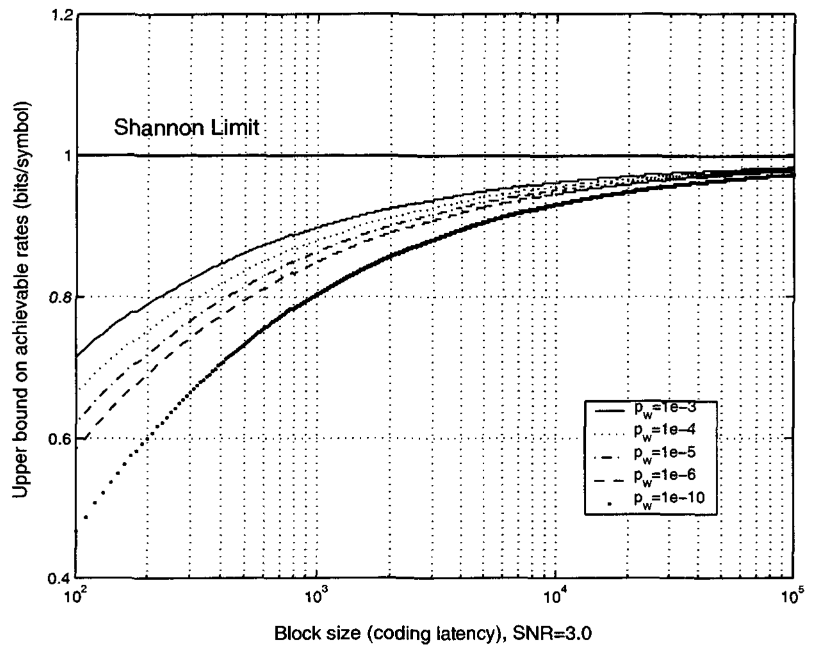


**Figure 2.3:** Upper bound on $(\epsilon, n)$-capacity of the sphere-packing bound for codes with varying block length $n$, operating over a continuous-input AWGN channel at a SNR of 3.0 (4.7712 dB), and $p_w = 10^{-3}$, $10^{-4}$, $10^{-5}$, $10^{-6}$, $10^{-10}$, respectively.

Plotted in Figs. 2.4 and 2.5 are performance comparisons of two imaginary codes, one of which is assumed to achieve the sphere-packing bound and the other the spherical random-coding bound, with the target error probability $p_w = 10^{-6}$ and with varying block lengths. In particular, information on the upper and lower bounds on the $(\epsilon, n)$-capacity is shown in Fig. 2.5. In this specific setting and for $n \geq 100$, the upper and lower bounds on the $(\epsilon, n)$-capacity are close enough together, thereby delivering precise information concerning the $(\epsilon, n)$-capacity, whereas when $n < 100$, the upper bound and the lower bound are apart and the question of determining $(\epsilon, n)$-capacity for this block-length region still remains open.

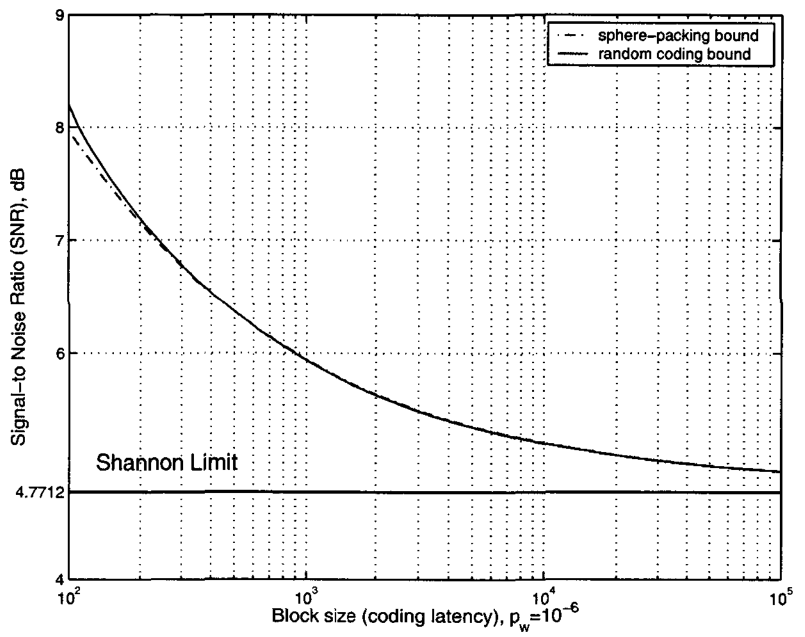


**Figure 2.4:** Signal-to-noise ratio of the spherical random-coding bound (as compared with the sphere-packing bound) for codes with varying block length $n$, operating over a continuous-input AWGN channel at a SNR of 3.0 and $p_w = 10^{-6}$.

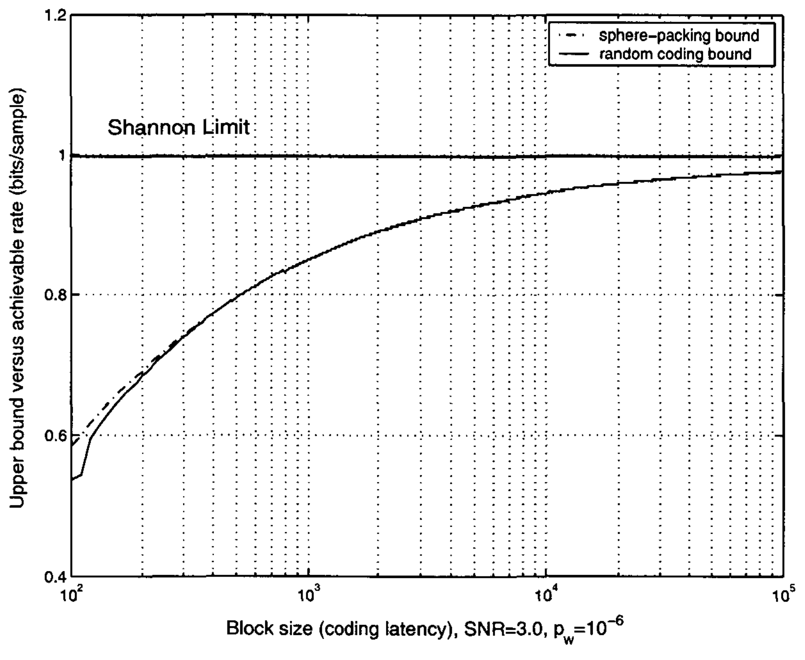


**Figure 2.5:** Achievable rates of the spherical random-coding bound (as compared with the sphere-packing bound) for codes with varying block length $n$, operating over a continuous-input AWGN channel at a SNR of 3.0 and $p_w = 10^{-6}$. Note that the discontinuity in the short-block-length region is caused by numerical instability.

Additional insight into the implications of Fig. 2.4 and Fig. 2.5 may be obtained by re-examining the definitions of the sphere-packing bound and the spherical random-coding bound. As we know, the performance limit corresponding to the sphere-packing bound would be reached with equality only if the code were a *perfect* spherical code for the continuous-input AWGN, i.e., if equal-sized cones could be drawn around every codeword so as to completely fill the $n$-dimensional space without intersecting. Actually this is impossible for all $n > 2$ and $M > 2$. On the other hand, we observe empirically that the spherical random-coding bound is virtually indistinguishable from the sphere-packing bound for block lengths greater than a few hundreds. [3] Therefore it is tempting to construct probabilistic codes to "approximate" the ensemble of spherical random codes, rather than search for a deterministic "best" code.

## 2.3 Channel Coding Theorem Revisited

As discussed previously, the major motivation to construct probabilistic codes is to mimic the ensemble of spherical random codes, thereby closely approaching the sphere-packing bound over a wide range of block lengths and rates. This idea is fundamentally different from the traditional effort to search for a deterministic best code.

**Definition 2.2** *Probabilistic code with independent codewords—A $(n, M)$ probabilistic code for a certain DMC with channel input space $\mathcal{X}$ under average power constraint $P$ consists of the following:*

- *For each encoding block, generate $M$ codewords $X_1^n, X_2^n, \ldots, X_M^n$, that satisfy the power constraint $P$, i.e., for every codeword*

$$\sum_{i=1}^{n} x_{i,w}^2 \leq nP, \qquad w = 1, 2, \ldots, M, \tag{2.12}$$

  *where codewords $X_w^n = (x_{1,w}, x_{2,w}, \ldots, x_{n,w})$ are created by independent identically distributed (i.i.d.) random variables $X_w \in \mathcal{X}$, subject to a common probability distribution $p_X$ [58] maximizing input-output mutual information $I(X;Y)$, e.g., for an AWGN channel, $p_X \sim \mathcal{N}(0, P)$.*

[3]More empirical evidence on the binary erasure channel (BEC), the binary symmetric channel and the AWGN channel can be found in [60, 61]. It is worth mentioning that the discrepancy between the random-coding bound and sphere-packing bound depends not only on the block length $n$, but also on the rate $R$ of the code. More precisely, it will be shown in Section 2.5 that the discrepancy decreases exponentially with the product of $nR$.

- *An encoding function $X : \{1, 2, \dots, M\} \to \mathcal{X}^n$, selecting one codeword from the codebook and passing it through the channel.*

- *A synchronization scheme between the encoder and decoder that guarantees that the decoder will generate an exact copy of the codebook of encoder for each block. In other words, the receiver has perfect knowledge of the random sources $X_w$.*

- *A decoding function*

$$g : \mathcal{Y}^n \to \{1, 2, \dots, M\}, \tag{2.13}$$

  *which is a deterministic rule that assigns an estimate to each possible received vector.*

The definition of probabilistic codes has the effect of "combined coding and modulation" (baseband), i.e. the encoder feeds its output directly to the noise channel. In our notation, the transmission rate is measured as $R = \frac{\log_2 M}{n}$, which can be made readily larger than 1 bit per sample simply by generating a large number of codewords such that $M = 2^{nR}$. As there is no explicit coding redundancy involved, the probabilistic code is also referred to as probabilistic signaling.

The probabilistic code is inherently time-varying, i.e., the codebook varies from block to block and the codeword with the same index $w$ does not remain the same for different blocks. This scheme differs from the traditional concept wherein a *deterministic* codebook is selected once and used repetitively. The time-varying nature ensures that the channel input resembles a stochastic process with an appropriate distribution, which maximizes the mutual information of the channel input and output.

The probabilistic code should not be confused with the standard method of proof of coding theorems based on a *random-coding argument* [62]. Whereas a probabilistic code constitutes a communication technique, a random-coding argument is a proof technique often used to establish the existence of a (single) deterministic code which yields good performance on a specific channel without actually constructing the code. This is done by introducing a probability mass function (pmf) on an ensemble of codes, computing the corresponding average performance over such an ensemble, and then invoking the argument to show that if this average performance is good, then there must exist at least one code in the ensemble with good performance. In contrast, a probabilistic code constitutes a communication technique, the implementation of which requires the availability of a common source of randomness at the transmitter and receiver.

Armed with this formal definition of the probabilistic code, it is now straightforward to prove that a probabilistic code is a capacity-achieving code.

**Theorem 2.1** *The Shannon limit of a Gaussian channel with power constraint $P$ and noise variance $\sigma^2$,*

$$C = \frac{1}{2}\log_2(1 + \frac{P}{\sigma^2})$$

*can be attained by a probabilistic code.*

**Proof**: We will use the same decoding rule as in [18], namely, joint typicality decoding.

1. *Generation of probabilistic codebook.* For every encoding block, we generate codewords with each element i.i.d. according to a normal distribution with variance $P - \epsilon$. For large $n$, we have $\frac{1}{n}\sum x_{i,w}^2 \to P-\epsilon$, then the probability that a codeword does not satisfy the power constraint is quite small. Assume $x_{i,w}, i = 1, 2, \dots, n, w = 1, 2, \dots, 2^{nR}$ be i.i.d. subject to $\mathcal{N}(0, P - \epsilon)$, which form codewords $X_1^n, X_2^n, \dots, X_{2^{nR}}^n \in \mathcal{R}^n$. Thus for each particular encoding block, we have $2^{nR}$ codewords as the *rows* of a matrix

$$\begin{bmatrix} x_{1,1} & x_{2,1} & \cdots & x_{n,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,2^{nR}} & x_{2,2^{nR}} & \cdots & x_{n,2^{nR}}. \end{bmatrix}_{2^{nR}\times n}$$

2. *Encoding.* To send the message index $w$, the transmitter sends the $w$th codeword $X_w^n$ in the codebook.

3. *Decoding.* The receiver first regenerates the same codebook as that of the transmitter of the corresponding block, then searches the list of codewords $\{X_w^n\}$ for one that is jointly typical with the received vector. If there is one and only one such codeword, the receiver declares it to be the transmitted codeword. Otherwise the receiver declares an error. The receiver also declares an error if the chosen codeword does not satisfy the power constraint.

4. *Probability of error.* By the symmetry of the code construction, the probability of error $p_w$ does not depend on the particular index that was sent. Without loss of generality, assume that codeword 1 was sent. Thus $Y^n = X_1^n + Z^n$, where $Z^n$ is the channel noise vector.

Define the following events:

$$E_0 = \left\{\frac{1}{n}\sum_{i=1}^{n} x_{i,1}^2 > P\right\}.$$

and

$$E_i = \{(X_i^n, Y^n) \quad \text{is in joint typical set } A_\epsilon^{(n)}\}.$$

An error occurs if $E_0$ occurs (the power constraint is violated) or $E_1^c$ occurs (the transmitted codeword and the received sequence are not jointly typical) or $E_2 \cup E_3 \cup \cdots \cup E_{2^nR}$ occurs (an incorrect codeword is jointly typical with the received sequence). Hence the word error probability can be expressed as

$$\begin{aligned} p_w &= \Pr(E|w=1) = \Pr(E_0 \cup E_1^c \cup E_2 \cup E_3 \cup \cdots \cup E_{2^{nR}}) \\ &\le \Pr(E_0) + \Pr(E_1^c) + \sum_{i=2}^{2^{nR}} \Pr(E_i) \end{aligned}$$

by the union bound for probabilities. Applying the law of large numbers, $\Pr(E_0) \to 0$ as $n$ tends to infinity. Now, by the joint asymptotic equipartition property (AEP), $\Pr(E_1^c) \to 0$, and hence

$$\Pr(E_1^c) \le \epsilon \text{ for sufficiently large } n.$$

By the code generation process we know that $X_1^n$ and $X_i^n$ are independent, $Y^n$ and $X_i^n$, $\forall i \neq 1$, are also independent. Hence, the probability that $X_i^n$ and $Y^n$ will be jointly typical is $\le 2^{-n(I(X:Y)-3\epsilon)}$ by the joint AEP [18]. Thus

$$\begin{aligned} p_w &= \Pr(E|w=1) = \Pr(E_0 \cup E_1^c \cup E_2 \cup E_3 \cup \cdots \cup E_{2^{nR}}) \\ &\le \Pr(E_0) + \Pr(E_1^c) + \sum_{i=2}^{2^{nR}} \Pr(E_i) \\ &\le \epsilon + \epsilon + \sum_{i=2}^{2^{nR}} \Pr(E_i) \\ &\le 2\epsilon + \sum_{i=2}^{2^{nR}} 2^{-n(I(X:Y)-3\epsilon)} \\ &= 2\epsilon + (2^{nR} - 1)2^{-n(I(X;Y)-3\epsilon)} \\ &\le 2\epsilon + 2^{-n(I(X;Y)-R-3\epsilon)} \\ &\le 3\epsilon \end{aligned}$$

for sufficiently large $n$ and $R < I(X;Y) - 3\epsilon$ which, together with Eq. (2.4), concludes the proof. ■

One might note that the above proof bears much resemblance with the random-coding argument in [18]. This is actually not surprising because in a probabilistic code, the average is done over time (block-by-block) and each block of the probabilistic code (termed member code) is an instance of the ensemble of random codes, that is to say, the average done over member codes of a probabilistic code is equivalent to the average over the ensemble of random codes. Unlike the random-coding argument where care must be taken to avoid trivial codes, there is no need in the probabilistic code to distinguish good or bad member codes.

## 2.4 The Optimum Decoder

In the preceding section, we constructed a time-varying probabilistic code and showed that it is able to achieve the Shannon limit asymptotically via *typical set decoding* for the AWGN channel. Although typical set decoding is asymptotically optimal and conceptually easy to analyze, the drawback is two-fold: firstly it is not optimal in the sense of minimizing the probability of error; secondly, it is merely a statistical term and somewhat cumbersome to obtain error probability performance.

The optimum procedure to minimize the probability of error is the maximum *a-posteriori* probability (MAP) decoding, i.e., the receiver should choose the *a-posteriori* most likely index. Using Bayes' rule, the posterior probabilities may be expressed as

$$p(X_w^n|Y^n) = \frac{p(Y^n|X_w^n)p(X_w^n)}{p(Y^n)}, \tag{2.14}$$

where $p(Y^n|X_w^n)$ is the conditional probability density function (pdf) of the observed vector given $X_w^n$, and $p(X_w^n)$ is the *a-priori* probability of the $w$-th codeword being transmitted. Assume that the $2^{nR}$ codewords are equally probable *a-priori*, i.e., $p(X_w^n) = 2^{-nR}$ for all $w$. Furthermore, note that the denominator in Eq. (2.14) is independent of which codeword is transmitted. Consequently, the decision rule based on finding the codeword that maximizes $p(X_w^n|Y^n)$ is equivalent to finding the codeword that maximizes $p(Y^n|X_w^n)$. It is evident that a detector based on the MAP criterion and the one that is based on the maximum likelihood criterion make the same decisions as long as *a-prior* probabilities $p(X_w^n)$ are all equal.

In the case of the AWGN channel, the logarithm likelihood function of $p(Y^n|X_w^n)$ is given by

$$\ln p(Y^n|X_w^n) = -\frac{1}{2}n\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{k=1}^{n}(y_k - x_{k,w})^2. \tag{2.15}$$

Hence, the maximum of $p(Y^n|X_w^n)$ over $X_w^n$ is equivalent to finding the index $w$ that minimizes the Euclidean distance

$$D(Y^n, X_w^n) = \sum_{k=1}^{n}(y_k - x_{k,w})^2. \tag{2.16}$$

We call the normalized version of $D(Y^n, X_w^n)$ by distance metrics $d(w)$, i.e.,

$$d(w) = \frac{1}{n} D(Y^n, X_w^n),$$

for $w = 1, 2, \ldots, 2^{nR}$. This decision rule is usually referred to as the *minimum distance criterion*. It should be mentioned that each codeword in the probabilistic code is characterized by a discrete stochastic process that maximizes the input-output mutual information. Although the average power remains the same, the energies of particular codewords may not necessarily be the same. Hence, a correlation detector is no longer optimal for the probabilistic code, particularly for small block lengths.

Suppose the codeword 1, $X_1^n$, is transmitted, then the $2^{nR}$ decision variables $d(w)$ are

$$\left\{ \begin{array}{lcl} d(1) & = & \frac{1}{n} \sum_{k=1}^{n} z_k^2 \\ d(2) & = & \frac{1}{n} \sum_{k=1}^{n} (x_{k,1} + z_k - x_{k,2})^2 \\ \vdots & & \vdots \\ d(2^{nR}) & = & \frac{1}{n} \sum_{k=1}^{n} (x_{k,1} + z_k - x_{k,2^{nR}})^2 \end{array} \right. \tag{2.17}$$

where $z_k$ is the channel noise. The decision is made in favor of the codeword $X_w^n$ having the least decision value $d(w)$ among the whole set of all codewords. As all codewords are mutually i.i.d. Gaussian variables and independent of the channel noise, the decision variable $d(1)$ has the normalized chi-square distribution with $n$ degrees of freedom (see Appendix B)

$$f_1(y) = \frac{1}{(2\sigma^2)^{n/2}\Gamma(n/2)} n^{n/2} y^{n/2-1} e^{-ny/2\sigma^2}, \quad y \geq 0 \tag{2.18}$$

and all other decision variables $d(w), w \neq 1$, yield the normalized chi-square distribution of

$$f_w(y) = \frac{1}{[2(\sigma^2 + 2P)]^{n/2}\Gamma(n/2)} n^{n/2} y^{n/2-1} e^{-ny/2(\sigma^2+2P)}, \quad y \geq 0. \tag{2.19}$$

It is interesting to note that each decision variable in Eq. (2.17) is precisely an estimate of the variances of the Gaussian processes — the correct index has the minimum variance of $\sigma^2$ and all other have variances of $\sigma^2 + 2P$. By the Shannon's noisy channel coding theorem, if the rate is less than the channel capacity and as block length $n$ goes to infinity, these estimates tend to be increasingly accurate and the probability of making an error exponentially vanishes.

It is mathematically convenient to first derive the probability that the detector makes a correct decision. This is the probability that $d(1)$ is greater than each of the other $2^{nR} - 1$ decision variables $d(2), d(3), \ldots, d(2^{nR})$. This probability may be expressed as

$$p_c = \int_0^\infty \Pr[d(2) > r, d(3) > r, \cdots, d(2^{nR}) > r | d(1) = r] f_1(r) dr, \tag{2.20}$$

where $\Pr[d(2) > r, d(3) > r, \ldots, d(2^{nR}) > r | d(1) = r]$ denotes the joint probability that $d(2)$, $d(3)$, ..., $d(2^{nR})$ are all greater than $r$, where $r \geq 0$. This joint probability is then averaged over all $r$. Unfortunately, the values of $d_w$, $w = 1, 2, \ldots, 2^{nR}$, are not statistically independent, and the evaluation of the influence of correlations is rather complicated, even impossible. Therefore, we resort to an approximation expression in which the correlations in decision variables are neglected.[4] One approach is to factor the joint probability into a product of $2^{nR} - 1$ marginal probabilities, yielding

$$p_c \approx \int_0^{\infty} [\Pr(d(w) > r | d(1) = r)]^{2^{nR}-1} f_1(r) dr. \tag{2.21}$$

Thus the probability of word error is

$$p_w = 1 - p_c. \tag{2.22}$$

Under the condition that $d(1) = r$, the decision variable $d(w)$ has the normalized chi-square distribution of

$$\frac{1}{[2(r+2P)]^{n/2}\Gamma(n/2)} n^{n/2} y^{n/2-1} e^{-ny/2(r+2P)}, \quad y \geq 0.$$

Thus the conditional probability $\Pr[d(w) > r | d(1) = r]$ yields

$$\Pr[d(w) > r | d(1) = r] = 1 - \int_0^r \frac{1}{[2(r+2P)]^{n/2}\Gamma(n/2)} n^{n/2} y^{n/2-1} e^{-ny/2(r+2P)} dy. \tag{2.23}$$

Without loss of generality, we assume that $n$ is an even integer, the integral can be expressed in closed form by repeated integration by parts, which yields

$$\Pr[d(w) > r | d(1) = r] = e^{-nr/2(2P+r)} \sum_{k=0}^{n/2-1} \frac{1}{k!} \left[\frac{nr}{2(2P+r)}\right]^k. \tag{2.24}$$

Substituting Eq. (2.24) into Eq. (2.21) and Eq. (2.22), the probability of word error can be expressed as

$$p_w \approx 1 - \int_0^{\infty} \frac{1}{(2\sigma^2)^{n/2}\Gamma(n/2)} n^{n/2} r^{n/2-1} e^{-nr/2\sigma^2} \left\{ e^{-nr/2(2P+r)} \sum_{k=0}^{n/2-1} \frac{1}{k!} \left[\frac{nr}{2(2P+r)}\right]^k \right\}^{2^{nR}-1} dr. \tag{2.25}$$

For moderate to large block lengths $n$, the evaluation of Eq. (2.25) becomes rather difficult due to the existence of the factorial of a large number. Hence we derive an asymptotic formula for probabilistic codes, which is based on the connection between probabilistic codes and spherical random codes. Note that the only difference between the probabilistic code and

[4] The correlations in decision variables due to the channel noise $\{z_k\}$ vanish asymptotically as the signal-to-noise ratio increases.

the ensemble of spherical random codes lies in the power constraint. In Shannon's spherical random codes, each codeword is required to lie exactly on the surface of the sphere of radius $\sqrt{nP}$, but in the probabilistic code, all codewords are required to have a power of $P$ in a probabilistic manner. As stated by the law of large numbers, as the block length $n$ tends to infinity, the probability that a codeword deviates from this power constraint can be made arbitrarily small. Thus one may presume that the asymptotic performance of both codes is nearly the same. In fact, the probabilistic code can be transformed to be an instance of the ensemble of spherical random codes by expanding one sample in the block length. Suppose we have a probabilistic code with each block of length $n$, and all codewords satisfy the power constraint $\sum\limits_{k=1}^{n} x_{k,w}^2 \leq nP$. To each codeword, add a further element of such value that the $n+1$ dimensional codeword lies exactly on the $n+1$ sphere surface. Specifically, the added $n+1$ element will have the value

$$x_{n+1,w} = \sqrt{(n+1)P - \sum_{k=1}^{n} x_{k,w}^2}\ . \tag{2.26}$$

This yields a snapshot of spherical random codes with $2^{nR}$ codewords of block length $n+1$, and the overall transmit rate becomes $\frac{n}{n+1}R$. Asymptotically as $n$ tends to infinity, this rate loss is negligible. This justification enables us to use Eq. (2.10) to calculate the asymptotic performance of the probabilistic code based on the ensemble of spherical random codes, namely

$$p_w \approx Q_{sp}(\theta)\left(1 + \frac{AG(\theta)\sin^2\theta - \cos\theta}{2\cos\theta - AG(\theta)\sin^2\theta}\right), \tag{2.27}$$

where $\theta$ is defined in Eq. (2.6). Considering the empirical evidence that the performance of a sphere-packing code is almost indistinguishable from that of the ensemble of spherical random codes for moderate to large block lengths, it is plausibly intuitive that a probabilistic code with independent codewords closely approaches the $(\epsilon, n)$-capacity correspondingly.

In order to quantify how close the probabilistic code with independent codewords approaches the $(\epsilon, n)$-capacity, we will establish an explicit relationship between the probabilistic code and the sphere-packing code later on.

## 2.5 Probabilistic Code with Correlated Codewords

The problem of finding the best possible code is often formulated as a sphere-packing problem, and protection against various kinds of disturbances can be measured in terms of Euclidean distance. The code design problem is therefore equivalent with the geometrical problem of packing a largest number of equal spheres within a given sphere in the $n$-dimensional Euclidean space. Alternatively, one can fix the power of codewords ($P$), and fix the number of codewords ($M = 2^{nR}$), and fix the dimension $n$, then the code design problem is to maximize the pairwise Euclidean distance between codewords. In particular, if a code have a uniform pairwise Euclidean distance between any pair of codewords and the uniform pairwise Euclidean distance attains the largest possible, then such a code is called the sphere-packing code.

The objective of this section is to construct a family of asymptotic sphere-packing codes as the block length tends to infinity.

## 2.5.1 Normalized Euclidean Distance

Recognizing the fact that the codewords $\{X_w^n\}$ are independent of the channel noise, the expected (normalized) decision variables in Eq. (2.17) can be written as

$$\begin{cases} \overline{d(1)} &= \frac{1}{n}\sum_{k=1}^{n} z_k^2 \\ \overline{d(2)} &= \frac{1}{n}\sum_{k=1}^{n}(x_{k,1}-x_{k,2})^2 + \frac{1}{n}\sum_{k=1}^{n} z_k^2 \\ \vdots & \quad \vdots \\ \overline{d(2^{nR})} &= \frac{1}{n}\sum_{k=1}^{n}(x_{k,1}-x_{k,2^{nR}})^2 + \frac{1}{n}\sum_{k=1}^{n} z_k^2 \end{cases} \tag{2.28}$$

Clearly the probability of making an error is closely related to the normalized Euclidean distances of codewords, i.e. the set

$$d_{i,j} = \frac{1}{n}\sum_{k=1}^{n}(x_{k,i}-x_{k,j})^2, \quad \text{for all } i \neq j. \tag{2.29}$$

Because all codewords in a probabilistic code with independent codewords are generated via Gaussian random variables in a time-varying manner, the normalized Euclidean distances between codewords are essentially random variables subject to the normalized chi-square distribution

$$f_D(d) = \frac{1}{(4P)^{n/2}\Gamma(n/2)} n^{n/2} d^{n/2-1} e^{-nd/4P}, \quad d \geq 0 \tag{2.30}$$

whose cumulative distribution function is illustrated in Fig. 2.6 for various $n$.

The normalized Euclidean distance of codewords, $d_{i,j}$, is closely related to the pairwise error probability (PEP). Denote by $\{X_i^n \to X_j^n\}$ a decoder *pairwise error event*, i.e., the decoder chooses $X_j^n$ when $X_i^n$ is the transmitted codeword and $X_i^n$ and $X_j^n$ are the only two possible decoding outcomes. Also denote by $p(X_i^n \to X_j^n)$ the *pairwise error probability*. Then

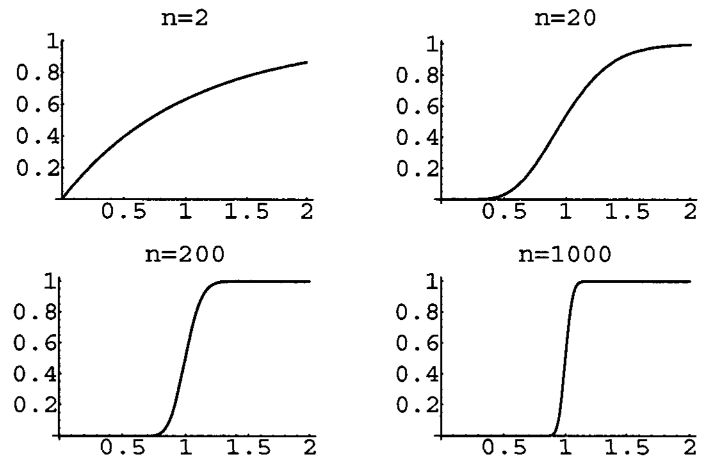


**Figure 2.6:** Cumulative distribution function (cdf) of the normalized Euclidean distance with varying block length $n$, where $P = 0.5$. As $n$ increases, its cdf becomes a step function meaning that the probability mass concentrates on its expectation $2P$.

for an AWGN channel,

$$p(X_i^n \to X_j^n) = Q\left(\sqrt{\frac{nd_{i,j}}{4\sigma^2}}\right), \tag{2.31}$$

where

$$Q(\alpha) = \frac{1}{\sqrt{2\pi}}\int_\alpha^\infty e^{-\beta^2/2}d\beta, \tag{2.32}$$

which is usually referred to as the (Gaussian) $Q$-function. It suffices to see that the probability of error between two codewords decreases exponentially with $d_{i,j}$, owing to the upper-bound

$$Q\left(\sqrt{\frac{nd_{i,j}}{4\sigma^2}}\right) \le \frac{1}{2}e^{-nd_{i,j}/8\sigma^2}. \tag{2.33}$$

As $n$ goes to infinity, we obtain the asymptotic normalized Euclidean distance set [5] $d_{i,j}^{\text{asy}}$ as

$$d_{i,j}^{\text{asy}} \overset{\text{def}}{=} \lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}(x_{k,i} - x_{k,j})^2, \quad i \neq j. \tag{2.34}$$

Under the assumption of independent codewords treated in the previous section, the asymp-

[5] By definition, the asymptotic (normalized) Euclidean distance denotes mathematical expectation over the block length. Thus it can also be interpreted as expected (normalized) Euclidean distance.

totic normalized Euclidean distance between codeword $X_i^n$ and codeword $X_j^n$ becomes

$$\begin{aligned} d_{i,j}^{\text{asy}} &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n}(x_{k,i}-x_{k,j})^2 \\ &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} x_{k,i}^2 + \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} x_{k,j}^2 - \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} 2x_{k,i}x_{k,j} \\ &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} x_{k,i}^2 + \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} x_{k,j}^2 \\ &= 2P, \end{aligned} \tag{2.35}$$

which holds for all $i \neq j$. Note that in Eq. (2.35) the third term $\lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n} 2x_{k,i}x_{k,j}$ falls to zero under independence conditions. We observe that in a probabilistic code with independent codewords, the asymptotic normalized Euclidean distance set $d_{i,j}^{\text{asy}}$ is uniform with a unique value of $2P$ that is precisely twice the transmit power.

As $d_{i,j}$ is an average sum of independent random variables, we can then use the Chernoff bound to give an exponential bound on the probability that $d_{i,j}$ will deviate from its mean $2P$ by more than a fraction $\epsilon$. A stronger bound can be obtained if the elements of codewords are amplitude-limited, or say, under peak-power constraint.

**Lemma 2.1** *Let $\{x_{k,i}\}$ be peak-power constrained, i.e., $\forall\ x_{k,i}^2 < E_p$, then*

$$\Pr\{|d_{i,j} - 2P| > \epsilon\} \leq e^{-\frac{1}{2E_p}\epsilon^2 n}.$$

*Proof:* The proof follows in the exactly the same way as that of the Appendix of [63]. ■

The above Lemma provides a straightforward justification of the probabilistic construction. Because of unlimited uses of member codes of the probabilistic code and the generalized law of large numbers, the performance of the probabilistic code will concentrate exponentially on that of a member code. As the block length $n$ is sufficiently large, the performance of a member code of the probabilistic codes should fall within the $\epsilon$ region of that of the average with probability close to 1.

### 2.5.2 Asymptotically Sphere-Packing Code

The asymptotic normalized Euclidean distance can be made even larger to improve the error probability performance. From the above derivation we know that, if the $i$-th codeword $\{x_{k,i}\}$ has a certain correlation with the $j$-th codeword $\{x_{k,j}\}$, it is also possible for the third term in Eq. (2.35) to contribute a positive value to the asymptotic normalized Euclidean distance.

In this subsection we will present a new construction of probabilistic codes with correlated codewords, which incidently turns out to be a sphere-packing code in the asymptotic sense.

Let us assume that $X_j, j = 1, 2, \ldots, M$ and $M = 2^{nR}$, are i.i.d. Gaussian random variables, which are used to generate independent codewords $X_1^n, X_2^n, \ldots, X_M^n$. These Gaussian random variables are mutually independent and with zero-mean, variances of $P_1, P_2, \ldots, P_M$ whose values will be determined later. Let $\mathbf{M}$ denote the $M \times M$ covariance matrix which is diagonal, i.e. $\mathbf{M} = \text{diag}(P_1, P_2, \cdots, P_M)$. Let $X$ denote the $M \times 1$ column vector consisting of random variables $X_j, j = 1, 2, \ldots, M$. The joint pdf of the Gaussian random vector $X$ is then defined as

$$p(X) = \frac{1}{(2\pi)^{M/2}(\det \mathbf{M})^{1/2}} \exp[-\frac{1}{2}X'\mathbf{M}^{-1}X], \tag{2.36}$$

where $\mathbf{M}^{-1}$ denotes the inverse of $\mathbf{M}$ and $X'$ the transpose of $X$.

Now let us consider a linear transformation [64, pp. 49–52] of $M$ independently Gaussian random variables $X$

$$Y = \mathbf{A}X, \tag{2.37}$$

where $Y$ is an $M \times 1$ column vector, $\mathbf{A}$ is an orthogonal matrix ($\mathbf{A}' = \mathbf{A}^{-1}$). The Jacobian of this transformation is $J = 1/\det(\mathbf{A})$. As $X = \mathbf{A}^{-1}Y$, we may substitute for $X$ in Eq. (2.36) and thus obtain the joint pdf of $Y$ in the form

$$\begin{aligned} p(Y) &= \frac{1}{(2\pi)^{M/2}(\det \mathbf{M})^{1/2}\det \mathbf{A}} \exp[-\frac{1}{2}(\mathbf{A}^{-1}Y)'\mathbf{M}^{-1}(\mathbf{A}^{-1}Y)] \\ &= \frac{1}{(2\pi)^{M/2}(\det \mathbf{Q})^{1/2}} \exp[-\frac{1}{2}Y'\mathbf{Q}^{-1}Y], \end{aligned}$$

where the covariance matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = \mathbf{A}\mathbf{M}\mathbf{A}'. \tag{2.38}$$

This leads to the following lemma:

**Lemma 2.2** *A set of correlated Gaussian random variables $Y$ can be obtained via linear transformation from a set of statistically independent Gaussian random variables $X$.*

The above analysis suggests a method to generate a probabilistic code with correlated codewords via linear transformation from a probabilistic code with independent codewords, which is summarized as follows.

**Definition 2.3** *Probabilistic code with correlated codewords*

- *At each time index $k$, $k = 1, 2, \dots, n$, generate a random (column) vector $X^k = (x_{k,1}, x_{k,2}, \cdots, x_{k,M})'$, where $x_{k,j}$, $j = 1, 2, \dots, M$ and $M = 2^{nR}$, are instances of i.i.d. Gaussian variables whose covariance matrix* $\mathbf{M}$ *equals* $\text{diag}(P_1, P_2, \cdots, P_M)$.
- *At each time index $k$, $k = 1, 2, \dots, n$, compute column vector $Y^k$ via linear transformation $Y^k = \mathbf{A}X^k$.*
- *Finally the codewords are formed by the rows of the $M$-by-$n$ matrix $(Y^1, Y^2, \cdots, Y^n)$.*

Now we shall determine values of $P_j$, $j = 1, 2, \dots, M$ such that the codewords, i.e. the rows of the $M$-by-$n$ matrix $(Y^1, Y^2, \cdots, Y^n)$, satisfy the average power constraint. By application of basic matrix theory, the matrix $\mathbf{A}$ consists of rows that are the eigenvectors of the covariance matrix $\mathbf{Q}$, and $\mathbf{M}$ is a diagonal matrix with elements equal to the eigenvalues of $\mathbf{Q}$. To satisfy the average energy constraint, $P$ should be chosen as the main diagonal elements of the matrix $\mathbf{Q}$, meaning that each codeword has an average power of $P$. To maintain the symmetry, we impose a constraint of equal correlation coefficients in $\mathbf{Q}$, which gives

$$\mathbf{Q} = \begin{bmatrix} P & \rho P & \cdots & \rho P \\ \rho P & P & \cdots & \rho P \\ \vdots & \vdots & \ddots & \vdots \\ \rho P & \rho P & \cdots & P \end{bmatrix}_{M\times M}, \tag{2.39}$$

where $-1 \le \rho \le 1$. Hence, the asymptotic normalized Euclidean distance in Eq. (2.35) yields

$$\begin{aligned} d_{i,j}^{\text{asy}} &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} y_{k,i}^2 + \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} y_{k,j}^2 - \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} 2y_{k,i}y_{k,j} \\ &= P + P - 2\rho P \\ &= 2(1-\rho)P. \end{aligned} \tag{2.40}$$

It is thus evident that, in order to maximize the asymptotic normalized Euclidean distance, we need to make $\rho$ as small as possible. However, there exists a tight lower bound on $\rho$ that induces the covariance matrix $\mathbf{Q}$ to be positive definite.

**Lemma 2.3** *The covariance matrix* $\mathbf{Q}$ *is positive definite if and only if $\rho$ is greater than $\frac{-1}{M-1}$.*

*Proof:*

$$\begin{aligned} \det(\lambda\mathbf{I} - \mathbf{Q}) &= \det \begin{bmatrix} \lambda - P & -\rho P & \cdots & -\rho P \\ -\rho P & \lambda - P & \cdots & -\rho P \\ \vdots & \vdots & \ddots & \vdots \\ -\rho P & -\rho P & \cdots & \lambda - P \end{bmatrix}_{M\times M} \\ &= [\lambda - (1 + (M-1)\rho)P][\lambda - (1-\rho)P]^{M-1}, \end{aligned}$$

thus the matrix $\mathbf{Q}$ has $M-1$ duplicate eigenvalues of $(1-\rho)P$ and one of $[1+(M-1)\rho]P$. In order to guarantee that $\mathbf{Q}$ is positive definite, we obtain the following conditions

$$\begin{cases} (1-\rho)P & > 0 \\ [1+(M-1)\rho]P & > 0, \end{cases} \tag{2.41}$$

which simplifies to $\rho > -1/(M-1)$. Thus completes the proof. ■

Since $\mathbf{M}$ is a diagonal matrix with elements equal to the eigenvalues of $\mathbf{Q}$, we obtain the following lemma

**Lemma 2.4** *The diagonal matrix* $\mathbf{M} = \mathrm{diag}(P_1, P_2, \cdots, P_M)$ *should be*

$$\mathrm{diag}\{\underbrace{(1-\rho)P, \cdots, (1-\rho)P}_{M-1}, [1+(M-1)\rho]P\},$$

*where $P$ is the average transmit power. As $\rho$ tends to $-1/(M-1)$, we have*

$$\lim_{\rho \to \frac{-1}{M-1}} \mathbf{M} = \mathrm{diag}\{\underbrace{\frac{MP}{M-1}, \cdots, \frac{MP}{M-1}}_{M-1}, 0\}.$$

Substituting the minimum (in the limiting case) value of $\rho$ into Eq. (2.40), the asymptotic normalized Euclidean distance becomes

$$d_{i,j}^{\mathrm{asy}} = \frac{2MP}{M-1} = \frac{2^{nR} \cdot 2P}{2^{nR}-1} \tag{2.42}$$

for all $i \neq j, \in 1, 2, \ldots, M$. It should be pointed out that, with increasing $n$, the improvement due to the introduction of correlated codewords becomes negligible while the additional complexity of linear transformation becomes prohibitive. However, it is important to note that the probabilistic code with correlated codewords of $\rho = -\frac{1}{M-1}$ is asymptotically the best code in terms of the normalized Euclidean distance. We are now in a position to prove this argument.

**Theorem 2.2** *The probabilistic code with correlated codewords of $\rho = -\frac{1}{M-1}$ achieves asymptotically the largest possible pairwise normalized Euclidean distance, leading to an asymptotic sphere-packing code.*

*Proof.* First we derive an upper bound on the *average* asymptotic normalized Euclidean distance over all distinct codewords $\{x_{k,j}\}$.

$$\begin{aligned}
\bar{D} &= \frac{1}{M(M-1)} \sum_{i \neq j} \lim_{n\to\infty} \left[ \frac{1}{n} \sum_{k=1}^{n} (x_{k,i} - x_{k,j})^2 \right] \\
&= \frac{1}{M(M-1)} \sum_{i,j} \lim_{n\to\infty} \left[ \frac{1}{n} \sum_{k=1}^{n} (x_{k,i} - x_{k,j})^2 \right] \\
&= \frac{1}{M(M-1)} \sum_{i,j} \lim_{n\to\infty} \left[ \frac{1}{n} \sum_{k=1}^{n} (x_{k,i}^2 - 2x_{k,i}x_{k,j} + x_{k,j}^2) \right] \\
&= \frac{1}{M(M-1)} \lim_{n\to\infty} \frac{1}{n} \left( \sum_{i,j,k} x_{k,i}^2 - 2\sum_{k}\sum_{i,j} x_{k,i}x_{k,j} + \sum_{i,j,k} x_{k,j}^2 \right) \\
&= \frac{1}{M(M-1)} \lim_{n\to\infty} \frac{1}{n} \left( 2M \sum_{i,k} x_{k,i}^2 - 2\sum_{k} \left( \sum_{i} x_{k,i} \right)^2 \right) \\
&\leq \frac{1}{M(M-1)} \lim_{n\to\infty} \left( 2M \sum_{i,k} x_{k,i}^2 \right) \\
&= \frac{2MP}{M-1}.
\end{aligned} \tag{2.43}$$

Referring to Eqs. (2.31) and (2.33), the pairwise error probability of any pair of distinct codewords decreases exponentially with the pairwise (normalized) Euclidean distance. Under the Gaussian noise assumption and the condition of equal use of codewords, the best code is exactly the code that has the maximum-minimum Euclidean distance among codewords. However, maximum-minimum distance among a specific set (each point in the set is a codeword in the $n$-dimensional Euclidean space) must be less than or equal to the average distance, otherwise this would lead to a paradox. It is thus clear that the probabilistic code with correlated codewords of $\rho = -\frac{1}{M-1}$ is asymptotically (as $n$ goes to infinity) the sphere-packing code with a *uniform* normalized Euclidean distance of

$$d_{i,j}^{\text{asy}} = \frac{2MP}{M-1}$$

for all $i \neq j$, which achieves the upper bound of $\bar{D}$. This completes the proof. ■

The fact that the probabilistic code with correlation $\rho = -\frac{1}{M-1} = -\frac{1}{2^{nR}-1}$ asymptotically attains the property of uniformly maximizing the normalized Euclidean distance of signal points under constrained average energy over the AWGN channel indicates that it is precisely the sphere-packing code envisioned by Shannon 50 years ago, but not yet being explicitly constructed so far.

In a special case of very low rate $R = 1/n$, i.e. $M = 2$, the bound in Eq. (2.43) becomes $4P$ and is simply equal to the Euclidean distance of an optimum antipodal signaling and it is also equal to the asymptotic Euclidean distance of the probabilistic code with correlated codewords of $\rho = -1$ as $n$ goes to infinity. In contrast, if one uses a typical random code or the probabilistic code with independent codewords, the asymptotic Euclidean distance is only $2P$, which is obviously 3 dB worse. For any fixed rate $R$ and as $n \to \infty$, thus $M \to \infty$, the bound in Eq. (2.43) approaches $2P$ quickly, which equals the asymptotic Euclidean distance of a probabilistic code with independent codewords. The implication is that, for large $M$ or say $2^{nR}$, only very little performance improvement is gained by the use of correlated codewords. The performance improvement vanishes exponentially as the product of the block length and the rate grows. It reveals to us that, for moderate to large $nR$, the probabilistic code with independent codewords performs almost as well as the best code.

The characterization of the probabilistic code with correlation $\rho = -\frac{1}{M-1}$ as the sphere-packing code in the asymptotic case sheds light on the discrepancy of the random-code bound and the sphere-packing bound. Considering the following limit

$$\lim_{nR \to \infty} \frac{M}{M-1} = \lim_{nR \to \infty} \frac{2^{nR}}{2^{nR}-1} = 1, \tag{2.44}$$

it can be deduced that the discrepancy decreases exponentially with the product of the block length and the rate of the code. Except for the situations with small $n$ and/or $R$, it is safe to say that the random-coding bound and the sphere-packing bound are indistinguishable from each other, thereby the probabilistic code closely approaches the $(\epsilon, n)$-capacity.

## 2.6 Summary

In this chapter, we have identified the maximum achievable rates over noisy, finite-block-length constrained AWGN channels, referred to as $(\epsilon, n)$-capacity $C_\epsilon^n$, with $\epsilon$ denoting the target (block) error probability and $n$ the block length. We have also investigated a family of time-varying block codes based on a probabilistic construction that closely approaches the $(\epsilon, n)$-capacity and provably achieves the Shannon limit over the AWGN channel. The encoding/decoding complexity of probabilistic codes with independent or correlated codewords grows exponentially with the product of block length and the rate, whereas they are the right codes capable of approaching the $(\epsilon, n)$-capacity closely except for small values of $nR$. The theoretical characterization of the $(\epsilon, n)$-capacity and corresponding probabilistic codes shed light on what is the ultimate limit under a critical coding-latency constraint together with a target (block) error probability and how optimum block codes look like.

This work leaves several interesting open questions. To name a few, one is the theoretical characterization of the $(\epsilon, n)$-capacity for extremely short block lengths and small rates where the sphere-packing bound and the random-coding bound typically diverge. Another open problem is quite pragmatic — can one find an encoding/decoding scheme with linear-time complexity or feasible complexity to approach the $(\epsilon, n)$-capacity in the case of relatively short block lengths? This issue will be attacked in the context of error-correcting codes defined on sparse graphs, which is the focus of subsequent chapters.

# Chapter 3

# Regular and Irregular Progressive Edge-Growth Tanner Graphs

## 3.1 Introduction

Error-correcting codes defined on sparse graphs [5, 43, 46–50, 65–70][1] have attracted considerable attention owing to their capacity-approaching performance and low-complexity iterative decoding. The prime examples of such codes are the low-density parity-check (LDPC) codes. It is known that the belief-propagation (BP) or sum-product algorithm (SPA) over cycle-free Tanner graphs [5] provides optimum decoding, hence it is natural to try to minimize the influence of the cycles in the iterative decoding process. This approach has been adopted for both LDPC codes [4] and turbo codes [6] by using rather long block lengths. If the cycles are made long enough, the decoding algorithm may run several iterations without being affected by them while the error rate decreases exponentially with the number of independent iterations. Using the incidence matrix associated with a graph, Gallager proposed an explicit construction [4, pp. 81-89] that guarantees independent decoding iterations up to a lower bound for his LDPC codes. Unfortunately this construction is valid only for regular LDPC codes, and appears to be computationally infeasible for relatively large block lengths.

For most existing classes of LDPC codes, the Tanner graph is randomly constructed by avoiding cycles of length 4 [9, 10, 52, 71]. To date, randomly constructed LDPC codes have largely relied on the sparsity of the parity-check matrix to avoid short cycles in the Tanner

[1]See also the special issue on codes and graphs and iterative algorithms of IEEE Transactions on Information Theory, Feb. 2001, edited by B. J. Frey, R. Koetter, G. D. Forney, Jr., F. R. Kschischang, R. J. McEliece, and D. A. Spielman.

graph. Although random graphs have been used to construct LDPC codes with impressive performance [54, 71], there is no guarantee that any given random graph defines a good code having a suitable shortest cycle (girth) that facilitates iterative decoding as well as having a respectable minimum distance that enhances decoding performance in a high signal-to-noise (SNR) regime. In particular, for short block lengths, the probability of choosing an unfavorable random graph is surprisingly high. The minimum distance issue becomes critical if an irregular degree sequence is used.

Construction of LDPC codes based on finite geometries has been reported in [72,73]. Finite geometry LDPC codes have relatively good minimum distances and their Tanner graphs do not contain cycles of length 4. They can be put in either cyclic or quasi-cyclic form so that the encoding can be achieved in linear time by using simple feedback shift registers. With high-rate and very long block-length, these codes perform very well under iterative decoding, only a few tenths of a decibel away from the Shannon limit. For further results, see [74–77].

Since the early work of Gallager, the first significant work in constructing LDPC codes based on *graph-theoretic* algebraic approach was reported in [42]. In [78, 79], explicit group-theoretic constructions of graphs were proposed. These graphs have girth approaching or exceeding the Erdös–Sachs bound [80], which is a non-constructive lower bound on the girth of random graphs and has the same significance as the Gilbert–Varshamov bound does in the context of minimum distance of linear codes. The notion of graph expansion was first introduced as an analysis tool in coding theory by Sipser and Spielman [43]. Three families of explicit expander graphs built on Cayley graphs of the non-Abelian group $PSL_2(\mathbb{F}_q)$ were used to construct generalized LDPC codes coupled with Hamming subcodes as constraints [81]. Recently, the algebraic approach has been pursued even further, with emphasis on constructing LDPC codes having almost the largest girth possible [82–84]. However, having large girth is only one necessary condition to yield good codes; design of good irregular degree sequence turns out to be equally or more important. Furthermore, practical code-design aspects, such as flexible construction with arbitrary code rates and block sizes as well as linear-time encoding, often conflict with algebraic construction methods.

In this chapter [2] we present a simple but efficient method for constructing Tanner graphs having a large girth in a best-effort sense by progressively establishing edges between symbol and check nodes in an edge-by-edge manner, called the Progressive Edge-Growth (PEG) construction. Given the number of symbol nodes $n$, the number of check nodes $m$, and the symbol-node-degree sequence of the graph, an edge-selection procedure is started such that

[2]This work has been submitted to IEEE Trans. Information Theory, co-authored with E. Evangelos and D. Arnold.

the placement of a new edge on the graph has as small an impact on the girth as possible. After a best-effort edge has been determined, the graph with this new edge is updated, and the procedure continues with the placement of a next edge. The PEG construction presented here is a general, non-algebraic method for constructing graphs with large girth that asymptotically guarantees a girth at least as large as an analogy of the Erdös–Sachs bound. Simulation results show that the resulting LDPC codes of progressive PEG Tanner graphs significantly outperform randomly constructed ones at relatively short block lengths.

In general, LDPC codes based on randomly constructed Tanner graphs do not lend themselves to easy characterization in terms of girth distribution, minimum distance, and number of nearest neighbors. In contrast, a Tanner graph constructed using the PEG principle has elegant properties with respect to the girth of the graph and minimum distance of the induced binary LDPC code. In particular, lower bounds on the girth and on the minimum distance are derived in terms of parameters of the underlying PEG Tanner graph.

It was shown in [51–53] that carefully designed irregular LDPC codes can outperform regular ones. Using the density evolution technique, irregular graphs have been designed by optimizing the degree-distribution pairs via linear programming or differential evolution [10, 51]. This approach proved to be very successful for designing random long block-length LDPC codes; but it is not yet clear whether it is applicable to short block-length codes and to the PEG construction. We describe an empirical Monte–Carlo approach using a variant of the "downhill simplex" search algorithm [85] to design irregular PEG graphs for short codes with fewer than a thousand bits, serving as a complementary procedure to the asymptotic density evolution technique. In our approach we focus on irregular PEG graphs whose degree of parity-check nodes is made as uniform as possible, so-called right-concentrated sequences [86,87]. The conventional downhill simplex algorithm [85, 88] is a nonlinear optimization procedure that operates on a multidimensional simplex. A similar optimization approach has been chosen in [89] to construct *random* irregular LDPC codes with two empirical cost functions: one based on empirical decoding trials, the other based on simulation decoding. In contrast, we design irregular LDPC codes based on the PEG construction and a nonlinear cost function that depends on the block-error rate at a certain $E_b/N_0$. It is demonstrated by simulations that our irregular PEG LDPC codes outperform significantly regular ones at short block lengths. Furthermore, it is shown that the irregular degree sequences obtained by the density-evolution approach, if properly chosen, work very well with the PEG construction too.

LDPC codes have been considered as a serious competitor of Turbo codes. One major drawback so far has been their apparent high encoding complexity that grows quadratically with the block length. The use of a cascade of subcodes to accomplish linear-time encoding complexity was proposed in [51,90]. As each subcode is, in general, considerably smaller that the overall code, a performance loss is expected. Another approach, proposed in [91], is to force the parity-check matrix to have an approximate lower or upper triangular form. Similar in spirit is the approach to incorporate a pattern, referred to as the zig-zag pattern, in the parity-check matrix [92,93]. In [94] it is proved that optimized LDPC codes can be encoded in linear time. Specifically, one can choose a randomly-generated LDPC code with an optimized irregular degree sequence pair, and rearrange the parity-check matrix of this code to have an almost upper-triangular structure which allows linear-time encoding. The significance of [94] lies in the existence proof of optimal LDPC codes with almost upper-triangular structure. Note that the rearrangement of the parity-check matrix does not change the LDPC code itself and the complexity of rearrangement is quadratic in general. Thus it is possible and preferable to construct such LDPC codes directly.

We investigate the construction of LDPC codes having (almost) triangular structure, good girth property and an optimum irregular degree sequence. It is shown that the linear-time encoding property can be combined with the PEG construction in a very natural way while maintaining equally good performance. To this end we demonstrate that our short block-length, linear-time-encodable LDPC codes offer comparable performance to the standardized Turbo codes in third-generation high-speed wireless data services with essentially the same encoding/decoding complexity.

Finally, we investigate the regular and irregular LDPC codes by using the same PEG construction but allowing symbol nodes to assume values from a finite field with more than two elements. In particular, Galois fields with $2^b$ elements denoted as GF($2^b$), $b > 1$, have been considered. Such constructions were first proposed in [5], and performance results of randomly-constructed LDPC codes over GF($2^b$) were reported in [89,95]. The fact that one can obtain consistently improved performance with increasing field size while decreasing the average column weight, is demonstrated by simulation results, which extends the observations of [95]. We report a short block-length (1008 bit) rate-1/2 irregular PEG LDPC code over $GF(2^6)$ with a block error rate $< 10^{-4}$ at $E_b/N_0 = 2$ dB, which appears to have the best performance under iterative decoding at this short block length to date.

The remainder of this chapter is organized as follows. Section 3.2 introduces the necessary definitions and notations on graphs. Section 3.3 describes the principle and the detailed algorithm of the PEG construction. In Section 3.4 we deal with graph properties of PEG Tanner graphs; in particular, the lower bounds on the girth and on the minimum distance are derived. Section 3.5 deals with the asymptotic analysis of the PEG Tanner graphs based on a series of relaxations. We focus on the distance (weight) distribution function of the resulting LDPC codes. We prove the asymptotic optimality of the PEG construction by showing the asymptotic optimality of its weakened version. Section 3.6 describes the variant of the "downhill simplex" search algorithm used to design near-optimum symbol-node-degree distributions for irregular PEG graphs. Section 3.7 presents simulation results comparing the performance of regular and irregular LDPC codes defined on PEG Tanner graphs with that of randomly constructed ones. We address the linear-time encoding based on the PEG principle in Section 3.8 and compare the performance of linear-time-encodable PEG LDPC codes with that of Turbo codes in Section 3.9. In Section 3.10 we extend the binary LDPC codes defined on PEG Tanner graphs to codes over a finite field with more than two elements. Finally, 3.11 concludes this chapter.

## 3.2 Definitions and Notations

An LDPC code is a linear code defined by a sparse parity check matrix $H$ having dimension $m \times n$. A bipartite graph with $m$ check nodes in one class and $n$ symbol nodes in the other can be created using $H$ as the integer-valued incidence matrix for the two classes. Such a graph is also called Tanner graph [5]. As a Tanner graph defines a unique parity check matrix and a parity check matrix corresponds to a unique Tanner graph, we shall use the terms Tanner graph and code interchangeably if no confusion arises. Formally, a Tanner graph is denoted as $(V, E)$, with $V$ the set of vertices (nodes), $V = V_c \cup V_s$, where $V_c = \{c_0, c_1, \ldots, c_{m-1}\}$ is the set of check nodes and $V_s = \{s_0, s_1, \ldots, s_{n-1}\}$ the set of symbol nodes. $E$ is the set of edges such that $E = V_c \times V_s$, with edge $(c_i, s_j) \in E$ if and only if $h_{i,j} \neq 0$, $h_{i,j} \in H$, $0 \leq i \leq m-1, 0 \leq j \leq n-1$. A Tanner graph is called $(d_s, d_c)$*-regular* if every symbol node participates in $d_s$ check nodes and every check node involves $d_c$ symbol nodes; otherwise it is called *irregular*. Denote the symbol degree sequence by $D_s = \{d_{s_0}, d_{s_1}, \ldots, d_{s_{n-1}}\}$, in which $d_{s_j}$ is the degree of symbol node $s_j$, $0 \leq j \leq n-1$, in nondecreasing order, i.e., $d_{s_0} \leq d_{s_1} \ldots \leq d_{s_{n-1}}$, and the parity-check degree sequence by $D_c = \{d_{c_0}, d_{c_1}, \ldots, d_{c_{m-1}}\}$, in which $d_{c_j}$ is the degree of parity-check node $c_j$, $0 \leq j \leq m-1$, and $d_{c_0} \leq d_{c_1} \ldots \leq d_{c_{m-1}}$. Let also the set of edges $E$ be partitioned in terms of $V_s$ as $E = E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{n-1}}$, with $E_{s_j}$ containing all edges incident on symbol node $s_j$. Further, denote the $k$-th edge incident on $s_j$ by $E^k_{s_j}$, $0 \leq k \leq d_{s_j} - 1$. Fig. 3.1 shows an example of a $D_s = \{2, 2, 2, 2, 3, 3, 3, 3\}$ irregular Tanner graph, in which the check degree sequence is uniformly of degree 5, i.e., $D_c = \{5, 5, 5, 5\}$. Note that the definition of a Tanner graph is quite general in that the check node may be a simple parity check, a parity check over Galois field, or even a subcode such as BCH code.

A graph is called *simple* if 1) it does not have a self loop that is an edge joining a vertex to itself, 2) there is at most one edge between the same pair of vertices, and 3) all edges are non-directed. In a simple graph, we say that vertices $x$ and $y$ are *adjacent* if $(x, y)$ is an edge. The set consisting of all vertices that are adjacent to $x$ is called $x$'s *neighbor*. A *subgraph* of a graph $G = (V, E)$ is a graph whose vertex and edge set are subsets of those of $G$. Note that, if $G' = (V', E')$ is a subgraph of $G$, then for every edge $e \in E'$, it must hold that both the vertices of $e$ lie in $V'$. A sequence of distinct vertices, starting from $x$ and ending with $y$, is called a *path* between $x$ and $y$, if any two consecutive vertices in the sequence are adjacent. A closed path with edges starting from $x$ and ending with itself is called a *cycle* of $x$. If there exists at least one path between $x$ and $y$, then $x$ and $y$ is called *connected* or $x$ is *reached* by $y$ and vice versa. If two vertices $x$ and $y$ in the graph are connected, their distance $d(x, y)$ is then defined as the length (number of edges) of the shortest path joining them. A subgraph is *tree-like* if there is no cycle inside.

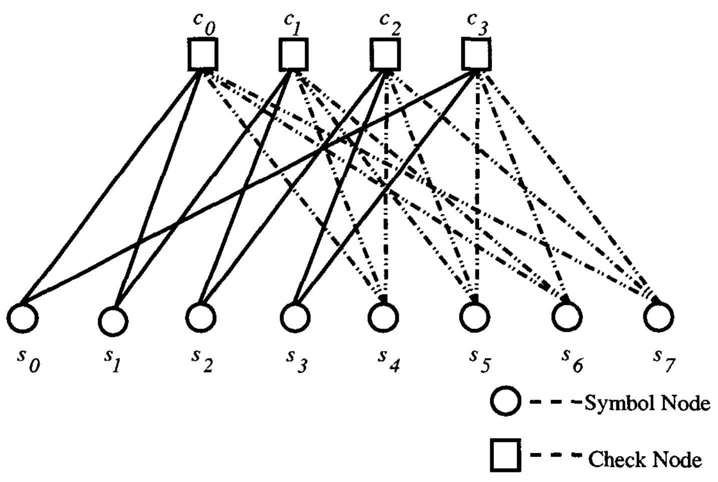


**Figure 3.1:** An example of symbol node degree $D_s = \{2, 2, 2, 2, 3, 3, 3, 3\}$ irregular Tanner graph.

In general, an ensemble of bipartite or Tanner graphs is characterized by degree distribution pairs. In the case of symbol node, the degree distribution is defined as $\Lambda(x) = \sum_{i \geq 2}^{d_s^{\max}} \Lambda_i x^i$, where $\Lambda_i$ is the fraction of symbol nodes connected to exactly $i$ check nodes; $d_s^{\max}$ is the largest entry in $D_s = \{d_{s_0}, d_{s_1}, \ldots, d_{s_{n-1}}\}$, and $\sum_{i \geq 2}^{d_s^{\max}} \Lambda_i = 1$. Similarly, in the case of parity-check node, the degree distribution is defined as $\Phi(x) = \sum_{i \geq 2}^{d_c^{\max}} \Phi_i x^i$, where $\Phi_i$ is the fraction of parity-check nodes connected to exactly $i$ symbol nodes; $d_c^{\max}$ is the largest entry in $D_c = \{d_{c_0}, d_{c_1}, \ldots, d_{c_{m-1}}\}$, and $\sum_{i \geq 2}^{d_c^{\max}} \Phi_i = 1$.

For a given symbol node $s_j$, define its *neighbor within depth* $l$, $\mathcal{N}_{s_j}^l$, as the set consisting of all check nodes reached by a subgraph (or a tree) spreading from symbol node $s_j$ within depth

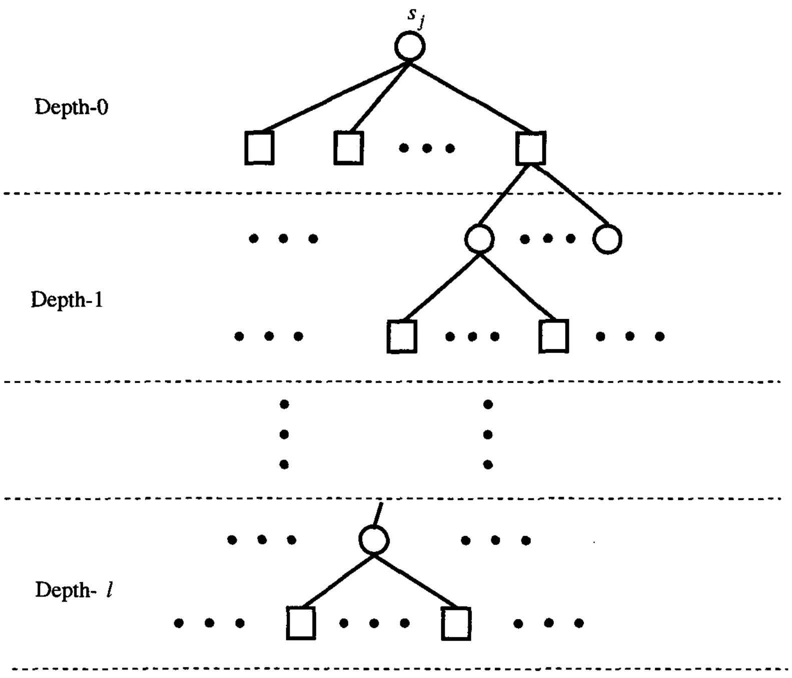


**Figure 3.2:** A breadth-first-search (BFS) subgraph spreading from symbol node $s_j$.

$l$, as shown in the example in Fig. 3.2. Its complementary set, $\bar{\mathcal{N}}^l_{s_j}$, is defined as $V_c \backslash \mathcal{N}^l_{s_j}$, or equivalently $\mathcal{N}^l_{s_j} \cup \bar{\mathcal{N}}^l_{s_j} = V_c$. The subgraph rooted from $s_j$ is generated by means of unfolding the Tanner graph in a breadth-first way. That is to say, we start from $s_j$, and traverse all edges incident on $s_j$; let these edges be $(s_j, c_{i_1}), (s_j, c_{i_2}), \ldots, (s_j, c_{i_{d_{s_j}}})$. Then explore all other edges incident on vertices $c_{i_1}, c_{i_2}, \ldots, c_{i_{d_{s_j}}}$ excluding $(s_j, c_{i_1}), (s_j, c_{i_2}), \ldots, (s_j, c_{i_{d_{s_j}}})$. This process continues until the depth can not increase further. Note that in the subgraph, duplicate vertices or edges may occur. Referring to Fig. 3.2, any symbol node residing at depth-$l$ has a distance to $s_j$ of $2l$, and any check node residing at depth-$l$ has a distance to $s_j$ of $2l+1$. Therefore $\mathcal{N}^l_{s_j}$ can be alternatively defined as the check node subset of distance (relative to $s_j$) smaller than or equal to $2l+1$.

Similarly, for a given parity-check node $c_i$, define its neighbor with depth $l$, $\mathcal{N}^l_{c_i}$, as the set consisting of all parity-check nodes reached by a subgraph (or a tree) spreading from $c_i$ within depth-$l$, as shown in Fig. 3.3.

The set $\mathcal{N}^l_{s_j}$ and its counterpart $\bar{\mathcal{N}}^l_{s_j}$ can be efficiently evaluated in a recursive manner. Initially mark the symbol node $s_j$ with integer 0 and set $d = 0$. While any vertex, either symbol node or parity-check node, was marked at the preceding stage, 1) look at all vertices marked $d$ and mark all unmarked neighbors of such vertices with $d+1$; 2) replace $d$ by $d+1$. At the termination of this algorithm, the mark of marked vertices is their distance to $s_j$, and the unmarked vertices, if exist, do not have a path connecting to $s_j$. Obviously $\mathcal{N}^l_{s_j}$ can be

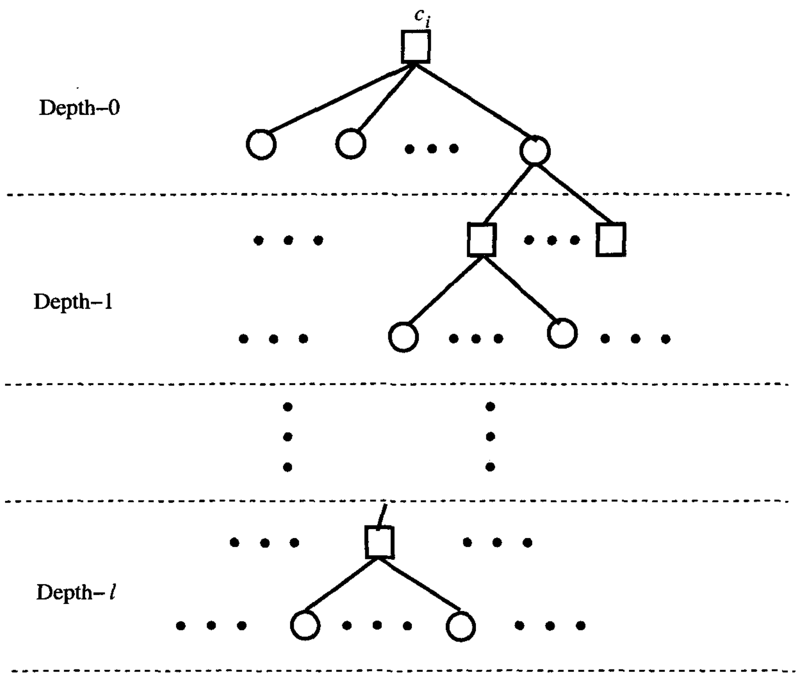


**Figure 3.3:** A BFS subgraph spreading from parity-check node $c_i$.

obtained by simply examining the check nodes having a mark less than or equal to $2l+1$.

In graph theory, girth $g$ refers to the length of the shortest cycle in a graph. For each symbol node $s_j$, we define a local girth $g_{s_j}$ as the length of the shortest cycle passing through that symbol node. The set of local girth $\{g_{s_j}\}$ is referred to as girth histogram. It follows, by definition, that $g = \min_j \{g_{s_j}\}$.

## 3.3 Progressive Edge-Growth (PEG) Algorithm

Constructing a Tanner graph with the largest possible girth is a rather difficult combinatorial problem. Nevertheless, a sub-optimum algorithm to construct a Tanner graph with a relatively large girth is feasible in practice. One such algorithm is the PEG algorithm that we present here, in which the local girth of a symbol node is maximized whenever a new edge is added to this symbol node. Suppose we have finished constructing edges of the first $j$ symbol nodes on a Tanner graph, i.e., edges $E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{j-1}}$ have been established. Let $g^t$ be the temporary girth under the current graph setting with edges $E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{j-1}}$. In other words, $g^t = \min\{g_{s_0}, g_{s_1}, \cdots, g_{s_{j-1}}\}$. The problem of constructing a graph having a large girth lies in how to select the edge set $E_{s_j}$ of symbol node $s_j$ such that adding these new edges to the current graph setting does not impair the current $g^t$ excessively. This boils down to optimizing $d_{s_j}$ edges of $E_{s_j}$ to maximize the local girth $g_{s_j}$ because, if adding $E_{s_j}$ to the current graph results in a cycle shorter than $g^t$, this new short cycle must pass through symbol node $s_j$. Unfortunately the complexity of exhaustively searching for the optimum set $E_{s_j}$ is governed by $\binom{m}{d_{s_j}}$, where $\binom{m}{d_{s_j}}$ is a binomial coefficient. Thus, we propose a best-effort algorithm known as progressive edge-growth, in which $d_{s_j}$ edges of $E_{s_j}$ are added to the current graph *on an edge-by-edge basis*, and the length of the shortest cycle passing through symbol node $s_j$ is maximized whenever a new edge originating in $s_j$ is being added. This can be accomplished simply by first expanding the tree originating in symbol node $s_j$ up to depth $l$ each time a new edge of $s_j$ is being determined, such that $\bar{\mathcal{N}}_{s_j}^l \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} = \varnothing$ or the cardinality of $\mathcal{N}_{s_j}^l$ stops increasing but is smaller than $m$, and then placing an edge between $s_j$ and a check node selected from the set $\bar{\mathcal{N}}_{s_j}^l$. In this way, the shortest cycle passing through this new edge under the current graph setting is guaranteed to be no shorter than $2(l+2)$. We summarize the proposed algorithm as follows.

**Progressive Edge-Growth Algorithm**:

```
for $j = 0$ to $n-1$ do
begin
  for $k = 0$ to $d_{s_j} - 1$ do
  begin
    if $k = 0$
      $E_{s_j}^0 \longleftarrow$ edge $(c_i, s_j)$, where $E_{s_j}^0$ is the first edge incident to $s_j$ and $c_i$ is
      one check node such that it has the lowest check degree under the current
      graph setting $E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{j-1}}$.
    else
      expanding a tree from symbol node $s_j$ up to depth $l$ under the current
      graph setting such that $\bar{\mathcal{N}}_{s_j}^l \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} = \varnothing$, or the cardinality of $\mathcal{N}_{s_j}^l$
      stops increasing but is less than $m$, then $E_{s_j}^k \longleftarrow$ edge $(c_i, s_j)$, where $E_{s_j}^k$
      is the $k$-th edge incident to $s_j$ and $c_i$ is one check node picked from the
      set $\bar{\mathcal{N}}_{s_j}^l$ having the lowest check-node degree.
  end
end
```

There is a subtle point in the PEG algorithm that needs further comment. Whenever we encounter multiple choices for connecting to symbol node $s_j$, i.e., multiple check nodes exist in $\bar{\mathcal{N}}_{s_j}^l$, we select the one having the smallest number of incidence edges under the current graph setting. Such a selective strategy renders the resulting PEG Tanner graphs as check-node-degree uniform as possible and it tends to favor graphs with a non-zero degree in parity-check nodes (parity-check-node regular graphs), or concentrated graphs with two nonzero consecutive degrees otherwise. There is strong evidence indicating that such a degree sequence is optimum in some weak sense [86, 87].

Even so, we may still face a situation in which multiple choices exist because multiple check nodes in $\bar{\mathcal{N}}_{s_j}^l$ might have the same lowest degree, particularly during the start-up period. There are two main approaches to solve this problem. The first is to randomly select one of these check nodes. The other is to always select one according to its position in the order of $c_0, c_1, \ldots, c_{m-1}$. For instance, we can first sort the check nodes in $\bar{\mathcal{N}}_{s_j}^l$ that have the same lowest degree according to their subscripts, and then always pick the first one. Here we adopt the first approach; however, note that the second may also be of interest because of its deterministic nature.

We close this section by pointing out the following remarks.

1. *Complexity* — The computational load in evaluating the set $\bar{\mathcal{N}}_{s_j}^l$ depends primarily on the degree sequences $D_s$ and $D_c$ as well as the depth $l$. In a sparse graph the elements of $D_s$ and $D_c$ are small numbers compared with the block length $n$, and $l$ grows at most logarithmically with the block length $n$ (we will see this point later). This compares much favorably with other random constructions that aim to achieve large girth whose complexity generally grows exponentially with $n$.

2. *Non-greedy version* — The version presented above is greedy in the sense that the spreading subgraph of $s_j$ proceeds as deep as possible, i.e., the depth $l$ is maximized such that $\bar{\mathcal{N}}_{s_j}^l \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} = \varnothing$. This approach proves to be favorable if the minimum distance is at a premium, particularly for short block-length and/or high-rate codes [96–98]. However, for long block-length, low-rate codes in which the minimum distance is not a critical issue, it might be favorable to limit $l$ to a certain value $l_{\max}$, 1) to make the check-node degree sequence concentrated in the strict sense, 2) possibly to reduce the *diameter* of the graph — the maximum distance of distinct vertex pairs, such that fewer decoding iterations are required. This kind of variant is called *non-greedy* PEG algorithm. Note that if one sets $l_{\max} = g_t/2 - 2$, where $g_t$ is the target girth, then this variant bears some resemblance to the "bit-filling" algorithm appeared in the independent work [99].

3. *Look-ahead-enhanced version* — The PEG principle refers to constructing edges in stages, where at each stage we make the choice for an edge emanating from a symbol node which locally optimizes the shortest cycle passing through the assumed edge, by which we move closer to our final goal. Obviously, this short-sighted local optimization does not usually produce the best overall solution. One can enhance the greedy PEG algorithm by looking one-step ahead. In the look-ahead-enhanced version, the same procedure as

in the greedy PEG algorithm is applied, except when multiple choices exist for placing the $k$ edge of $s_j$. In this case, rather than randomly select one, an additional test is introduced to distinguish multiple choices. That is, for each candidate in $\bar{\mathcal{N}}_{s_j}^l$, we evaluate the maximum depth $l$ of $s_j$ *assuming this candidate edge being put onto the graph*, and associate the maximum depth to each candidate, and finally select the one having the largest value as the parity-check node that the $k$ edge of $s_j$ joins.

4. *Flexibility and Scalability* — The idea underlying the PEG algorithm is quite flexible and scalable. The PEG algorithm can be used to construct good regular or irregular bipartite graphs. Later we will also show how to extend the PEG algorithm to construct linear-time encodable LDPC codes. The PEG algorithm is not necessarily limited to constructing bipartite graphs; it can be extended — although not treated here, in a straightforward manner to constructing arbitrary graphs with large girth by applying the PEG principle enunciated above.

## 3.4 Graph Properties

A randomly-constructed Tanner graph does not guarantee a meaningful lower bound on the girth and the minimum distance. In contrast, a Tanner graph constructed with the PEG algorithm has elegant properties with respect to the girth and the minimum distance.

### 3.4.1 Girth Bound

We need the following Lemmas to establish a lower bound on the girth of PEG Tanner graphs.

**Lemma 3.1** *Let $(V, E)$ be a regular Tanner graph with $d_c$ and $d_s$ edges incident with each parity-check and symbol node, respectively. If $\mathcal{N}_{s_j}^l$ denotes the depth-$l$ neighbor of a symbol node $s_j$ such that $\mathcal{N}_{s_j}^l \subset V_c$ and $\mathcal{N}_{s_j}^{l+1} = V_c$, then $l$ is lower bounded by*

$$l \geq \lfloor t \rfloor \,, \tag{3.1}$$

*where $t$ satisfies*

$$t = \frac{\log\left(md_c - \frac{md_c}{d_s} - m + 1\right)}{\log[(d_s-1)(d_c-1)]} - 1, \tag{3.2}$$

$\lfloor \cdot \rfloor$ *indicates the floor of a real number, and $m$ denotes the cardinality of the set $V_c$ of parity-check nodes.*

*Proof:* Consider a depth-$l$ subgraph spreading from any symbol node $s_j$, $s_j \in V_s$, such that $\mathcal{N}_{s_j}^l \subset V_c$ and $\mathcal{N}_{s_j}^{l+1} = V_c$. By definition the depth-0 subgraph contains $d_s$ parity-check nodes, each giving rise to $(d_s-1)(d_c-1)$ parity-check nodes in the next round of spreading. Thus, there are $d_s(d_s-1)(d_c-1)$ check nodes at depth-1. Similarly, there are $d_s(d_s-1)^l(d_c-1)^l$ check nodes at depth $l$. In principle, duplicate parity-check nodes may occur in the subgraph during the spreading process. Let $l'$ be the largest integer such that

$$d_s + d_s(d_s-1)(d_c-1) + \cdots + d_s(d_s-1)^{l'}(d_c-1)^{l'} \leq m, \tag{3.3}$$

which can be simplified to

$$\frac{d_s\left[(d_s-1)^{l'+1}(d_c-1)^{l'+1}-1\right]}{(d_s-1)(d_c-1)-1} \leq m. \tag{3.4}$$

Let $t$ be the floating-point solution of the equation

$$\frac{d_s\left[(d_s-1)^{t+1}(d_c-1)^{t+1}-1\right]}{(d_s-1)(d_c-1)-1} = m. \tag{3.5}$$

Then $l \geq l' \geq \lfloor t \rfloor$, After simple manipulations Eq. (3.5) yields Eq. (3.2), which completes the proof. ■

**Lemma 3.2** *Let $(V, E)$ be an irregular Tanner graph in which $d_c^{\max}$ and $d_s^{\max}$ are the largest degrees of the degree sequences $D_c$ and $D_s$, respectively. If $\mathcal{N}_{s_j}^l$ denotes the depth-$l$ neighbor of a symbol node $s_j$ such that $\mathcal{N}_{s_j}^l \subset V_c$ and $\mathcal{N}_{s_j}^{l+1} = V_c$, then $l$ is lower bounded by*

$$l \geq \lfloor t \rfloor, \tag{3.6}$$

*where $t$ satisfies*

$$t = \frac{\log\left(md_c^{\max} - \frac{md_c^{\max}}{d_s^{max}} - m + 1\right)}{\log[(d_s^{\max}-1)(d_c^{\max}-1)]} - 1, \tag{3.7}$$

*and $m$ denotes the cardinality of the set $V_c$ of parity-check nodes.*

*Proof:* The proof is identical to that of Lemma 3.1, except that $d_s$ and $d_c$ are replaced by $d_s^{\max}$ and $d_c^{\max}$, respectively. ■

We now establish a lower bound on the girth of a PEG Tanner graph.

**Theorem 3.1** *Let $(V, E)$ be an irregular PEG Tanner graph in which $d_c^{\max}$ and $d_s^{\max}$ are the largest degrees of the degree sequences $D_c$ and $D_s$, respectively. The girth $g$ of this graph is lower bounded by*

$$g \geq 2(\lfloor t \rfloor + 2), \tag{3.8}$$

*where $t$ satisfies*

$$t = \frac{\log\left(md_c^{\max} - \frac{md_c^{\max}}{d_s^{max}} - m + 1\right)}{\log[(d_s^{\max} - 1)(d_c^{\max} - 1)]} - 1, \tag{3.9}$$

*and $m$ denotes the cardinality of the set $V_c$ of parity-check nodes. If $(V, E)$ is a regular PEG Tanner graph with $d_c$ and $d_s$ edges incident with each parity-check and symbol node, $d_c^{\max}$, $d_s^{max}$ in Eq.* (3.9) *are then replaced by $d_c, d_s$ respectively.*

*Proof*: Suppose that the closed path $(s_{j_0}, c_{i_0})$, $(c_{i_0}, s_{j_1})$, $(s_{j_1}, c_{i_1})$, $(c_{i_1}, s_{j_2})$, $\ldots$, $(s_{j_{g/2-1}}, c_{i_{g/2-1}})$, $(c_{i_{g/2-1}}, s_{j_0})$ is among the ones that provide the shortest cycle in a Tanner graph $(V, E)$ constructed with the PEG algorithm where, without loss of generality, $j_{g/2-1}$ is the largest index among $j_0, j_1, \cdots, j_{g/2-1}$. Then the length of the shortest cycle in the graph, i.e., girth $g$, is equal to the local girth of symbol node $s_{j_{g/2-1}}$, i.e., $g = g_{j_{g/2-1}}$. As $j_{g/2-1}$ is the largest index, $g_{j_{g/2-1}}$ can be viewed as the girth of the symbol node $s_{j_{g/2-1}}$ in the intermediary graph with edges in the set $E_0 \cup E_1 \cup \cdots \cup E_{j_{g/2-1}}$. Clearly, the edges in the complementary set $E_{j_{g/2}} \cup \cdots \cup E_{n-1}$ have no impact on the local girth of $s_{j_{g/2-1}}$. Recall now the successive procedure in the PEG algorithm for placing edges in the set $E_{j_{g/2-1}}$. By construction the shortest possible cycle passing through symbol node $s_{j_{g/2-1}}$ has length $2(l + 2)$ where $l$ corresponds to the depth-$l$ neighbor $\mathcal{N}^l_{s_{j_{g/2-1}}}$ such that $\bar{\mathcal{N}}^l_{s_{j_{g/2-1}}} \neq \varnothing$, but $\bar{\mathcal{N}}^{l+1}_{s_{j_{g/2-1}}} = \varnothing$. It can readily be seen that in our case $l + 1 = g/2 - 1$. Therefore, by making use of Lemma 3.2 we obtain

$$g/2 - 2 \geq \lfloor t \rfloor \tag{3.10}$$

which implies $g \geq 2(\lfloor t \rfloor + 2)$, where $t$ is subject to 3.9. ■

The bound on the girth provides a justification of the effort of the PEG algorithm to keep the check-node degree as uniform as possible. The more uniform the Tanner graph, the smaller the values of $d_s^{\max}$ and $d_c^{\max}$, thereby improving the lower bound.

An upper bound on the girth of a general Tanner graph can readily be derived based on the idea in [4, pp. 81–82].

**Lemma 3.3** *Let $(V, E)$ be a regular Tanner graph with $d_c$ and $d_s$ edges incident with each parity-check and symbol node, respectively. The girth $g$ of this graph is upper bounded by*

$$g \leq 4\lfloor t \rfloor + 4, \tag{3.11}$$

*where $t$ is given by*

$$t = \frac{\log\left[(m-1)(1 - \frac{d_s}{d_c(d_s-1)}) + 1\right]}{\log[(d_c - 1)(d_s - 1)]}, \tag{3.12}$$

*and $m$ denotes the cardinality of the set $V_c$ of parity-check nodes.*

*Proof*: Consider a depth-$l$ subgraph spreading from any parity-check node $c_i$, $c_i \in V_c$, such that $\mathcal{N}_{c_i}^l \subseteq V_c$ and $\mathcal{N}_{c_i}^{l+1} = V_c$. Furthermore, let's assume that there are no duplicate parity-check nodes within the subgraph of depth-$l$ but there are duplicate parity-check nodes within depth-$(l+1)$. Under these conditions the girth $g$ must be smaller than or equal to $4l+4$. The value $l$ can be upper bounded based on the fact that the total number of parity-check nodes within the subgraph of depth-$l$ must be smaller than or equal to the cardinality of $V_c$, namely $m$ . Observe that there is only one parity-check node at depth-0, and there are $d_c(d_s-1)$ parity-check nodes at depth-1, $d_c(d_c-1)(d_s-1)^2$ at depth-2, and so forth. Thus, we have

$$1 + d_c(d_s-1) + d_c(d_c-1)(d_s-1)^2 + \cdots + d_c(d_c-1)^{l-1}(d_s-1)^l \leq m, \tag{3.13}$$

which reduces to

$$\frac{d_c(d_s-1)[(d_c-1)^l(d_s-1)^l-1]}{(d_c-1)(d_s-1)-1} + 1 \leq m. \tag{3.14}$$

If $t$ is the solution of the equation

$$\frac{d_c(d_s-1)[(d_c-1)^t(d_s-1)^t-1]}{(d_c-1)(d_s-1)-1} + 1 = m. \tag{3.15}$$

then $t \geq \lfloor t \rfloor \geq l$. Therefore

$$g \leq 4l+4 \leq 4\lfloor t \rfloor + 4 \tag{3.16}$$

where $t$ is defined by Eq. (3.12). ■

**Lemma 3.4** *Let $(V,E)$ be a regular PEG Tanner graph with $d_c$ and $d_s$ edges incident with each parity-check and symbol node, respectively. The girth $g$ of this graph is always larger than or equal to the half of the upper bound.*

*Proof*: To prove the argument, it suffices to show that the lower bound on the girth of a PEG Tanner graph is larger than or equal to half of the upper bound of a general Tanner graph. The ratio of the lower bound to the upper bound is given by

$$\begin{aligned}\frac{2\left(\left\lfloor \frac{\log\left(md_c - \frac{md_c}{d_s} - m + 1\right)}{\log[(d_s-1)(d_c-1)]} - 1 \right\rfloor + 2\right)}{4\left\lfloor \frac{\log\left[(m-1)(1-\frac{d_s}{d_c(d_s-1)})+1\right]}{\log[(d_c-1)(d_s-1)]} \right\rfloor + 4} &= \frac{2\left(\left\lfloor \frac{\log\left(\frac{md_cd_s - md_c - md_s}{d_s} + 1\right)}{\log[(d_s-1)(d_c-1)]} \right\rfloor + 1\right)}{4\left(\left\lfloor \frac{\log\left[\frac{md_cd_s - md_c - md_s}{d_c(d_s-1)} + 1 - (1-\frac{d_s}{d_c(d_s-1)})\right]}{\log[(d_c-1)(d_s-1)]} \right\rfloor + 1\right)} \\ &\geq \frac{1}{2},\end{aligned}$$

in which inequality holds because in a nontrivial parity-check code we have $d_c > d_s$ and $d_s > 1$.
■

The asymptotic Erdös–Sachs bound [80] states that a randomly generated regular graph with $n$ vertices and of degree $k$ has girth larger than or equal to $\log_{k-1} n$ with probability approaching 1 as $n \to \infty$. This lower bound is exactly half of an upper bound (asymptotically) that can be obtained in the same manner as we did in the proof of Lemma 3.3. Therefore, Lemma 3.4 implies that the girth of a PEG Tanner graph surpasses or meets an analogy of the Erdös–Sachs bound.

We derive the subsequent two lemmas to tighten the upper bound on the girth of a general Tanner graph.

**Lemma 3.5** *Let $(V, E)$ be a regular Tanner graph with $d_c$ and $d_s$ edges incident with each parity-check and symbol node, respectively. The girth $g$ of the graph is upper bounded by*

$$g \leq 4\lfloor t \rfloor + 4, \tag{3.17}$$

*where $t$ is given by*

$$t = \frac{\log\left[(m-1)(1-\frac{d_s}{d_c(d_s-1)})+1\right]}{\log[(d_c-1)(d_s-1)]}, \tag{3.18}$$

*and $m$ denotes the cardinality of the set $V_c$ of parity-check nodes. Furthermore, if*

$$[(d_c-1)(d_s-1)]^{\lfloor t \rfloor} > m - 1 - \frac{d_c(d_s-1)\{[(d_c-1)(d_s-1)]^{\lfloor t \rfloor}-1\}}{(d_c-1)(d_s-1)-1}, \tag{3.19}$$

*then*

$$g \leq 4\lfloor t \rfloor + 2. \tag{3.20}$$

*Proof*: Consider a depth-$l$ subgraph spreading from any parity-check node $c_i$, $c_i \in V_c$, such that $\mathcal{N}_{c_i}^{l} \subseteq V_c$ and $\mathcal{N}_{c_i}^{l+1} = V_c$. Furthermore, let's assume that there are no duplicate parity-check nodes within the subgraph of depth-$l$ but there are duplicate parity-check nodes within depth-$(l+1)$. Under these conditions the girth $g$ must be smaller than or equal to $4l+4$. The value $l$ can be upper bounded $\lfloor t \rfloor$ where $t$ is determined by Eq. (3.18).

Now let's check the total number of symbol nodes residing at depth-$\lfloor t \rfloor$. On the one hand, there are $d_c[(d_c-1)(d_s-1)]^{\lfloor t \rfloor}$ symbol nodes at depth-$\lfloor t \rfloor$, because there are $d_c(d_c-1)^{\lfloor t \rfloor-1}(d_s-1)^{\lfloor t \rfloor}$ distinct parity-check nodes residing at depth-$\lfloor t \rfloor$. On the other hand, since only $m-(1+d_c(d_s-1)+d_c(d_c-1)(d_s-1)^2+\cdots+d_c(d_c-1)^{\lfloor t \rfloor-1}(d_s-1)^{\lfloor t \rfloor})$ distinct check nodes are not present in the depth-$\lfloor t \rfloor$ subgraph and each may lead to $d_c$ symbol nodes, there are at most $md_c - d_c(1+d_c(d_s-1)+d_c(d_c-1)(d_s-1)^2+\cdots+d_c(d_c-1)^{\lfloor t \rfloor-1}(d_s-1)^{\lfloor t \rfloor})$ distinct symbol nodes residing at depth-$\lfloor t \rfloor$. Therefore, if

$$\begin{aligned} d_c[(d_c-1)(d_s-1)]^{\lfloor t \rfloor} > md_c - d_c(1 &+ d_c(d_s-1) + d_c(d_c-1)(d_s-1)^2 \\ &+ \cdots + d_c(d_c-1)^{\lfloor t \rfloor-1}(d_s-1)^{\lfloor t \rfloor}), \end{aligned} \tag{3.21}$$

then there must be at least one duplicate symbol residing at depth-$\lfloor t \rfloor$, which implies

$$g \leq 4\lfloor t \rfloor + 2. \tag{3.22}$$

After simple algebra manipulations, Eq. (3.21) yields Eq. (3.19), which completes the proof. ■

**Lemma 3.6** *Let $(V, E)$ be a regular Tanner graph with $d_c$ and $d_s$ edges incident with each parity-check and symbol node, respectively, $m$ and $n$ be the cardinality of the set $V_c$ and $V_s$, respectively. The girth $g$ of the graph is upper bounded by*

$$g \leq \min\{g_1, g_2\}, \tag{3.23}$$

*where*

$$g_1 = \begin{cases} 4\lfloor t_1 \rfloor + 2 & \text{if } \mathcal{I}_1 = 0 \\ 4\lfloor t_1 \rfloor + 4 & \text{otherwise} \end{cases} \qquad g_2 = \begin{cases} 4\lfloor t_2 \rfloor + 2 & \text{if } \mathcal{I}_2 = 0 \\ 4\lfloor t_2 \rfloor + 4 & \text{otherwise} \end{cases} \tag{3.24}$$

*in which*

$$t_1 = \frac{\log\left[(m-1)(1 - \frac{d_s}{d_c(d_s-1)}) + 1\right]}{\log[(d_c-1)(d_s-1)]}, \tag{3.25}$$

$$t_2 = \frac{\log\left[(n-1)(1 - \frac{d_c}{d_s(d_c-1)}) + 1\right]}{\log[(d_c-1)(d_s-1)]}, \tag{3.26}$$

*and $\mathcal{I}_1$ is equal to 0 if and only if*

$$[(d_c-1)(d_s-1)]^{\lfloor t_1 \rfloor} > m - 1 - \frac{d_c(d_s-1)\{[(d_c-1)(d_s-1)]^{\lfloor t_1 \rfloor} - 1\}}{(d_c-1)(d_s-1) - 1}, \tag{3.27}$$

*and $\mathcal{I}_2$ is equal to 0 if and only if*

$$[(d_c-1)(d_s-1)]^{\lfloor t_2 \rfloor} > n - 1 - \frac{d_s(d_c-1)\{[(d_c-1)(d_s-1)]^{\lfloor t_2 \rfloor} - 1\}}{(d_c-1)(d_s-1) - 1}. \tag{3.28}$$

*Proof*: The new results in this lemma compared to Lemma 3.5 can be obtained by a duality of the proof of Lemma 3.5. The duality lies in the fact that one can exchange $V_c$ and $V_s$ and the same arguments apply. ■

Fig. 3.4 depicts both the lower bound on a PEG Tanner graph and the upper bound on a general Tanner graph for regular $d_s = 3, d_c = 6$ codes with varying $m$ (in this case $n = 2m$). It is observed that the lower bound for the entire range of block lengths is above half the upper bound. Compared to Gallager's construction [4, pp. 81-89] for large girth, the PEG construction achieves essentially the same performance on the girth but with less

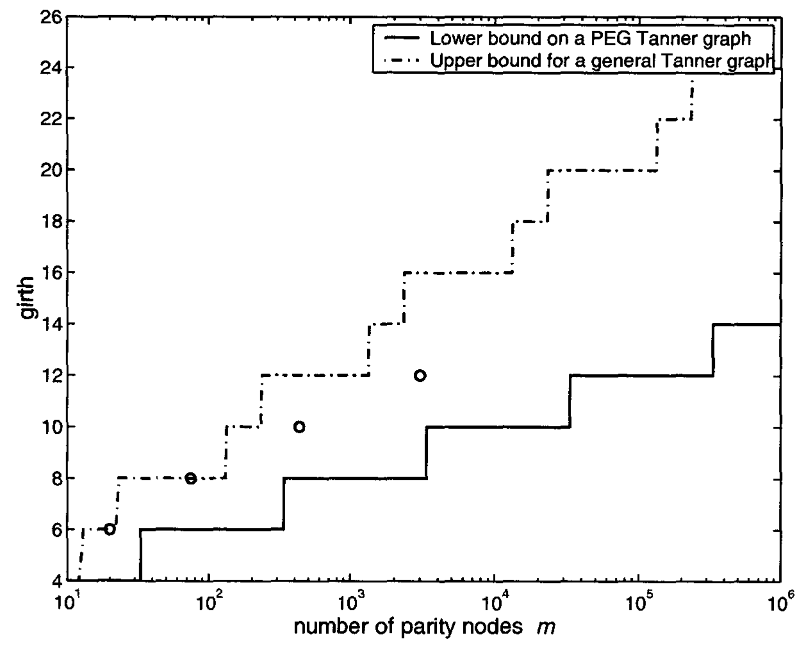


**Figure 3.4:** Lower and upper bounds on a PEG regular Tanner graph with $d_s = 3, d_c = 6$.

complexity. Recall that for large block lengths the complexity of Gallager's construction becomes prohibitively large owing to the so-called emergency procedure contained therein. Also, the PEG construction can be applied to irregular graphs, whereas Gallager's construction only applies to regular graphs. Later it will be shown that the PEG algorithm can be extended to design linear-time-encodable LDPC codes in a very natural way. Fig. 3.5 compares the upper and lower bounds for regular $d_s = 4, d_c = 8$ codes. It is worthwhile to point out that the lower bound on the PEG graphs can be exceeded by the PEG algorithm. We observe, through experiments, that with some parameters $d_s, d_c$ and $m$, the non-greedy PEG variant or look-ahead-enhanced variant can achieve larger girth than the lower bound, some cases of which were shown in Fig. 3.4 as small circles.

## 3.4.2 Minimum Distance Bound

Assume $V_s$ takes on values from the binary alphabet $\{0, 1\}$ and $V_c$ is a set of simple parity checks (SPC), the Tanner graph then translates into a Gallager's binary LDPC code. The randomly constructed $(d_s, d_c)$-regular code for $d_s \geq 3$ has a minimum distance that increases linearly with the block length $n$, for $d_s$ and $d_c$ constant [4]. This is only valid for relatively large block lengths, however, and a code with a low minimum distance will be impaired in its performance at high SNRs. Although finding the minimum distance turns out to be a difficult task, some bounds on the minimum distance of a general Tanner graph have been established

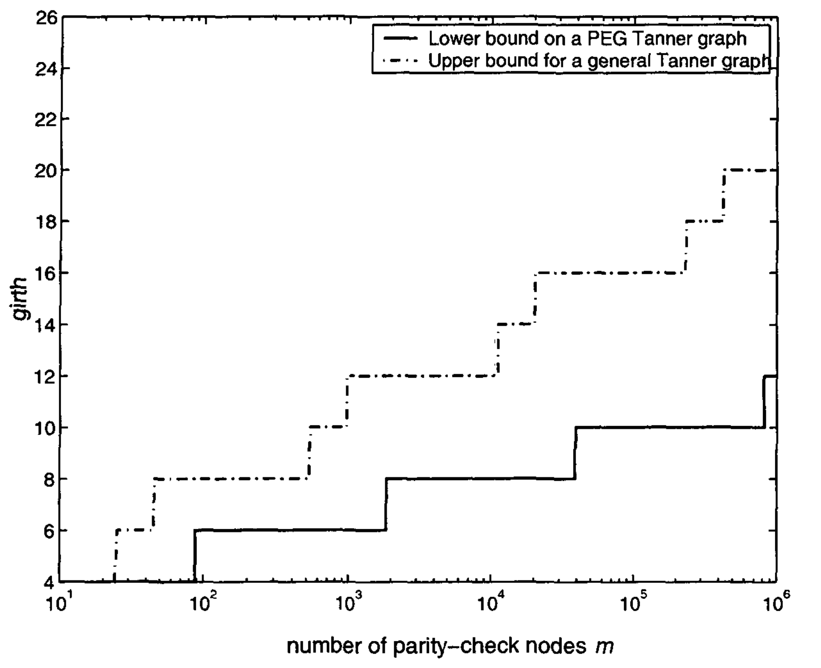


**Figure 3.5:** Lower and upper bounds on a PEG regular Tanner graph with $d_s = 4, d_c = 8$.

in [100]. For a PEG Tanner graph, it is possible to derive a lower bound on the minimum distance in a similar way.

The lower bound on the minimum distance of a PEG Tanner-graph code is based on the properties of the subgraph induced by a minimum weight codeword in the code. Adopting the notation of [100], a symbol node whose associated value in the minimum weight codeword is nonzero will be called an *active* symbol node. The edges incident to active symbol nodes will be called *active* edges, and the check nodes with at least one active incident edge will be called *active* check nodes. Note that in a binary LDPC code, any edge incident to an active symbol node must be active, and any active check node must be incident to *even* active edges.

**Lemma 3.7** *Given a symbol-node regular Tanner graph with $d_s$ ($d_s = 2$) edges incident to each symbol node. Assume the girth of the graph is $g$. Then its minimum distance $d_{\min}$ satisfies*

$$d_{\min} = g/2\ . \tag{3.29}$$

*Proof*: This lemma follows directly from the fact that the shortest cycle in such a graph forms a complete active tree, i.e. a valid codeword, and the fact that it involves exactly $g/2$ active symbol nodes. ■

An application of Lemma 3.7 is to establish an upper bound on the minimum distance of an LDPC code with irregular Tanner graph. Consider a subgraph of the underlying Tanner graph of an LDPC code, which consists of all degree-2 symbol nodes and their associated edges and check nodes, and assume the girth of this subgraph is $g_{\text{sub}-2}$. It then follows that $g_{\text{sub}-2}/2$ is an effective upper bound of the LDPC code.

**Lemma 3.8 [A Variation of Tanner'81]:** *Given a regular Tanner graph with $d_c$ edges incident to each check node and $d_s$ ($d_s \geq 3$) to each symbol node. Assume the girth of the graph is $g$, $g > 4$. Then its minimum distance $d_{\min}$ satisfies*

$$d_{\min} \geq 1 + \frac{d_s[(d_s-1)^{\lfloor (g-2)/4 \rfloor} - 1]}{d_s - 2}. \tag{3.30}$$

*Furthermore, if $g/2$ is even, the lower bound on $d_{\min}$ can be made even tighter:*

$$d_{\min} \geq 1 + \frac{d_s[(d_s-1)^{\lfloor (g-2)/4 \rfloor} - 1]}{d_s - 2} + (d_s - 1)^{\lfloor (g-2)/4 \rfloor}. \tag{3.31}$$

*Proof*: Consider a subgraph induced by a minimum-weight codeword in the graph. The subgraph consists of $n'$ active symbol nodes, $m'$ active check nodes, and all (active) edges incident on $n'$ active symbol nodes. Note that $d_{\min}$ is exactly equivalent to the number of active symbol nodes, $n'$. Now start a depth-$l$ tree from any active symbol node as shown in Fig. 3.6, where $l = \lfloor (g-2)/4 \rfloor$.

Any symbol node, check node, and edge resident in the tree must be active and mutually different, except that the last row of check nodes does not have to meet the condition of being mutually different if $g/2$ is odd. The number of active edges incident to an active check node can in principle be 2, 4, ..., etc.; however only a degree-2 (active) check node is considered to generate a minimum expansion for the number of active symbol nodes, which in turn gives rise to a lower bound on the minimum distance. There is only one active symbol node at depth 0, and there are $d_s$ active symbol nodes at depth 1, $d_s(d_s-1)$ at depth 2, and similarly $d_s(d_s-1)^{l-1}$ at depth $l$. As the girth of the underlying graph is given by $g$, it is clear that all active symbol nodes within this depth-$\lfloor (g-2)/4 \rfloor$ tree must be mutually different, otherwise there always exists a closed path shorter than $g$, leading to a paradox. Thus $d_{\min}$ is lower bounded by the number of active symbol nodes within the depth-$\lfloor (g-2)/4 \rfloor$ tree, that is

$$\begin{aligned} d_{\min} &\geq 1 + d_s + d_s(d_s-1) + \cdots + d_s(d_s-1)^{l-1} \\ &= 1 + \frac{d_s[(d_s-1)^l - 1]}{d_s - 2} \\ &= 1 + \frac{d_s[(d_s-1)^{\lfloor (g-2)/4 \rfloor} - 1]}{d_s - 2}, \end{aligned} \tag{3.32}$$

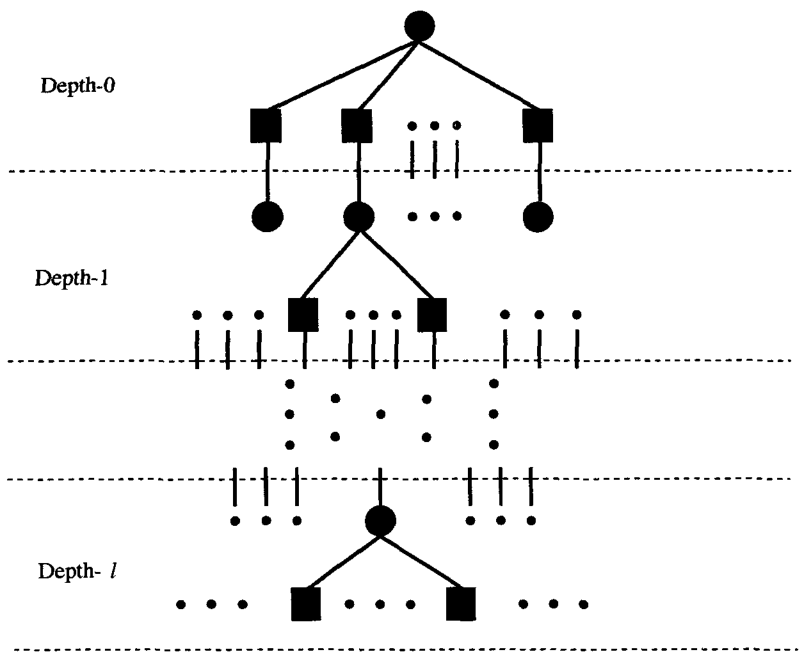


**Figure 3.6:** An "active" tree induced by a minimum-weight codeword.

which proves the first claim. If $g/2$ is even, the tree can proceed one step beyond depth $\lfloor (g-2)/4 \rfloor$, resulting in $d_s(d_s-1)^l$ active symbol nodes. These $d_s(d_s-1)^l$ active nodes are, however, not necessarily mutually different; a lower bound on the number of different active symbol nodes is $d_s(d_s-1)^l$ divided by $d_s$,[3] yielding the additive term $(d_s-1)^{\lfloor (g-2)/4 \rfloor}$. ■

Tanner derived a minimum-distance lower bound for general bipartite graphs [5], which, in the case that all subcodes are simple parity checks, turns out to be

$$d_{\min} \geq \begin{cases} \frac{2\left[(d_s-1)^{(g-2)/4}-1\right]}{d_s-2} + \frac{2}{d_s}[d_s-1]^{(g-2)/4} & \text{for } g/2 \text{ odd} \quad , \\ \frac{2\left[(d_s-1)^{g/4}-1\right]}{d_s-2} & \text{for } g/2 \text{ even} \ . \end{cases} \tag{3.33}$$

Comparing with Eq. (3.33), Eq. (3.30)) and Eq. (3.31) are slightly stronger as $d_s \geq 3$ in general.

Note that for the case of $g = 6$, Eq. (3.30) reduces to the conventional bound $d_{\min} \geq 1 + d_s$, and for $g = 8$, we know $d_{\min} \geq 2d_s$.[4] One might postulate that the PEG construction does not guarantee the uniformity of the check-node-degree sequence, although the resulting PEG Tanner graph is nearly check-node-degree uniform. Our experiments of short codes indicate that uniformity of check nodes can actually be achieved through multiple trials or using non-greedy variants. Nonetheless, the above lower bound can easily be extended to those cases where the degree sequence of check nodes is not uniform.

[3]As each symbol node has at most $d_s$ edges.

[4]This case was observed and pointed out to us by M. P. C. Fossorier.

**Theorem 3.2** *Given a symbol-node-uniform PEG Tanner graph with $d_s$ ($d_s \geq 3$) edges incident to each symbol node, let $d_c^{\max}$ be the largest degree of check nodes. The minimum distance $d_{\min}$ satisfies*

$$d_{\min} \geq \begin{cases} 1 + \frac{d_s[(d_s-1)^{\lfloor (g-2)/4 \rfloor}-1]}{d_s-2} & \text{if } g/2 \text{ is odd} \\ 1 + \frac{d_s[(d_s-1)^{\lfloor (g-2)/4 \rfloor}-1]}{d_s-2} + (d_s-1)^{\lfloor (g-2)/4 \rfloor} & \text{if } g/2 \text{ is even,} \end{cases} \tag{3.34}$$

*where the girth $g$ is lower bounded by*

$$g \geq 2(\lfloor t \rfloor + 2), \tag{3.35}$$

*in which*

$$t = \frac{\log(md_c^{\max} - \frac{md_c^{\max}}{d_s} - m + 1)}{\log[(d_s-1)(d_c^{\max}-1)]} - 1. \tag{3.36}$$

*Proof.* The proof follows directly from Theorem 3.1 and Lemma 3.8. ■

Note that the above bound on the minimum distance is still a weak bound, although it always furnishes a meaningful bound on graphs having a large girth. There are two reasons for the weakness of this lower bound. The first is the assumption that all active check nodes are satisfied by exactly two symbol nodes, which weakens the estimate of the minimum distance. The second is that the condition that the last row of check nodes must be satisfied with additional active symbol nodes has not been taken into account.

The above theorem can readily be extended and applied to analyze the minimum distance of irregular PEG Tanner-graph codes. As a working example, suppose a simple symbol-node-degree distribution defined as $0.8x^2 + 0.2x^{20}$, and $n$ is chosen as 3000 and $m$ as 2000. As specified by the PEG construction, the edges of degree-2 symbol nodes are established first, and the resulting girth of the subgraph containing all degree-2 symbol nodes turns out to be 44, which means that the minimum distance involving only degree-2 symbol nodes is 22, a result derived directly from the argument that the shortest cycle involving only degree-2 symbol nodes leads to a minimum weight codeword. Next, the edges of degree-20 symbol nodes are constructed, and finally the global girth is found to be 6. By applying Eq. (3.32), an active tree stemming from a degree-20 symbol node will contain at least 21 symbol nodes, which means that a minimum codeword involving a degree-20 symbol node will have a weight of 21 at least. Therefore one can know that the minimum distance of the resultant PEG Tanner-graph code is lower-bounded by 21, or more specifically, in our example it can be either 21 or 22. This argument justifies the effort in the PEG construction to order the symbol degree sequence $D_s$ in a *nondecreasing* order and hence construct the edges of lower-degree symbol nodes earlier, improving the minimum distance induced by the subgraph of lower-degree symbol nodes, particularly degree-2 symbol nodes.

It is worth pointing out that the girth of Tanner graphs can also lead to a lower bound on the size of the smallest stopping set [101]. The finite-length error probability of iterative decoding over binary erasure channels (BEC) has been determined via stopping sets in the Tanner graph representation of an LDPC code [102, 103]. The size of the smallest stopping set has a direct impact on the error-floor behavior.

## 3.5 Asymptotic Analysis of Ensemble Codes

In the preceding section, we derived a lower bound on the minimum distance of binary LDPC codes underlying PEG Tanner graphs that is mainly based on the girth property. However, the analysis resting solely on the girth property seems insufficient to demonstrate that families of codes are asymptotically good. Next, we shall prove that the ensemble of PEG codes is indeed asymptotically good, i.e., its distance distribution asymptotically approaches that of the equiprobable ensemble of parity-check codes. In order to simplify the analysis, we will extend (to say exactly, relax) the PEG algorithm such that the PEG Tanner graphs degrade to a general sparse random ensemble and in turn all variants of PEG graphs can be visualized as expurgated random ensembles in which bad graphs with short cycles and/or small minimum distance codewords are avoided.

The equiprobable random ensemble of parity-check codes has been analyzed in [4]; it is a capacity-approaching family of codes for many practical channels, specifically binary-symmetric stationary ergodic channels [62, 104]. The equiprobable random ensemble of parity-check codes of rate $R$ and block length $n$ is defined as the ensemble in which the $m \times n$, $R = 1 - \frac{m}{n}$ [5], parity-check matrix is filled with statistically independent equiprobable binary digits $\{0, 1\}$. Averaged over the equiprobable ensemble of parity-check codes, the minimum distance is a random variable whose distribution function can be represented by the following lemma.

**Lemma 3.9 [Gallager'63]:** *Over the equiprobable ensemble of parity-check codes of length*

[5]The actual rate in the ensemble may have a rate slightly higher than $R$, since the rows of a matrix in this ensemble are not necessarily independent over the modulo 2 field. Nonetheless, this minor difference does not cause serious problem, see [4, 9, 10] for instance.

*n and rate R, the minimum distance distribution function $Pr(D \leq \delta n)$ is given by*

$$Pr(D \leq \delta n) = \sum_{j=1}^{\delta n} \binom{n}{j} 2^{-n(1-R)} \tag{3.37}$$

*Proof*: See [4, Eq. (2.6), p. 13]. ∎

Now we describe the procedure to relax the PEG construction to facilitate simple analysis: Recalling that in the PEG algorithm, we partition the entire set of check nodes into two subsets whenever a new edge of a symbol node ($s_j$) is being added: one is the *prohibitive* subset, on which if the new edge is placed, there exits a short cycle — this check-node set corresponds to the depth-$l$ neighbors of the symbol node $\mathcal{N}_{s_j}^l$; the other is the *feasible* subset from which each element is selected with *equal* probability to get connected with the symbol node, which is the complement set of $\mathcal{N}_{s_j}^l$. If we relax the greedy PEG algorithm such that the prohibitive subset only contains the existing connected parity checks, i.e., the expanding tree halts at depth-0 ($l = 0$), then the PEG construction reduces to a simple random construction,[6] each column in the parity-check matrix containing exactly $d_s$ 1's. It is clear that the PEG codes are a particular subset of these random matrices, from which bad codes (graphs) containing small minimum distance (short cycles) are successfully avoided.

It still turns out to be a tedious task to analyze the distance distribution function of the random construction discussed previously. One can relax it further by means of removing the constraint that the 1's in each column of parity-check matrix be exactly $d_s$, instead, one can define a new ensemble of parity-check matrices, i.e. sparse random matrices $H$ in which each entry in a column takes on 1 with probability $p$ and 0 with $1-p$, $p = d_s/m, 0 < p < 1/2$, $p$ is called *density*. Such an ensemble is naturally a broader random construction and contains the previous random construction as its own subset. Note that this relaxed random construction with density $p$ is asymptotically the same as the old one, and also it is a weaker extension since it includes some bad codes (graphs) again.

In summary, we obtain a series of relaxations of PEG constructions. These PEG Tanner graphs can naturally be viewed as *expurgated* ensembles of the sparse random graphs with density $p$, from which bad graphs having short cycles and/or small minimum distance are effectively precluded. Our asymptotic analysis of PEG constructions relies on an extreme degradation of PEG Tanner graphs, i.e, the sparse random matrix $H$ with density $p$ described previously.

[6]This ensemble is equivalent to that of Section II-A.(3) in [71].

**Lemma 3.10** *Let $H$ be a random $n(1-R) \times n$ matrix over GF(2), with a density of $p$, $0 < p < 1/2$. The minimum distance distribution function $Pr(D \leq \delta n)$ is given by*

$$Pr(D \leq \delta n) = \sum_{j=1}^{\delta n} \binom{n}{j} \left[\frac{2}{1+(1-2p)^j}\right]^{-n(1-R)} \tag{3.38}$$

*Proof*: Fix some nonzero vector $V = [v_0, v_1, \ldots, v_{n-1}] \in [\mathrm{GF}(2)]^n$, *with exactly $j$ nonzero* coordinates, i.e., the Hamming weight of this vector is $j$. Without loss of generality, assume that the first $j$ coordinates of $V$ are nonzero. Let $P_j$ be the probability that $\sum_{i=1}^{j} v_i h_{k,i} = 0$, where each $h_{k,i}$ is the entry of $k$-th row of the matrix $H$ and is thus chosen according to the uniform distribution of $p$. As there is a total of $n(1-R)$ independent check equations, the probability that $V$ is a valid codeword is given by $P_j^{n(1-R)}$.

Given that the $v_i$'s are fixed, the following recursion holds for $P_j$.

$$P_j = P_{j-1}(1-p) + (1-P_{j-1})p \, , \tag{3.39}$$

with the initial condition $P_0 = 1$. Denote $Q_j = P_j - 1/2$. It follows from Eq. (3.39) that

$$Q_j = Q_{j-1}(1-2p) \, , \tag{3.40}$$

with $Q_0 = 1/2$. Hence we obtain

$$Q_j = \frac{1}{2}(1-2p)^j \, , \tag{3.41}$$

and, consequently,

$$P_j = \frac{1}{2} + \frac{1}{2}(1-2p)^j \, . \tag{3.42}$$

The number of vectors $V$ with exactly $j$ nonzero coordinates is given by $\binom{n}{j}$; the minimum distance distribution function $\Pr(D \leq \delta n)$ can then be written as

$$\begin{aligned} \Pr(D \leq \delta n) &= \sum_{j=1}^{\delta n} \binom{n}{j} P_j^{n(1-R)} \\ &= \sum_{j=1}^{\delta n} \binom{n}{j} \left[\frac{1}{2} + \frac{1}{2}(1-2p)^j\right]^{n(1-R)} \\ &= \sum_{j=1}^{\delta n} \binom{n}{j} \left[\frac{2}{1+(1-2p)^j}\right]^{-n(1-R)} . \end{aligned}$$

This completes the proof. ■

**Lemma 3.11** *Let $H$ be a random $n(1-R) \times n$ matrix over GF(2), with a density of $p$, $0 < p < 1/2$. The minimum distance grows at least linearly with block length $n$.*

*Proof*: It suffices to prove that, given any positive $p$, there exists a positive number $\epsilon$ such that $\Pr(d_{\min} \leq \epsilon n) \xrightarrow{exp} 0$. Starting with Eq. (3.38), we have

$$\begin{aligned}\Pr(d_{\min} \leq \epsilon n) &= \sum_{j=1}^{\epsilon n} \binom{n}{j} \left[\frac{2}{1+(1-2p)^j}\right]^{-n(1-R)} \\ &\leq \sum_{j=1}^{\epsilon n} \binom{n}{j} \left[\frac{2}{1+(1-2p)}\right]^{-n(1-R)} .\end{aligned}$$

Recall the following inequality [4, (2.7), p. 13]

$$\sum_{j=1}^{\epsilon n} \binom{n}{j} \leq \binom{n}{\epsilon n} \frac{1-\epsilon}{1-2\epsilon} . \tag{3.43}$$

We obtain

$$\Pr(d_{\min} \leq \epsilon n) \leq \binom{n}{\epsilon n} \frac{1-\epsilon}{1-2\epsilon} \left[\frac{1}{1-p}\right]^{-n(1-R)} . \tag{3.44}$$

In view of the following bound on $\binom{n}{\epsilon n}$ [62, p. 530]

$$\frac{1}{\sqrt{8n\epsilon(1-\epsilon)}} 2^{nH_2(\epsilon)} \leq \binom{n}{\epsilon n} \leq \frac{1}{\sqrt{2\pi n\epsilon(1-\epsilon)}} 2^{nH_2(\epsilon)} . \tag{3.45}$$

We have

$$\Pr(d_{\min} \leq \epsilon n) \leq \frac{1}{1-2\epsilon} \sqrt{\frac{1-\epsilon}{2\pi n\epsilon}} 2^{n[H_2(\epsilon)-(1-R)\log_2 \frac{1}{1-p}]} . \tag{3.46}$$

As $H_2(\epsilon)$ is monotonically increasing within $(0, 1/2)$, there must be a small positive number $\epsilon$, for any positive $p \in (0, 1/2)$, that satisfies

$$H_2(\epsilon) - (1-R)\log_2 \frac{1}{1-p} \leq 0 .$$

Thus,

$$\Pr(d_{\min} \leq \epsilon n) \longrightarrow 0 \qquad \text{exponentially with } n . \tag{3.47}$$

■

**Theorem 3.3** *Given a random matrix with density $p$, i.e., a degradation of a PEG ensemble of uniform-symbol-degree $d_s$ ($d_s \geq 3$), its minimum distance distribution function approaches closely that of the equiprobable ensemble of parity-check codes, for sufficiently large $n$, fixed $R$ and fixed $p$, where $0 < p < 1/2$, $p = d_s/[n(1-R)]$.*

*Proof*: In view of Eq. (3.47), the minimum distance distribution function of the sparse random Tanner graph with fixed density $p$ can be rewritten as

$$\begin{aligned}
\Pr(D \le \delta n) &= \Pr(\epsilon n \le D \le \delta n) \\
&= \sum_{j=\epsilon n}^{\delta n} \binom{n}{j} \left[\frac{2}{1+(1-2p)^j}\right]^{-n(1-R)} \\
&\le \sum_{j=\epsilon n}^{\delta n} \binom{n}{j} \left[\frac{2}{1+(1-2p)^{\epsilon n}}\right]^{-n(1-R)} \\
&= \sum_{j=\epsilon n}^{\delta n} \binom{n}{j} 2^{-n(1-R)} [1+(1-2p)^{\epsilon n}]^{n(1-R)} . \qquad (3.48)
\end{aligned}$$

To prove Eq. (3.48) approaches asymptotically Eq. (3.37), it remains to show

$$\lim_{n\to\infty} [1+(1-2p)^{\epsilon n}]^{n(1-R)} = 1 . \qquad (3.49)$$

Using $1+x \le e^x$ twice we get

$$\begin{aligned}
[1+(1-2p)^{\epsilon n}]^{n(1-R)} &\le \left[1+e^{-2p\epsilon n}\right]^{n(1-R)} \\
&\le e^{n(1-R)e^{-2p\epsilon n}} . \qquad (3.50)
\end{aligned}$$

Therefore

$$\begin{aligned}
\lim_{n\to\infty} [1+(1-2p)^{\epsilon n}]^{n(1-R)} &\le \lim_{n\to\infty} e^{n(1-R)e^{-2p\epsilon n}} \\
&= e^{\lim_{n\to\infty} n(1-R)e^{-2p\epsilon n}} \\
&= e^0 = 1 . \qquad (3.51)
\end{aligned}$$

Combining the fact that $[1+(1-2p)^{\epsilon n}]^{n(1-R)} \ge 1$, we obtain Eq. (3.49). ■

Because the maximum-likelihood decoding performance of a linear code depends uniquely on the minimum distance distribution function, this theorem effectively states that the binary low-density parity codes underlying PEG Tanner graphs are asymptotically good, i.e., capable of achieving reliable communication at rates up to the capacity of the binary-symmetrical stationary ergodic channels, *when optimally decoded.*

We can prove that the binary random or PEG codes also attain the (asymptotic) Gilbert–Varshamov bound with high probability. A code with block length $n$ and rate $R$ satisfies the Gilbert–Varshamov minimum-distance bound if the relative minimum distance $\delta = d_{\min}/n$ satisfies

$$H_2(\delta) = 1 - R , \qquad (3.52)$$

where $H_2(\delta)$ is the binary entropy function

$$H_2(\delta) = -\delta \log_2 \delta - (1-\delta)\log_2(1-\delta) . \qquad (3.53)$$

**Theorem 3.4** *Given a random matrix with density p, i.e., a degradation of the PEG ensemble of uniform-symbol-degree $d_s$ ($d_s \geq 3$), the binary LDPC code underlying this graph attains the Gilbert–Varshamov minimum-distance bound with high probability.*

*Proof*: First, recall the following inequality [4, (2.7), p. 13]

$$\sum_{j=\epsilon n}^{\delta n} \binom{n}{j} \leq \sum_{j=1}^{\delta n} \binom{n}{j} \leq \binom{n}{\delta n} \frac{1-\delta}{1-2\delta} . \tag{3.54}$$

By inserting Eq. (3.54) in Eq. (3.48), we obtain

$$\Pr(D \leq \delta n) \leq \binom{n}{\delta n} \frac{1-\delta}{1-2\delta} 2^{-n(1-R)} \left[1 + (1-2p)^{\epsilon n}\right]^{n(1-R)} . \tag{3.55}$$

Recall the following bound on $\binom{n}{\delta n}$ [62, p. 530]

$$\frac{1}{\sqrt{8n\delta(1-\delta)}} 2^{nH_2(\delta)} \leq \binom{n}{\delta n} \leq \frac{1}{\sqrt{2\pi n\delta(1-\delta)}} 2^{nH_2(\delta)} . \tag{3.56}$$

Thus 3.55 is upper bounded by

$$\Pr(D \leq \delta n) \leq \frac{1}{1-2\delta} \sqrt{\frac{1-\delta}{2\pi n\delta}} 2^{nH_2(\delta)} 2^{-n(1-R)} \left[1 + (1-2p)^{\epsilon n}\right]^{n(1-R)} . \tag{3.57}$$

Using Eq. (3.49) for sufficiently large $n$, we obtain

$$\Pr(D \leq \delta n) \leq \frac{1}{1-2\delta} \sqrt{\frac{1-\delta}{2\pi n\delta}} 2^{n[H_2(\delta)-(1-R)]} . \tag{3.58}$$

For increasing $n$, the bound of $\Pr(D \leq \delta n)$ as a function of $\delta$ approaches a step function with the step at $\delta_0$ such that $H_2(\delta_0) = (1-R)$. In particular if $\delta = \delta_0$, then

$$\Pr(D \leq \delta_0 n) \leq \begin{cases} \frac{1}{1-2\delta_0} \sqrt{\frac{1-\delta_0}{2\pi n\delta_0}} & \\ = 0 & \text{for } n \to \infty \end{cases} \tag{3.59}$$

Furthermore, recognizing that $H_2(\delta)$ is monotonically increasing with $\delta \in (0, 1/2)$, we conclude that for any $\epsilon > 0$, the probability for which $D < n(\delta_0 - \epsilon)$ exponentially approaches 0 with $n$. This completes the proof. ■

## 3.6 Irregular PEG Tanner Graphs

For very long block lengths, the authors of [9, 52] prove a concentration theorem that states that in the limit large random graphs can be assumed to be effectively cycle-free. This allows the calculation of precise convergence thresholds under the sum-product decoding algorithm via the density evolution approach. An ensemble of random codes with optimum degree-distribution pairs can then be carefully chosen to optimize its threshold. This approach has been used to find good irregular LDPC codes based on the random construction that exhibit a performance extremely close to the Shannon limit on typical memoryless channels.

However, two issues complicate the analysis and design of codes having short block lengths. The first is that the concentration theorem applies only to the asymptotic case; for short block lengths, random graphs experience significant deviation as pointed out in [105]. The second is that the cycle-free assumption is no longer valid, which is a complication the density evolution cannot handle adequately.

In this section, we briefly describe an empirical Monte–Carlo approach using a variant of the "downhill simplex" optimization technique to design symbol-node-degree distributions. The resulting degree distribution is used to optimize a LDPC code with specific parameters $n$ and $m$ whose Tanner graph is constructed by the PEG algorithm.

The "downhill simplex" algorithm in general is a nonlinear unconstrained optimization procedure that operates on a multidimensional simplex consisting of several vertices [85, 88]. Here we adopt a simple variant of the "downhill simplex" method that can handle constraints. Each vertex represents a feasible symbol-node-degree distribution, and its cost function is defined as the block-error rate at a $E_b/N_0$ threshold. The worst vertex is replaced by a better vertex according to three basic operations: contraction, expansion, and reflection, resulting in a new simplex. This step is repeated until the diameter of the simplex is less than a preselected tolerance.

## 3.6.1 Constraints

The optimization parameters are the symbol-node-degree distribution that must be a probability vector. Assume there are $l$ nonzero coefficients in the symbol-node-degree distribution $\Lambda(x)$, and define

$$\Lambda(x) = \sum_{i=1}^{l} \Lambda_i x^{d_i},$$

where $d_i$ is the preselected degree of the $i$-th nonzero entry in the degree distribution. Then we have the following constraints:

$$\sum_{i=1}^{l} \Lambda_i = 1.0, \tag{3.60}$$

where $0.0 < \Lambda_i < 1.0$, for $i = 1, 2, \ldots, l$. To improve the searching efficiency, we can reduce this $l$-dimensional searching problem to a $(l-1)$-dimensional one, i.e., we form a constrained optimization problem with an $(l-1)$-dimensional parameter vector $\mathbf{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_{l-1})$ under the following inequality constraints:

$$\begin{cases} 0.0 < \Lambda_i < 1.0 & \text{for } i = 1, 2, \cdots, l-1 \\ 0.0 < \sum_{i=1}^{l-1} \Lambda_i < 1.0. & \end{cases} \tag{3.61}$$

The $l$-th coefficient of $\Lambda(x)$ is computed by

$$\Lambda_l = 1.0 - \sum_{i=1}^{l-1} \Lambda_i.$$

### 3.6.2 Cost Function

We need a cost function to assess the quality of a valid symbol-node-degree distribution $\mathbf{\Lambda}$, and then minimize this function by manipulating the simplex. The most natural cost function is the block-error rate at a certain SNR per bit $(E_b/N_0)$. In this approach we first construct a PEG graph in accordance with $\mathbf{\Lambda}$, and then evaluate the average block-error rate at a certain $E_b/N_0$ using Monte Carlo simulations with a prescribed number of blocks. The value of $E_b/N_0$ is initialized with a relatively large number, then gradually decreases in small steps whenever a zero block-error rate is encountered. The final $E_b/N_0$, at which there is no PEG graph with a symbol-node-degree distribution $\mathbf{\Lambda}$ having a zero block-error rate, is called the $E_b/N_0$ threshold.

### 3.6.3 Constrained "Downhill Simplex" Search

The method we have adopted in designing irregular PEG Tanner graphs is a constrained "downhill simplex" nonlinear optimization procedure that uses only cost function evaluations. A simplex in our setting is a geometrical manifold consisting of $2(l-1)$ vertices, with each vertex representing a valid symbol-node-degree distribution. We summarize the algorithm as follows:

**The Constrained "Downhill Simplex" Algorithm:**

**Step 1: Initializing a simplex**

**1a.** Find a vertex $\mathbf{\Lambda}^1$ that meets all constraints. This can easily be done manually.

**1b.** Determine the other $2l-3$ vertices according to:

1.
   ```
   for $j = 2$ to $2l-2$
       for $i = 1$ to $l-1$
           $\Lambda_i^j \longleftarrow$ Random[0, 1]
       end
   end
   ```
   where $\Lambda_i^j$ is the $i$-th element of the $j$-th vertex $\Lambda^j$ in the simplex, and Random[0, 1] is a uniform [0, 1] random number generator.

2. Check whether $\sum_i^{l-1} \Lambda_i^j < 1.0$ in the sequence of $j = 1, 2, \ldots, 2l-2$. Suppose the first $j$ vertices all meet this constraint, and the $(j+1)$-th does not, then

$$\Lambda^{j+1} = (\Lambda^{j+1} + \mathbf{t})/2,$$

   where

$$\mathbf{t} = \frac{1}{j} \sum_{k=1}^{j} \Lambda^k.$$

   Repeat this check procedure until all vertices $\Lambda^j$, $j = 1, 2, \ldots, 2l-2$, satisfy the constraint.

**1c.** Evaluate cost functions of the vertices in the simplex, i.e., the block-error rates $P_e^j$ of codes associated with irregular PEG graphs generated with $\Lambda^j$, where $j = 1, 2, \ldots, 2l-2$.

**Step 2: Forming a new simplex**

**2a.** Determine the worst vertex $P_e^W$ and the second worst vertex $P_e^w$,

$$P_e^W \overset{def}{=} \max_{1 \le j \le 2l-2} P_e^j$$

and

$$P_e^w \overset{def}{=} \max_{1 \le j \le 2l-2, j \ne W} P_e^j.$$

**2b.** Compute the reflection vertex $\Lambda^r$ of the worst vertex $\Lambda^W$ in the current simplex.

$$\Lambda^r = (1+\alpha)\Lambda^R - \alpha\Lambda^W,$$

in which

$$\Lambda^R = \frac{1}{2l-3} \sum_{j=1, j \ne W}^{2l-2} \Lambda^j$$

**2c.** Search a new vertex $\Lambda^n$ based on $\Lambda^r$ to replace the worst vertex $\Lambda^W$, forming a new simplex. The search procedure is as follows.

```
1. for i=1 to $l-1$
        if $\Lambda_i^r < 0.0$ then $\Lambda_i^r = \Lambda_i^r + \delta$
        if $\Lambda_i^r > 1.0$ then $\Lambda_i^r = \Lambda_i^r - \delta$
   end
```

   Repeat this "**for**" loop until $0.0 < \Lambda_i^r < 1.0$ for all $i = 1, 2, \dots, l-1$. $\delta$ is usually set to $10^{-5}$.

2. Check whether $\sum_i^{l-1} \Lambda_i^r < 1.0$. If not,

$$\Lambda^r = (\Lambda^R + \Lambda^r)/2,$$

   and then go back to **2c** in **Step 2**.

3. Evaluate the cost function $P_e^r$ of the vertex $\Lambda^r$. If $P_e^r < P_e^w$, then

$$\Lambda^n = \Lambda^r, \; P_e^n = P_e^r.$$

   Otherwise,

$$\Lambda^r = (\Lambda^R + \Lambda^r)/2,$$

   and then go back to **2c** in **Step 2**.

**2d.** Compute the average distance of these vertices in the new simplex. If the average distance is smaller than a prescribed value, the optimum vertex is chosen as the one having the smallest block-error rate among the simplex, and the optimization algorithm terminates. Otherwise, go back to **2a** in **Step 2** and start a new round to evolve the simplex.

It can be seen that the computational complexity depends primarily on the evaluation of the cost function of a valid symbol-node-degree distribution. Experimental results show that in general approximately 100 evaluations of the block-error rate at the $E_b/N_0$ threshold will be sufficient for the constrained "downhill simplex" algorithm to yield a satisfactory symbol-node-degree distribution. Note that the computational load can be greatly reduced by means of efficient implementations of the sum-product algorithm for decoding LDPC codes [106–108]. Further reductions may be possible by using other, easily computable cost functions, see for example [89].

Note that the Monte–Carlo scheme can be viewed as being complementary to the density evolution scheme in the sense that the former is an empirical approach and its accuracy depends primarily on the amount of computation power available, whereas the latter is a theoretical approach and is based on asymptotic analysis. The density evolution approach proves to be very successful for long block-length codes and for the random construction; but it is not clear yet whether it is still applicable to short block-length codes and to the PEG construction; the Monte–Carlo approach is also indispensable for non-binary LDPC codes where the density evolution approach is not yet available.

## 3.7 Code Performance

### 3.7.1 Regular Codes

In this section we study the performance of PEG Tanner graphs applied to binary LDPC codes by means of computer simulations. For comparison purposes, we use the rate-1/2 ($n = 504$, $m = 252$) code of MacKay in [109], which is based on a regular Tanner graph with $d_s = 3, d_c = 6$. Currently this code is one of the best codes with these parameters. A PEG Tanner graph of 504 symbol and 252 check nodes is generated with uniform degree 3 for each symbol node. The resulting graph is nearly check-node uniform with degree 6, except for 8 check nodes with a degree of 7, and 8 with a degree of 5. We also use a randomly constructed rate-1/2 (504, 252) code, in which the degree of symbol nodes is 3 and the positions of 1s in a column is determined by a random integer generator uniformly distributed among the set $\{0, 1, \ldots, m-1\}$. Additional tests are implemented to avoid 4 cycles in the graph representation.

Fig. 3.7 compares the girth histogram of the PEG, MacKay's, and random graph codes. In the PEG Tanner graph, each symbol node has a local girth of 8, except for three symbol nodes with a local girth of 10. In MacKay's code, 63% of the symbol nodes have a local girth of 6 and 37% one of 8. In the random graph, 79% of the symbol nodes have a local girth of 6 and 21% one of 8. The average local girth of these three graphs is 8.01, 6.74, and 6.42, respectively. Clearly, based on the girth histogram, the PEG Tanner graph has an advantage over its two counterparts.

Fig. 3.8 compares the bit- and block-error rates for the three codes after 80 iterations over an additive white Gaussian noise (AWGN) channel,[7] and reveals that the performance of the random graph is much worse than that of the other two codes, perhaps mainly because of its poor girth histogram. We observe that the LDPC code based on the PEG Tanner graph is always slightly better than MacKay's code. With 80 iterations and at a block-error rate of $5 \times 10^{-5}$, the LDPC code based on the PEG Tanner graph outperforms MacKay's code by 0.2 dB. The significance of this result should not be underestimated, considering that, to the

[7]Throughout, the channel model is assumed to be the binary-input AWGN channel.

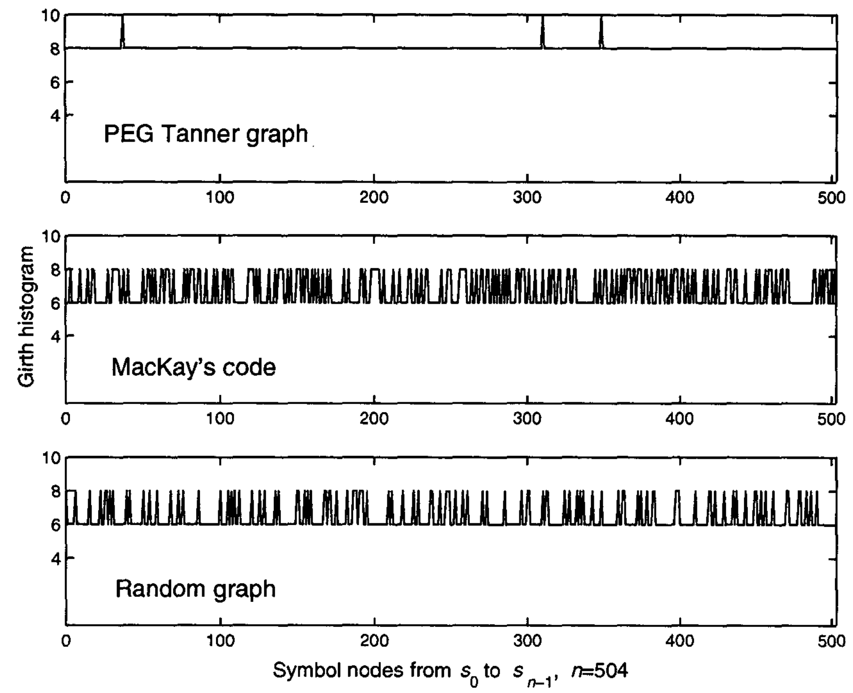


**Figure 3.7:** Girth histograms of a PEG Tanner graph, MacKay's code, and a random graph, with parameters $n = 504, m = 252, d_s = 3, d_c = 6$.

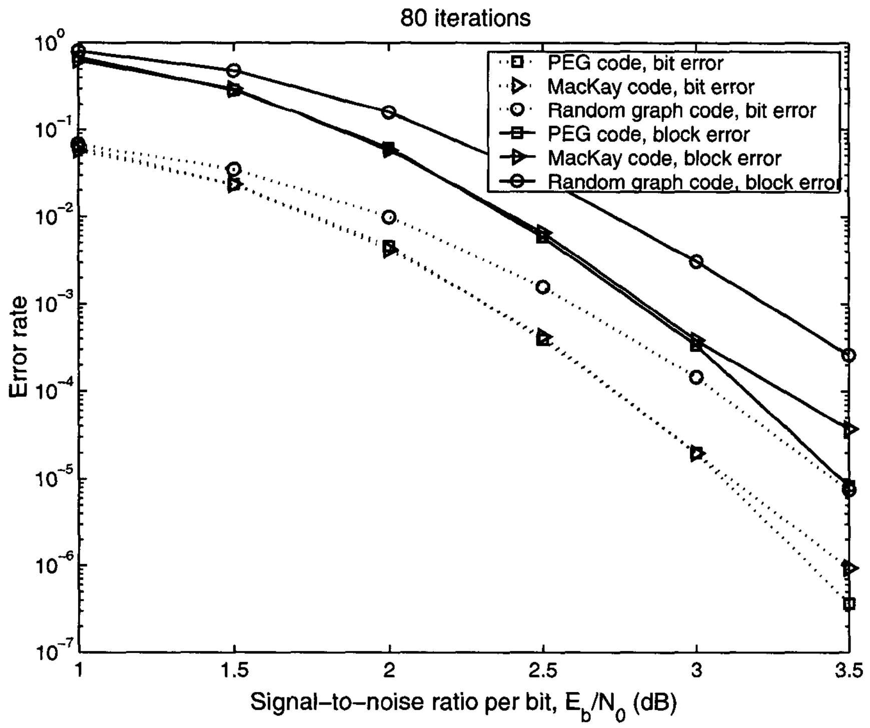


**Figure 3.8:** Bit- and block-error rates of PEG Tanner-graph code, MacKay's code, and random graph code, with parameters $n = 504, m = 252, d_s = 3, d_c = 6$.

best of our knowledge, MacKay's codes still are the best codes for short and medium block lengths.

Note that although both MacKay's code and the random graph have a global girth of 6, the performance of the latter degrades significantly. This suggests that in reality it is not only the girth but also the girth histogram that dominate the performance of the iterative decoding. In [105] the average of the girth histogram is used as a heuristic tool to select good codes from random graphs for short block lengths.

Fig. 3.9 compares the girth histograms of the PEG, MacKay's and random graphs with parameters $n = 1008, m = 504, d_s = 3, d_c = 6$. In the PEG Tanner graph, 17% of the symbol nodes have a local girth of 8 and 83% one of 10. In MacKay's code, 39.5% of the symbol nodes have a local girth of 6 and 60.3% one of 8. In the random graph, 55.6% of the symbol nodes have a local girth of 6 and 44.2% one of 8. The average local girth of these three graphs is 9.66, 7.214, and 6.892, respectively. Fig. 3.10 compares the bit and block-error rates for these three (1008, 504) codes after 80 iterations. Again, we observe that the LDPC code based on the PEG Tanner graph is much better than the LDPC code based on the random graph, and slightly better than MacKay's code.

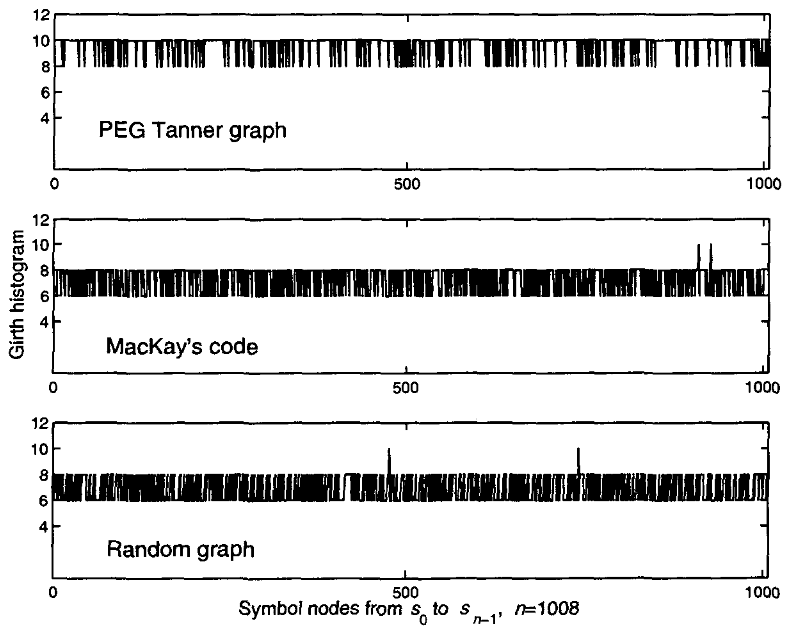


**Figure 3.9:** Girth histograms of a PEG Tanner graph, MacKay's code, and a random graph, with parameters $n = 1008, m = 504, d_s = 3, d_c = 6$.

We report that the look-ahead-enhanced PEG construction yields a Tanner graph of girth 10 for $n = 1008, m = 504, d_s = 3, d_c = 6$. This represents an improvement in terms of girth relative to the generic PEG construction, in which the girth is 8. Moreover, by applying

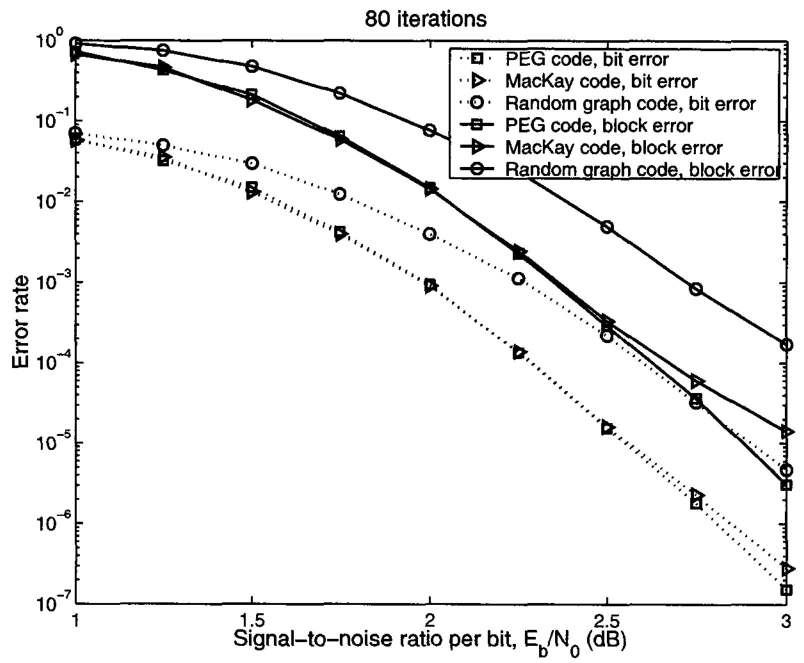


**Figure 3.10:** Bit- and block-error rates of a PEG Tanner-graph code, MacKay's code, and random graph code, with parameters $n = 1008, m = 504, d_s = 3, d_c = 6$.

Lemma 6 with these parameters we know that in this case the upper bound on the girth is 12, indicating that the look-ahead-enhanced PEG is close to the globally optimum.

### 3.7.2 Irregular Codes

In this subsection we report on the code performance of near-optimum irregular symbol-node-degree distributions for the PEG construction, obtained by the Monte–Carlo approach together with constrained "downhill simplex" algorithm described above. The symbol-node-degree distribution is $\Lambda(x) = 0.47532x^2+0.279537x^3+0.0348672x^4+0.108891x^5+0.101385x^{15}$, which is optimized for $n = 504, m = 252$. As required by the PEG construction, the check-node-degree distribution is automatically concentrated and thus uniquely determined by the symbol-node-degree distribution and the code rate [8]. The PEG LDPC code we considered is of rate-1/2, and its performance is depicted in Fig. 3.11. As can be seen this code outperforms MacKay's rate-1/2 (504, 252) code by about 0.4 to 0.5 dB in the entire range of $E_b/N_0$ after 80 iterations. Also plotted is the performance of a randomly constructed code using the same degree distribution,[9] exhibiting a very early "error floor" effect. This clearly shows that the

[8]Whenever a check-node-degree distribution is not explicitly shown, it is assumed to be concentrated.

[9]The irregular random code is constructed column by column with appropriate degree, and the 1s in each column are determined by a uniform integer generator. Efforts have been taken to avoid 4-cycles.

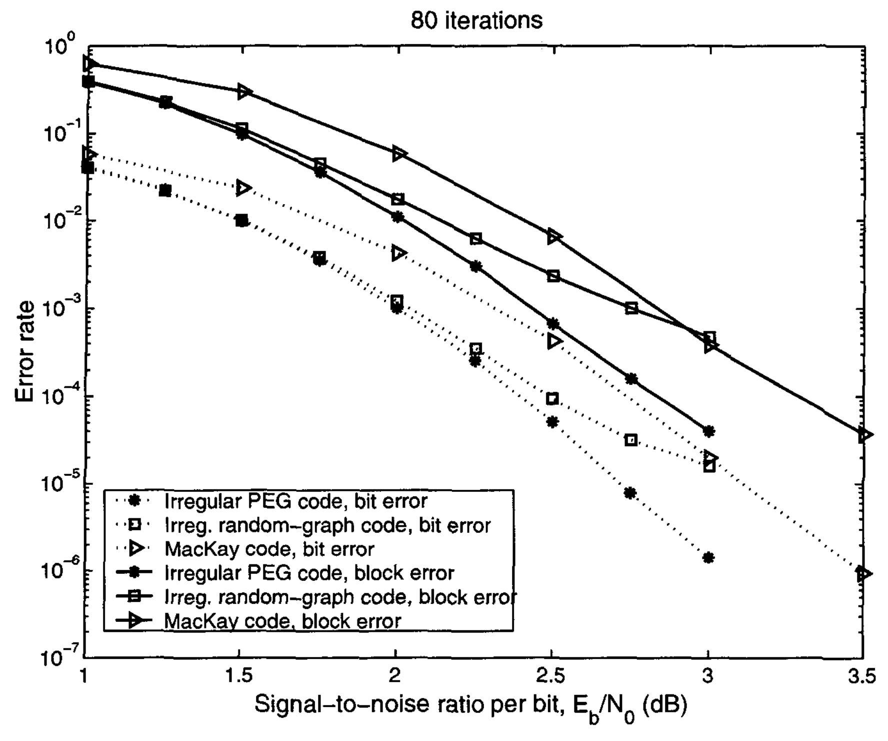


**Figure 3.11:** Bit- and block-error rates of the irregular PEG Tanner-graph code, irregular random-graph code, and MacKay's code (regular) with parameters $n = 504, m = 252$. The symbol-node-degree distribution for the irregular PEG Tanner-graph code is $\Lambda(x) = 0.47532x^2 + 0.279537x^3 + 0.0348672x^4 + 0.108891x^5 + 0.101385x^{15}$.

PEG construction provides a significant advantage over the random construction, particularly for short block lengths and for cases in which the figure of merit is the block error rate.

We also investigate this degree distribution applied to the case $n = 1008, m = 504$. The performance of the resulting LDPC code constructed with the PEG algorithm is shown in Fig. 3.12. This figure indicates nearly 0.5 dB gain over MacKay's rate-1/2 (1008, 504) code. To the best of our knowledge, the irregular PEG LDPC codes reported here are the best ones to date in terms of block-error rate at short block lengths.

Density evolution has proven to be an efficient and effective approach to design good irregular-degree-distribution pairs with which LDPC codes based on random construction exhibit a performance extremely close to the Shannon limit for sufficiently long block lengths. It is thus tempting to combine the PEG construction with the symbol-node-degree distribution optimized by the density evolution approach to design LDPC codes. We investigate the performance of symbol-node-degree distributions [10] in [10, Tables I and II] under the PEG

[10]We do not need the check-node distribution as the check-degree sequence is made as uniform as possible

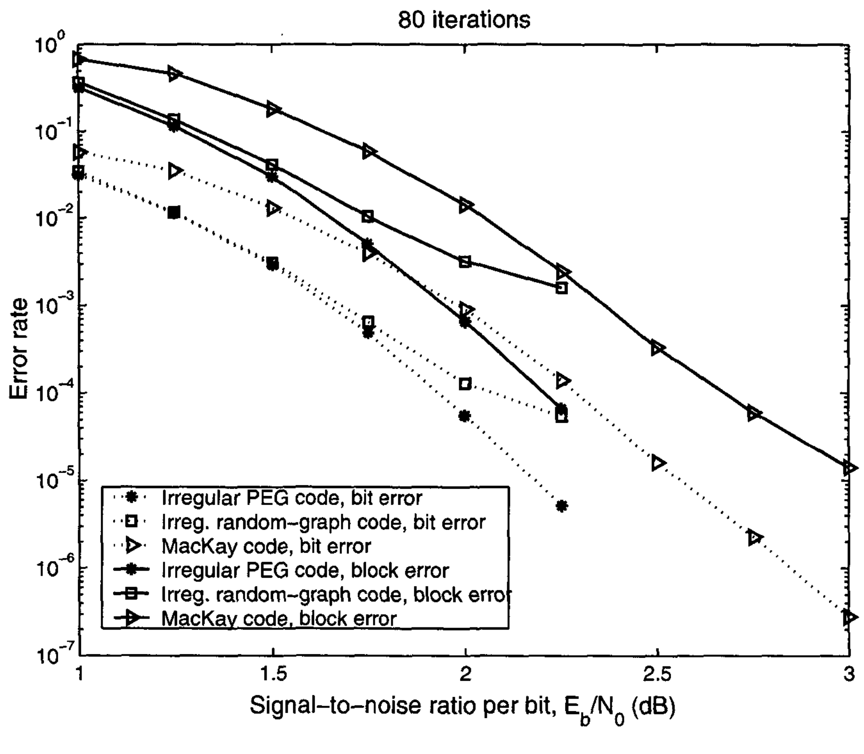


**Figure 3.12:** Bit- and block-error rates of the irregular PEG Tanner-graph code, irregular random-graph code, and MacKay's code (regular) with parameters $n = 1008, m = 504$. The symbol-node-degree distribution for the irregular PEG Tanner-graph code is $\Lambda(x) = 0.47532x^2 + 0.279537x^3 + 0.0348672x^4 + 0.108891x^5 + 0.101385x^{15}$.

construction with parameters $n = 504, m = 252$. Among these symbol-node distributions with maximum symbol-node degrees 4, 5, 7, 9, 11, 15, 30, 50, the one with maximum degree 15 achieves the best performance, which is essentially the same performance as that of the one optimized by the empirical Monte–Carlo approach, see Fig. 3.13. This confirms that even for short block lengths, the degree distributions designed by the density evolution approach are still nearly optimum if the PEG construction is used. Note that the degree distribution with maximum degree 50 has a very good "threshold," which is only 0.06 dB away from the Shannon capacity in the limiting case, but suffers from a significant degradation due to the short block length. Also plotted is the performance of LDPC codes based on random construction with the same degree distributions. Once again, we observe that the PEG construction significantly outperforms the random construction, particularly in the high-SNR region.

The Monte–Carlo approach together with the constrained downhill simplex method to design near-optimum degree distributions is indispensable for two reasons: first it confirmed

in the PEG algorithm.

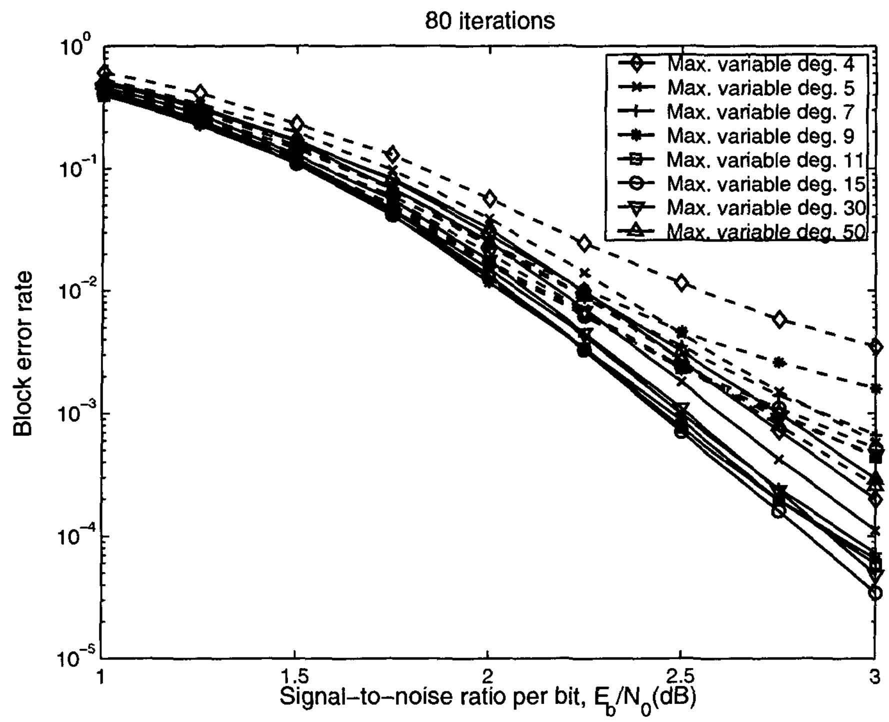


**Figure 3.13:** Block-error rates of irregular PEG (solid lines) and irregular random-graph codes (dashed lines) with density-evolution-optimized degree distributions; code parameters are $n = 504, m = 252$.

that the irregular degree distributions optimized by the density evolution approach match well with the PEG construction for short block lengths; this contrasts the old belief that irregular degree distributions do not work well with the random construction for short block lengths. Secondly, it can also be applied to the cases of non-binary LDPC codes (in Section 3.10) where a density-evolution approach is not available yet.

## 3.8 Linear-Time Encoding

The complexity per bit of iterative decoding using BP or SPA on a Tanner graph has been shown to be independent of the block length $n$, but the encoding complexity generally scales as $n^2$. Several publications address this issue, see [51,90–94] for instance. The most common idea is to exploit the sparseness of the parity-check matrix $H$ and its corresponding graph to obtain an efficient encoding format, namely, a triangular or almost triangular parity-check matrix. For example, in [92] the codeword $w$ and the parity-check matrix $H$ are partitioned into $w = [p, d]$ and $H = [H^p, H^d]$, respectively, such that

$$[H^p, H^d]w^T = 0. \tag{3.62}$$

It was found empirically that a good choice for $H^p$ is the $m \times m$ square matrix

$$H^p = \begin{pmatrix} 1 & & & 0 \\ 1 & 1 & & \\ & \ddots & \ddots & \\ 0 & & 1 & 1 \end{pmatrix}_{m\times m} . \tag{3.63}$$

The matrix $H^d$ is then created by constructing a random graph such that no 4-cycles are generated. Using Eq. (3.62) and Eq. (3.63), the parity-check bits $p = \{p_i\}$ can be computed by

$$p_1 = \sum_{j=1}^{n-m} h_{1,j}^d d_j \text{ and } p_i = p_{i-1} + \sum_{j=1}^{n-m} h_{i,j}^d d_j, \ i > 1, \quad (\text{mod}2), \tag{3.64}$$

where $d = \{d_i\}$ is the systematic part of the codeword, and $H^d = \{h_{i,j}^d\}$ is the $m \times (n-m)$ component of the partitioned parity-check matrix $H$. Clearly, the encoding process has become much simpler because the Gaussian elimination step is avoided. Moreover, computation and storage requirements in the encoder are also reduced because $H^d$ is sparse by design. This approach clearly guarantees a linear-time encoding, but in general may result in some loss in performance. The structure of Eq. (3.63) gives rise to a special pattern on a Tanner graph that was called the "zigzag" pattern in irregular repeat accumulate (IRA) codes [93].

We show that this simple idea can also be applied in a natural way to a PEG Tanner graph, rendering linear-time encoding possible while maintaining a large girth and equally good performance. To facilitate linear-time encoding, we partition the symbol node set $V_s$ into two disjoint subsets: a redundant subset $V_s^p$ and an information subset $V_s^d$. The redundant subset contains the first $m$ symbol nodes, which are redundant bits in a codeword and whose associated edges are prespecified as the simple "zigzag" pattern shown in Fig. 3.14 that corresponds to the square matrix $H^p$ in Eq. (3.62). The information subset $V_s^d$ contains the other $n - m$ symbol nodes, which correspond to information (systematic) bits in a codeword and whose associated edges are established by the PEG algorithm with the zigzag edges already in the graph. Note that the prescribed zigzag edges do not contain a single cycle before the PEG algorithm starts, thereby imposing no negative impact on the girth histogram.

One of the most important results on encoding of LDPC codes appears in [94] where it is proved that optimized LDPC codes can be encoded in linear time. In particular, one can choose a randomly-generated LDPC code with an optimized irregular degree sequence

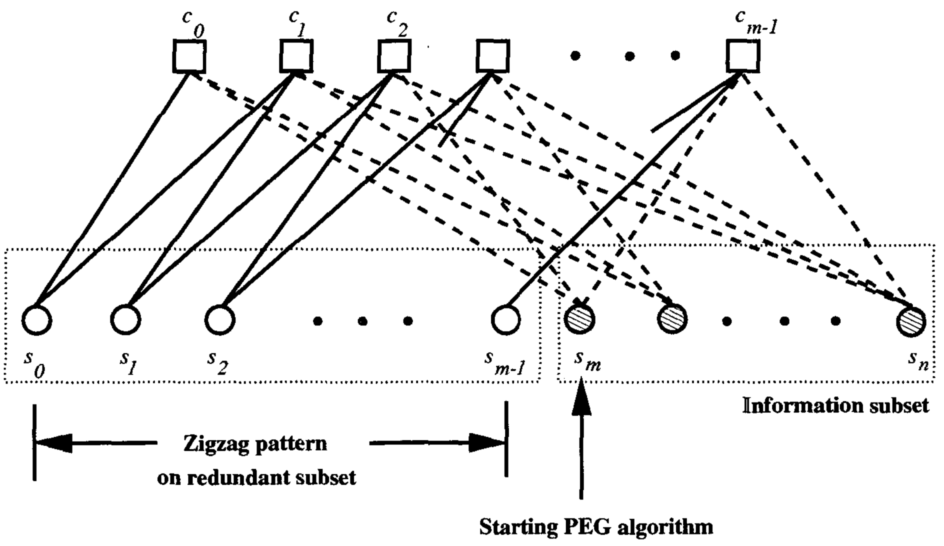


**Figure 3.14:** Linear-time-encodable PEG Tanner graph with zigzag pattern. The solid edges correspond to the zigzag pattern, which is specified before the PEG algorithm starts. The dashed lines correspond to the edges established by the PEG algorithm.

pair, and then the parity-check matrix of this code can be rearranged to have an (almost) upper-triangular structure that allows linear-time encoding. The significance of [94] lies in the existence proof of optimal LDPC codes with (almost) upper-triangular structure. Note that the rearrangement of a parity-check matrix does not change the LDPC code itself and the complexity of rearrangement is generally quadratic, it is thus preferable to construct such LDPC codes directly.

The PEG algorithm can be easily tailored to construct LDPC codes having (almost) triangular structure, good girth properties and optimum irregular degree sequence. For the sake of clarity we focus on parity-check matrices with upper-triangular structure. In this case the $m \times m$ component $H^p = \{h^p_{i,j}\}$ of the parity-check matrix can be written as

$$H^p = \begin{pmatrix} 1 & h^p_{1,2} & \cdots & h^p_{1,m} \\ 0 & 1 & & \\ \vdots & \vdots & \ddots & \vdots \\ & & 1 & h^p_{m-1,m} \\ 0 & \cdots & 0 & 1 \end{pmatrix}_{m\times m}, \tag{3.65}$$

and the parity bits are computed according to

$$p_i = \sum_{j=i+1}^{m} h^p_{i,j} p_j + \sum_{j=1}^{n-m} h^d_{i,j} d_j, \quad (\text{mod}2), \tag{3.66}$$

where Eq. (3.66) is computed recursively from $i = m$ to $i = 1$. By combining the PEG construction method with the constraint that the resulting $H$ matrix be of lower or upper triangular form, we obtain powerful, linear-time encodable, irregular LDPC codes with large girth and very good performance. As before, we partition the symbol node set $V_s$ into the redundant subset $V_s^p$ and the information subset $V_s^d$ containing the first $m$ symbol nodes (parity bits) and the other $n-m$ symbol nodes (systematic information bits), respectively. The edges of the symbol nodes are then established by means of the PEG algorithm while observing the special pattern in Eq. (3.65), so that a good girth histogram is obtained. As the procedure of establishing the edges of $n-m$ information bits follows the construction of edges of redundant subset $V_s^p$ and is exactly the same as that described in Section 3.3, we focus only on the modified PEG algorithm for constructing edges of $V_s^p$.

```
PEG Algorithm for Establishing Edges of $V_s^p$:
for $j = 0$ to $m-1$ do
begin
  for $k = 0$ to $d_{s_j} - 1$ do
  begin
    if $k = 0$
      $E_{s_j}^0 \longleftarrow$ edge $(c_j, s_j)$, where $E_{s_j}^0$ is the first edge incident to $s_j$. This edge
      corresponds to the '1' in the diagonal line of matrix $H^p$.
    else
      expanding a tree from symbol node $s_j$ up to depth $l$ under the cur-
      rent graph setting such that $\bar{\mathcal{N}}_{s_j}^l \cap \{c_0, c_1, \dots, c_{j-1}\} \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} \cap$
      $\{c_0, c_1, \dots, c_{j-1}\} = \varnothing$, or the cardinality of $\mathcal{N}_{s_j}^l$ stops increasing, then
      $E_{s_j}^k \longleftarrow$ edge $(c_i, s_j)$, where $E_{s_j}^k$ is the $k$-th edge incident to $s_j$ and $c_i$ is
      one check node picked from the set $\bar{\mathcal{N}}_{s_j}^l \cap \{c_0, c_1, \dots, c_{j-1}\}$ having the
      lowest check-node degree.
  end
end
```

As an example we consider an irregular PEG Tanner-graph code whose parity-check matrix is forced into an upper triangular form. Fig. 3.15 compares the bit and block-error rates of the irregular PEG Tanner-graph code presented in Section 3.7.2, an irregular PEG Tanner-graph code with a parity-check matrix in upper diagonal form, and MacKay's code, all with parameters $n = 1008, m = 504$. The symbol-node-degree distribution for both irregular PEG Tanner-graph codes is $\Lambda(x) = 0.47532x^2 + 0.279537x^3 + 0.0348672x^4 + 0.108891x^5 + 0.101385x^{15}$. When the parity check matrix is forced into an upper triangular form there is only one symbol

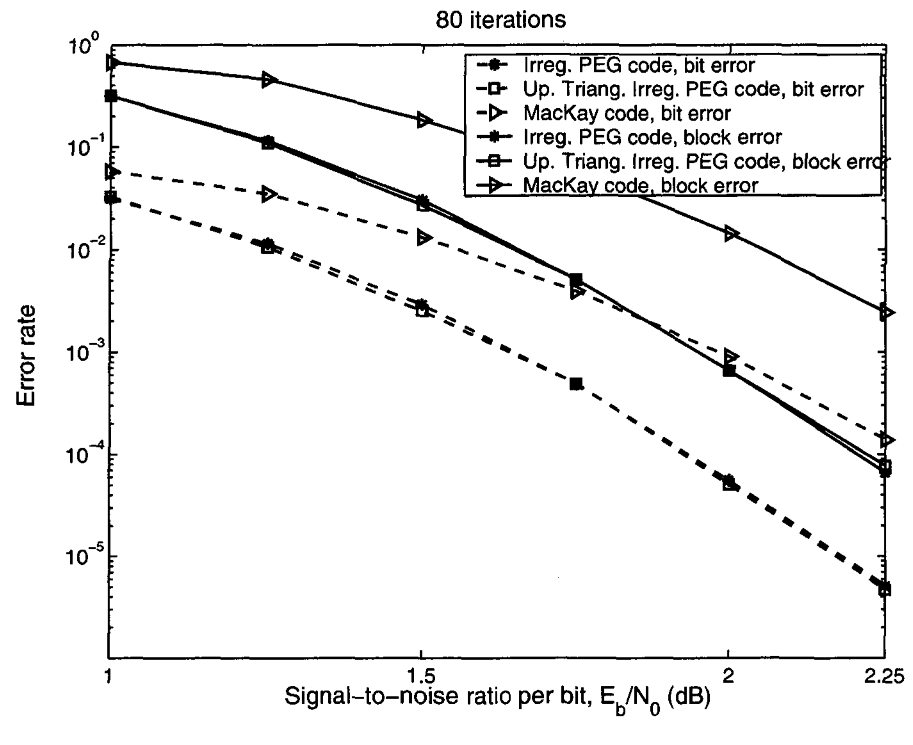


**Figure 3.15:** Bit- and block-error rates of an irregular PEG Tanner-graph code, an upper-triangular PEG Tanner-graph code, and MacKay's code.

node of degree 1. As can be seen, the two irregular codes designed according to the PEG algorithm have essentially the same performance and are about 0.5 dB better than MacKay's rate-1/2 ($n = 1008, m = 504$) code. Hence it is shown that linear-time encoding can be achieved without noticeable performance degradation under the PEG construction.

## 3.9 Performance of PEG Codes versus Turbo Codes

Irregular LDPC codes with large block lengths have been shown to outperform Turbo codes [10,54]. However, it appears to be a common belief that for relatively short block lengths Turbo codes outperform LDPC codes. Here we take a closer look at the performance of linear-time-encodable PEG LDPC codes (of relatively short block lengths) as compared with that of Turbo codes in the CDMA2000 standard. The turbo encoder consists of two systematic, recursive, 8-state convolutional encoders concatenated in parallel, with an interleaver. The transfer function for the encoder is $G(D) = \left[1 \ \frac{n_0(D)}{d(D)} \ \frac{n_1(D)}{d(D)}\right]$, where $d(D) = 1{+}D^2{+}D^3$, $n_0(D) = 1{+}D{+}D^3$, and $n_1(D) = 1 + D + D^2 + D^3$. Depending on different puncturing patterns, rate-1/2, 1/3, and 1/4 codes can be obtained. Note that for each block a total of six bits are used to terminate the two convolutional encoders. For a detailed description of the standardized Turbo interleaver, we refer the reader to [110]. The LDPC codes and Turbo codes are compared at exactly the same rate and block length. For decoding Turbo codes, 12 iterations with the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [111] in log-domain are employed; for decoding LDPC codes, 80 iterations using the low-complexity SPA of [107] are used so that the decoding complexity remains approximately the same. In designing linear-time-encodable binary LDPC codes, we take the irregular degree distribution as $\Lambda(x) = 0.477081x^2 + 0.280572x^3 + 0.0349963x^4 + 0.0963301x^5 + 0.0090884x^7 + 0.00137443x^{14} + 0.100558x^{15}$, chosen from [10].

Fig. 3.16 and Fig. 3.17 show the bit- and block-error rates over the AWGN channel for linear-time-encodable PEG LDPC codes and the CDMA2000 Turbo codes, with rate $\simeq 1/2$ and block lengths of 1024 and 2048, respectively. Both kinds of codes have comparable performance at exactly the same block length and rate; in the low-SNR region, the CDMA2000 Turbo codes are slightly better, whereas in the high-SNR region, the LDPC codes outperform the CDMA2000 Turbo codes. The advantage of the LDPC codes becomes more pronounced if the figure of merit is the block-error rate.

In conclusion, our linear-time-encodable PEG LDPC codes offer comparable performance to the standardized Turbo codes with essentially the same encoding/decoding complexity. It is anticipated that if better degree distributions are available from the analysis of finite block-

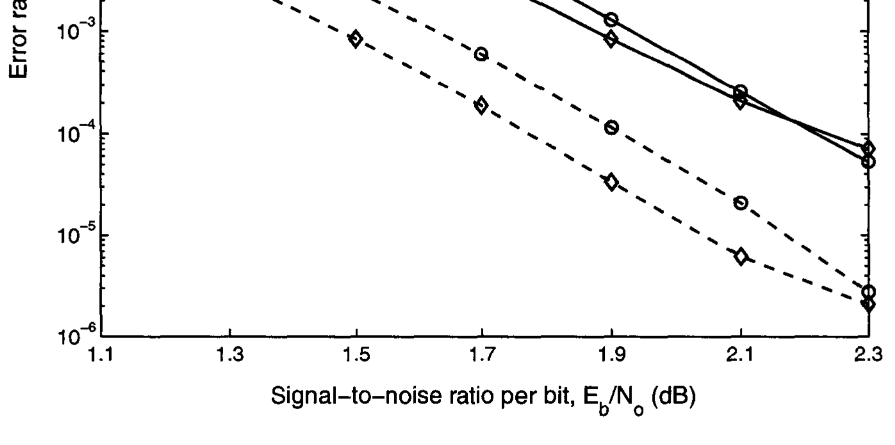


**Figure 3.16:** Bit- and block-error rates of a linear-time-encodable PEG LDPC code and a CDMA2000 turbo code, with block length 1024 and rate (512-6)/1024.

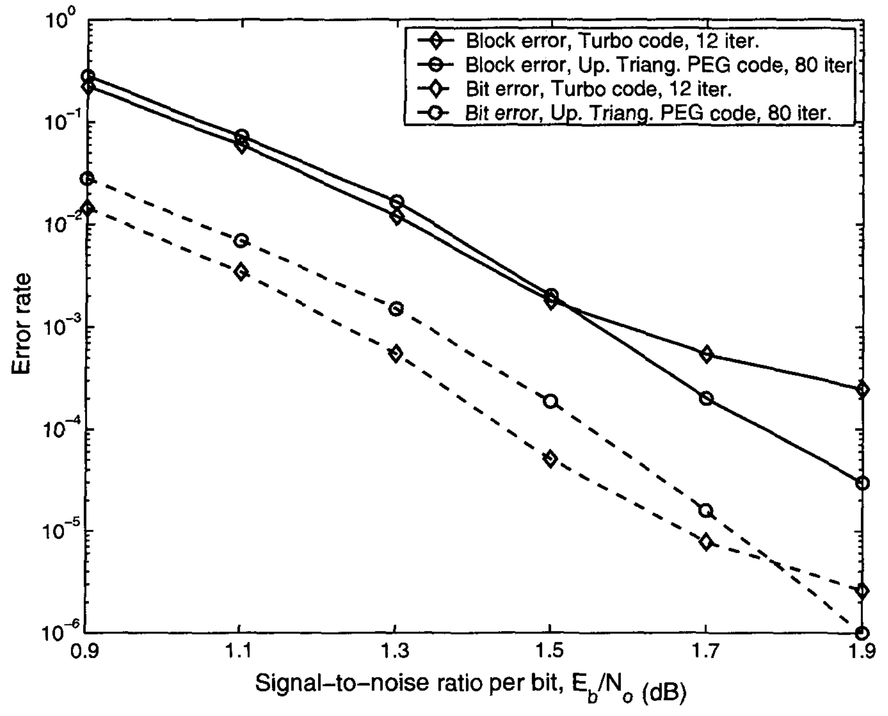


**Figure 3.17:** Bit- and block-error rates of a linear-time-encodable PEG LDPC code and a CDMA2000 turbo code, with block length 2048 and rate (1024-6)/2048.

length LDPC codes [102], the performance of such LDPC codes can be further improved. To close this section, we point out that, as the iterative decoding of LDPC codes is fully parallelizable and could be accomplished at significantly greater speeds, the linear-time-encodable PEG LDPC codes show promise for extremely-high-speed applications such as magnetic recording, optical communication, and broadband wireline/wireless access technologies.

## 3.10 PEG Tanner-Graph Codes over GF($2^b$)

So far we have primarily considered binary LDPC codes represented by binary parity-check matrices or their corresponding bipartite graphs constructed using the PEG algorithm. These codes can easily be generalized to finite fields GF($q$) in the same natural way as in [89, 95], i.e., by allowing the symbol nodes to assume values from the finite field. Specifically, fields with $q = 2^b$, $b > 1$, elements that are important in practical applications, and codes over GF($2^b$) with linear constraints defined on non-binary sparse $H$ matrices are considered in this section. In GF($2^b$) each encoded symbol $w_i$ consists of $b$ binary bits. Since a symbol from the field GF($q$), $q = 2^b$ for some integer $b$, may be represented as a binary string of $b$ bits, we can use such codes with binary-input channels, transmitting one $q$-ary symbol for every $b$

uses of the binary channel. The decoder interprets $b$ bits $(y_0, y_1, \dots, y_{b-1})$ from the channel as a single $2^b$-ary symbol and sets the prior information of that symbol by assuming a product distribution for the values of each constituent bit. Namely

$$f^x := \prod_{i=0}^{b-1} f_{y_i}^{x_i^b}$$

where $f_{y_i}^{x_i^b}$ is the likelihood the $i$th constituent bit is equal to $x_i^b$, where $(x_0^b, x_1^b, \dots, x_{b-1}^b)$ is the binary representation of the transmitted symbol $x$.

The motivation that we use GF($q$) LDPC codes with binary-input channels is based on the following arguments: Iterative decoding performance of an LDPC code is in general dominated by two competing factors. One is the Hamming weight spectrum which requires the density of parity-check matrix to be higher; the other is the performance loss due to iterative decoding for which lower density of parity-check matrix is favorable. The two conflicting requirements must be well balanced, particularly for short block lengths. The binary interpretation of codes over GF($q$) provides interesting possibilities for designing binary codes, as the binary minimum distance can be much larger than the symbol minimum distance [112, p. 112-113], [113, 114].

We describe the construction of PEG GF($q$) LDPC codes briefly: given the number of symbol nodes $n$, the number of parity-check nodes $m$, and the symbol-node-degree sequence of the graph, the PEG algorithm is started first, in exactly the same manner as the binary case, such that the placement of a new edge on the graph has as small an impact on the girth of the graph as possible. In this way a PEG Tanner graph is obtained that not only has a large girth but also a good girth histogram. To form a GF($q$) parity-check matrix, the positions of nonzero entries are determined by the PEG Tanner graph, whereas the values of the nonzero entries of the parity-check matrix are selected randomly from a uniform distribution among nonzero elements of GF($q$).

Table 3.1 shows optimized rate-1/2 irregular PEG Tanner-graph codes over GF($2^b$) and their corresponding symbol-node-degree distributions. The optimization of the degree sequences was accomplished with the variant of the "downhill simplex"method described in Section 3.6. We compare codes having a block length of $n$ symbols over GF($2^b$) with binary codes of length $nb$ bits.

Fig. 3.18 shows the performance of the irregular PEG Tanner-graph codes over a binary-input Gaussian channel. Five codes of rate 1/2 over GF(2), GF(8), GF(16), GF(32), and GF(64) are shown. All codes correspond to block lengths of 1008 bits (except the irregular PEG Tanner-graph code over GF(32), which has a block length of 202 symbols or 1010 bits). Also shown is the performance of the rate-1/2, $n = 1008$, $m = 504$ binary MacKay code as well

**Table 3.1:** Optimized symbol-node-degree distributions for rate-1/2 PEG codes over GF($2^b$). The block length in binary bits is $nb$.

| Galois field | $(n, m)$ | Symbol-node-degree distribution | Ave. symbol degree |
|---|---|---|---|
| GF(2) | (1008, 504) | $0.47532x^2 + 0.279537x^3 + 0.0348672x^4 + 0.108891x^5$ $+0.101385x^{15}$ | 3.994 |
| GF(8) | (336, 168) | $0.643772x^2 + 0.149719x^3 + 0.193001x^4 + 0.013508x^5$ | 2.5762 |
| GF(16) | (252, 126) | $0.772739x^2 + 0.102863x^3 + 0.113797x^4 + 0.010601x^5$ | 2.3623 |
| GF(32) | (202, 101) | $0.84884x^2 + 0.142034x^3 + 0.009126x^4$ | 2.1603 |
| GF(64) | (168, 84) | $0.94x^2 + 0.05x^3 + 0.01x^4$ | 2.07 |

as the sphere-packing bound for this block length. As can be seen, an improvement of 0.25 dB is obtained by moving from binary to GF($2^6$) PEG construction. Furthermore, the overall gain of the GF($2^6$) PEG construction compared with the binary MacKay code is approximately 0.75 dB. Finally, the rate-1/2 irregular PEG code over GF($2^6$) shows a block-error rate $< 10^{-4}$

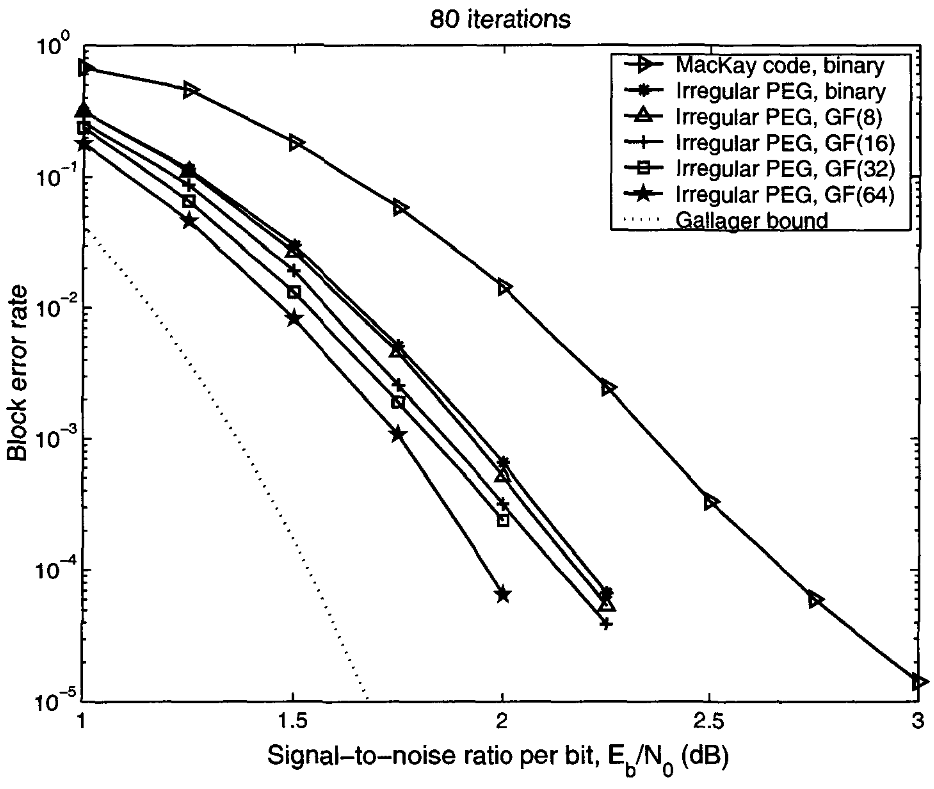


**Figure 3.18:** Bit- and block-error rates of irregular LDPC codes over GF(2), GF(8), GF(16), GF(32), and GF(64), based on PEG Tanner graph with the parameters given in Table 3.1.

at $E_b/N_0$ 2 dB, i.e., a performance that is only 0.4 dB from the Shannon–Gallager–Berlekamp sphere-packing bound[11] [see Appendix C] of a binary-input AWGN channel [62,115], which appears to be the best-known performance at this block length to date.

The performance results indicate that the PEG Tanner-graph codes over higher-order fields outperform the binary ones. Furthermore, owing to the PEG construction that aims at large girth as well as good irregular degree sequence we have observed a *monotonic* improvement with increasing field order, which contrasts with the observations of [95]. Interestingly enough, the irregularity feature seems to be unnecessary if the higher-order field is sufficiently large, and the optimum graph tends to favor a regular one of degree-2 in all symbol nodes, termed cycle graph or cycle code, which is the lowest density graph in the context of iterative decoding.

## 3.11 Summary

A general method for constructing Tanner graphs having large girth and minimum distance has been presented. The main principle in this construction is to optimize the placement of a new edge, connecting a particular symbol node to specific check node on the graph such that the largest possible local girth is achieved. In this way, the underlying graph grows in an edge-by-edge manner, optimizing each local girth, and is thus referred to as a progressive edge-growth (PEG) Tanner graph.

Upper and lower bounds on the girth of a PEG Tanner graph have been derived. These bounds depend on the number of symbol nodes and on that of check nodes, as well as on the maximum values of the symbol- and check-node degrees of the underlying graph. By comparing the lower bound and the upper bound on the girth, we claim that the girth of a PEG Tanner graph meets or surpasses an analogy of the Erdös–Sachs bound. In addition, a lower bound on the minimum distance of binary LDPC codes defined on PEG Tanner graphs has also been derived.

The advantages of the PEG construction over a random one are twofold. First, it yields a much better girth distribution, thereby facilitating the task of the BP or SPA during the iterative decoding process. Second, it leads to (or guarantees) a meaningful lower bound on the minimum distance, providing insight into the performance of the code at high signal-to-noise ratios. Simulation results confirmed that using the PEG algorithm for constructing short-block-length LDPC codes results in a significant improvement compared to randomly

[11]Bear in mind, however, that no undue significance should be attached to this 0.4 dB gap, as at this block length the formula for computing the Shannon–Gallager–Berlekamp bound might not be sufficiently precise.

constructed codes.

We have described an empirical Monte–Carlo approach using a variant of the "downhill simplex" search algorithm to design irregular PEG graphs for small codes with fewer than a thousand bits, which can be viewed as a complementary tool for the asymptotic analysis "density evolution". We found that even for small block lengths such as $n = 504$, there is a good degree distribution from the density evolution approach that works perfectly with the PEG construction. Simulation results showed that in conjunction with optimum symbol-node-degree distributions (obtained from either the empirical or the density evolution approach) our PEG construction yields the best binary LDPC codes at short block lengths to date. Linear-time-encodable LDPC codes have also been constructed by slightly modifying the PEG algorithm to yield a Tanner graph containing a zigzag pattern or (almost) triangular format. This easy encoding property is attained without noticeable performance degradation. We have also demonstrated that in third-generation high-speed wireless data services, the linear-time-encodable PEG LDPC codes offer comparable performance to standardized Turbo codes with essentially the same encoding/decoding complexity.

Finally, regular and irregular LDPC codes have been generalized by using the PEG construction but allowing the symbol nodes to take values over higher-order finite fields. This work confirms that by moving to higher-order fields short-block-length codes can be constructed that operate very close to the Gallager bound when decoded with the sum-product algorithm. We reported a short block-length (1008 bit) rate-1/2 irregular PEG LDPC code over $GF(2^6)$ with a block error rate $< 10^{-4}$ at $E_b/N_0 = 2$ dB, which appears to have the best-known performance at this short block length to date.

# Chapter 4

# Cycle Tanner-Graph Codes

## 4.1 Introduction

Gallager's binary LDPC codes have been shown to achieve near-Shannon-limit performance when decoded using the belief-propagation (also called sum-product) algorithm [4, 10, 71]. Binary LDPC codes may be generalized to finite Galois fields GF($q$), as first suggested by Tanner [5] in the context of codes on graphs and investigated empirically by Davey and MacKay [89, 95] over the binary-input AWGN channel. Sparse graph codes over GF($q$) can alternatively be defined in terms of a nonsystematic LDPC matrix $H$ whose nonzero entries are selected randomly from nonzero elements of GF($q$) with uniform probability $p/(q-1)$, where $p$ is a small number referred to as the *density* of $H$. As a symbol from the field GF($q$), $q = 2^b$, for some integer $b$, may be represented as a binary string of $b$ bits, one can use such codes with binary-input-constrained channels, transmitting one $q$-ary symbol for every $b$ uses of the binary channel. The decoder interprets $b$ bits $(y_0, y_1, \cdots, y_{b-1})$ from the channel as a single $2^p$-ary symbol and sets the prior information of that symbol by assuming a product distribution for the values of each constituent bit.

The binary interpretation of codes over GF($2^b$) provides interesting possibilities for designing binary error-correcting codes [112]. In particular, binary interpretations of RS codes over GF($2^b$) have been investigated in [113]. In [114, 116, 117] some famous nonlinear binary codes were described as binary interpretations of linear codes over the ring $\mathbb{Z}_4$. These codes include the Nordstrom–Robinson, Preparata and Kerdock codes, which are of theoretical interest but often difficult to decode.

For sparse graph codes over GF($2^b$) under iterative decoding, it was empirically observed in [95] that by moving from binary to GF($2^b$) while keeping the same binary block length, one can obtain a performance improvement of about 0.3 dB. It was also noted in [95] that there is not always a monotonic improvement with increased field order over the binary-input AWGN channel. In [97] we demonstrate a monotonic improvement with increased field order using the PEG construction and a good irregular degree sequence optimized by the Monte-Carlo approach. Very interestingly, it turns out that the irregularity feature seems to be unnecessary if the higher-order field grows sufficiently large, and the optimum graph tends to favor a regular feature of degree-2 in all symbol nodes, i.e., $d_s = 2$. Abusing notation, we denote the ensemble of regular sparse Tanner graphs having $d_s = 2$ by TG(2, $d_c$).

Binary LDPC codes defined on TG(2, $d_c$) were dismissed by Gallager and considered not to be truly practical because their minimum distance grows only logarithmically with block length $n$ [4, Theorem 2.5]. Nevertheless they are quite interesting from the point of view of iterative decoding because their simple structure makes their analysis easier than that of the other low-density codes, and they are variously called "graph-theoretic," "circuit," or "cycle" codes [118, Section 5.8], [119]. Among all codes that are amenable to iterative decoding, codes on TG(2, $d_c$) are therefore the codes with lowest possible density in their parity-check matrices, and the underlying Tanner graph TG(2, $d_c$) has *sparsest* connectivity in the sense of iterative decoding.

TG(2, $d_c$) has long been understood as an ensemble of sparse graphs for which the iterative decoding is close to or essentially equal to the optimum decoding, e.g. see [46, pp. 54-57]. It has been shown in [104,120,121] that the threshold probability for ML or typical-pair decoding of an ensemble of binary LDPC codes on TG(2, $d_c$) over the binary symmetric channel (BSC) is given by

$$p^*(d_c) = \frac{1}{2}\left(1 - \sqrt{1 - \frac{1}{(d_c - 1)^2}}\right). \tag{4.1}$$

Later it was shown in [122] that Eq. (4.1) is exactly the threshold of iterative decoding, indicating that the asymptotic performance of iterative and optimum decoding coincide.

In this chapter, we investigate the Hamming weight spectrum of the binary interpretation of ensembles of random GF($2^b$) codes of sparse Tanner graphs. Our results visually show that if the field order $b$ is sufficiently large, the Hamming weight spectrum of the sparse-graph code ensemble asymptotically approaches the classical binomial distribution of the Shannon equiprobable random ensemble. This observation justifies the advantages of GF($2^b$) codes defined over the cycle Tanner graph TG(2, $d_c$) in both senses: minimum distance and iterative decoding. More specifically, one can minimize the performance loss due to iterative decoding by resorting to the sparsest Tanner graph TG(2, $d_c$), and, simultaneously improve the code itself and let it behave like an equiprobable random code by moving to a sufficiently large field. Therefore, cycle Tanner-graph codes defined over GF($2^b$), with sufficiently large $b$, are heuristically both "good codes for optimum decoding" and "good codes for iterative decoding".

The close connection of the iterative decoding of GF($2^b$) codes on sparse Tanner graphs to the Kikuchi approximation is also outlined. It is worth mentioning that the Kikuchi approximation often yields better performance than the Bethe approximation does in the presence of cycles in statistical inference problems, and the stationary point of the Bethe approximation is known to be equivalent to the fixed point of ordinary belief propagation (BP) or SPA.

## 4.2 Weight-Spectrum Analysis

In this section we investigate the average weight spectrum of a random ensemble of regular sparse Tanner graph codes over GF($q$). We are particularly interested in the binary interpretation of GF($q$) codes on sparse Tanner graphs when they are applied to binary channels. Formally, such codes are defined as follows: given some $p$, $0 < p < 1/2$, we choose the entries of each column in the matrix $H$ independently, so that 0 is attained with probability $1 - p$, and each nonzero element in GF($q$) with equal probability $p/(q-1)$. Note that if $q = 2$, this reduces to the random ensemble of binary LDPC codes in Section II-A.(3) of [71].

Define the symbol distance between two codewords in a linear code over GF($q$) as the number of positions in which the codewords differ. The symbol weight of a codeword is the number of nonzero digits or the distance from the all-zero codeword. The symbol distance function $N_s(j)$ of a code is defined as the number of codewords of symbol weight $j$. It follows from the group properties of such a code that $N_s(j)$ is the number of codewords at distance $j$ from any given codeword.

**Theorem 4.1** *Let $\overline{N_s(j)}$ be the average number of codewords of symbol weight $j$ in a code averaged over the sparse random ensemble $H$ based on some $p$ and defined over GF($q$). The parity-check codes are of symbol length $n/\log_2 q$ ($n$ in binary bits) and rate $R$. Then for $j$, $0 < j \le n/\log_2 q$,*

$$\overline{N_s(j)} = \alpha \binom{\frac{n}{\log_2 q}}{j} (q-1)^j \left[ \frac{q-1}{q}(1 - \frac{qp}{q-1})^j + \frac{1}{q} \right]^{\frac{n(1-R)}{\log_2 q}}, \tag{4.2}$$

*where $\alpha$ is the normalizing constant such that $\sum_j \overline{N_s(j)} = 2^{nR}$ and $\overline{N_s(0)} = 1$.*

*Proof.* Fix some nonzero vector $V = [v_0, v_1, \dots, v_{\frac{n}{\log_2 q}-1}] \in [\mathrm{GF}(q)]^{\frac{n}{\log_2 q}}$, with exactly $j$ nonzero coordinates, i.e. a symbol weight of $j$. Without loss of generality assume that the first $j$ coordinates of $V$ are nonzero. Let $P_j$ be the probability that $\sum_{i=1}^{j} v_i h_{k,i} = 0$, where each $h_{k,i}$ is the entry of the $k$-th row of the matrix $H$ and is thus chosen according to the uniform distribution of $p/(q-1)$ among nonzero elements of GF$(q)$. As there are in total $n(1-R)/\log_2 q$ independent check equations, the probability that $V$ is a valid codeword is given by $P_j^{\frac{n(1-R)}{\log_2 q}}$.

For fixed $v_i$, the following recursion [1] holds for $P_j$:

$$P_j = P_{j-1}(1-p) + \frac{(1-P_{j-1})p}{q-1}\,, \tag{4.3}$$

with the initial condition $P_0 = 1$. Let $Q_j = P_j - 1/q$. It follows from Eq. (4.3) that

$$Q_j = Q_{j-1}\left(1 - \frac{qp}{q-1}\right)\,, \tag{4.4}$$

with $Q_0 = (q-1)/q$. Hence we obtain

$$Q_j = \frac{q-1}{q}\left(1 - \frac{qp}{q-1}\right)^j\,, \tag{4.5}$$

and, consequently,

$$P_j = \frac{q-1}{q}\left(1 - \frac{qp}{q-1}\right)^j + \frac{1}{q}\,. \tag{4.6}$$

The number of vectors $V$ with exactly $j$ nonzero coordinates is given by

$$\binom{\frac{n}{\log_2 q}}{j}(q-1)^j\,.$$

The average number of codewords of symbol weight $j$ is then given by

$$\overline{N_s(j)} = \alpha\binom{\frac{n}{\log_2 q}}{j}(q-1)^j\left[\frac{q-1}{q}(1-\frac{qp}{q-1})^j + \frac{1}{q}\right]^{\frac{n(1-R)}{\log_2 q}}\,. \tag{4.7}$$

Finally note that there is only one all-zero codeword and that the number of all codewords should sum up to $q^{\frac{nR}{\log_2 q}} = 2^{nR}$. This completes the proof. ■

**Lemma 4.1** *Let $\overline{N(k)}$ be the average number of codewords of Hamming weight $k$ in a code averaged over the sparse random ensemble $H$ based on some $p$ and defined over GF(2). Then, for $0 < k \leq n$,*

$$\overline{N(k)} = \alpha\binom{n}{k}\left[\frac{1}{2}(1-2p)^k + \frac{1}{2}\right]^{n(1-R)}\,, \tag{4.8}$$

*where $\alpha$ is the normalizing constant such that $\sum_k \overline{N(k)} = 2^{nR}$ and $\overline{N(0)} = 1$.*

[1]This recursion has been used to analyze the rank properties of a sparse random $n \times n$ matrix over GF$(q)$ in [123].

*Proof*: This is obtained by inserting $q = 2$ into Eq. (4.2). ■

**Lemma 4.2** *Let $\overline{N(k)}$ be the average number of codewords of Hamming weight $k$ in a code averaged over the Shannon equiprobable random ensemble $H$ defined over GF(2). Then for $0 < k \le n$,*

$$\overline{N(k)} = \binom{n}{k} 2^{-n(1-R)} . \tag{4.9}$$

*Proof*: This is obtained by inserting $q = 2, p = 1/2$ into Eq. (4.2). This result first appeared in [4] and is known as the binomial distribution. ■

**Theorem 4.2** *Let $\overline{N(k)}$ be the average number of codewords of Hamming weight $k$ in a code averaged over the sparse random ensemble $H$ based on some $p$ and defined over $GF(q)$, $q > 2$. The parity-check codes are of symbol length $n/\log_2 q$ ($n$ in binary bits) and rate $R$. Then for $0 < k \le n$,*

$$\overline{N(k)} = \beta \sum_{j \ge k/\log_2 q}^{\min\{k, n/\log_2 q\}} \overline{N_s(j)} \cdot \frac{\binom{j \log_2 q}{k} [1 - (1 - \frac{k}{j \log_2 q})^{\log_2 q}]^j}{(q-1)^j} , \tag{4.10}$$

*where $\beta$ is the normalizing factor such that $\sum_k \overline{N(k)} = 2^{nR}$, and $\overline{N(0)} = 1$.*

*Proof*: The sparse random ensemble of parity-check codes over GF($q$) of length $n/\log_2 q$ and rate $R$ can be visualized as an ensemble of binary codes of length $n$ and rate $R$ when used with binary channels, thus there are $2^{nR}$ codewords in total, leading to $\sum_k \overline{N(k)} = 2^{nR}$. It is clear that the number of all-zero codewords is unique, i.e. $\overline{N(0)} = 1$.

Suppose a valid codeword has a symbol weight of $j$, $0 < j \le n/\log_2 q$. We need to evaluate the probability that this codeword contains $k$, $j \le k \le j \log_2 q$, 1's in its binary representation. Without loss of generality, assume that each nonzero symbol in the codeword is uniformly distributed among the nonzero elements of GF($q$). In total, there are $(q-1)^j$ patterns the $j$ nonzero symbols can assume. On the other hand, there are $\binom{j \log_2 q}{k}$ possibilities that the 1's in their binary representation sum up to exactly $k$, of which only a fraction, i.e. $[1 - (1 - \frac{k}{j \log_2 q})^{\log_2 q}]^j$, satisfy the condition that each symbol be nonzero. Accordingly, the probability that a codeword of symbol weight $j$ contains $k$ 1's in its binary representation is given by

$$\frac{\binom{j \log_2 q}{k} [1 - (1 - \frac{k}{j \log_2 q})^{\log_2 q}]^j}{(q-1)^j} .$$

From this, we obtain Eq. (4.10). ■

Fig. 4.1 shows the Hamming weight spectra of binary random ensembles with different block lengths but keeping the same density and rate. As the block length increases, the Hamming weight spectrum of the binary *sparse* random ensemble closely approaches that of the Shannon *equiprobable* random ensemble — the binomial distribution.

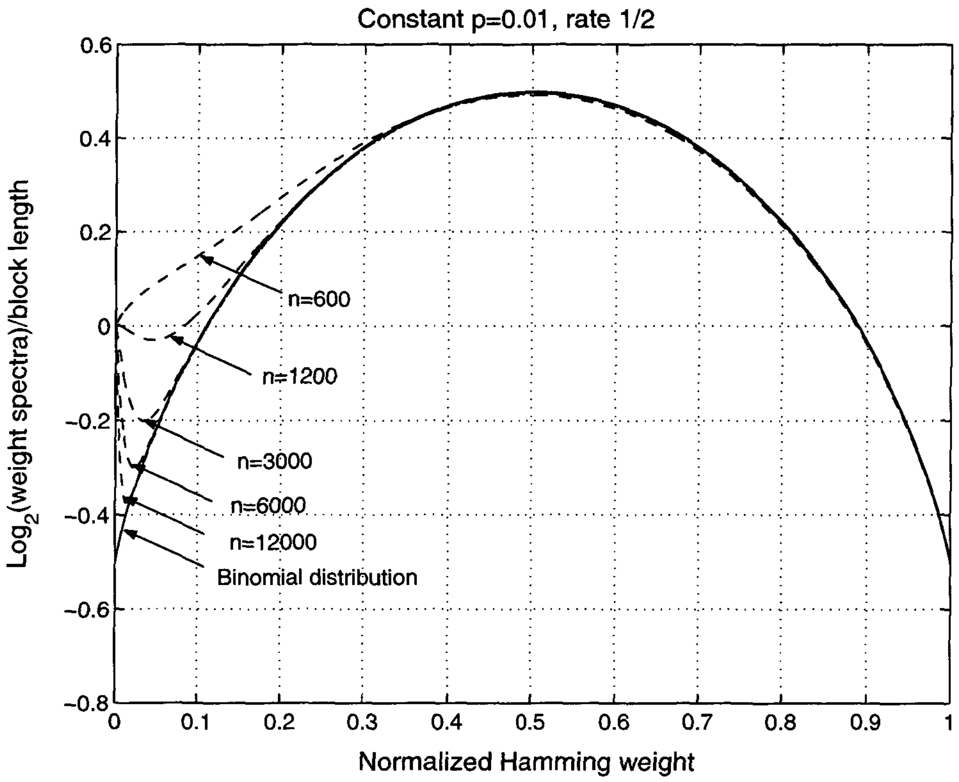


**Figure 4.1:** Hamming weight spectra of binary sparse random ensembles with different block lengths $n$.

Fig. 4.2 shows the Hamming weight spectra of binary random ensembles having different densities, from $p = 0.005$ to $p = 0.5$. It can be seen that, as the density increases, an LDPC code closely approaches the equiprobable binary random code, which is the best code we can imagine but unfortunately is computationally infeasible to decode [61]. In contrast, an LDPC code can be decoded iteratively with the SPA whose complexity is essentially linear to the block length. On the one hand, one would like to increase the density of an LDPC code such that it can closely approach the best code; on the other hand, iterative decoding requires the density of the underlying Tanner graph to be small enough so that the suboptimum iterative decoding comes close to the optimum decoding. These two conflicting requirements must be well balanced, and this poses significant design challenges, particularly in the case of relatively short block lengths.

Fig. 4.3 shows the Hamming weight spectra of low-density random ensembles over GF(2),

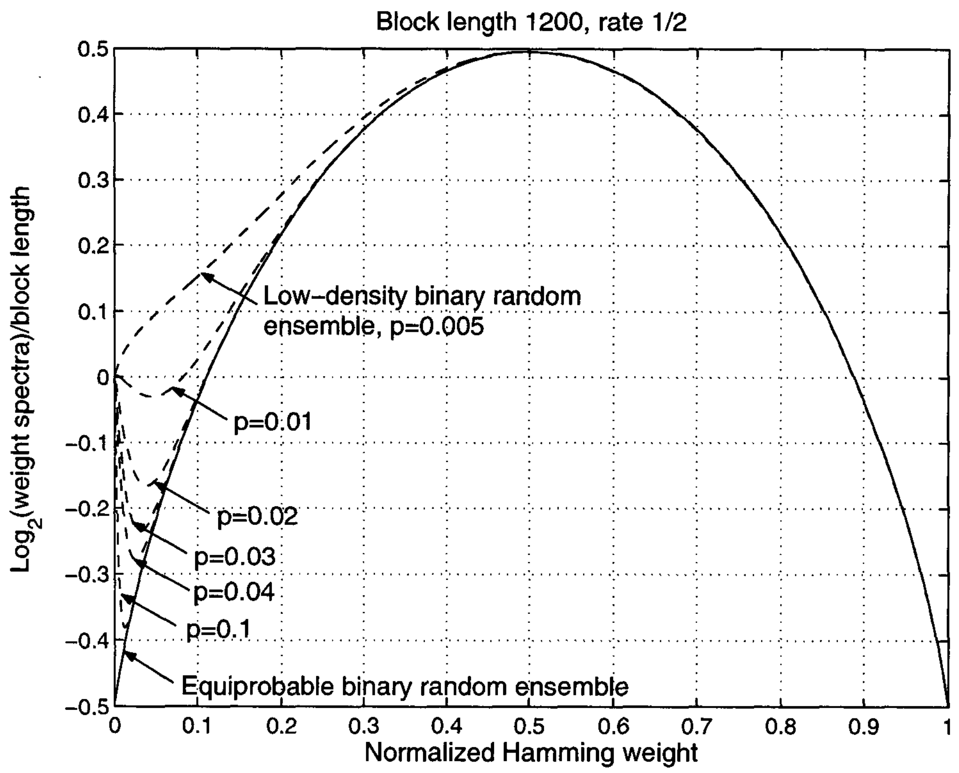


**Figure 4.2:** Hamming weight spectra of binary random ensembles as a function of density $p$.

GF($2^2$), GF($2^3$), GF($2^4$), GF($2^6$), and GF($2^{10}$) with the same block length of $n = 1200$ bit, and the same density of $p = 0.005$. Also plotted is the spectrum of the equiprobable binary random ensemble. As the size of the Galois field increases, the Hamming weight spectrum closely approaches that of the equiprobable binary random ensemble, while keeping the same (binary) block length, rate, and density of the parity-check matrix. Fig. 4.3 indicates that by fixing the density of Tanner graphs (parity-check matrices), one can always improve the Hamming weight spectrum by consistently increasing the field order while keeping the same binary block length. This allows the use of extremely sparse graphs of codes which previously were believed to be too weak in the sense of minimum distance, particularly the GF($2^b$) codes on the cycle Tanner graph TG($2, d_c$). It is well-known that the sparser the underlying Tanner graph, the smaller the deleterious effect of cycles in the graph and the closer the iterative decoding approaches the optimum decoding, therefore the binary interpretation of GF($2^b$) cycle Tanner-graph codes, if $b$ is sufficiently large, would perform very well under iterative decoding, as opposed to the conventional binary LDPC codes of the same block length. It has been reported in [97] that a GF($2^6$) (almost) cycle Tanner-graph code, with block length 1008 and rate 1/2, achieves the best performance (under iterative decoding) known to date, beating its counterparts of turbo codes and binary LDPC codes of the same (binary) block length and rate.

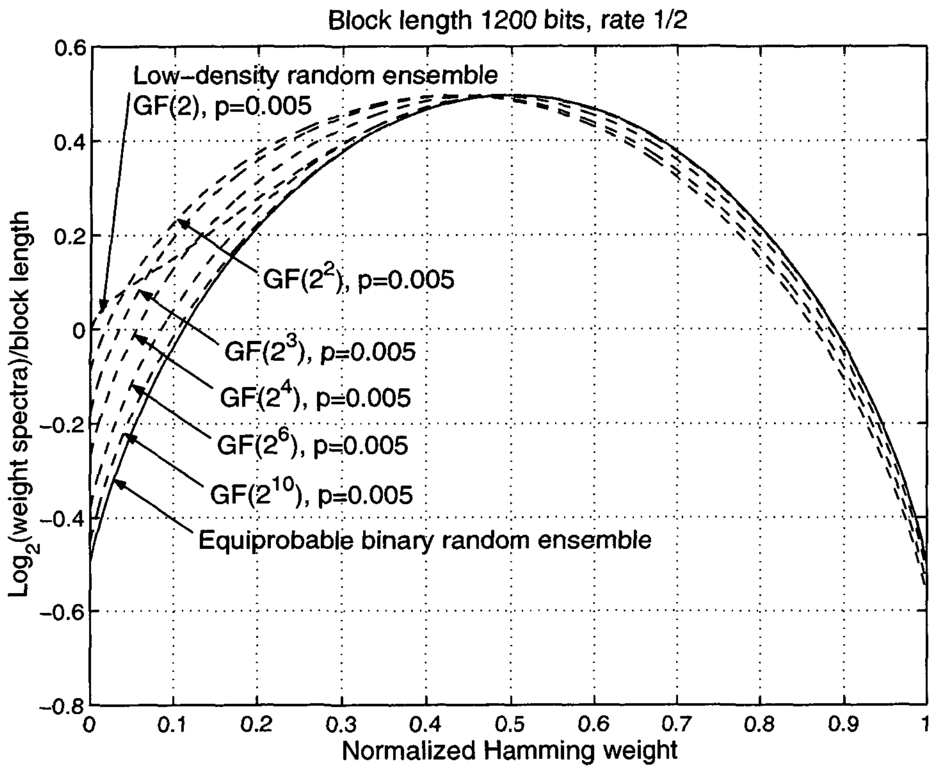


**Figure 4.3:** Hamming weight spectra of random ensembles over GF($q$) with the same block length $n$ of 1200 bits and the same density $p$ of 0.005.

## 4.3 A Perspective on Iterative Decoding from the Kikuchi Approximation

The iterative decoding of GF($2^b$) graphical codes in their binary interpretation is fundamentally different from that of ordinary binary LDPC codes. In the former case, the decoder interprets $b$ bits $(y_0, y_1, \ldots, y_{b-1})$ from the channel as a single $2^b$-ary symbol and sets the prior information of that symbol by assuming a product distribution for the values of each constituent bit. Namely

$$f^x := \prod_{i=0}^{b-1} f_{y_i}^{x_i^b}$$

where $f_{y_i}^{x_i^b}$ is the likelihood the $i$th constituent bit is equal to $x_i^b$, where $(x_0^b, x_1^b, \ldots, x_{b-1}^b)$ is the binary representation of the transmitted symbol $x$. The decoding algorithm is then a generalized belief propagation algorithm in which the messages exchanged over the graph are probabilities of clusters of bits, hereafter called cluster belief propagation (CBP). In contrast, in the ordinary BP algorithm for decoding binary LDPC codes, the messages are merely probabilities of individual bits, not a cluster of bits. Next we shall provide a justification that the CBP algorithm can be fundamentally advantageous over the ordinary BP algorithm by establishing connections to the Kikuchi and Bethe approximations in statistical physics.

For the purpose of analyzing the CBP algorithm, we consider a pairwise Markov Random Field (MRF) model of the Tanner graph. Each parity-check node in the Tanner graph is converted into a hidden node (the same as a symbol node) in a pairwise MRF with an observable node hanging off it. For instance, Fig. 4.4 shows the pairwise MRF corresponding to a TG(2, 3). Now we briefly describe the CBP algorithm in the context of the MRF model for decoding the binary interpretation of GF($2^b$) graphical (regular) codes. Let Roman letters like $j$ denote the symbols, i.e., clusters of $b$ bits, and Greek letters like $\alpha$ denote the check nodes. In a GF($2^b$) graphical code, the symbol node $j$ can be in one of $2^b$ states denoted by $x_j$, while the check nodes can be in one of $2^{d_c b}$ states denoted by $x_\alpha$, as each check node involves $d_c$ symbol nodes. If the $j$-th symbol is connected to the $\alpha$-th check node, then we use the notation $x_\alpha(j)$ to denote the state the $j$-th node should be in so as to correspond with the $\alpha$-th check node being in state $x_\alpha$. The pairwise potentials (compatibility function) $\psi(x_j, x_\alpha)$ are set to one if $x_\alpha(j) = x_j$ and zero otherwise. The singleton potentials are set to $\psi_j(x_j) = f^{x_j}$ for the symbol nodes, while $\psi_\alpha(x_\alpha)$ is one if $x_\alpha$ satisfies the check-node constraint and zero otherwise. The following update rules on this Markov graph give the CBP algorithm for iteratively decoding the binary interpretation of GF($2^b$) graphical codes:

$$m_{\alpha j}(x_j) \leftarrow \sum_{x_\alpha : x_\alpha(j)=x_j} \psi_\alpha(x_\alpha) \prod_{k\in\mathcal{N}(\alpha)\backslash j} m_{k\alpha}(x_\alpha) \tag{4.11}$$

$$m_{j\alpha}(x_\alpha) \leftarrow \psi_j(x_\alpha(j)) \prod_{\beta\in\mathcal{N}(j)\backslash\alpha} m_{\beta j}(x_\alpha(j)) \tag{4.12}$$

$$b_j(x_j) \leftarrow \psi_j(x_j) \prod_{\alpha\in\mathcal{N}(j)} m_{\alpha j}(x_j) \tag{4.13}$$

$$b_\alpha(x_\alpha) \leftarrow \psi(x_\alpha) \prod_{j\in\mathcal{N}(\alpha)} m_{j\alpha}(x_\alpha) \tag{4.14}$$

where $\mathcal{N}(\alpha)\backslash j$ means all symbol nodes neighboring check node $\alpha$, except $j$; $\mathcal{N}(j)\backslash\alpha$ means all check nodes neighboring symbol node $j$, except $\alpha$. Here $m_{j\alpha}$ ($m_{\alpha j}$) refers to the message that node $j$ ($\alpha$) sends to node $\alpha$ ($j$), and $b_j$ is the belief (approximate marginal posterior probability) at node $j$, obtained by multiplying all incoming messages to that node by the local evidence.

It has been pointed out [124] that an error-correcting code can be represented as an Ising spin glass model, revealing an elegant marriage of information theory and statistical physics. In particular maximum-likelihood decoding is equivalent to finding the ground state of the corresponding spin system [124], and maximum a-posteriori symbol probability decoding is equivalent to computing the thermal average at the Nishimori temperature [125–127]. Decod-

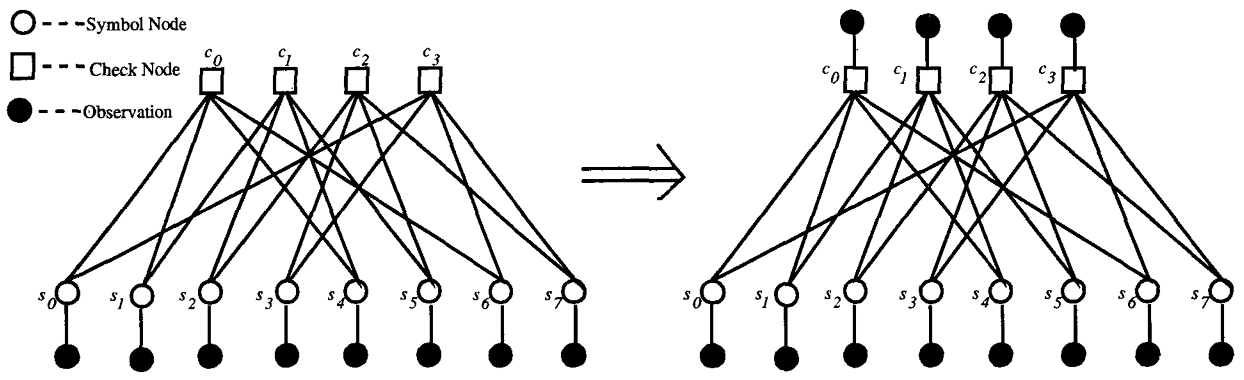


**Figure 4.4:** The pairwise Markov random graph model of a Tanner-graph code.

ing of an error-correcting code from the point of view of statistical physics is often understood as solving the stationary conditions of the approximate Gibbs free energy [128]. In Appendix C of [71], MacKay reported empirical decoding results of binary LDPC codes using the variational free energy minimization (mean field) algorithm, which performs worse than BP algorithm. It was shown in [129, 130] that, although the BP algorithm for graphical models with cycles is not guaranteed to converge, the BP fixed point corresponds to the stationary point of an approximate free energy, known as the Bethe free energy, which is a better approximation to Gibbs free energy than mean field approximation. Kikuchi developed a method, known as cluster variational method, of deriving approximations that improve on and generalize the Bethe approximation to the Gibbs free energy [131]. In a general Kikuchi approximation, the free energy is approximated as a sum of the free energies of basic clusters of nodes, minus the free energy of over-counted cluster intersections, minus the free energy of the over-counted intersections of intersections, and so on [130]. The Bethe approximation is the simplest example of the more complicated Kikuchi free energy: for that case, the basic cluster are all the connected pairs of nodes. In general, increasing the size of the basic clusters improves the approximation to the Gibbs free energy. In the limit where a basic cluster covers all the nodes in the system, the Kikuchi approximation becomes exact.

Just as BP only converges to a stationary point of the Bethe free energy, we can expect that CBP only converges to a stationary point of the Kikuchi free energy, as stated in the following lemma.

**Lemma 4.3** *A set of messages and beliefs are fixed points of the CBP decoding algorithm* *for the binary interpretation of GF($2^b$) codes if and only if they are stationary points of*

$$\begin{aligned} F_{\text{Kikuchi}} &= -\sum_j \sum_{x_j} b_j(x_j) \ln \psi_j(x_j) - \sum_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln \psi_\alpha(x_\alpha) \\ &\quad + \sum_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln b_\alpha(x_\alpha) - \sum_j (d_j - 1) \sum_{x_j} b_j(x_j) \ln b_j(x_j), \end{aligned} \tag{4.15}$$

*subject to the constraint that $b_\alpha(x_\alpha)$ marginalize down to $b_j(x_j)$ for all $j \in \mathcal{N}(\alpha)$. $d_j$ is the number of neighbors of symbol node $j$, i.e. the degree of symbol node $j$.*

*Proof:* The proof follows directly from claim 1 in [130]. This lemma can also be viewed as an extension of Corollary 2 of [130] with a minor difference: the corresponding free energy here is of Kikuchi-type because each symbol node $j$ represents a cluster of $b$ binary bits. If $b = 1$, it reduces to the Bethe free energy. ■

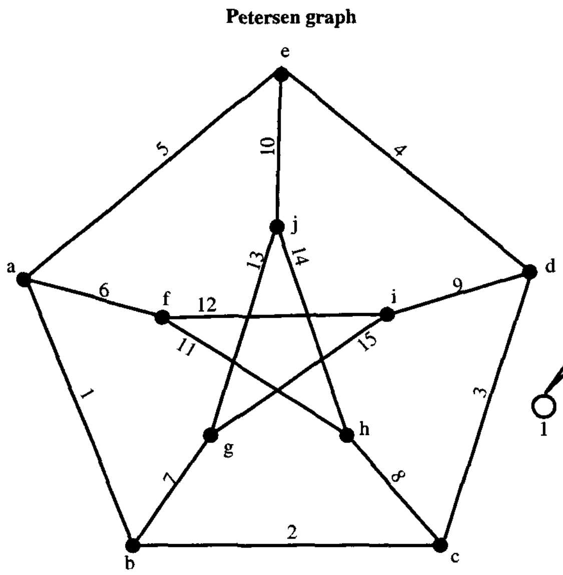
The connections between BP and the Bethe approximation, and CBP and the Kikuchi approximation reveal that, in the case of error-correcting codes with strictly short block lengths where the graphical model under consideration has substantially short cycles, the binary interpretation of GF($2^b$) codes may potentially be advantageous as their corresponding iterative decoding algorithm CBP can outperform ordinary BP.

## 4.4 Construction of Cycle Tanner-Graph GF($2^b$) Codes

Constructing a good cycle Tanner-graph GF($2^b$) code can be performed in two steps. First one could construct a generic cycle Tanner graph TG(2, $d_c$) having large girth and then associate each edge with a nonzero field element randomly selected from field GF($2^b$). The PEG algorithm [96] can be used to construct TG(2, $d_c$) with girth exceeding an analogy of the Erdös–Sachs bound and can be used for any combination of parameters such as the block length and the rate. Nonetheless our focus here is on the constructions of TG(2, $d_c$) deriving from the graph-theoretic approach.

Tanner [74] pointed out that two graph parameters are relevant to the iterative decoding performance, namely the graph girth $g$ (i.e. the length of the shortest cycle) and the graph diameter $r$ (i.e. the maximum over all pairs of vertices of the length of the shortest path between them), and that the most favorable graph for iterative decoding may be the one that has the largest girth and smallest diameter. Therefore, a special class of $k$-regular graphs, i.e. every vertex has $k$ connections, called Moore graphs, appears to the most interesting.

**Theorem 4.3** *[132, p. 181]: Of the following conditions on a graph G, any two imply the third:*



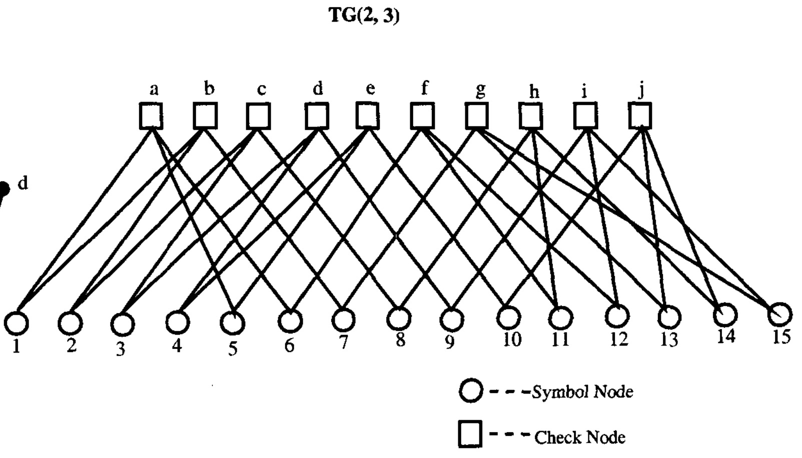


**Figure 4.5:** The Petersen graph (left) and its induced TG(2, 3) (right).

- *G is connected with maximum degree $d$ and diameter $r$;*
- *G has minimum degree $d$ and girth $g = 2r + 1$;*
- *G has $1 + d((d-1)^r - 1)/(d-2)$ vertices.*

A graph satisfying these three conditions [2] is called a Moore graph of diameter $r$ and degree $d$. The famous Petersen graph happens to be the unique Moore graph of diameter 2 and degree 3, see Fig. 4.5. Its fame stems from the fact that it is a counterexample to a large number of conjectures in graph theory.

Note that each edge in a $d$-regular graph has two endpoints (vertices), and that if we translate each edge into a symbol node and its two endpoints (vertices) into two check nodes connected to it, we obtain a TG(2, $d$). In this way, from a $d_c$-regular graph G on $m$ vertices, we derive a TG(2, $d_c$) with $d_c m/2$ symbol nodes on one side and $m$ check nodes on the other. The rate of the resulting code is $\geq d_c/2$.

**Definition 4.1 (edge-vertex incidence graphs, [43])** : *Let G be a graph with edge set E and vertex set V. The edge-vertex incidence graph of G is a cycle Tanner graph with vertex set $E \cup V$ and edge set*

$$\{(e, v) \in E \cup V : v \text{ is an endpoint of } e\}.$$

**Example 4.1** *see Fig. 4.5.*

[2]The first two conditions show that a Moore graph is regular.

**Lemma 4.4** *Let the girth of a graph G be $g_G$, then the girth of the edge-vertex incidence graph of G is $2g_G$.*

*Proof*: Consider a specific smallest cycle in G having $g_G$ edges. By definition each edge in the cycle corresponds to a symbol node in the resulting edge-vertex incidence graph, and evolves into two new edges connecting to its two endpoints. Then the corresponding cycle in the edge-vertex incidence graph has $2g_G$ edges. Therefore the girth of the edge-vertex incidence graph of G is $2g_G$. ■

In practice, it is often desirable to have a flexible combination of block length and rate. Unfortunately, it turns out that Moore graphs are very rare. For instance, if there is a Moore graph of diameter of 2 and degree $d$, then $d$ must be 2, 3, 7, or 57, and the number of vertices must be 5, 10, 50, or 3250, respectively. Alternatively, a larger family, the Ramanujan graphs [78, 79], which are $d$-regular graphs, can be used to construct TG(2, $d$) by their edge-vertex incidence graphs. A Ramanujan graph is defined by the property that the second largest eigenvalue of the adjacency matrix is no larger than $2\sqrt{d-1}$, and thus is known to have good expansion properties, large girth and small diameter. In particular, the girth of Ramanujan graphs is asymptotically a factor of 4/3 better than the Erdös-Sachs bound, which appears to be the best known $d$-regular graphs in terms of girth. Note that Ramanujan graphs have been used in several instanced to construct good LDPC codes, see [43, 81, 82] for instance.

**Theorem 4.4** *[78, 79]: Let s, t be distinct primes, and s is a quadratic nonresidue modulo t. Then $X^{s,t}$ are $(s+1)$-regular Cayley graphs of the projective general linear group of degree 2 over the field of t elements, $PGL(2, \mathbb{F}_t)$, with the number of vertices of $t(t^2-1)$ and of a girth of at least*

$$g_G \geq 4\log_s t - \log_s 4. \tag{4.16}$$

*Moreover, the edge-vertex incidence graphs of $X^{s,t}$ have a girth of at least*

$$g_{TG} \geq 8\log_s t - 2\log_s 4. \tag{4.17}$$

## 4.5 Simulation Results

It is interesting to compare the girth properties of cycle TG(2, $d_c$) of various possible construction approaches, including Ramanujan, PEG (non-greedy version), and look-ahead-enhanced PEG Tanner graphs. Fig. 4.6 depicts both the lower bound on a PEG Tanner graph (Theorem 3.1) and the upper bound (Lemma 3.6) on any Tanner graph for regular $d_s = 2, d_c = 4$

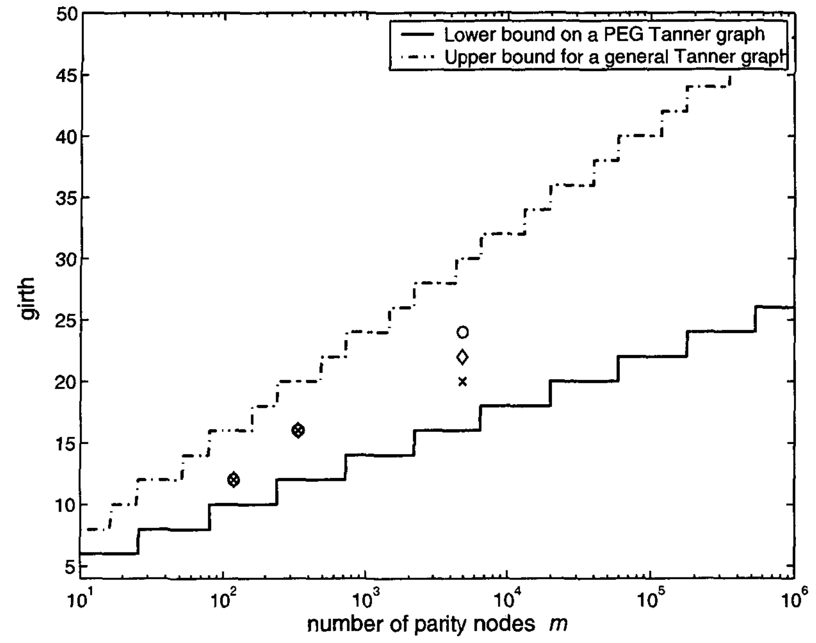


**Figure 4.6:** Lower and upper bounds on the girth of cycle TG(2, 4). The symbols $\circ$, $\diamond$, $\times$ correspond to Ramanujan, look-ahead-enhanced PEG, and PEG Tanner graphs, respectively.

codes with varying $m$ (in this case $n = 2m$). We consider three particular cases of Ramanujan graphs, $X^{3,5}$ with $m = 120$ and $n = 240$, $X^{3,7}$ with $m = 336$ and $n = 672$, and $X^{3,17}$ with $m = 4896$ and $n = 9792$, whose resulting cycle TG(2, 4) have girth 12, 16, and 24, respectively. With the same $m$, $n$, $d_s$ and $d_c$, all three candidates attain the same girth in the first two cases, and in the last case, where the graph size is relatively large, the edge-vertex incidence graph of the Ramanujan graph outperforms its counterparts in terms of girth. Note that Ramanujan graphs only exist for a few specific combinations of parameters.

Fig. 4.7 shows the performance of cycle Tanner-graph GF($2^b$) codes as a function of field order $b$. The cycle Tanner graph is the edge-vertex incidence graph of the Ramanujan graph $X^{3,7}$ with $m = 336$ and $n = 672$ and, accordingly, the actual binary block length of cycle Tanner-graph GF($2^b$) codes is $nb$ in bits. The binary interpretation of cycle Tanner-graph GF($2^b$) codes is used over the binary-input AWGN channel and decoded by the CBP algorithm up to 80 iterations. We compare the performance of cycle Tanner-graph GF($2^b$) codes with the Shannon–Gallager–Berlekamp sphere-packing bound [3] [15, 115] of the same binary block length $nb$ over the binary-input AWGN channel, plotted in Fig. 4.7 as dotted lines (shown in sequence from right to left are $b = 1, 2, 3, 4, 5, 6$, and 8). As can be seen, the gap decreases significantly as $b$ grows larger; in particular the performance of the cycle Tanner-graph GF($2^8$)

[3]In computing this bound, the term $o(n)$ is neglected because of the difficulty to calculate it.

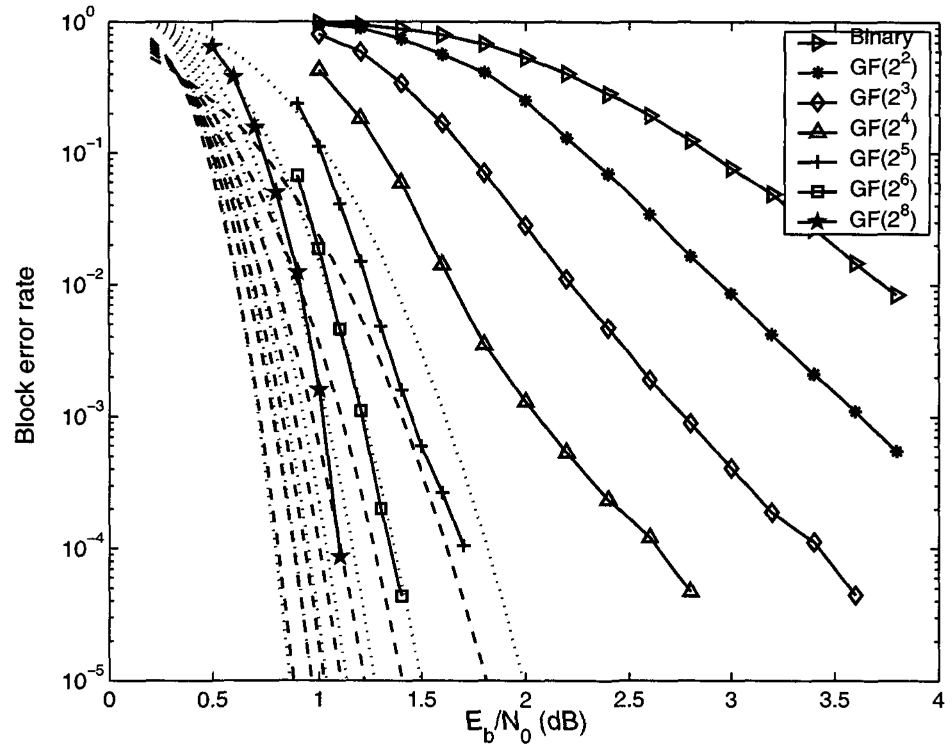


**Figure 4.7:** Block-error rate of cycle Tanner-graph GF($2^b$) codes under iterative decoding. The cycle Tanner graph is the edge-vertex incidence graph of the Ramanujan graph $X^{3,7}$ with $m = 336$ and $n = 672$. The actual (binary) block length of cycle Tanner-graph GF($2^b$) codes is $nb$ in bits.

code is within 0.4 dB from the Shannon–Gallager–Berlekamp sphere-packing bound of the same binary block length, at a block-error rate of $10^{-4}$.

The Shannon–Gallager–Berlekamp sphere-packing bound is often used as a lower bound for the block-error rate of the best code at moderate to large block lengths (its tightness also depends on the rate); and at short block lengths it becomes slightly loose as a consequence of neglecting the $o(n)$ term. Here we propose an alternative approach to estimate the performance of the asymptotically best code under ML decoding, which is tighter at short block lengths. One can think of an ensemble of Shannon equiprobable random codes, whose average performance under ML decoding is known to be "typical" by the generalized law of large numbers. As the distance spectrum of the Shannon equiprobable random ensemble is known (the binomial distribution), one can apply the best-known bounding technique — the Tangential Sphere Bound (TSB) [133–135] — to yield a tight upper bound on the block-error rate of the asymptotically best ensemble under optimum decoding. The TSBs of the Shannon equiprobable random ensemble of various block lengths over the binary-input AWGN channel are plotted in Fig. 4.7 as dashed lines (shown from right to left are $b = 1, 2, 3, 4, 5, 6$, and

8). It can be seen that at $b = 1$ (block length 667) the TSB is tighter than the Shannon–Gallager–Berlekamp bound by 0.19 dB with a block-error rate of $10^{-4}$, at $b = 2$ (block length $667 \times 2$) the TSB is tighter by 0.1 dB, and finally, the TSB is essentially the same as the Shannon–Gallager–Berlekamp bound as the block length grows larger.

## 4.6 Summary

The iterative decoding performance of an LDPC code is in general dominated by two competing factors. One is the Hamming weight spectrum, which requires the density of the parity-check matrix to be higher, the other is the performance loss due to iterative decoding, for which a low density of the parity-check matrix is favorable. These two conflicting requirements must be well balanced, particularly for short block lengths. It is demonstrated that the Hamming weight spectrum can be significantly improved by moving from the binary field to fields of higher order with lowest possible density. Therefore cycle Tanner-graph codes defined over GF($2^b$), with sufficiently large $b$, are heuristically both "good codes for optimum decoding" and "good codes for iterative decoding". Simulation results of cycle Tanner-graph GF($2^b$) codes based on the edge-vertex incidence graphs of Ramanujan graphs confirmed this claim.

# Chapter 5

# Decoding Binary LDPC Codes

## 5.1 Introduction

Efficient hardware implementation of the sum-product algorithm (SPA) for decoding binary LDPC codes has become a topic of increasing interest. Analog networks of transistors for turbo decoding or SPA have been investigated by the groups of Hagenauer [136], and of Loeliger [137], exploiting a natural match between probability theory and transistor physics. The building block is the analog 'boxplus' circuit, with which any network of sum-product modules can be directly implemented in analog VLSI. The major difficulty in analog circuits is not the complexity of hyperbolic tangent function in 'boxplus' operation, but the stability and synchronization issues.

In digital hardware design, it has been shown that direct implementation of the original form of SPA is sensitive to quantization effect [138]. Furthermore, it is demonstrated in [138] that using likelihood ratios can substantially reduce the quantization level required. A simplification of the SPA that reduces the complexity of the parity-check node update at the cost of some loss in performance has been proposed in [139]. This simplification has been derived by operating in the log-likelihood-ratio (LLR) domain. Recently, a low-complexity implementation of the SPA that also operates entirely in the log-likelihood domain has been presented in [106,140]. This scheme bridges the gap between the optimal SPA and the simplified approach in [139]. Low-complexity software and hardware implementations of an iterative decoder for LDPC codes suitable for multiple access applications have been presented in [141]. Improving the performance of [139] by normalization and offset has been proposed in [142,143].

In this chapter, we investigate low-delay and low-complexity digital implementations of the LLR-SPA from both the architectural and the algorithmic point of view and describe new derivatives thereof [1]. Log-likelihood ratios are used as messages between symbol nodes and parity-check nodes. It is known that in practical systems, using LLRs offers implementation advantages over the use of probabilities or likelihood ratios, because multiplications are replaced by additions and the normalization step is eliminated. Serial and parallel architectures for realizing the party-check node update are investigated, leading to trellis and tree topologies, respectively. In both cases, specific *core operations* similar to the special operations defined in the log-likelihood algebra of [144] are used. An application of the forward-backward procedure on the trellis topology leads precisely to the low-complexity implementation of the SPA described in [106]. The tree topology together with its corresponding core operation yields a new edge-level parallelism which is much more promising for extremely high-speed applications. The introduction of serial and parallel architecture not only leads to low-complexity LDPC decoding algorithms that can be implemented with simple comparators and adders but also provides the ability to make up the loss in performance by utilizing simple look-up tables or constant correction factors. The unified treatment of decoding techniques for LDPC codes presented in this chapter provides flexibility in selecting the appropriate design point in high-speed applications in terms of performance, latency, and computational complexity.

The remainder of this chapter is organized as follows. In Section 5.2, the SPA in the log-likelihood domain is described and the issues associated with a brute-force implementation are discussed. In Section 5.3, a trellis topology for carrying out the parity-check node update is derived. The core operation on this trellis is the LLR of the exclusive OR (XOR) function of two binary independent random variables [144], instead of the hyperbolic tangent operation used in the brute-force implementation. This core operation can either be implemented very accurately by using the max* operation [145] or approximately by using the so called sign-min operation. In either case, the check-node updates can be efficiently implemented on the trellis by the well known forward-backward algorithm. Section 5.4 is devoted to parallel processing at edge level inside the check node update, and a simple tree topology with a new core operation is proposed. It is shown that such an implementation offers smaller latency at the cost of a more complex core operation. In practice, this core operation can be realized by employing a simple eight-segment piecewise linear function. In Section 5.5, simulation results are presented, comparing the performance of the various alternative implementations of the various LLR-SPA alternatives. Section 5.6 consists of three approximations of the LLR-SPA for decoding binary LDPC codes, emphasizing simplicity at the expense of accuracy and correctness of the LLR-SPA. Finally, Section 5.7 contains a summary of the results and conclusions.

[1]Part of this work was incorporated into a joint paper with J. Chen, A. Dholakia, E. Eleftheriou and M.P.C. Fossorier, and will be submitted to IEEE Trans. Communications.

## 5.2 SPA in the Log-Likelihood Domain

Following a notation similar to [71, 139], let $\mathcal{C}(s)$ denote the set of check nodes connected to symbol node $s$, i.e., the positions of 1's in the $s$th column of the parity-check matrix $H$, and let $\mathcal{S}(c)$ denote the set of symbol nodes that participate in the $c$th parity-check equation, i.e., the positions of 1's in the $c$th row of $H$. Furthermore, $\mathcal{S}(c)\backslash s$ represents the set $\mathcal{S}(c)$, excluding the $s$th symbol node, and similarly, $\mathcal{C}(s)\backslash c$ represents the set $\mathcal{C}(s)$, excluding the $c$th check node. In addition, $q_{s\to c}(x)$, $x \in \{0,1\}$, denotes the message that the symbol node $s$ sends to the check node $c$ indicating the probability of symbol $s$ being 0 or 1, based on all the checks involving $s$ except $c$. Similarly, $r_{c\to s}(x)$, $x \in \{0,1\}$, denotes the message that the $c$th check node sends to the $s$th symbol node indicating the probability of symbol $s$ being 0 or 1, based on all the symbols checked by $c$ except $s$. Finally, $\mathbf{y} = [y_1, y_2, \cdots, y_n]$ denotes the received word corresponding to the transmitted codeword $\mathbf{u} = [u_1, u_2, \cdots, u_n]$.

The LLR of a binary valued random variable $U$ is defined as

$$L(U) \overset{def}{=} \log \frac{P(U=1)}{P(U=0)}, \tag{5.1}$$

where $P(U = x)$ denotes the probability that the random variable $U$ takes the value $x$. Furthermore, let us define the LLRs $\lambda_{s\to c}(u_s) \overset{def}{=} \log(q_{s\to c}(1)/q_{s\to c}(0))$ and $\Lambda_{c\to s}(u_s) \overset{def}{=} \log(r_{c\to s}(1)/r_{c\to s}(0))$. The LLR-SPA is then summarized as follows.

*Initialization:* Each symbol node $s$ is assigned an *a posteriori* LLR $L(u_s) = \log\{P(u_s = 1|y_s)/P(u_s = 0|y_s\}$. In case of equiprobable inputs on an AWGN channel, $L(u_s) = 2y_s/\sigma^2$, where $\sigma^2$ is the noise variance. For every position $(c, s)$ such that $H_{c,s} = 1$,

$$\begin{aligned} \lambda_{s\to c}(u_s) &= L(u_s), \\ \Lambda_{c\to s}(u_s) &= 0. \end{aligned}$$

*Step (i) (check-node update):* For each $c$, and for each $s \in \mathcal{S}(c)$, compute

$$\Lambda_{c\to s}(u_s) = 2\tanh^{-1}\{\prod_{s'\in\mathcal{S}(c)\backslash s} \tanh[\lambda_{s'\to c}(u_{s'})/2)]\}. \tag{5.2}$$

*Step (ii) (symbol-node update):* For each $s$, and for each $c \in \mathcal{C}(s)$, compute

$$\lambda_{s\to c}(u_s) = L(u_s) + \sum_{c'\in\mathcal{C}(s)\backslash c} \Lambda_{c'\to s}(u_s).$$

For each $s$, compute

$$\lambda_s(u_s) = L(u_s) + \sum_{c\in\mathcal{C}(s)} \Lambda_{c\to s}(u_s).$$

*Step (iii) (decision):* Quantize $\hat{\mathbf{u}} = [\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_n]$ such that $\hat{u}_s = 1$ if $\lambda_s(u_s) \geq 0$, and $\hat{u}_s = 0$ if $\lambda_s(u_s) < 0$. If $\hat{\mathbf{u}}H^T = \mathbf{0}$, then halt the algorithm with $\hat{\mathbf{u}}$ as the decoder output; otherwise go to *Step (i)*. If the algorithm does not halt within some maximum number of iterations, then declare a decoder failure.

The check-node updates are computationally the most complex part of the LLR-SPA. Two issues influence their complexity: i) the topology used in computing the messages that a particular check node sends to the symbol nodes associated with it, and ii) the implementation of the core operation needed for computing these messages. For example, the core operation of the check-node update computation in *Step (i)* is the hyperbolic tangent function, which looks apparently difficult to implement in digital hardware. Furthermore, in a brute-force implementation of the check-node update Eq. (5.2), $d_c(d_c - 1)$ multiplications are necessary per check node, with the multiplicands in each multiplication requiring the evaluation of the hyperbolic tangent core operation. Clearly, the higher the rate of the code, the higher the row-degree $d_c$, thus leading to a larger number of multiplications. Therefore, the brute-force topology and its corresponding core operation is not suited for high-speed digital applications.

## 5.3 Serial Implementation: Trellis Topology

### 5.3.1 Check-Node Updates

Consider a particular check node $c$ with $d_c$ connections from symbol nodes in $\mathcal{S}(c) = (s_1, s_2, \ldots, s_{d_c})$. The incoming messages are then $\lambda_{s_1 \to c}(u_{s_1})$, $\lambda_{s_2 \to c}(u_{s_2}), \ldots$, $\lambda_{s_{d_c} \to c}(u_{s_{d_c}})$. The goal is to efficiently compute the outgoing messages $\Lambda_{c \to s_1}(u_{s_1})$, $\Lambda_{c \to s_2}(u_{s_2}), \ldots$, $\Lambda_{c \to s_{d_c}}(u_{s_{d_c}})$.

Let us define two sets of auxiliary binary random variables $f_1 = u_{s_1}, f_2 = f_1 \oplus u_{s_2}$, $f_3 = f_2 \oplus u_{s_3}, \ldots$, $f_{d_c} = f_{d_c - 1} \oplus u_{s_{d_c}}$, and $b_{d_c} = u_{s_{d_c}}$, $b_{d_c - 1} = b_{d_c} \oplus u_{s_{d_c - 1}}$, $\ldots, b_1 = b_2 \oplus u_{s_1}$, where $\oplus$ denotes the binary XOR operation. It can easily be seen that for statistically independent binary random variable $V_1$ and $V_2$ [144],

$$L(V_1 \oplus V_2) = \log \frac{1 + e^{L(V_1) + L(V_2)}}{e^{L(V_1)} + e^{L(V_2)}}. \tag{5.3}$$

In [144] the so-called 'boxplus' operation $\boxplus$ is defined as

$$L(V_1) \boxplus L(V_2) = L(V_1 \oplus V_2), \tag{5.4}$$

and it can be easily verified that

$$\begin{aligned} L(V_1) \boxplus \infty &= L(V_1) \\ L(V_1) \boxplus -\infty &= -L(V_1) \\ L(V_1) \boxplus 0 &= 0. \end{aligned}$$

Using Eq. (5.3) repeatedly, we can obtain $L(f_1), L(f_2), \ldots, L(f_{d_c})$ and $L(b_1), L(b_2), \ldots, L(b_{d_c})$ in a *recursive* manner based on the knowledge of $\lambda_{s_1 \to c}(u_{s_1})$, $\lambda_{s_2 \to c}(u_{s_2}), \ldots$, $\lambda_{s_{d_c} \to c}(u_{s_{d_c}})$. Using the parity-check node constraint $u_{s_1} \oplus u_{s_2} \oplus \ldots \oplus u_{s_{d_c}} = 0 \mod 2$, we obtain

$$u_{s_i} = f_{i-1} \oplus b_{i+1} \tag{5.5}$$

for every $i \in \{2, \ldots, d_c - 1\}$. As the value of $L(f_{i-1} \oplus b_{i+1})$ does not depend on the incoming message from symbol node $s_i$, i.e. $\lambda_{s_i \to c}(u_{s_i})$, the outgoing (extrinsic) message from check node $c$ to symbol node $s_i$ can be simply expressed as

$$\begin{aligned} \Lambda_{c \to s_i}(u_{s_i}) &= L(f_{i-1} \oplus b_{i+1}), \quad i = 2, 3, \ldots, d_c - 1, \\ \Lambda_{c \to s_1}(u_{s_1}) &= L(b_2) \\ \Lambda_{c \to s_{d_c}}(u_{s_{d_c}}) &= L(f_{d_c - 1}). \end{aligned} \tag{5.6}$$

The total computational load consists of the forward recursive computation of $L(f_i)$, the backward recursive computation of $L(b_i)$, and the final pairwise part in Eq. (5.6), which amounts to $3(d_c - 2)$ core operation $L(V_1 \oplus V_2)$. This should be compared to $d_c(d_c - 1)$ hyperbolic tangent operations for the check-node updates of the brute-force topology. Clearly, the above procedure is exactly the forward-backward algorithm on a single-state trellis, as shown in Fig. 5.1. In Section 5.3.3, an efficient implementation of the core operation will be described.

### 5.3.2 Symbol-Node Updates

In the log-likelihood domain, the symbol-node updates consist only of additions of incoming messages. It is more convenient to compute the posterior LLR for the symbol $u_s$, given by

$$\lambda_s(u_s) = L(u_s) + \sum_{i=1}^{d_s} \Lambda_{c_i \to s}(u_s),$$

where $\Lambda_{c_i \to s}(u_s)$, $i = 1, \ldots, d_s$ are the incoming LLRs from the parity-check nodes $\mathcal{C}(s) = (c_1, c_2, \ldots, c_{d_s})$ connected to the symbol node $s$. Then, the outgoing messages from symbol node $s$ are obtained as

$$\lambda_{s \to c_i}(u_s) = \lambda_s(u_s) - \Lambda_{c_i \to s}(u_s), \quad i = 1, \ldots, d_s. \tag{5.7}$$

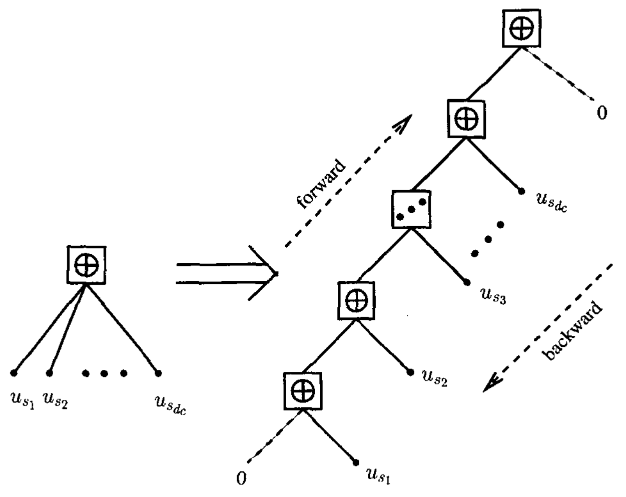


**Figure 5.1:** A serial configuration for computing the check-node updates.

The total computational load for a symbol-node update is $2d_s$ additions (subtractions). Note that this computational complexity figure includes the number of operations needed to obtain the posterior LLR use in *Step (iii)* of LLR-SPA.

## 5.3.3 Efficient Implementation of the Core Operation $L(U \oplus V)$

In this section, two versions of efficient implementation of the core operation $L(U \oplus V)$ are described, both of which are amenable to efficient VLSI design.

The first version is analogous to the max* operation used in turbo codes [145, 146]. By using the Jacobian logarithm twice, we obtain

$$\begin{aligned}
& L(U \oplus V) = \log \frac{1+e^{L(U)+L(V)}}{e^{L(U)}+e^{L(V)}} \\
= \; & \log[1+e^{L(U)+L(V)}] - \log[e^{L(U)}+e^{L(V)}] \\
= \; & \max[0, L(U)+L(V)] + \log(1+e^{-|L(U)+L(V)|}) \\
& - \max[L(U), L(V)] - \log(1+e^{-|L(U)-L(V)|}) \qquad (5.8)
\end{aligned}$$

in which the terms $\log(1+e^{-|L(U)+L(V)|})$ and $\log(1+e^{-|L(U)-L(V)|})$ can be implemented by a look-up table. Fig. 5.2 shows a plot of the function $g(x) = \log(1+e^{-|x|})$. A 3-bit coarse quantization table of $g(x)$ is given in Table 5.1. The maximum approximation error is less than 0.05.

The function $g(x)$ can also be approximated more accurately by a piecewise linear function where the multiplying factors are powers of two and therefore simple to implement in hardware with shift operations. Table 5.2 shows a piecewise linear approximation of $g(x)$ with only

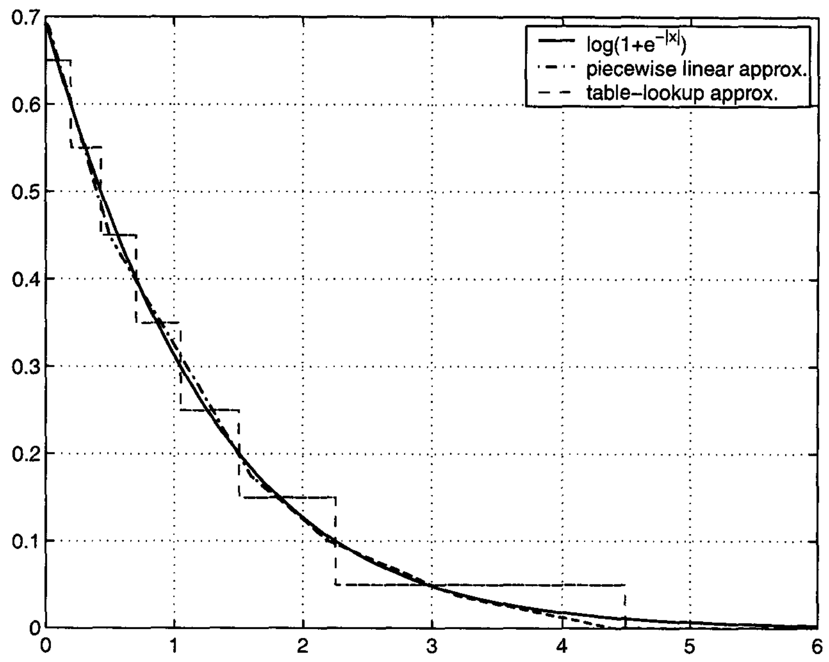


**Figure 5.2:** The function $g(x) = \log(1 + e^{-|x|})$.

eight regions. Fig. 5.2 also shows the piecewise linear approximation plot corresponding to Table 5.2. As can be seen, the piecewise linear function offers almost a perfect match to the original function.

The core operation $L(U \oplus V)$ can also be approximated as [144]

$$\begin{aligned} L(U \oplus V) &= \log \frac{1 + e^{L(U)+L(V)}}{e^{L(U)} + e^{L(V)}} \\ &\approx \operatorname{sign}[L(U)]\operatorname{sign}[L(V)] \\ &\quad \cdot \min[|L(U)|, |L(V)|], \end{aligned} \tag{5.9}$$

which is called herein as the sign-min approximation. The advantage of using the sign-min approximation lies in its simplicity. No additions are needed for the check-node updates, but only two-way comparisons, hence requiring a very small number of logic gates.

**Table 5.1:** Quantization table for $g(x) = \log(1 + e^{-|x|})$.

| $\lvert x\rvert$ | $\log(1 + e^{-\lvert x\rvert})$ | $\lvert x\rvert$ | $\log(1 + e^{-\lvert x\rvert})$ |
|---|---|---|---|
| [0, 0.196) | 0.65 | [1.05, 1.508) | 0.25 |
| [0.196, 0.433) | 0.55 | [1.508, 2.252) | 0.15 |
| [0.433, 0.71) | 0.45 | [2.252, 4.5) | 0.05 |
| [0.71, 1.05) | 0.35 | [4.5, $+\infty$) | 0.0 |

**Table 5.2:** Piecewise linear function approximation for $g(x) = \log(1 + e^{-|x|})$.

| $\|x\|$ | $\log(1+e^{-\|x\|})$ | $\|x\|$ | $\log(1+e^{-\|x\|})$ |
|---|---|---|---|
| [0, 0.5) | $-\|x\| * 2^{-1} + 0.7$ | [2.2, 3.2) | $-\|x\| * 2^{-4} + 0.2375$ |
| [0.5, 1.6) | $-\|x\| * 2^{-2} + 0.575$ | [3.2, 4.4) | $-\|x\| * 2^{-5} + 0.1375$ |
| [1.6, 2.2) | $-\|x\| * 2^{-3} + 0.375$ | [4.4, $+\infty$) | 0.0 |

It can be shown that the following equality holds:

$$\begin{aligned} &\max[0, L(U) + L(V)] - \max[L(U), L(V)] \\ &= \operatorname{sign}[L(U)]\operatorname{sign}[L(V)] \cdot \min[|L(U)|, |L(V)|]. \end{aligned} \tag{5.10}$$

Therefore, the difference between the operations max* and sign-min is given by the term $\log(1 + e^{-|L(U)+L(V)|})-\log(1 + e^{-|L(U)-L(V)|})$, called the 'correction factor $c$' in [106]. That is

$$\log \frac{1 + e^{-|u+v|}}{1 + e^{-|u-v|}} = \begin{cases} -c & \text{if } |u+v| > 2|u-v| \text{ and } |u-v| < 2 \\ c & \text{if } |u-v| > 2|u+v| \text{ and } |u+v| < 2 \\ 0 & \text{otherwise.} \end{cases} \tag{5.11}$$

It is shown in [106] that this correction factor can be approximated by a single constant without incurring significant loss in performance with respect to the SPA. It is interesting to note that this correction factor has been re-invented in [147] with a slight modification.

## 5.4 Parallel Implementation: Tree Topology

For applications with high throughput requirements, recursive algorithms such as the forward-backward algorithm may not be well suited. In this section, a simple tree topology that enables fast check-node updates is described. The symbol-node updates remain the same as in Eq. (5.7).

We begin by defining an auxiliary binary random variable $S_c = \sum_{i=1}^{d_c} \oplus u_{s_i}$. The LLR of $S_c$ at a particular check node $c$ can be computed in parallel with the tree topology shown in Fig. 5.3 using the efficient implementation of $L(U \oplus V)$ described in the previous section. The speed-up factor over the trellis topology is of order of $O[d_c/\log(d_c)]$.

The rest of this section is devoted to a simple and efficient way for computing the LLRs $\Lambda_{c\to s_i}(u_{s_i})$ by means of the auxiliary random variable $S_c$.

Let us consider

$$\begin{aligned} L(S_c) &= L(\sum_{i=1}^{d_c} \oplus u_{s_i}) = L(u_{s_i} \oplus \sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j}) \\ &= \log \frac{1 + e^{L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j}) + L(u_{s_i})}}{e^{L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j})} + e^{L(u_{s_i})}}. \end{aligned} \tag{5.12}$$

In the context of iterative decoding, the term $L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j})$ is exactly equivalent to the outgoing (extrinsic) message $\Lambda_{c\to s_i}(u_{s_i})$ from check node $c$ to any symbol node $u_{s_i} \in \{u_{s_1}, u_{s_2}, \ldots, u_{s_{d_c}}\}$, and $L(u_{s_i})$ is exactly $\lambda_{s_i\to c}(u_{s_i})$. Thus Eq. (5.12) becomes

$$L(S_c) = \log \frac{1 + e^{\Lambda_{c\to s_i}(u_{s_i}) + \lambda_{s_i\to c}(u_{s_i})}}{e^{\Lambda_{c\to s_i}(u_{s_i})} + e^{\lambda_{s_i\to c}(u_{s_i})}}.$$

After some algebra, we finally obtain

$$\Lambda_{c\to s_i}(u_{s_i}) = \log \frac{e^{\lambda_{s_i\to c}(u_{s_i}) + L(S_c)} - 1}{e^{\lambda_{s_i\to c}(u_{s_i}) - L(S_c)} - 1} - L(S_c). \tag{5.13}$$

We define

$$\Lambda_{c\to s_i}(u_{s_i}) \overset{def}{=} L(u_{s_i} \ominus S_c), \quad i = 1, \ldots, d_c.$$

Clearly, for each $i \in \{1, 2, \ldots, d_c\}$, the extrinsic information $\Lambda_{c\to s_i}(u_{s_i})$ can be computed simultaneously by a parallel implementation of the new core operation $L(u_{s_i} \ominus S_c)$ as shown in Fig. 5.3.

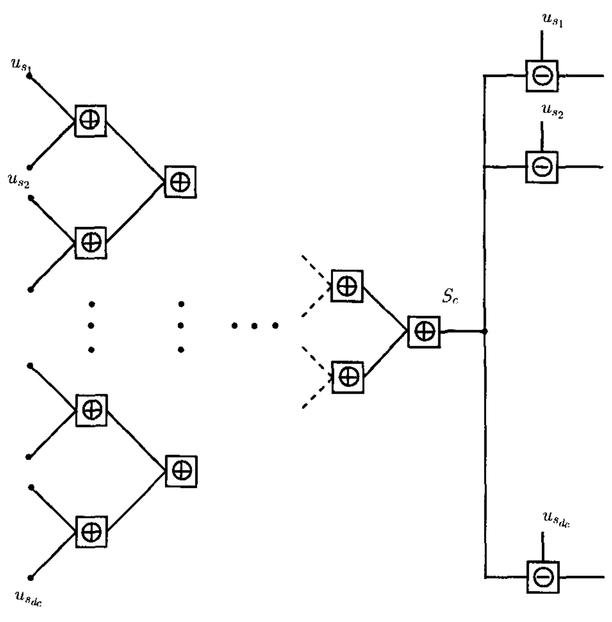


**Figure 5.3:** A parallel configuration for computing the check-node updates.

Observe that Eq. (5.13) can be written as

$$\begin{aligned}\Lambda_{c\to s_i}(u_{s_i}) &= \log\left|e^{\lambda_{s_i\to c}(u_{s_i})+L(S_c)}-1\right| \\ &\quad -\log\left|e^{\lambda_{s_i\to c}(u_{s_i})-L(S_c)}-1\right|-L(S_c).\end{aligned} \tag{5.14}$$

In Eq. (5.14) the calculation of the function $h(x) = \log|e^x - 1|$ can be implemented either by a look-up table or can be approximated by a piecewise-linear function. A plot of $h(x)$ is given in Fig. 5.4.

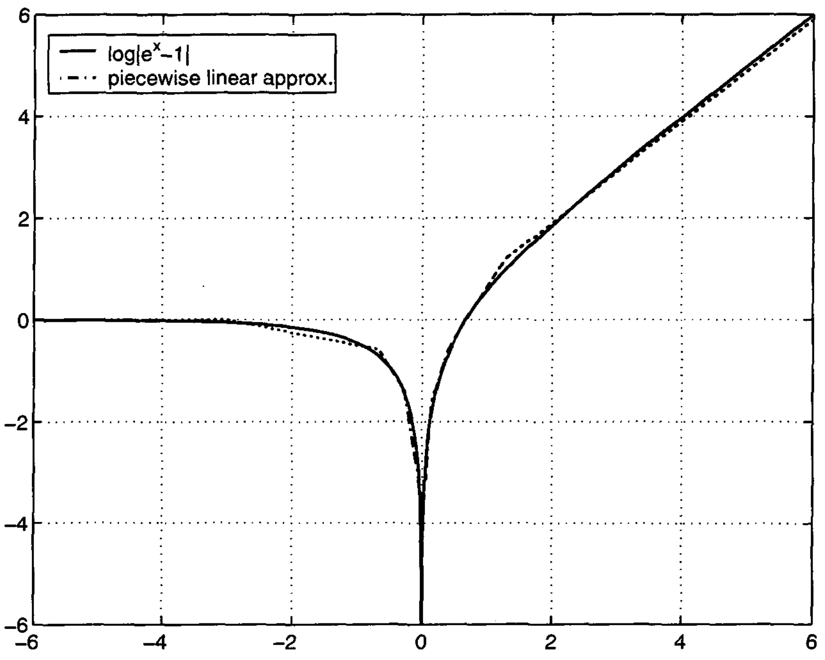


**Figure 5.4:** The function $h(x) = \log|e^x - 1|$.

Clearly, only $d_c - 1$ core operations of type $L(U \oplus V)$ and $d_c$ core operations of type $L(U \ominus V)$ are necessary for a particular check-node update. As can be seen in Fig. 5.4, as $x$ approaches zero, the function $h(x)$ approaches $-\infty$. This behavior makes it difficult to use a look-up table with a small number of quantization levels for implementing the new core operation $L(U \ominus V)$. Nevertheless, $h(x)$ can easily be approximated by a piecewise linear function where the multiplying factors are powers-of-two and therefore simple to implement in hardware with shift operations. Table 5.3 is a very accurate piecewise linear approximation of $h(x)$ with only eight regions.

**Table 5.3:** Piecewise linear function approximation for $h(x) = \log|e^x - 1|$.

| $\|x\|$ | $\log\|e^x - 1\|$ | $\|x\|$ | $\log\|e^x - 1\|$ |
|---|---|---|---|
| [-∞,-3) | 0 | [0, 0.15) | $2^4x - 4.0$ |
| [-3,-0.68) | $-2^{-2}x - 0.75$ | [0.15, 0.4) | $2^2x - 2.2$ |
| [-0.68,-0.27) | $-2x - 1.94$ | [0.4, 1.3) | $2x - 1.4$ |
| [-0.27, 0.0) | $-2^3x - 3.56$ | [1.3, +∞) | $x - 0.1$ |

## 5.5 Simulation Results

Fig. 5.5 and Fig. 5.6 show the bit error rate performance of an $[n = 1008, k = 504]$ LDPC code from [109] and an $[n = 6000, k = 3000]$ PEG LDPC code, respectively, assuming an AWGN channel. Simulation results are presented for the following LDPC decoding algorithms: the SPA, the LLR-SPA using the trellis topology for the check-node update (designated as 'LLR-SPA1'), and the LLR-SPA using the tree topology for the check-node update (designated 'LLR-SPA2'). Furthermore, the correction term $\log(1 + e^{-|L(U)+L(V)|}) - \log(1 + e^{-|L(U)-L(V)|})$ in the core operation $L(U \oplus V)$ of the LLR-SPA1 has been computed using either the look-up table shown in Table 5.1 or the piecewise linear function shown in Table 5.2. In addition, further simplifications of the LLR-SPA1 have been simulated in which the correction factor in the core operation $L(U \oplus V)$ is approximated by a fixed constant or eliminated entirely.

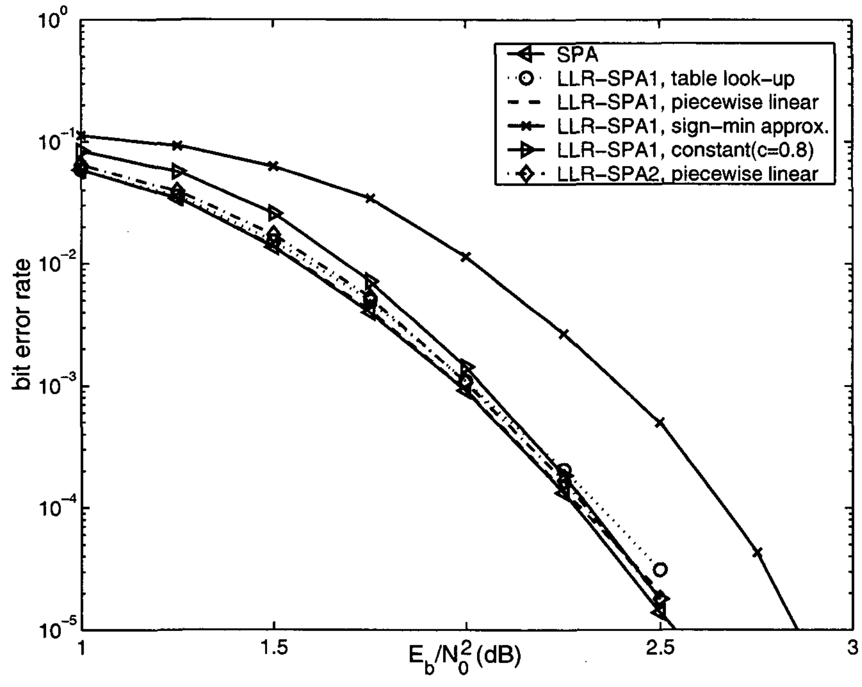


**Figure 5.5:** Performance of $[n = 1008, k = 504]$ LDPC code of MacKay.

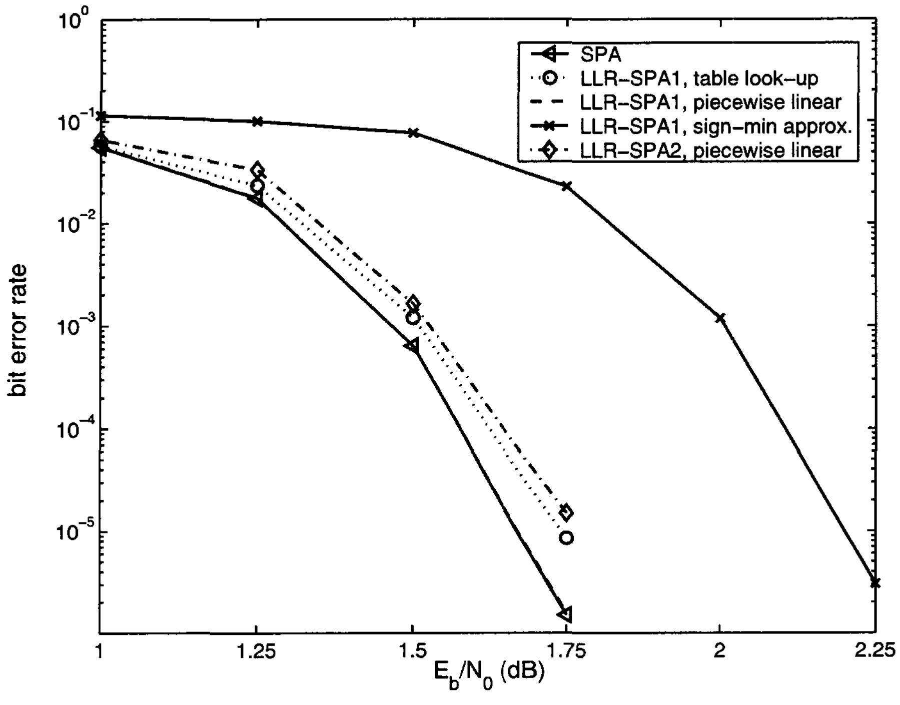


**Figure 5.6:** Performance of $[n = 6000, k = 3000]$ PEG LDPC code.

The last case corresponds to the aforementioned sign-min approximation. Finally, the core operation involved in LLR-SPA2 is implemented using the piecewise linear function shown in Table 5.3. The results are obtained via Monte Carlo simulations where the maximum number of iterations is fixed to 80 in all cases.

For both codes, we observe that at a bit error rate of $10^{-5}$, the simple sign-min approximation suffers a performance penalty of 0.3 to 0.5 dB. It appears that the loss in performance is greater as the number of parity-check equations of the LDPC code increases. On the other hand, all the other reduced-complexity variants of the LLR-SPA perform very close to the conventional SPA. In particular, the piecewise linear approximations of the core operations in LLR-SPA1 or LLR-SPA2 appear to suffer no loss (essentially less than 0.05 dB) in performance even in the case of the $[n = 6000, k = 3000]$ PEG LDPC code which involves 3000 parity-check equations. Furthermore, as can be seen in Fig. 5.5, the simple LLR-SPA1 algorithm that uses a constant correction term ($c = 0.8$) is also able to achieve the performance of the conventional SPA, in particular at higher SNRs.

## 5.6 Approximations of the LLR-SPA

In Section 5.3 and Section 5.4, we developed the trellis topology and the tree topology together with the corresponding core operations $L(U \oplus V)$ and $L(U \ominus S)$. The trellis topology enables the well-known forward-backward algorithm for a minimum usage of the core operation $L(U \oplus V)$, and the tree topology leads to an edge-level parallelism in the check-node update, which is favorable for extremely high-speed applications, but with the more difficult core operation $L(U \ominus S)$. We also addressed the issues of efficiently implementing the core operations $L(U \oplus V)$ and $L(U \ominus S)$ in digital circuits using only adders and comparators, focusing on the accuracy and correctness of the LLR-SPA.

In this section we will propose three approximations of the LLR-SPA for decoding binary LDPC codes, emphasizing simplicity at the expense of accuracy and correctness of the LLR-SPA. The first approximation is for the core operation $L(U \oplus V)$ on the trellis topology, and the second is for the core operation $L(U \ominus S)$ on the tree topology. The third is a graph-based approximation in the sense that only a few incoming messages of least reliability are kept and all other messages are tentatively decided. As the third version reduces the size of the trellis or tree in the check-node update so that fewer core operations $L(U \oplus V)$ and/or $L(U \ominus S)$ are needed, it can be combined with the first two approximations to yield extremely simple methods for decoding LDPC codes.

### 5.6.1 Separation Principle

Before presenting these mentioned three approximations of LLR-SPA, we establish the following separation principle.

**Lemma 5.1** *The sign and the magnitude of any outgoing (extrinsic) message in the check-node update are separable in the sense that the sign of an outgoing message depends only on the signs of incoming messages and the magnitude depends only on the magnitudes of incoming messages.*

*Proof:* Recall the fact that, in the trellis topology shown in Fig. 5.1, the computation of any specific outgoing message is equivalent to applying a sequence of core operations — $L(U \oplus V)$. In order to prove the lemma, it then suffices to show that the sign computation and the magnitude computation of $L(U \oplus V)$ are separable.

It is known that the core operation $L(U \oplus V)$ can be expressed as follows:

$$\begin{aligned} L(U \oplus V) &= L(U) \boxplus L(V) \\ &= 2\tanh^{-1}\left[\tanh(L(U)/2)\tanh(L(V)/2)\right] \\ &= \operatorname{sign}(L(U))\operatorname{sign}(L(V)) \cdot 2\tanh^{-1}\left[\tanh(|L(U)|/2)\tanh(|L(V)|/2)\right] \\ &= \operatorname{sign}(L(U))\operatorname{sign}(L(V)) \cdot [|L(U)| \boxplus |L(V)|]. \end{aligned} \tag{5.15}$$

Using Eq. (5.3) we have

$$\begin{aligned} |L(U)| \boxplus |L(V)| &= \log \frac{1 + e^{|L(U)|+|L(V)|}}{e^{|L(U)|} + e^{|L(V)|}} \\ &\geq 0, \end{aligned} \tag{5.16}$$

where the inequality holds because

$$[1 + e^{|L(U)|+|L(V)|}] - [e^{|L(U)|} + e^{|L(V)|}] = (e^{|L(U)|} - 1)(e^{|L(V)|} - 1) \geq 0. \tag{5.17}$$

Therefore Eq. (5.15) can be decomposed into

$$\operatorname{sign}\left[L(U \oplus V)\right] = \operatorname{sign}[L(U)]\operatorname{sign}[L(V)] \tag{5.18}$$

$$|L(U \oplus V)| = |L(U)| \boxplus |L(V)|. \tag{5.19}$$

It is clear from Eqs. (5.18) and (5.19) that the computation of the sign of $L(U \oplus V)$ depends only on the signs of $L(U)$ and $L(V)$, and the computation of the magnitude of $L(U \oplus V)$ depends only on the magnitudes of $L(U)$ and $L(V)$, which completes the proof. ■

The separation principle allows us to separate the circuit of computing the sign from that of computing the magnitude. The circuit of computing the sign of an outgoing message is rather simple. The simplest way is to first calculate $\operatorname{sign}(S_c) = \prod \operatorname{sign}[L(u_{s_i})]$ and then $\operatorname{sign}(S_c) \cdot \operatorname{sign}[L(u_{s_i})]$. An instance of the circuit follows precisely the style of Fig. 5.3, for which the input is replaced with 0 or 1 representing the signs, and $\oplus$ and $\ominus$ are replaced with exclusive OR.

One of the main advantages of the separation lies in that one can derive simpler update equations for the check-node update, which will be discussed later.

**Lemma 5.2** *Let $L(S_c) := L(\sum_{i=1}^{d_c} \oplus u_{s_i})$. Then the magnitude of $L(S_c)$ is less than or equal to the minimum of the magnitudes of $L(u_{s_i})$, namely $|L(S_c)| \leq \min_{\forall i}\{|L(u_{s_i})|\}$.*

*Proof:* First we prove that $|L(u_{s_i})| \boxplus |L(u_{s_j})| \leq \min\{|L(u_{s_i})|, |L(u_{s_j})|\}$, where $i, j \in \{1, 2, ...d_c\}$. Using Eqs. (5.8) and (5.10), we obtain

$$\begin{aligned} |L(u_{s_i})| \boxplus |L(u_{s_j})| &= \min\{|L(u_{s_i})|, |L(u_{s_j})|\} + \log(1 + e^{-||L(u_{s_i})|+|L(u_{s_j})||}) \\ &\quad - \log(1 + e^{-||L(u_{s_i})|-|L(u_{s_j})||}). \end{aligned} \tag{5.20}$$

It can be easily checked that

$$\log(1+e^{-||L(u_{s_i})|+|L(u_{s_j})||}) - \log(1+e^{-||L(u_{s_i})|-|L(u_{s_j})||}) \leq 0,$$

thus $|L(u_{s_i})| \boxplus |L(u_{s_j})| \leq \min\{|L(u_{s_i})|, |L(u_{s_j})|\}$. Now consider

$$\begin{aligned}
|L(S_c)| &= |L(\sum_{i=1}^{d_c} \oplus u_{s_i})| \\
&\overset{a}{=} |\sum_{i=1}^{d_c} \boxplus L(u_{s_i})| \\
&\overset{b}{=} \sum_{i=1}^{d_c} \boxplus |L(u_{s_i})| \\
&\overset{c}{\leq} \min(((|L(u_{s_1})|, |L(u_{s_2})|), |L(u_{s_2}|)), \cdots, |L(u_{s_{d_c}})|) \\
&= \min_{\forall i}\{|L(u_{s_i})|\}, \qquad (5.21)
\end{aligned}$$

where $a$ holds by definition, $b$ by the separation lemma, and $c$ by the associative property of $\boxplus$ and min operations, which completes the proof. ■

Before leaving this subsection, it is worthwhile to mention the original Gallager's approach [4, pp. 43] for decoding binary LDPC codes. Its key idea is based on the following observation:

$$\begin{aligned}
|L(U)| \boxplus |L(V)| &= 2\tanh^{-1}\left[\tanh(|L(U)|/2)\tanh(|L(V)|/2)\right] \\
&= f(f(|L(U)|) + f(|L(V)|)), \qquad (5.22)
\end{aligned}$$

where

$$f(x) = \log\frac{e^x+1}{e^x-1}, \qquad (5.23)$$

which is an involution transform, i.e., has the property $f(f(x)) = x$. Generally

$$\sum_{i=1}^{d_c-1} \boxplus |L(u_{s_i})| = f(\sum_{i=1}^{d_c-1} f(|L(u_{s_i})|)). \qquad (5.24)$$

Using the same idea of tree topology in Section 5.4, one might think of a seemingly simple implementation of Gallager's approach for the check-node update: the transform defined by Eq. (5.23) is first done on all incoming messages, actually LLRs, of a parity-check node. Then additions can be done with the terms needed, or preferably summing all terms then subtracting the individual ones to obtain the individual pseudo messages from which the outgoing messages are obtained by means of the involution transform.

This transform approach looks promising in extremely high-speed applications because it is conceptually simple and highly parallelizable. However, the digital implementation of the transform poses a significant challenge. It can easily be verified that, as $x$ goes to zero, $f(x)$ tends to infinity, requiring an extremely large range to be quantized. At the starting period of iterative decoding, the values of $x$ are often very small and thus this approach is unavoidably sensitive to quantization errors. From a private communication with a senior engineer at Texas Instrument Corp., we learned that at least 8 quantization bits are required for this approach.

## 5.6.2 Approximation of $|L(U \oplus V)|$

The complexity of the LLR-SPA on the trellis topology depends solely on the implementation of the core operation $L(U \oplus V)$. When the signs are handled separately, the core operation can be simplified as follows:

$$\begin{aligned} |L(U \oplus V)| &= L(|U| \oplus |V|) \\ &= \log \frac{1 + e^{|L(U)|+|L(V)|}}{e^{|L(U)|} + e^{|L(V)|}} \\ &\approx \log \frac{e^{|L(U)|+|L(V)|}}{e^{|L(U)|} + e^{|L(V)|}}, \end{aligned} \tag{5.25}$$

where the approximation is made on the assumption that $e^{|L(U)|+|L(V)|} \gg 1$. Without loss of generality, one can assume $|L(U)| < |L(V)|$ [2], then Eq. (5.25) can be rewritten as

$$|L(U \oplus V)| \approx |L(U)| - \log\left(1 + e^{-(|L(V)|-|L(U)|)}\right) . \tag{5.26}$$

Care must be taken if the righthand side of Eq. (5.26) is less than 0, as in this case 0 should be used to approaximate $|L(U \oplus V)|$ instead. In principle one can use the quantization table in Table 5.1 or the piecewise linear function approximation in Table 5.2 to approximate $g(x) = \log(1 + e^{-x}), x \geq 0$, accurately. However, one can also use the coarse approximations, either a simpler piecewise linear function such as

$$g_1(x) = \begin{cases} 0.6 - 2^{-2} * x & \text{if } x < 2.4 \\ 0 & \text{otherwise,} \end{cases} \tag{5.27}$$

(designated by LLR-SPA1-PLF), or a constant

$$g_2(x) = \begin{cases} 0.375 & \text{if } x < 1.8 \\ 0 & \text{otherwise,} \end{cases} \tag{5.28}$$

(designated by LLR-SPA1-C), shown in Fig. 5.7, to obtain a greedy simplification in mathematical operations. Note that the approximation of $g(x)$ by means of a constant as in Eq. (5.28) can be implemented using only 20 transistors, analogously to the simplified MAX* operation in turbo decoders [148].

[2]Finding the minimum over the set $\{|L(U)|, |L(V)|\}$ can be obtained as a by-product in computing the difference $|L(V)| - |L(U)|$ by checking its sign, and in Eq. (5.26) the difference $|L(V)| - |L(U)|$ is required anyway.

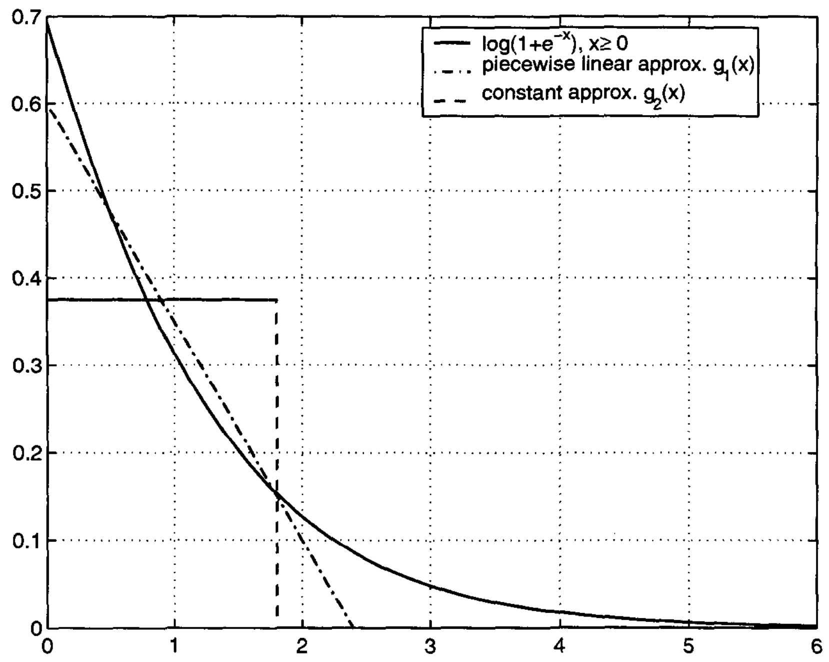


**Figure 5.7:** The coarse approximation of the function $g(x) = \log(1 + e^{-x}), x \geq 0$.

## 5.6.3 Density Evolution

The density evolution technique has been proposed to predict the performance of iterative decoders utilizing the LLR-SPA, assuming the underlying graphical model to be cycle-free [9, 54]. It has recently been extended to predict the performance of the min-sum decoding algorithm and other variants [143, 147, 149].

Let $v$ be a LLR message from a degree-$d_v$ variable node to a parity-check node and $u$ a LLR message from a parity-check node to a variable node. Under sum-product decoding, $v$ is equal to the sum of all incoming LLRs; i.e.,

$$v = \sum_{i=0}^{d_v-1} u_i, \tag{5.29}$$

where $u_i$, $i = 1, \dots, d_v - 1$, are the incoming LLRs from the neighbors of the variable node except the parity-check node that receives the message $v$, and $u_0$ is the observed (channel) LLR of the output bit associated with the variable node.

The message update rule for a parity-check node having degree $d_c$ can be represented as

$$u = \sum_{j=1}^{d_c-1} \boxplus v_j, \tag{5.30}$$

where $v_j$, $j = 1, \dots, d_c - 1$, are the incoming LLRs from $d_c - 1$ neighbors, and $u$ is the message sent to the remaining neighbor.

To perform density evolution numerically, we need to discretize the densities properly. Let $\mathcal{Q}(w)$ be the quantized message of $w$, i.e.,

$$\mathcal{Q}(w) = \begin{cases} \lfloor \frac{w}{\Delta} + \frac{1}{2} \rfloor \cdot \Delta & \text{if } w \geq \frac{1}{2\Delta} \\ \lceil \frac{w}{\Delta} - \frac{1}{2} \rceil \cdot \Delta & \text{if } w \leq -\frac{1}{2\Delta} \\ 0 & \text{otherwise,} \end{cases} \tag{5.31}$$

where $\mathcal{Q}$ is the quantization operator, $\Delta$ is the quantization step size, $\lfloor x \rfloor$ is the largest integer not greater than $x$, and $\lceil x \rceil$ is the smallest integer not smaller than $x$.

Discretized sum-product decoding is defined as sum-product decoding with all input and output messages quantized in this way. Under discretized sum-product decoding, Eq. (5.29) becomes $\bar{v} = \sum_{i=0}^{d_v-1} \bar{u}_i$, where $\bar{v} = \mathcal{Q}(v)$ and $\bar{u}_i = \mathcal{Q}(u_i)$ for $i = 0, \cdots, d_v - 1$. Denote the probability mass function (pmf) of a quantized message $w$ by $p_w[k] = \Pr(\bar{w} = k\Delta)$ for $k \in \mathbb{Z}$. Then, $p_v$ is related to its input pmf's by

$$p_v = \otimes_{i=0}^{d_v-1} p_{u_i}, \tag{5.32}$$

where $p_v$ is the pmf of $\bar{v}$, $p_{u_i}$ is the pmf of $\bar{u}_i$, and $\otimes$ is discrete convolution. As the $\bar{u}_i$'s are i.i.d. for $1 \leq i < d_v$, the above equation can be rewritten as

$$p_v = p_{u_0} \otimes (\otimes^{d_v-1} p_u), \tag{5.33}$$

where $p_u = p_{u_i}$, $1 \leq i < d_v$.

We define the following two-input operator $\mathcal{R}$:

$$\mathcal{R}(a, b) = \mathcal{Q}(a \boxplus b), \tag{5.34}$$

where $a$ and $b$ are real random variables. Note that this can be done efficiently using a pre-computed table. Using this operator, the quantized message $\bar{u}$ of Eq. (5.30) can be calculated as follows:

$$\bar{u} = \mathcal{R}(\bar{v}_1, \mathcal{R}(\bar{v}_2, \cdots, \mathcal{R}(\bar{v}_{d_c-2}, \bar{v}_{d_c-1}) \ldots)), \tag{5.35}$$

where we assume that discretized sum-product decoding at parity-check nodes is done pairwise. Clearly, if the operation $\boxplus$ is implemented approximately, the quantized message $\bar{u}$ models the behavior of discretized approximate LLR-SPA decoding.

Let $c = \mathcal{R}(a, b)$. The pmf $p_c$ of $c$ is given by

$$p_c[k] = \sum_{(i,j):k\Delta = \mathcal{R}(i\Delta, j\Delta)} p_a[i] p_b[j]. \tag{5.36}$$

We write this as $p_c = \mathcal{R}(p_a, p_b)$ for the sake of simplicity.

As the $p_{v_i}$'s are all equal, we define $p_v = p_{v_i}$ for any $1 \leq i < d_c$, and write $p_u = \mathcal{R}(p_v, \mathcal{R}(p_v, \cdots, \mathcal{R}(p_v, p_v)\ldots))$ as $p_u = \mathcal{R}^{d_c-1}p_v$.

Consider a random ensemble of irregular codes specified by the two degree distributions $\lambda(x)$ and $\rho(x)$, where $\lambda(x) = \sum_{i=2}^{d_l} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{j=2}^{d_r} \rho_j x^{j-1}$. Here $\lambda_i$ is the fraction of edges that belong to degree-$i$ variable nodes, $\rho_j$ is the fraction of edges that belong to degree-$j$ parity-check nodes, $d_l$ is the maximum variable degree, and $d_r$ is the maximum parity-check degree. By defining $\lambda(p) = \sum_{i=2}^{d_l} \lambda_i \otimes^{i-1} p$ and $\rho(p) = \sum_{j=2}^{d_r} \rho_j \mathcal{R}^{j-1} p$ for any pmf $p$, we can write the discretized density evolution as follows:

**Lemma 5.3** [**Chung** *et al.* [54]] *The discretized density evolution is given by* $p_u^{l+1} = \rho(p_{u_0} \otimes \lambda(p_u^l))$, *where the initial pmf is*

$$p_u^0 = \delta_0 = \begin{cases} 1 & \text{at } 0 \\ 0 & \text{otherwise.} \end{cases}$$

The discretized density evolution continues until either the density of $u$ tends to the "point mass at infinity" (equivalently, the probability of error tends to zero) or it converges to a density having a finite error probability, which is defined as the probability of $u$ being negative. The *threshold* is defined as the maximum noise level such that the error probability tends to zero as the number of iterations tends to infinity.

Using the discretized density evolution approach described in Lemma 5.3, we compare approximations of LLR-SPA with the min-sum algorithm and the exact LLR-SPA. The results are summarized in Table 5.4. These results show a performance degradation of about 0.18 to 1.13 dB for the min-sum algorithm, ranging from rate 9/10 to 1/4. The lower the rate, the greater the performance loss. However, the proposed LLR-SPA1-C is capable of re-gaining almost the entire performance loss due to the min-sum algorithm, for instance, at rate 1/4 the performance loss is only 0.09 dB and at rate 9/10 only 0.02 dB. It is worth emphasizing that the core operation $|L(U \oplus V)|$ in LLR-SPA1-C needs two additions(minus) and one comparison only but no look-up table.

As pointed out in [54], the discretized density evolution also predicts the performance of a practical decoder that operates on quantized messages. Using this approach, we study the performance of approximations of LLR-SPA with quantization errors. This is of practical interest, as it can guide us in choosing appropriate quantization parameters, particularly in selecting the number of bits for a digital hardware implementation. For simplicity, we

**Table 5.4:** Threshold $(E_b/N_0)$ of LDPC codes with LLR-SPA and approximation of LLR-SPA.

| $d_s$ | $d_c$ | rate | LLR-SPA | Min-Sum | LLR-SPA1-PLF | LLR-SPA1-C |
|---|---|---|---|---|---|---|
| 3 | 4 | 1/4 | 0.957448 dB | 2.0834 dB | 1.01048 dB | 1.04428 dB |
| 3 | 5 | 2/5 | 0.889071 dB | 1.67778 dB | 0.919921 dB | 0.946202 dB |
| 3 | 6 | 1/2 | 1.10195 dB | 1.69908 dB | 1.12473 dB | 1.14649 dB |
| 3 | 7 | 4/7 | 1.33526 dB | 1.82377 dB | 1.35387 dB | 1.3728 dB |
| 3 | 9 | 2/3 | 1.7477 dB | 2.12468 dB | 1.76215 dB | 1.77769 dB |
| 3 | 12 | 3/4 | 2.22494 dB | 2.52392 dB | 2.23645 dB | 2.24934 dB |
| 3 | 15 | 4/5 | 2.58599 dB | 2.84375 dB | 2.59601 dB | 2.60738 dB |
| 3 | 30 | 9/10 | 3.63246 dB | 3.81485 dB | 3.63964 dB | 3.64805 dB |

restrict ourselves to uniform and time-invariant quantization. The results are summarized in Tables 5.5 and 5.6, showing the quantization effects with 5- and 6-bit (all including the sign bit) implementations. For each case a reasonably good quantization step size $\triangle$ is chosen by multiple trials. It can be seen that, with appropriate choices of combinations of quantization step size and number of quantization bits, 5 or 6 bits appear to be sufficient for attaining a performance close to that of the LLR-SPA. At first sight it seems strange that the threshold of rate-1/4 with 5- or 6-bit quantization outperforms that of LLR-SPA. We remind ourselves, however, this is most likely due to the alias effect of the coarse quantization of the density evolution.

**Table 5.5:** Threshold $(E_b/N_0)$ of LDPC codes decoded with 5-bit quantization.

| $d_s$ | $d_c$ | rate | stepsize $\triangle$ | Min-Sum | LLR-SPA1-PLF | LLR-SPA1-C |
|---|---|---|---|---|---|---|
| 3 | 4 | 1/4 | $7/2^4$ | 2.08673 dB | 0.97023 dB | 0.911324 dB |
| 3 | 5 | 2/5 | $9/2^4$ | 1.70142 dB | 0.932132 dB | 0.969805 dB |
| 3 | 6 | 1/2 | $10/2^4$ | 1.728 dB | 1.21325 dB | 1.17419 dB |
| 3 | 7 | 4/7 | $10/2^4$ | 1.86396 dB | 1.4371 dB | 1.40389 dB |
| 3 | 9 | 2/3 | $10/2^4$ | 2.22244 dB | 1.87435 dB | 1.84647 dB |
| 3 | 12 | 3/4 | $13/2^4$ | 2.56641 dB | 2.40816 dB | 2.56641 dB |
| 3 | 15 | 4/5 | $15/2^4$ | 2.87912 dB | 2.73067 dB | 2.87912 dB |
| 3 | 30 | 9/10 | $17/2^4$ | 3.86394 dB | 3.75487 dB | 3.86394 dB |

## 5.6.4 Approximation of $|L(u_{s_i} \ominus S_c)|$

On the tree topology in the check-node update, we define

$$L(S_c) = L(\sum_{i=j}^{d_c} \oplus u_{s_j}), \tag{5.37}$$

**Table 5.6:** Threshold $(E_b/N_0)$ of LDPC codes decoded with 6-bit quantization.

| $d_s$ | $d_c$ | rate | stepsize $\Delta$ | Min-Sum | LLR-SPA1-PLF | LLR-SPA1-C |
|---|---|---|---|---|---|---|
| 3 | 4 | 1/4 | $14/2^5$ | 2.09977 dB | 0.969665 dB | 0.910469 dB |
| 3 | 5 | 2/5 | $17/2^5$ | 1.69726 dB | 0.930979 dB | 0.924047 dB |
| 3 | 6 | 1/2 | $17/2^5$ | 1.71604 dB | 1.13702 dB | 1.13274 dB |
| 3 | 7 | 4/7 | $17/2^5$ | 1.8383 dB | 1.36628 dB | 1.36281 dB |
| 3 | 9 | 2/3 | $17/2^5$ | 2.13623 dB | 1.77382 dB | 1.77101 dB |
| 3 | 12 | 3/4 | $17/2^5$ | 2.53374 dB | 2.24732 dB | 2.2447 dB |
| 3 | 15 | 4/5 | $17/2^5$ | 2.85376 dB | 2.60707 dB | 2.60458 dB |
| 3 | 30 | 9/10 | $17/2^5$ | 3.85203 dB | 3.67198 dB | 3.66975 dB |

and then

$$\begin{aligned}|L(S_c)| &= \sum_{j=1}^{d_c} \boxplus |L(u_{s_j})| \\ &= \left(\sum_{j=1,j\neq i}^{d_c} \boxplus |L(u_{s_j})|\right) \boxplus |L(u_{s_i})| \\ &= \log \frac{1 + e^{|L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j})| + |L(u_{s_i})|}}{e^{|L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j})|} + e^{|L(u_{s_i})|}} \\ &\approx \log \frac{e^{|L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j})| + |L(u_{s_i})|}}{e^{|L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j})|} + e^{|L(u_{s_i})|}}. \end{aligned} \tag{5.38}$$

By definition we have

$$|L(u_{s_i} \ominus S_c)| = |L(\sum_{j=1,j\neq i}^{d_c} \oplus u_{s_j})|. \tag{5.39}$$

Solving $|L(u_{s_i} \ominus S_c)|$ from Eq. (5.38), we obtain

$$\begin{aligned}|L(u_{s_i} \ominus S_c)| &\approx \log \frac{e^{|L(S_c)| + |L(u_{s_i})|}}{e^{|L(u_{s_i})|} - e^{|L(S_c)|}} \\ &= |L(S_c)| - \log\left[1 - e^{-(|L(u_{s_i})| - |L(S_c)|)}\right]. \end{aligned} \tag{5.40}$$

Care must be taken if the right-hand side of Eq. (5.40) is less than 0; then 0 should be used to approaximate $|L(u_{s_i} \ominus S_c)|$ instead. According to Lemma 5.2, $|L(u_{s_i})| - |L(S_c)| \geq 0$, we need only consider the approximation of function $k(x) = -\log(1 - e^{-x})$ for $x \geq 0$. Shown in Fig. 5.8 are two piecewise linear approximations of $k(x)$, where

$$k_1(x) = \begin{cases} 1.5 - x & \text{if } x < 1.5 \\ 0 & \text{otherwise,} \end{cases} \tag{5.41}$$

and

$$k_2(x) = \begin{cases} 2.6 - 2^2 * x & \text{if } x < 0.4429 \\ 1.05 - 2^{-1} * x & \text{if } 0.4429 \leq x \leq 2.1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.42}$$

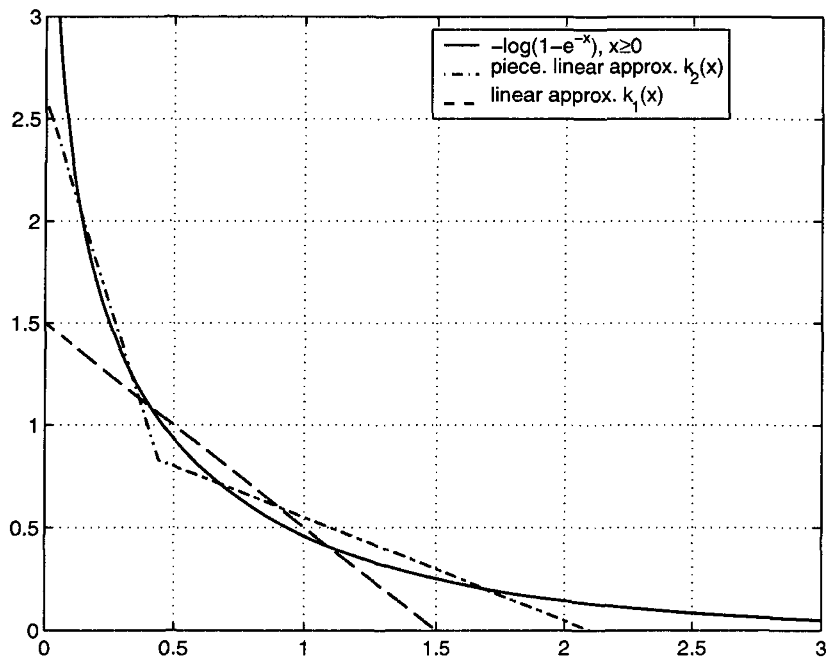


**Figure 5.8:** The coarse approximation of function $g(x) = -\log(1 - e^{-x}), x \geq 0$.

## 5.6.5 Graph-Based Approximation

In this subsection, we propose a complexity- and delay-efficient simplification of the LLR-SPA based on graph reduction according to the magnitude (reliability) of incoming messages (LLRs) [150]. The key feature of the new approximation consists of a modification of the complexity-intensive and delay-causing update equations at the check nodes of the factor graph of the LDPC code. The modified update equations at a check node are based on a simplified factor graph retaining several least reliability values of the incoming messages and on using a balanced tree topology to achieve optimum parallel processing. Furthermore, the complexity of the new algorithm can be adjusted: the least complex version of the algorithm corresponds to the so-called min-sum approximation, and the most complex version gives the full LLR-SPA.

Specifically, all incoming messages are split into the set of a few least reliable messages and the set of all other messages, which are treated as being fully reliable. In this way, the complexity of the update equations can essentially be reduced to the case of only a few incoming messages. The reduction in complexity is particularly pronounced for high-rate LDPC codes, where each check node is connected to a large number of symbol nodes.

The key idea of the new algorithm is to select a fixed number, say $z$, of least reliable symbol nodes $u_{o_1}, \ldots, u_{o_z}$ and to treat all remaining symbol nodes as being fully reliable, i.e.,

$$|L(u_{o_z+1})| = |L(u_{o_z+2})| = \cdots = |L(u_{o_{d_c}})| = \infty. \tag{5.43}$$

This amounts to keeping the original soft values for the $z$ least reliable incoming messages and approximating the original soft values for the other $d_c - z$ messages by the hard decisions $\pm\infty$, claiming the associated bits to have sign $\pm$.

The distinction of incoming messages with full information and with tentative decision can be reflected in the construction of a partially balanced binary factor graph for the updates of the check node corresponding to $u_{s_1} \oplus \cdots \oplus u_{s_{d_c}} = 0 \pmod 2$ (see Fig. 5.9). This partially balanced binary tree, which we will call a reliability-based balanced tree, is obtained as follows. From the least reliable incoming messages $u_{o_1}, \ldots, u_{o_z}$, we form a (partially) balanced binary sub-tree with the root given by the binary state variable $S' = u_{o_1} \oplus \ldots \oplus u_{o_z}$. Similarly, a (partially) balanced binary sub-tree with the root $S'' = u_{o_z+1} \oplus \ldots \oplus u_{o_{d_c}}$ is formed from the hard-decision symbol nodes $u_{o_z+1}, \ldots, u_{o_{d_c}}$. Eventually, the two sub-trees are connected via a binary parity-check node, which becomes the root of the reliability-based balanced tree.

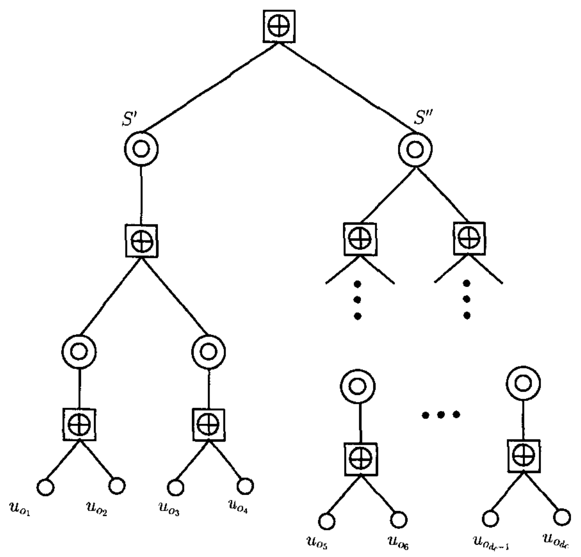


**Figure 5.9:** Reliability-based balanced tree separating the $z = 4$ least reliable (left sub-tree) and the $d_c - z$ most reliable incoming messages (right sub-tree) in a parity-check node $u_{o_1} \oplus \cdots \oplus u_{o_{d_c}} = 0 \pmod 2$. The double circle represents a binary intermediate variable that is a modulo-2 sum of symbol nodes.

The complexity of the LLR-SPA on the partially balanced factor graph is mainly determined by the complexity on the left sub-tree emanating from state node $S'$ because full soft reliability values are computed only on this sub-tree. On the right sub-tree, which emanates from state $S''$, only sign computations are needed because all leaves have LLRs that are $\pm\infty$ (note that $\infty$ is the neutral element with respect to the $\boxplus$-operation, i.e. $L(U) \boxplus \infty = L(U)$, and $L(U) \boxplus -\infty = -L(U)$).

Supposing that the sign computations of extrinsic messages are done separately, the partially balanced tree of Fig. 5.9 can be further simplified by keeping only the left sub-tree with the root $S'$. The magnitude of extrinsic messages corresponding to incoming messages residing on the right sub-tree is equal to the reliability of $S'$, namely $|L(S')|$. The resulting factor graph is shown in Fig. 5.10.

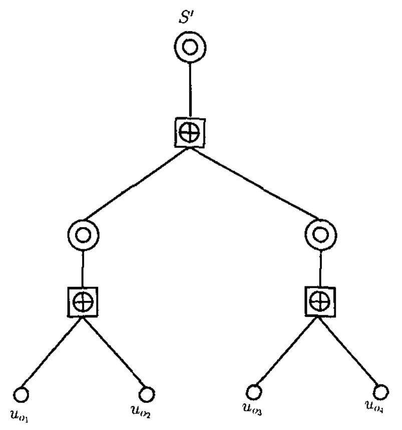


**Figure 5.10:** Balanced subtree of the $z = 4$ least reliable incoming messages.

We have several methods for the parity-check node update on this subtree. One common method is the forward-backward one, e.g., the forward pass includes the updates from the top down and then the backward pass includes the updates from the bottom up. We can also change the topology of this factor graph into a trellis or a tree, as shown in Section 5.3 or Section 5.4, to accomplish the check-node update.

For high-rate LDPC codes, the reliability-based check-node update provides substantial savings in complexity: compared with the $3(d_c - 2)$ $L(U \oplus V)$ operations for every check node in the full LLR-SPA, the simplified version needs about $3(z-2)$ $L(U \oplus V)$ operations, to which some overhead for partial-ordering has to be added. Both numbers are calculated based on the trellis implementation.

The partial-ordering problem, which finds the $z$ smallest elements in a set of $d_c$ real values, is a problem whose worst-case complexity is difficult to analyze. Here, we give a simple algorithm that provides a simple bound on the number of comparisons needed. The algorithm is based on merge-sorting [151], but with the following modification: in the ordered list obtained from the merging of two smaller ordered lists, only the $z$ smallest values are kept and all other (greater) values are deleted from the list. As a result, the maximum list size at each stage of the algorithm is $z$. A balanced tree with $d_c$ leaves contains a total of $d_c - 1$ inner nodes (that are not leaves). At these inner nodes, at most $z$ comparisons have to be done. Hence, the total complexity of partial ordering is upper bounded by $z(d_c - 1)$ comparisons. Note that when the comparisons are done in parallel at each level in the tree, the run time of the sorting algorithm corresponds to about $\log_2(d_c)$ comparison operations.

Another approach to find the $z$ smallest elements in a set of $d_c$ real values is to consecutively find the smallest one, using a balanced binary tree excluding the smallest one found in the preceding round. In this way, we find the smallest one at the first round, and the second smallest one at the second round, and so on. The total complexity of comparisons is about $z(d_c - 1)$ and the time complexity is about $z \log_2 d_c$.

For simulations on the AWGN channel, we have considered an LDPC code of length $n = 4489$ and rate 4158/4489, which is defined by $m = 335$ parity checks. Fig. 5.11 shows the bit-error-rate performance of this code. The following LDPC decoding algorithms have been used: the full LLR-SPA, and the reliability-based approximations of the LLR-SPA with $z = 2$ least reliable values, with $z = 3$ least reliable values, and with $z = 4$ least reliable values. The results are obtained using Monte–Carlo simulations, in which the maximum number of iterations is fixed to 80 in all cases. For the calculation of the core operation $L(U \oplus V)$, we use Eq. (5.3) in its full accuracy.

We observe that the simple min-sum approximation, which essentially corresponds to the reliability-based approximation of the LLR-SPA with $z = 2$ least reliable values, suffers a performance penalty of about 0.3 dB at a bit-error rate of $10^{-6}$. It is apparent from Fig. 5.11 that the loss in performance is recovered by increasing the number $z$ of remaining least reliable values. For instance, for $z = 3$ the performance loss falls within 0.1 dB, whereas for $z = 4$ it is only 0.04 dB from the full LLR-SPA.

Fig. 5.12 shows the bit-error-rate performance of MacKay's LDPC code of length $n = 1008$ and rate 1/2 on the AWGN channel. Again we see that the simple min-sum approximation suffers non-negligible performance loss relative to the full LLR-SPA, which can be regained by using $z = 3$ or $z = 4$.

Note that two improved BP-based reduced-complexity decoding algorithms presented in [142, 143] can be understood as the graph-based approximation with $z = 2$ followed by a normalization or offset procedure. By optimizing the normalization or offset factor, it has

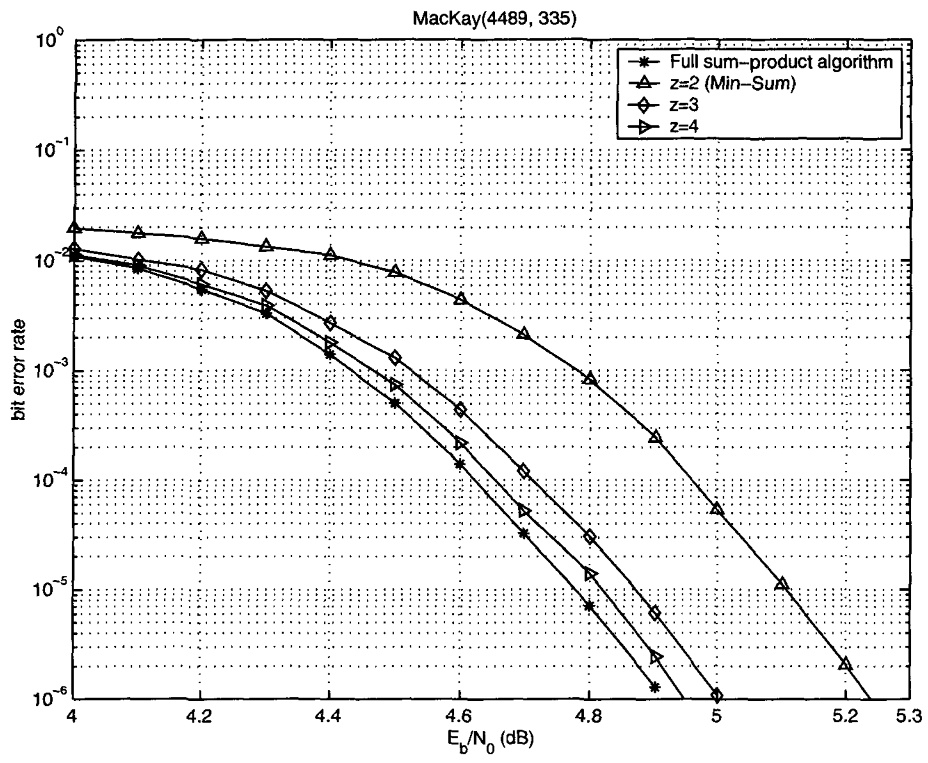


**Figure 5.11:** Performance of the LLR-SPA and its simplifications using $z = 2, 3$, and 4 least reliable values for a rate-4158/4489 code on the AWGN channel.

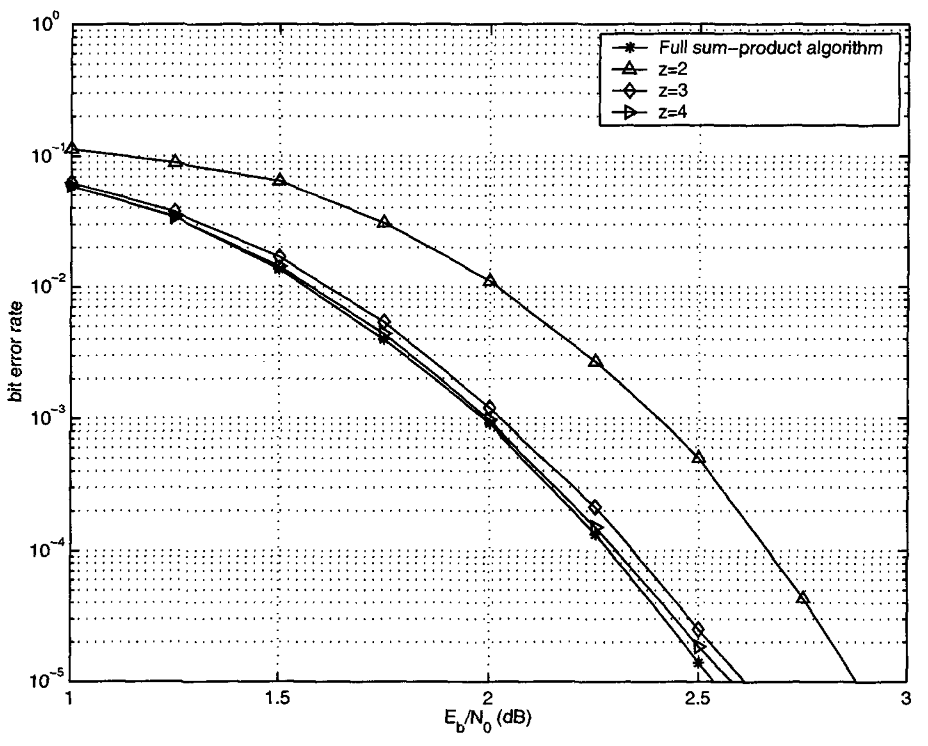


**Figure 5.12:** Performance of the LLR-SPA and its simplifications using $z = 2, 3$, and 4 least reliable values for a rate-1/2 code (1008, 504) on the AWGN channel.

been demonstrated that the performance of the full LLR-SPA can also be closely approached.

## 5.7 Summary

Efficient implementations of the SPA for decoding binary LDPC codes have been considered. A number of reduced-complexity implementations of the SPA based on using LLRs as messages between symbol nodes and parity-check nodes have been investigated. In particular, two different topologies for implementing the parity-check-node update, namely, trellis and tree topologies have been presented. In both cases, the correction terms in the core operations have been implemented via piecewise linear functions or look-up tables, or even by using a single constant, facilitating simple hardware design. Simulations results have shown that it is possible to achieve the performance of the conventional SPA extremely closely with a significant reduction in implementation complexity.

We have proposed three approximations of the LLR-SPA for decoding binary LDPC codes. The first two focus on reducing the complexity of core operations on the trellis or tree topology, sacrificing the correctness of the LLR-SPA. These two approximations are made available via the separation principle that the sign and magnitude computation of outgoing messages in the parity-check-node update can be implemented separately.

The main feature of the third approximation consists in partially ordering the reliability of the incoming messages at each parity-check node and consequently simplifying the factor graph representation of the parity-check node. The complexity of the graph-based approximation approach depends on the number $z$ of least reliable messages that are selected. When keeping only $z = 2$ least reliable messages, the approach reduces essentially to the well-known min-sum algorithm, which apparently has the least complexity. Simulation results have shown that for increasing values of $z$, the performance of the graph-based approximation quickly approaches that of the full LLR-SPA.

In conclusion, we investigated efficient implementations of the LLR-SPA for decoding binary LDPC codes from both the algorithmical and the architectural point of view, and derived new reduced-complexity variants thereof. All existing digital solutions turn out to be specific instances in our framework. The unified treatment of decoding techniques for binary LDPC codes provides flexibility in selecting the appropriate design point in high-speed applications in terms of performance, latency, and computational complexity.

# Chapter 6

# Turbo Equalization: A Message-Passing Scheduling Perspective

## 6.1 Introduction

The growth of the bit densities of magnetic recording systems by means of shrinking the bit size on the surface of the magnetic medium unavoidably leads to degradation in the signal-to-noise ratio (SNR), which in turn requires advanced signal-processing and coding techniques. Turbo codes [6, 152–154] and LDPC codes [4, 5, 8, 10, 71] are recent breakthroughs in coding theory that promise to push the areal density of the magnetic recording system to its limits. Considerable work in investigating turbo equalization schemes for the magnetic recording channel has recently been reported in [155–160]. In particular, the application of turbo codes and combined turbo decoding and turbo equalization to magnetic recording was presented in [155]. A simplified serial turbo decoder structure, in which the inner code is the precoded PR4 channel and the outer code is a single convolutional code, has been considered in [159,160].

Except for capacity-approaching decoding performance, LDPC codes have remarkable features, e.g. good minimum distance and a highly parallel, low latency decoder, that make them suitable for magnetic recording systems. Previous proposals for applying LDPC codes to the magnetic recording channel include [140, 158, 161–169]. In particular, [161, 165] proposed the use of high-rate LDPC codes as outer codes for magnetic recording systems. The maximum-transition-run (MTR) code is serially concatenated in reverse order with an LDPC code to avoid the need for soft decoding of the MTR code. Iterative (turbo) detection/decoding is then performed between the partial response (PR) channel and the LDPC code. Ref. [140] investigated low-complexity near-optimal detection algorithms for the PR channel and proposed reduced-complexity algorithms for decoding LDPC codes, where various joint decoding/detection schedules were also investigated. Refs. [166, 168] considered the applications of combinatorial constructions of high-rate LDPC codes to magnetic recording systems, [167] designed near-optimal degree sequences to construct irregular LDPC codes applied to the PR channel, and [169] proposed parallel bit-based and state-based message-passing algorithms to remove intersymbol interference incurred in the PR channel, at a cost of about 0.6 dB in a specific example to achieve the highest-level parallelism.

Factor graphs [67] prove to be a very powerful graphical model on which a single algorithm — the sum-product algorithm can encompass a large variety of practical algorithms. In particular, the forward-backward algorithm, the Viterbi algorithm, Pearl's belief propagation, the iterative turbo decoding algorithm, the Kalman filter, and even certain FFT algorithms can be visualized as instances of SPA with different message-passing schedules, given an appropriate definition of messages in each case. Ref. [164] made use of the factor graph model to investigate the performance of joint decoding/detection of high-rate LDPC codes and the PR channel, and a more general case was considered in [170]. Note that the parallel bit-based and state-based message-passing algorithm in [169] can alternatively be derived from the SPA with the so-called flooding schedule, which yields the least possible latency and was originally proposed for decoding LDPC codes. Likewise, concurrent turbo decoding [171, 172] can also be viewed as the SPA on a trellis with the flooding schedule.

In this chapter, we propose a new message-passing schedule, characterized by a parallel windowed forward-backward (PWFB) schedule over the PR channel, for joint iterative decoding of LDPC codes and PR channels. The decoding/detection latency is essentially independent of the block length, and proportional to the window size, which is adjustable. Simulation results show that the performance loss reported in [169] can be recovered with the same computational complexity.

## 6.2 Joint Factor Graph Representation

The basic magnetic recording system with an LDPC encoder and a serially concatenated detector/decoder configuration is shown in Fig. 6.1. A block of binary data $\mathbf{u} = [u_1, \ldots, u_k]$ is encoded by a rate-$k/n$ LDPC encoder into a binary codeword $\mathbf{b} = [b_1, \ldots, b_n]$ and then mapped to antipodal encoded data symbols $x_i = 2b_i - 1$, $i = 1, \ldots, n$, which are written on the disk at a rate of $1/T$. Adopting a linear model for the write-read process on a magnetic

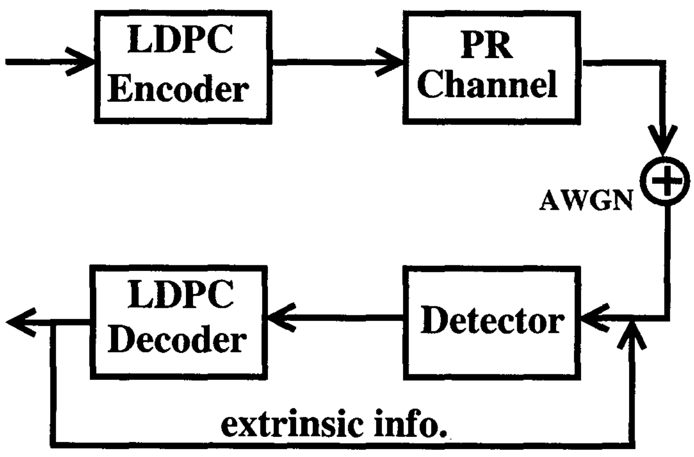


**Figure 6.1:** Block diagram of the LDPC-coded partial-response (PR) system

disk and assuming the presence of thermal noise only, the magnetic recording channel is often modeled by a Lorentzian channel model with additive white Gaussian noise (AWGN). The data signal is read back via a low-pass filter (LPF) and shaped such that the overall discrete-time transfer function, including the head/disk-medium characteristics, the analog LPF, and the sampler, closely matches the PR4, EPR4, or other generalized PR polynomials. The shaped channel output samples $y_i$ are then passed to the detector. The detector produces soft information about the encoded data symbols $x_i$, which is passed to the LDPC decoder. The extrinsic information produced by the LDPC decoder is then fed back to the channel detector. Such loops are often performed a few times before sending out hard decisions on $u_i$.

An LDPC code has an elegant factor graph representation, also known as bipartite graph or Tanner graph representation. An LDPC code is a linear code defined by a sparse parity-check matrix $H$ having dimensions $m \times n$. A factor graph (bipartite) with $m$ parity-check nodes in one class and $n$ symbol nodes in the other can be created using $H$ as the integer-valued incidence matrix for the two classes. Decoding of the LDPC code turns out to be the SPA operating on this graph. Messages of bit probabilities are alternately passed between two types of nodes, symbol (bit) nodes and parity-check nodes. The alternation of messages passing from symbol nodes to parity-check nodes and back to symbol nodes forms one LDPC decoding iteration. As there is no inherent time axis inside this factor graph representation, all messages are passed or exchanged simultaneously, called the flooding schedule [67], potentially leading to a very-high-speed decoder.

The PR channel is simply an intersymbol interference channel which can be modeled as a trellis. The factor graph representation of a trellis is a generalization of Wiberg's work [46]. Symbols, states, and constraints are represented by three different kinds of nodes. A state or

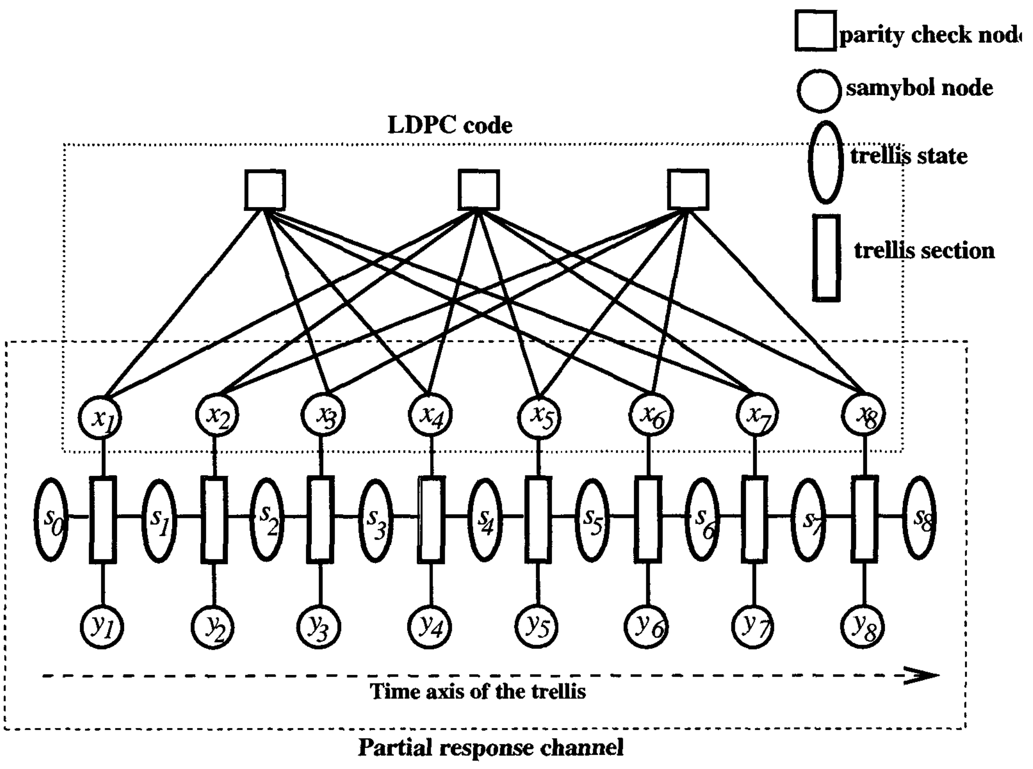

parity check nod
samybol node
trellis state
trellis section
LDPC code
$x_1$ $x_2$ $x_3$ $x_4$ $x_5$ $x_6$ $x_7$ $x_8$
$s_0$ $s_1$ $s_2$ $s_3$ $s_4$ $s_5$ $s_6$ $s_7$ $s_8$
$y_1$ $y_2$ $y_3$ $y_4$ $y_5$ $y_6$ $y_7$ $y_8$
Time axis of the trellis
Partial response channel

**Figure 6.2:** Joint factor graph representation of the LDPC-coded PR system.

symbol node is connected to a constraint node if the corresponding state or symbol is involved in the corresponding constraint. The constraints are often called *trellis section* which depicts the rule of state transitions. Fig. 6.2 shows an example of a joint factor graph representation of the LDPC-coded PR system. In the jargon of factor graph theory, the parity-check node and the trellis section are called *function node*, while the symbol node and the trellis state are called *variable node.* Each variable node takes on values from the binary set $\{1, -1\}$ (symbol node) or from the state set (trellis-state node) whose size depends on the memory of the PR channel. The messages along an edge in both directions are defined as a set of probability assignments of the associated variable node on its respective alphabet space.

Note that, unlike the factor graph representation of an LDPC code, the factor graph representation of a trellis has an obvious time axis and no cycles in itself.

Within the framework of joint factor graph representation of the LDPC code and the PR channel, it is convenient to define the message-update equations, maintaining soft information throughout the joint decoding/detection procedure and properly treating extrinsic information. As the SPA may be applied to arbitrary factor graphs, cycle-free or not, it is natural to apply the SPA for joint LDPC decoding and for PR detection.

The SPA operates according to the following simple rule [67]: The message sent from a node $v$ on an edge $e$ is the product of the function at $v$ (or the unit function if $v$ is a variable node) with all messages received at $v$ on edges *other than* $e$, summarized for the variable associated with $e$.

Let $\mu_{v_v \to v_f}(v_v)$ denote the message sent from variable node $v_v$ to function node $v_f$, and let $\mu_{v_f \to v_v}(v_v)$ denote the message sent from function node $v_f$ to node $v_v$. Also, let $\mathcal{N}(v)\backslash s$ denote the set of neighbors of a given node $v$, except for $s$. Then the message computations performed by the SPA may be expressed as follows:
**variable to function:**

$$\mu_{v_v \to v_f}(v_v) = \prod_{s \in \mathcal{N}(v_v) \backslash v_f} \mu_{s \to v_v}(v_v) \tag{6.1}$$

**function to variable:**

$$\mu_{v_f \to v_v}(v_v) = \sum_{\sim\{v_v\}} \left( f(V) \prod_{s \in \mathcal{N}(v_f) \backslash v_v} \mu_{s \to v_f}(s) \right), \tag{6.2}$$

where $\sum_{\sim\{v_v\}}$ means that variable $v_v$ is excluded from the summation, and $V = \mathcal{N}(v_f)$, neighbors of $v_f$, is the set of arguments of the local function of $v_f$. For a parity-check node, the local function $f(V)$ is defined as an indicator function of all its associated symbol nodes, with 1 if and only if (iff) the parity check is satisfied with a valid configuration. For a trellis section, the local function is defined as an indicator function of its left-hand state, right-hand state, the associated input and output, with 1 iff a configuration represents a valid trellis transition.

## 6.3 Message-Passing Schedules

The joint factor graph representation of a LDPC-coded PR system provides a unified approach for analyzing existing iterative receivers and to designing new alternatives of varying complexity, latency and performance. As the SPA turns out to be the unifying algorithm to define message-passing updating equations, the main part of the work in designing the iterative receiver is to define message-passing schedules.

We assume that messages are synchronized with a global discrete-time clock, with at most one message passed on any edge in any given direction at each clock tick. Any such message effectively replaces or overwrites previous messages that might have been sent on the edge in the same direction. Every node can send a message at each clock tick along every edge simultaneously. One iteration means that all messages have to be updated (only) once, and, irrespective of a specific message-passing schedule, one iteration implies the same computational complexity. With these conventions, a message-passing schedule in a factor graph is a specification of messages to be passed during each clock tick, and we focus on the latency of one iteration.

As shown in Fig. 6.2, the joint factor graph representation of a LDPC-coded PR system consists of a factor graph of the LDPC code and a factor graph of the trellis for the PR channel. As the LDPC code is often constructed randomly and there is no explicit time axis inside its factor graph, the so-called flooding schedule is commonly used, calling for a message to pass in each direction over each edge at each clock tick. As such, one decoding iteration corresponds to one clock tick. An appealing practical aspect of the flooding schedule is that it breaks down the decoding procedure into small, independent, and parallel decoding functions (i.e. the symbol nodes and the parity-check nodes), potentially leading to a very-high-speed hardware LDPC decoder [173,174]. This property is particularly important in magnetic recording applications, where the data rate is typically high and the decoding latency has to be extremely low.

In the factor graph of the trellis for the PR channel, there exists an explicit time axis in that the detection of later states depends on all the former states and inputs. The most efficient algorithm, in terms of computational complexity, is the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [111], which is equivalent to the SPA with a forward-backward message-passing schedule (also known as serial schedule in [67]). As shown in Fig. 6.3, the forward (backward) message-passing schedule involves a serial chain from left to right (right to left), with only one message being passed in one direction to its next neighbor at each clock tick. The forward-backward computations of messages are recursive, whereby the later message depends on the former one, thus the decoding latency of performing one iteration is equal to $n$ clock ticks [1], where $n$ is the block length, which poses a significant challenge for the low-latency requirement in magnetic recording channels.

To remedy this issue, [169] proposed parallel state-based message-passing algorithms for the PR channel detection. This algorithm can, in essence, be understood as the SPA with a flooding schedule (referring to Fig. 6.4) operating on the factor graph representation of the trellis of the PR channel, namely, each node (in the trellis part) sends out a message along every associated edge simultaneously, and consequently, one iteration of channel detector corresponds to one clock tick. However, it was reported by a specific example that about 0.6 dB is lost due to the flooding schedule.

[1]Note that trellis-state nodes are of degree two and perform no computation (no latency): a message arriving on one (incoming) edge is simply forwarded to the other (outgoing) edge.

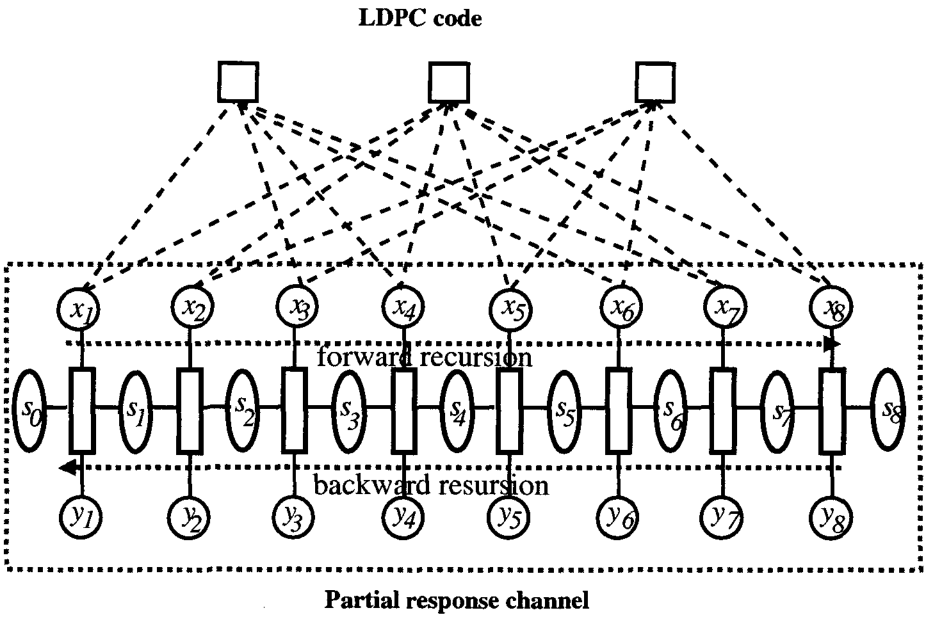


**Figure 6.3:** An example of the ordinary forward-backward message-passing schedule for the PR channel detection.

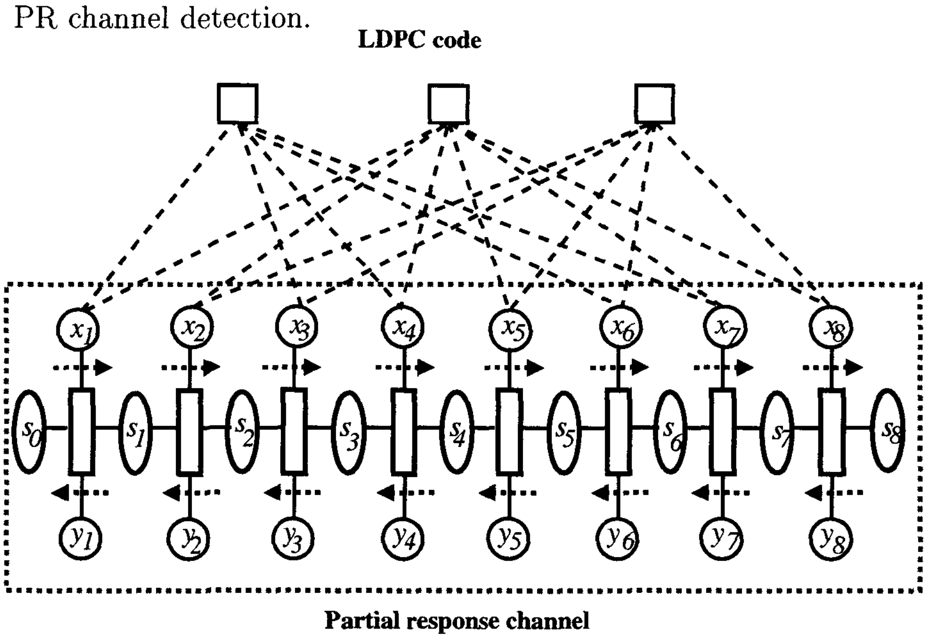


**Figure 6.4:** An example of the flooding message-passing schedule for the PR channel detection.

A possible cause of the performance degradation of the flooding schedule in the PR channel detection might be that the messages in the flooding schedule along the PR channel do not propagate as quickly as the forward-backward schedule does. For instance, within one iteration, each message can only have an influence on its neighbouring nodes in the flooding schedule. In contrast, these messages may have an influence on all nodes along the time axis in the forward-backward schedule, and moreover, the recursive updating of these messages yields the best possible channel detection if the interplay of the LDPC code is neglected. The drawback of the forward-backward schedule is its apparently high latency because of its serial nature.

In many cases it might be desirable to obtain a better balance between latency and performance. Here we propose a parallel windowed message-passing schedule for the PR channel detection that combines the advantages of the forward-backward schedule with the flooding schedule. The parallel windowed forward-backward (PWFB) schedule works as follows: Partition the factor graph of the PR channel into consecutive non-overlapped windows of size $w$. Within each window, the forward-backward message-passing schedule is used, and this schedule is synchronized and carried out in parallel on each window. Fig. 6.5 shows an example of the PWFB message-passing schedule with $w = 2$, where $w$ is counted in terms of the number of trellis sections inside each window. It is obvious that the computational complexity for one iteration of channel detection remains the same as those for the other two counterparts, and the latency for one iteration is equivalent to $w$ owing to the forward-backward schedule within each individual window.

Note that the flooding schedule and forward-backward schedule over the PR channel are two special cases of the PWFB message-passing schedule. If we select $w = 1$, the PWFB schedule reduces to the flooding schedule, and if we choose $w = n$, it becomes the ordinary forward-backward schedule.

The PWFB message-passing schedule considered so far used non-overlapped windows. It is, in principle, possible to incorporate overlapped windows in a parallel schedule. This would mean an increase in the computational complexity, as the overlapped nodes will be processed more than once in one iteration. However, the additional complexity may be justified by the increased performance gain. In [145], a sliding-window-based decoding schedule for MAP decoding of convolutional codes was described, with the aim of reducing memory requirements. The overlapped parts of the sliding windows in the forward as well as the backward direction were used to obtain refined messages for use in the non-overlapped parts, thereby delivering better performance.

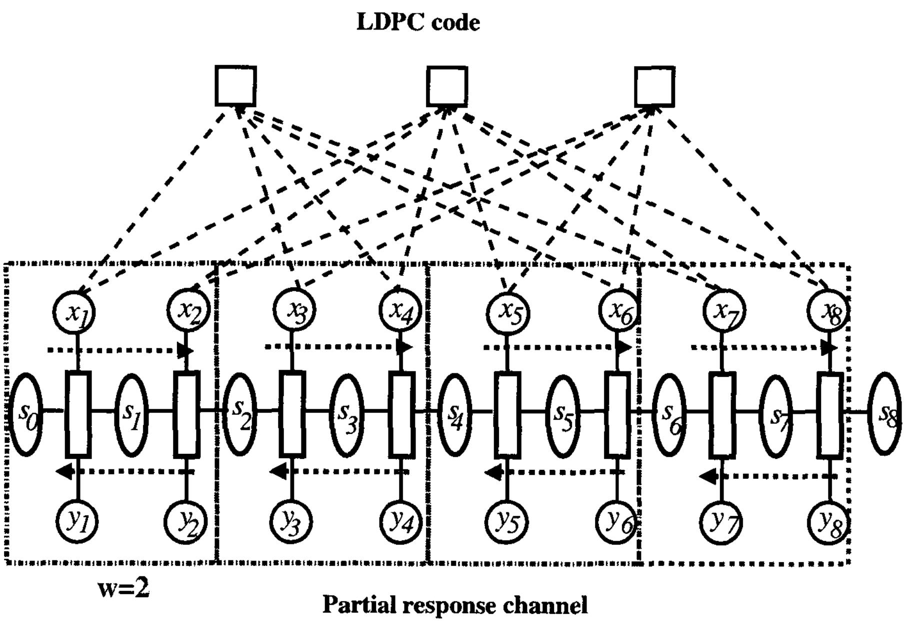

LDPC code
$x_1$ $x_2$ $x_3$ $x_4$ $x_5$ $x_6$ $x_7$ $x_8$
$s_0$ $s_1$ $s_2$ $s_3$ $s_4$ $s_5$ $s_6$ $s_7$ $s_8$
$y_1$ $y_2$ $y_3$ $y_4$ $y_5$ $y_6$ $y_7$ $y_8$
w=2
Partial response channel

**Figure 6.5:** An example of the parallel windowed forward-backward (PWFB) message-passing schedule for the PR channel detection with $w = 2$.

## 6.4 Simulation Results

The error-rate performance of the different message-passing schedules applied to the LDPC-coded PR system has been studied by computer simulations. The PR channel is assumed to be EPR4. Each round of iteration includes one iteration of the LDPC decoder and one iteration of the PR channel detector. For the LDPC decoder, the flooding schedule is assumed and aligned to the last clock tick of the PR channel detector. Therefore the total latency of the joint decoder/detector is dominated by that of channel detector. The joint iterative decoding/detection procedure is run for a maximum of 12 iterations, halting earlier if a valid codeword is found. Note that the computational complexity of each iteration remains the same and is independent of the specific message-passing schedule. The LDPC code is a regular one with column weight 3, of rate 7/8, and block length 495, and is from MacKay [109].

Fig. 6.6 compares the performance of the LDPC-coded PR channel for various message-passing schedules. Although the local updating equations remain the same, i.e., specified by the SPA, the message-passing schedules over the PR channel play an obvious role in determining the performance. It can be seen that the ordinary forward-backward schedule, although is of the largest latency, achieves the best performance, and the flooding schedule, which has the lowest latency, suffers about 0.6 dB of performance loss. Furthermore, the PWFB schedule with different window sizes, offers a series of trade-offs between the latency and performance. In particular, it is shown in this specific example that the 0.6 dB performance loss can be

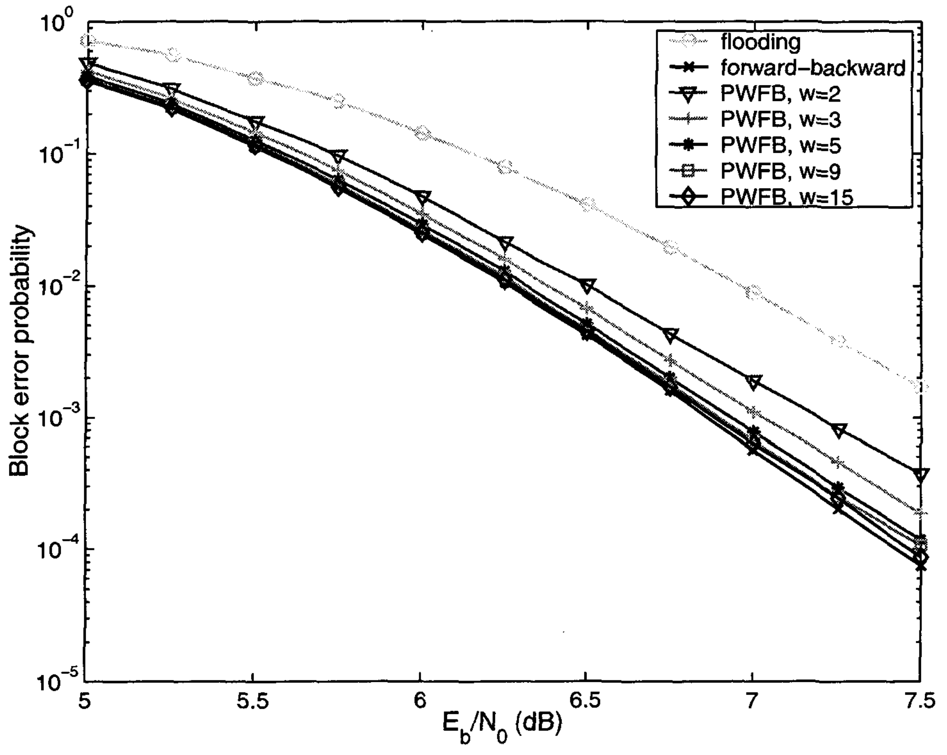


**Figure 6.6:** Performance of the LDPC-coded EPR4 channel for various message-passing schedules.

*recovered by the PWFB message-passing schedule with window size* $w = 3$ or $w = 5$.

The standard iterative decoding of turbo codes, the BCJR algorithm, can be visualized as an instance of the SPA with the forward-backward message-passing schedule over the factor graph representation of each constituent code [67], and concurrent turbo decoding [171] is an instance of the SPA with the flooding schedule to achieve low-latency decoding. Similarly, we can apply the PWFB message-passing schedule to the decoding of turbo codes, whose factor graphs are essentially two concatenated trellises. Here we studied the performance of turbo codes in the CDMA2000 standard with various message-passing schedules. The turbo encoder consists of two systematic, recursive, 8-state convolutional encoders concatenated in parallel, with an interleaver. The transfer function for the encoder is $G(D) = \begin{bmatrix} 1 & \frac{n_0(D)}{d(D)} & \frac{n_1(D)}{d(D)} \end{bmatrix}$, where $d(D) = 1 + D^2 + D^3$, $n_0(D) = 1 + D + D^3$, and $n_1(D) = 1 + D + D^2 + D^3$. For each block, a total of six information bits are used to terminate the two convolutional encoders. For a detailed description of the standardized Turbo interleaver, we refer the reader to [110]. We take a turbo code of rate 1/2, a (information) block length of 512 bits over the bi-AWGN channel and run the SPA with different schedules for up to 12 iterations. The computational complexity of each schedule for one iteration remains the same, and we focus on trade-offs of latency against performance for various message-passing schedules, as depicted in Fig. 6.7. As

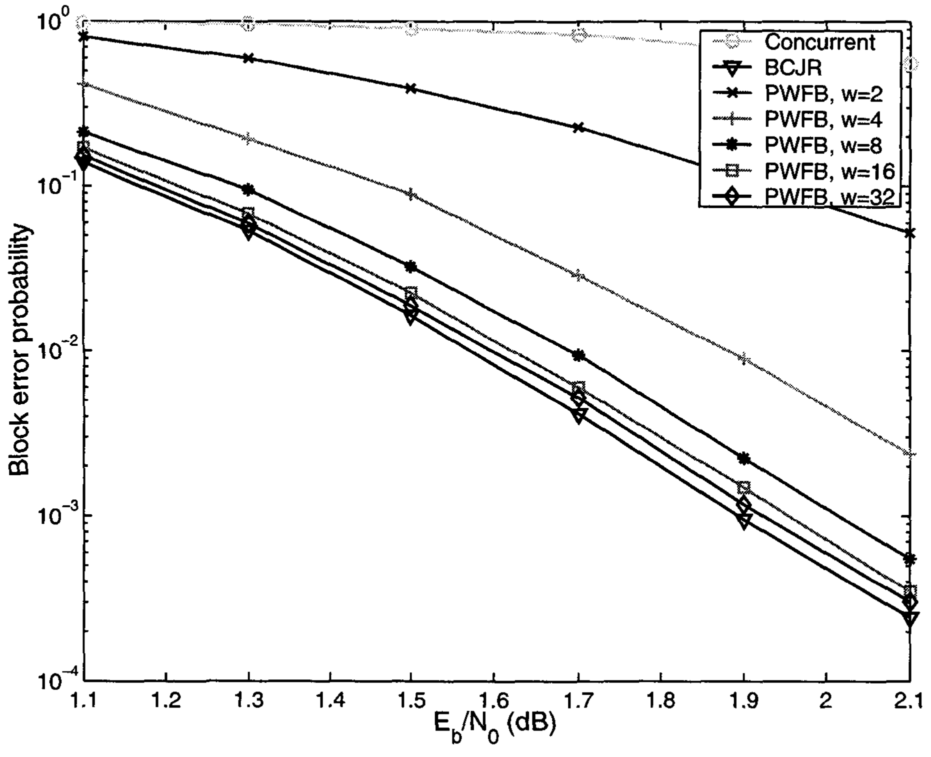


**Figure 6.7:** Performance of turbo decoding for various message-passing schedules over the AWGN channel.

expected, we observe a significant performance degradation when moving from the forward-backward (BCJR) schedule to the flooding (concurrent) schedule, and this performance loss can be recovered by the application of the PWFB message-passing schedule with a relatively small window size. Note that the parallel MAP algorithm for low-latency turbo decoding, recently proposed in [175], can be directly derived by the application of the PWFB message-passing schedule.

Although not explicitly investigated here, we would like to point out that the PWFB message-passing schedule can also be applied to the turbo-coded PR system, where the outer code is either a single convolutional code or a turbo code, and expect to achieve a good balance between latency, complexity and performance.

## 6.5 Summary

We have presented a new message-passing schedule for turbo equalization in the LDPC-coded PR system, called parallel windowed forward-backward message-passing schedule. The PWFB message-passing schedule combines the forward-backward schedule within a window with the flooding schedule on the window level. We have shown by simulation that it yields a good trade-off between latency, complexity, and performance, with a judicious selection of the window size, in the LDPC-coded PR system. In particular, a very low latency can be achieved without suffering a noticeable loss in the joint decoding/detection performance or an increase in computational complexity. The application of the PWFB message-passing schedule to turbo decoding/equalization has also been addressed.

While the joint detection/decoding example considered in the preceding section employed the EPR4 channel, we are also investigating PWFB message-passing schedule for more practical generalized PR channels, e.g., based on shaping polynomials of the form $(1-D^2)(1+p_1D+p_2D^2)$. We expect that the general approach outlined in this chapter will facilitate achieving the appropriate trade-off in terms of latency, computational complexity and performance.

# Chapter 7

# Concluding Remarks

## 7.1 Conclusion

Error-correcting codes are at the center of information theory since Shannon determined the capacity of ergodic channels. A major goal of coding theory is to find a class of error-correcting codes, together with the corresponding decoding algorithms, that realize the promise of Shannon's noisy channel-coding theorem in a practical way. The traditional picture was that good codes are easy to construct but difficult to decode. With the advent of turbo codes and LDPC codes together with iterative decoding, this old picture has evolved into a new one — asymptotically good codes are both easily constructable and decodable. It is a remarkable fact that all known practical, capacity-approaching coding schemes are now understood to be codes defined on graphs, particularly sparse graphs, together with the associated iterative decoding algorithm — the sum-product algorithm. A historic landmark was reached by Chung, Forney, Richardson, and Urbanke, who designed a random ensemble of rate-1/2 LDPC codes having a threshold within 0.0045 dB of the Shannon limit, and a performance within 0.036 dB of the Shannon limit at a bit-error rate of $10^{-6}$ with a block length of $10^7$. It seems unlikely that further improvements could have any practical significance in situations where very long block lengths are tolerable or affordable.

However, little is known on finite-block-length coding theory and practice, a situation that is tougher but of profound practical significance. In the finite-block-length "arena", the first theoretical issue is the characterization of the channel capacity — the ultimate limit of the maximum transmission rate. It turns out that the channel SNR, code block length, and target block-error rate (NOT bit error rate) are fundamental factors in determining the capacity of finite-block-length constrained channels. Moreover, good codes, in terms of optimum decoding, are again easily constructable if both the block length and the product of the block length and code rate are reasonably large.

There are two concentration theorems stating that asymptotically large random sparse graphs can be assumed to be effectively cycle-free and their performance concentrates on the average of the random ensemble, which are key ingredients in the design of capacity-approaching LDPC codes with extremely long block lengths. In the finite-block-length "arena", the concentration theorems weaken in the following two senses: random sparse graphs inevitably experience non-negligible deviation that depends on the block length and the sparsity of the underlying graph. The shorter the block length and the sparser the underlying graph, the more significant the deviation of the performance. Furthermore, the cycle-free assumption is no longer valid. The first argument advocates the invention of well-defined expurgated random ensembles in which "bad" deviation is precluded, and the second argument motivates the construction of good codes "for optimum decoding" and "for iterative decoding" simultaneously.

The ensemble of PEG Tanner graphs can be viewed as an expurgated random ensemble which precludes "bad" graphs containing short cycles and low-weight codewords in their resulting LDPC codes. The PEG algorithm is simple, flexible and yet powerful enough to generate good LDPC codes of short and moderate block lengths. The PEG construction essentially attains the same girth as Gallager's explicit construction for regular graphs, both of which meet or exceed an analogy of the Erdös–Sachs bound. Empirical results show that in conjunction with optimum symbol-node-degree distributions (obtained from either the empirical or the density evolution approach) the PEG construction yields the best binary LDPC codes (under iterative decoding) at short block lengths known to date.

Binary LDPC codes at short to moderate block lengths are destined not to be "good codes for optimum decoding" because iterative decoding requires them to have a rather sparse graph representation. The iterative decoding performance of a binary LDPC code is in general dominated by two conflicting factors. One is the Hamming weight spectrum, which requires the density of its parity-check matrix to be higher, the other is the performance loss due to iterative decoding in the presence of short cycles, for which the lower density of the parity-check matrix is favorable. The two conflicting requirements are often difficult to balance, particularly for short block lengths. It is, however, demonstrated that the Hamming weight spectrum can be improved by moving from binary field to fields of higher order while maintaining the underlying graph to be sparsest possible. Therefore cycle Tanner graph TG(2, $d_c$) codes defined over GF($2^b$), with sufficiently large $b$, are heuristically both "good codes for optimum decoding" and "good codes for iterative decoding".

Codes on sparse graphs are often decoded iteratively by the sum-product algorithm with low complexity. We investigate efficient digital implementations of the SPA for decoding binary LDPC codes in the log-likelihood ratios domain, and describe new reduced-complexity derivatives thereof. We examine LLR-SPA from both the architectural and the algorithmic point of view. Serial and parallel architectures for the parity-check-node update are proposed, leading to trellis and tree topologies, respectively. In both cases, specific core operations similar to the special operations defined in the log-likelihood algebra of Hagenauer are used. This framework not only leads to reduced-complexity LDPC decoding algorithms that can be implemented with simple comparators and adders but also provides the ability to compensate for the loss in performance by utilizing simple piecewise-linear approximations or constant correction factors. The unified treatment of decoding techniques for LDPC codes provides flexibility in selecting the appropriate design point in high-speed applications in terms of performance, latency, and computational complexity.

## 7.2 Outlook

This thesis presents a systematic exposition on the finite-block-length coding theory and practice, and many interesting issues are left open. We list here some opportunities for future research.

- The treatment of $(\epsilon, n)$-capacity in its current stage is preliminary. Very little is known concerning the capacity of the cases where $nR$ and/or $n$ are small. Quite certain is the fact that geometric constructions in these cases will yield better codes than the random construction, and thus the random coding bound for lower bounding the $(\epsilon, n)$-capacity becomes rather loose. In addition, our numerical examples rely heavily on the use of Shannon's spheric random coding bound and sphere-packing bound, which are limited to the AWGN channel. For the binary-input AWGN channel, little is known on how to evaluate the $(\epsilon, n)$-capacity.

- The asymptotic analysis of PEG construction relies on a series of relaxations. Although the relaxed PEG is shown to be asymptotically good, the relaxations weaken the luster of PEG in its minimum distance as an expurgated random ensemble in the finite-block-length cases. Direct analysis of the PEG ensemble is strongly expected to shed light on "expurgation".

- Compared with the decoding algorithm of binary LDPC codes, the decoding algorithm for GF($2^b$) codes on sparse Tanner graphs is much more complex. Reduced-complexity algorithms are thus desirable. In addition, a density evolution approach for designing good irregular degree sequences for GF($2^b$) codes on sparse Tanner graphs is still lacking.

- The determination of the minimum distance of an instance of an ensemble of LDPC codes is often thought to be a rather difficult task except in some trivial cases. This issue becomes a big concern in applications requiring extremely low block-error rates, for instance, magnetic recoding channels and optical transmission. A possible route might be to devise an approximation or randomized algorithm to search for minimum-weight codewords. As we finalized this thesis, we learned of the error-impulse method [176,177] which might be applicable to LDPC codes in order to yield a reasonably good estimate of the minimum distance.

# Appendix

## A: Evaluation of $\frac{\Gamma(\frac{n}{2}+1)}{\Gamma(\frac{n+1}{2})}$

It would be rather difficult to evaluate $\frac{\Gamma(\frac{n}{2}+1)}{\Gamma(\frac{n+1}{2})}$ for a large block length $n$ by direct calculation. In this appendix, we present an alternative method to evaluate it approximately.

By Stirling's formula, we have

$$\Gamma(z) \approx e^{-z} z^{z-\frac{1}{2}} (2\pi)^{\frac{1}{2}} \left[ 1 + \frac{1}{12z} + \frac{1}{288z^2} - \frac{139}{51840z^3} - \frac{571}{2488320z^4} + \cdots \right]$$

for $z \to \infty$. Substituting this formula into $\frac{\Gamma(\frac{n}{2}+1)}{\Gamma(\frac{n+1}{2})}$, and after some algebra, we obtain

$$\frac{\Gamma(\frac{n}{2}+1)}{\Gamma(\frac{n+1}{2})} \approx \frac{(\frac{n}{2}+1)^{1/2}(1+\frac{1}{n+1})^{n/2}}{e^{1/2}} \frac{[1+\frac{1}{12z}+\frac{1}{288z^2}-\frac{139}{51840z^3}-\frac{571}{2488320z^4}]_{z=\frac{n}{2}+1}}{[1+\frac{1}{12z}+\frac{1}{288z^2}-\frac{139}{51840z^3}-\frac{571}{2488320z^4}]_{z=\frac{n+1}{2}}}.$$

If $n$ is sufficiently large, we obtain a simplified asymptotic version

$$\begin{aligned} \frac{\Gamma(\frac{n}{2}+1)}{\Gamma(\frac{n+1}{2})} &\approx \frac{(\frac{n}{2}+1)^{1/2}(1+\frac{1}{n+1})^{n/2}}{e^{1/2}} \\ &\approx \sqrt{\frac{n}{2}} + o(1/\sqrt{n}). \end{aligned}$$

It is worth pointing out that the evaluation of $\frac{\Gamma(\frac{n}{2}+1)}{\Gamma(\frac{n+1}{2})}$ considered above also applies to non-integer $n$ as does the Stirling's formula.

## B: Normalized Chi-Square Distribution

Let $X$ be Gaussian distributed with zero mean and variance $\sigma^2$. Setting $Y = X^2$, we obtain the pdf of $Y$ in the form

$$p_Y(y) = \frac{1}{\sqrt{2\pi y}\sigma} e^{-y/2\sigma^2}, \quad y \geq 0. \tag{B.1}$$

The cdf of $Y$ is

$$\begin{aligned} F_Y(y) &= \int_0^y p_Y(u) du \\ &= \frac{1}{\sqrt{2\pi y}} \int_0^y \frac{1}{\sqrt{u}} e^{-u/2\sigma^2} du, \end{aligned} \tag{B.2}$$

which cannot be expressed in closed form. The characteristic function, however, can be determined in closed form. It is [64]

$$\phi(jv) = \frac{1}{(1 - j2v\sigma^2)^{1/2}}. \tag{B.3}$$

Now, suppose that the random variable $Y$ is defined as

$$Y = \frac{1}{n} \sum_{i=1}^{n} X_i^2, \tag{B.4}$$

where the $X_i$, $i = 1, 2, \cdots, n$, are statistically independent and identically distributed Gaussian random variables with zero mean and variance $\sigma^2$. As a consequence of the statistical independence of the $X_i$, the characteristic function of $Y$ is

$$\phi_Y(jv) = \frac{1}{(1 - j2v\sigma^2/n)^{n/2}}. \tag{B.5}$$

The inverse transform of this characteristic function yields the pdf

$$p_Y(y) = \frac{1}{(2\sigma^2)^{n/2}\Gamma(n/2)} n^{n/2} y^{n/2-1} e^{-ny/2\sigma^2}, \quad y \geq 0, \tag{B.6}$$

where $\Gamma(p)$ is the gamma function defined as

$$\begin{aligned} \Gamma(p) &= \int_0^\infty t^{p-1} e^{-t} dt. \quad p > 0 \\ \Gamma(p) &= (p-1)!, \quad p \text{ an interger}, \ p > 0 \\ \Gamma(\tfrac{1}{2}) &= \sqrt{\pi}, \quad \Gamma(\tfrac{3}{2}) = \tfrac{1}{2}\sqrt{\pi}. \end{aligned}$$

This pdf, which is a generalization of the chi-square distribution, is called a normalized chi-square pdf with $n$ degrees of freedom. It is illustrated in Fig. B.1.

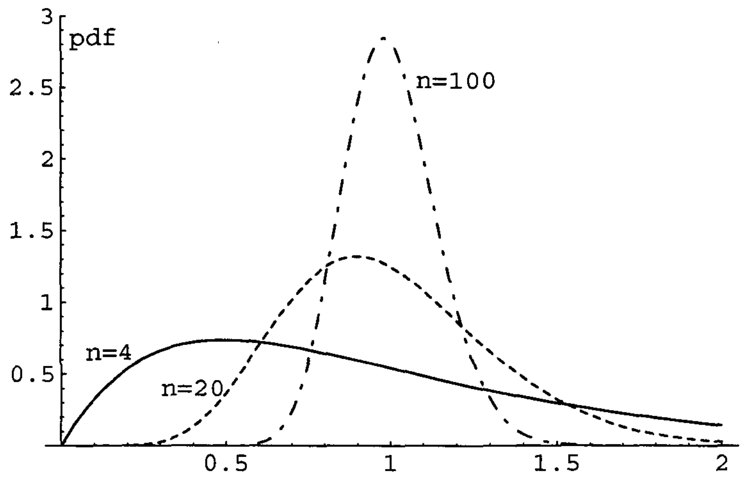


**Figure B.1:** The pdf of a normalized chi-square-distributed random variable for several degrees of freedom with $\sigma^2 = 1$.

The first two moments of $Y$ are

$$\begin{aligned} E(Y) &= \sigma^2 \\ E(Y^2) &= \frac{2\sigma^4}{n} + \sigma^4 \\ \sigma_y^2 &= \frac{2\sigma^4}{n}. \end{aligned}$$

The cdf of $Y$ is

$$F_Y(y) = \int_0^y \frac{1}{(2\sigma^2)^{n/2}\Gamma(n/2)} n^{n/2} u^{n/2-1} e^{-nu/2\sigma^2} du, \quad y \geq 0. \tag{B.7}$$

When $n$ is even, this integral can be expressed in closed form. Specifically, let $m = \frac{1}{2}$, where $m$ is an integer. Then, by repeated integration by parts, we obtain

$$F_Y(y) = 1 - e^{-my/\sigma^2} \sum_{k=0}^{m-1} \frac{1}{k!} \left(\frac{my}{\sigma^2}\right)^k, \quad y \geq 0. \tag{B.8}$$

Substituting $N$, $(N+2P)$, and $2P$ for $\sigma^2$ in Eq. (B.6) yields Eqs. (2.18), (2.19), and (2.30), respectively.

## C: Shannon–Gallager–Berlekamp Sphere-Packing Bound

One common way of providing a performance limit of finite-block-length codes over the binary-input AWGN channel is via the Shannon–Gallager–Berlekamp approach [15]. Let $n$ be the code length of a block, $R$ the code rate in bits/symbol, and $\epsilon$ the probability of making a block error at the decoder, i.e. decoding the wrong code word. For rates below the channel capacity, Shannon, Gallager, and Berlekamp showed that $\epsilon$ (of the best codes) is lower-bounded by

$$\begin{aligned} \epsilon &> 2^{-n(E_{sp}(R)+o(n))}; \\ E_{sp}(R) &= \max_q \max_{\rho \geq 0} [E_0(\rho, q) - \rho R], \end{aligned} \tag{C.1}$$

where $q$ is the distribution of the input symbols, and

$$E_0(\rho, q) = -\log_2 \int_y \left[ \sum_x q(x) p(y|x)^{1/(1+\rho)} \right]^{1+\rho}. \tag{C.2}$$

In Eq. (C.2), $x$ is the input signal to the channel, $y$ is the channel output signal, and $p(y|x)$ is the conditional probability of the channel output signal, given the input. Eq. (C.1) is known as the Gallager exponent.

For the AWGN channel, $y = x + z$, where $z$ is uncorrelated zero-mean Gaussian noise with variance $\sigma^2$ and

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{(y-x)^2}{2\sigma^2} \right). \tag{C.3}$$

In the case of binary phase-shift keying (BPSK) signaling, we have $x = \pm\sqrt{E_s}$ and it is known that the maximizing distribution $q(x)$ is given by $q(x) = 1/2$, for all $x$, and

$$E_0(\rho, q) = -\log_2 \int_y \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{y^2 + E_s}{2\sigma^2} \right) \left[ \cosh\left( \frac{2y\sqrt{E_s}}{2\sigma^2(1+\rho)} \right) \right]^{1+\rho}. \tag{C.4}$$

Note that the term $o(n)$ in Eq. (C.1) plays an appreciable role only for short block lengths, and that it is often difficult to calculate and thus neglected.

## D: Tangential Sphere Bound (TSB)

The tangential sphere bound is an upper bound on the block-error rate of a linear code under optimum decoding. It can be shown [134] that the TSB is always tighter than the tangential bound [178] and the union bound, especially for low and moderate values of $E_b/N_0$. Suppose that the signals transmitted through an AWGN channel for each message (represented by a codeword of a linear block code $C$) are of the same energy $E$. The energy of each signal is $E = nE_s$, where $n$ is the block length and $E_s$ is the energy transmitted per symbol. Denote the noise variance by $\sigma^2$. The derivation of TSB follows by the central inequality

$$\text{Prob}(A) \leq \text{Prob}(\underline{z} \in B, A) + \text{Prob}(\underline{z} \notin B), \tag{D.1}$$

where $A$ is an event that represents a decoding block error, $B$ is an $n$-dimensional cone with half angle $\theta$ and radius $r = \sqrt{nE_s}\tan\theta$, and $\underline{z}$ is the noise vector added to the transmitted signal.

The TSB on the block-error probability $\epsilon$ is based only on the distance spectrum $\{S_k\}$ of the binary linear block code $C$ and reads, following the notation of [135],

$$\begin{aligned}\epsilon \;\leq\; & \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(-z_1^2/2\sigma^2)} \left\{ \left[ 1 - \overline{\gamma}\left( \frac{n-1}{2}, \frac{r_{z_1}^2}{2\sigma^2} \right] \right) \right. \\ & \left. + \sum_{k:\delta_k/2<\alpha_k} S_k \left[ Q\left( \frac{\beta_k(z_1)}{\sigma} \right) - Q\left( \frac{r_{z_1}}{\sigma} \right) \right] \overline{\gamma}\left( \frac{n-2}{2}, \frac{r_{z_1}^2 - \beta_k^2(z_1)}{2\sigma^2} \right) \right\} dz_1, \end{aligned} \tag{D.2}$$

where $Q(\cdot)$ is the Gaussian $Q$-function defined by Eq. (2.32), $\overline{\gamma}(a, x)$ is the normalized incomplete Gamma function given by

$$\overline{\gamma}(a, x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt, \tag{D.3}$$

for positive values of $a$ and $x$. In Eq. (D.2), the following representations are also used.

$$\begin{aligned} r_{z_1} &= \left( 1 - \frac{z_1}{\sqrt{nE_s}} \right) r \\ \beta_k(z_1) &= \frac{\delta_k r_{z_1}}{2r\sqrt{1 - \frac{\delta_k^2}{4nE_s}}} \\ \alpha_k &= r\sqrt{1 - \frac{\delta_k^2}{4nE_s}}, \end{aligned} \tag{D.4}$$

where $\delta_k$ is defined to be the Euclidean distance between two signals whose corresponding codewords differ in $k$ symbols ($k \leq n$). Thus for the case of BPSK, $\delta_k = 2\sqrt{kE_s}$.

The upper bound Eq. (D.2) is valid for all positive values of $r$ and thus the optimal radius $r$ in the sense of achieving the tightest upper bound is determined by setting the derivative of the right side of the bound to zero, yielding the following optimization equation

$$\begin{aligned} \theta_k &= \cos^{-1}\left(\frac{\delta_k}{2r\sqrt{1-\frac{\delta_k^2}{4nE_s}}}\right) \\ \sum_{k:\delta_k/2<\alpha_k} S_k \int_0^{\theta_k} \sin^{n-3}\phi d\phi &= \frac{\sqrt{\pi}\Gamma(\frac{n-2}{2})}{\Gamma(\frac{n-1}{2})}, \end{aligned} \tag{D.5}$$

where

$$\Gamma(x) = \int_0^{\infty} t^{x-1}e^{-t}dt, \; x \geq 0 \tag{D.6}$$

designates the Gamma function.

# Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623-656, July and October 1948. Reprinted in *Claude Elwood Shannon: Collected Papers*, pp. 5-83, (N. J. A. Sloane and A. D. Wyner, eds.) Piscataway: IEEE Press, 1993.

[2] C. E. Shannon, "Communication in the presence of noise," *Proc. IRE*, vol. 37, pp. 10–21, Jan. 1949.

[3] R. G. Gallager, "Low-density parity-check code," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, 1962.

[4] R. G. Gallager, *Low Density Parity Check Codes.* MIT Press, Cambridge, MA, 1963.

[5] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sept. 1981.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Intl. Conf. Commun. (ICC), Geneva, Switzerland*, pp. 1064–1070, 1993.

[7] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in *Cryptography and Coding, 5th IMA Conference (Lecture Notes in Computer Science)*, pp. 100–111, 1995. C. Boyd, Ed. Berlin, Germany: Springer vol. 1025.

[8] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *IEE Elect. Lett.*, vol. 32, pp. 1645–1646, Aug. 1996. Reprinted in *Elect. Lett.*, vol. 33, pp. 457–458, Mar. 1997.

[9] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.

[10] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[11] E. N. Gilbert, "A comparison of signaling alphabets," *Bell System Technical Journal*, vol. 31, pp. 504–522, May 1952.

[12] P. Elias, “Coding for noisy channels,” *IRE Convention Record*, vol. 3, pp. 37–46, 1955.

[13] R. G. Gallager, “Simple derivation of the coding theorem and some applications,” *IEEE Trans. Inform. Theory*, vol. 11, pp. 3–18, Jan. 1965.

[14] A. D. Wyner, “Capacity of the band-limited Gaussian channel,” *Bell System Technical Journal*, vol. 45, pp. 359–371, Mar. 1965.

[15] C. E. Shannon, R. G. Gallager, and E. R. Berlekamp, “Lower bounds to error probability for coding on discrete memoryless channels: Part I and II,” *Inform. Control.*, vol. 10, pp. 65–103 and 527–552, January and May 1967.

[16] G. D. J. Forney, “Exponential error bounds for erasure, list, and decision-feedback schemes,” *IEEE Trans. Inform. Theory*, vol. 14, pp. 206–220, Mar. 1968.

[17] A. J. Viterbi, “Error bounds for white Gaussian and other very noisy memoryless channels with generalized decision regions,” *IEEE Trans. Inform. Theory*, vol. 15, pp. 279–287, Mar. 1969.

[18] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York: Wiley, 1991.

[19] R. W. Hamming, “Error detecting and error correcting codes,” *Bell System Technical Journal*, vol. 29, pp. 147–160, Apr. 1950.

[20] M. J. E. Golay, “Note on digital coding,” *Proc. IRE*, vol. 37, p. 657, June 1949.

[21] D. E. Muller, “Application of Boolean algebra to switching circuit design and to error detection,” *IRE Trans. Electronic Comput.*, vol. 3, pp. 6–12, Sept. 1954.

[22] I. S. Reed, “A class of multiple-error correcting codes and the decoding scheme,” *IRE Trans. Inform. Theory*, vol. 4, pp. 38–49, Sept. 1954.

[23] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *SIAM J.*, vol. 8, pp. 300–304, June 1960.

[24] A. Hocquenghem, “Codes correcteurs d’erreurs,” *Chiffres*, vol. 2, pp. 147–156, 1959.

[25] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Inform. Control*, vol. 3, pp. 68–79, Mar. 1960.

[26] R. C. Bose and D. K. Ray-Chaudhuri, “Further results in error correcting binary group codes,” *Inform. Control*, vol. 3, pp. 279–290, Sept. 1960.

[27] G. D. J. Forney, *Concatenated Codes.* MIT Press, Cambridge, Mass, 1966.

[28] V. D. Goppa, “New class of linear correcting codes,” *Probl. Peredach. Inform.*, vol. 6, pp. 24–30, 1970.

[29] V. D. Goppa, "Rational presentation of codes and (L, g)-codes," *Probl. Peredach. Inform.*, vol. 7, pp. 41–49, 1971.

[30] J. Justesen, "A class of constructive asymptotically good algebraic codes," *IEEE Trans. Inform. Theory*, vol. 18, pp. 652–656, Sept. 1972.

[31] W. W. Peterson, "Encoding and error-correction precedures for Base-Chauhuri codes," *IRE Trans. Inform. Theory*, vol. 6, pp. 459–470, Sept. 1960.

[32] R. T. Chien, "Cyclic decoding procedures for BCH codes," *IEEE Trans. Inform. Theory*, vol. 10, pp. 357–363, Oct. 1964.

[33] G. D. J. Forney, "On decoding BCH codes," *IEEE Trans. Inform. Theory*, vol. 11, pp. 549–557, Oct. 1965.

[34] J. L. Massey, "Step-by-step decoding of the BCH codes," *IEEE Trans. Inform. Theory*, vol. 11, pp. 580–585, Oct. 1965.

[35] E. R. Berlekamp, *Algebraic Coding Theory.* McGraw-Hill, New York, 1968.

[36] J. M. Wozencraft and B. Rieffen, *Sequential Decoding.* MIT Press, Cambridge, Mass., 1961.

[37] R. M. Fano, "A heuristic discussion of probabilistic coding," *IEEE Trans. Inform. Theory*, vol. 9, pp. 64–74, Apr. 1963.

[38] K. S. Zigangirov, "Some sequential decoding precedures," *Probl. Peredach. Inform.*, vol. 2, pp. 13–25, 1970.

[39] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM J. Res. Dev.*, vol. 13, pp. 675–685, Nov. 1969.

[40] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. 13, pp. 260–269, Apr. 1967.

[41] V. Zyablov and M. Pinsker, "Estimation of the error-correction complexity of Gallager low-density codes," *Probl. Peredach. Inform.*, vol. 11, pp. 23–26, 1975.

[42] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, pp. 71–78, 1982.

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

[44] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence: 29*, pp. 241–288, 1986.

[45] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* San Francisco, CA:Kaufmann, 1988. 2nd ed.

[46] N. Wiberg, *Codes and Decoding in General Graphs.* PhD thesis, Linköping Univ., Linköping, Sweden, 1996.

[47] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *European. Trans. Telecommun.*, vol. 6, pp. 513–526, Sept. 1995.

[48] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 140–152, Feb. 1998.

[49] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 219–230, Feb. 1998.

[50] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, vol. 46, pp. 325–343, Mar. 2000.

[51] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proc. 29th ACM Symp. Theory of Computing, El Paso, TX*, pp. 150–159, May 1997.

[52] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of low-density codes and improved designs using irregular graphs," in *Proc. 30th ACM STOC, Dallas, TX*, pp. 249–258, 1998.

[53] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes and using irregular graphs and belief propagation," in *Proc. IEEE Intl. Symp. Inform. Theory, Cambridge, MA*, p. 117, Aug. 1998.

[54] S.-Y. Chung, G. D. F. Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, pp. 58–60, Feb. 2001.

[55] B. J. Frey, R. Koetter, G. D. J. Forney, F. R. Kschischang, R. J. McEliece, and D. A. Spielman, "Special issue on codes and graphs and iterative algorithms," *IEEE Trans. Inform. Theory*, vol. 47, pp. 493–849, Feb. 2001.

[56] S. Verdu, "Fifty years of Shannon theory," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2057–2077, Oct. 1998.

[57] T. Berger, *Rate-Distortion Theory: A Mathematical Basis for Data Compression.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[58] S. Shamai (Shitz) and S. Verdu, "The empirical distribution of good codes," *IEEE Trans. Inform. Theory*, vol. 43, pp. 836–846, May 1997.

[59] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell System Technical Journal*, vol. 38, pp. 611–656, May 1959.

[60] S. Dolinar, D. Divsalar, and F. Pollara, "Code performance as a function of block size," *TMO Progress Report 42-133, Jet Propulsion Laboratory, Pasadena, CA*, pp. 1–23, May 1998.

[61] S. J. MacMullan and O. M. Collins, "A comparison of known codes, random codes, and the best codes," *IEEE Trans. Inform. Theory*, vol. 44, pp. 3009–3022, Nov. 1998.

[62] R. G. Gallager, *Information Theory and Reliable Communication.* New York: Wiley, 1968.

[63] N. Alon, J. Spencer, and P. Erdos, *The Probabilistic Method.* New York: Wiley, 1992.

[64] J. G. Proakis, *Digital Communications.* Third edition, New York: McGraw-Hill, 1995.

[65] N. Alon and M. Luby, "A linear-time erasure-resilient code with nearly optimal recovery," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1732–1736, Nov. 1996.

[66] R. Kötter and A. Vardy, "Factor graphs: Construction, classification and bounds," in *Proc. IEEE Intl. Symp. Inform. Theory, Cambridge, MA,*, p. 14, 1998.

[67] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum–product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[68] A. R. Calderbank, G. D. Forney, and A. Vardy, "Minimal tail-biting trellises: The Golay code and more," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1435–1455, July 1999.

[69] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Computation*, vol. 12, pp. 1–41, 2000.

[70] G. D. J. Forney, "Codes on graphs: News and views," in *Proc. 2nd Intl. Symp. on Turbo Codes and Related Topics, Brest, France*, pp. 9–16, Sept. 2000.

[71] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[72] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometrices: a rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711–2736, Nov. 2001.

[73] R. Lucas, M. P. C. Fossorier, Y. Kou, and S. Lin, "Iterative decoding of one-step majority logic decodable codes based on belief propagation," *IEEE Trans. Communications*, vol. 48, pp. 931–937, June 2000.

[74] R. Tanner, D. Srkdhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. of ISTA 2001, Ambleside, England*, 2001.

[75] J. L. Fan, "Array codes as low-density parity-check codes," in *Proc. 2nd Intl. Symp. on Turbo Codes and Related Topics, Brest, France*, Sept. 2000.

[76] B. Vasic, "Combinatorial constructions of structured low-density parity-check codes for iterative decoding," 2001. submitted for publication.

[77] S. J. Johnson and S. R. Weller, "Regular low-density parity-check codes from combinatorial designs," in *Proc. IEEE Inform. Theory Workshop, Cairns, Australia*, Sept. 2001.

[78] G. A. Margulis, "Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators," *Problemy Peredachi Inform.*, vol. 24, pp. 51–60, 1988.

[79] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs," *Combinatorica*, vol. 8, pp. 261–277, 1988.

[80] P. Erdös and H. Sachs, "Reguläre Graphen gegebene Taillenweite mit minimaler Knotenzahl," *Wiss. Z. Univ. Hall Martin Luther Univ. Halle - Wittenberg Math. - Natur. Reine*, pp. 12:251–257, 1963.

[81] J. Lafferty and D. Rockmore, "Codes and iterative decoding on algebraic expander graphs," in *Proc. IEEE Intl. Symp. Inform. Theory and Applications (ISITA), Honolulu, Hawaii, USA*, Nov. 2000.

[82] J. Rosenthal and P. O. Vontobel, "Construction of LDPC codes based on Ramanujan graphs and ideas from Margulis," in *Proc. 38th Annual Allerton Conf. on Communication, Computing and Control, Monticello, IL*, Oct. 2000.

[83] P. O. Vontobel and R. M. Tanner, "Construction of codes based on finite generalized quadrangles for iterative decoding," in *Proc. IEEE Intl. Symp. Inform. Theory, Washington, DC*, June 2001.

[84] D. J. C. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes," *Electronic Notes in Theoretical Computer Science*, vol. 74, 2002.

[85] S. L. S. Jacoby, J. S.Kowalik, J. T. Pizzo, and W. T. Veterling, *Iterative Methods for Nonlinear Optimization Problems.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.

[86] L. Bazzi, T. Richardson, and R. Urbanke, "Exact thresholds and optimal codes for the binary symmetric channel and Gallager's decoding algorithm A," *IEEE Trans. Inform. Theory.* to appear.

[87] M. A. Shokrollahi, "New sequence of linear time erasure codes approaching the channel capacity," in *Proc. 13th Intl. Sym. on Applied Algebra, Algebraic Algorithm and Error-correcting Codes*, 1999.

[88] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Veterling, *Numerical Recipes in C.* Cambridge, 1988.

[89] M. C. Davey, *Error-Correction Using Low-Density Parity-Check codes.* PhD thesis, University of Cambridge, Dec. 1999.

[90] D. Spielman, "Linear-time encodeable and decodable error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1723–1731, Nov. 1996.

[91] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of construction of irregular Gallager codes," *IEEE Trans. Communications*, vol. 47, pp. 1449–1454, Oct. 1999.

[92] L. Ping, W. K. Leung, and N. Phamdo, "Low density parity check codes with semi-random parity check matrix," *IEE Elect. Lett.*, vol. 35, pp. 38–39, Jan. 1999.

[93] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat–accumulate codes," in *Proc. 2nd Intl. Symp. on Turbo Codes and Related Topics, Brest, France*, pp. 1–8, Sept. 2000.

[94] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[95] M. C. Davey and D. MacKay, "Low-density parity-check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, pp. 165–167, June 1999.

[96] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM), San Antonio, Texas, USA*, Nov. 2001.

[97] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Irregular progressive edge-growth Tanner graphs," in *Proc. 4th Intl. ITG Conf. on Source and Channel Coding, Berlin*, Jan. 2002.

[98] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Irregular progressive edge-growth Tanner graphs," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[99] J. Campello, D. S. Modha, and S. Rajagopalan, "Designing LDPC codes using bit-filling," in *Proc. IEEE Intl. Conf. Commun. (ICC), Helsinki, Finland*, June 2001.

[100] R. M. Tanner, "Minimum distance bounds by graph analysis," *IEEE Trans. Inform. Theory*, vol. 47, pp. 808–821, Feb. 2001.

[101] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang, "Stopping sets and the girth of Tanner graphs," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[102] C. Di, D. Proietti, E. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570–1579, June 2002.

[103] J. Zhang and A. Orlitsky, "Finite-length analysis of LDPC codes with large left degrees," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[104] S. Aji, H. Jin, A. Khandekar, D. J. C. MacKay, and R. J. McEliece, "BSC thresholds for code ensembles based on "Typical Pairs"decoding," in *Codes, Systems, and Graphical Models :* the IMA volumes in mathematics and its applications, edited by B. Marcus and J. Rosenthal, Springer, pp. 195–210, 2001.

[105] Y. Mao and A. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," in *Proc. IEEE Intl. Conf. Commun. (ICC), Helsinki, Finland*, pp. 41–44, 2001.

[106] E. Eleftheriou, T. Mittelholzer, and A. Dholakia, "Reduced-complexity decoding algorithm for low-density parity-check codes," *IEE Elect. Lett.*, vol. 37, pp. 102–104, Jan. 2001.

[107] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum–product algorithm for decoding LDPC codes," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM), San Antonio, Texas, USA*, Nov. 2001.

[108] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Near optimal reduced-complexity decoding algorithms for LDPC codes," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[109] D. J. C. MacKay, *Online database of low-density parity-check codes.* `http://wol.ra.phy.cam.uk/mackay/codes/data.html`.

[110] *Standards for CDMA2000 Spread Spectrum Systems.* EIA/TIA IS-2000, 1-6.

[111] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[112] M. Bossert, *Channel Coding for Telecommunications.* John Wiley & Sons, 1999.

[113] A. Vardy and Y. Be'ery, "Bit-level soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 37, Mar. 1991.

[114] A. R. Hammons, P. V. Kumar, A. R. Calderbank, N. J. A. Sloane, and P. Sole, "The $\mathbb{Z}_4$-linearity of Kerdock, Preparata, Goethals, and related code," *IEEE Trans. Inform. Theory*, vol. 40, pp. 301–319, Feb. 1994.

[115] C. Schlegel and L. Perez, "On error bounds and turbo codes," *IEEE Commun. Lett.*, vol. 3, pp. 205–207, July 1999.

[116] A. A. Nechaev, "Kerdock codes in a cyclic form," *Discrete Math. Appl.*, vol. 1, pp. 365–384, Apr. 1991.

[117] A. R. Calderbank, G. McGuire, P. V. Kumar, and T. Helleseth, "Cyclic codes over $\mathbb{Z}_4$, locator polynomials and Newton's identities," *IEEE Trans. Inform. Theory*, vol. 42, pp. 217–226, Jan. 1996.

[118] W. W. Peterson and E. J. W. Jr., *Error-Correcting Codes.* Cambridge, Mass. The MIT Press, 1972.

[119] D. Jungnickel and S. A. Vanstone, "Graphical codes revisited," *IEEE Trans. Inform. Theory*, vol. 43, pp. 136–146, Jan. 1997.

[120] L. Decreusefond and G. Zémor, "On the error-correcting capabilities of cycle codes of graphs," *Combinatorics, Probability and Computing*, vol. 6, pp. 27–38, 1997.

[121] J.-P. Tillich and G. Zémor, "Optimal cycle codes constructed from Ramanujan graphs," *SIAM Journal on Discrete Math.*, vol. 10, pp. 447–459, Mar. 1997.

[122] G. Zémor, "On iterative decoding of cycle codes of graphs," in *Codes, Systems, and Graphical Models :* the IMA volumes in mathematics and its applications, edited by B. Marcus and J. Rosenthal, Springer, pp. 311–326, 2001.

[123] J. Blomer, R. Karp, and E. Welzl, "The rank of sparse random matrices over finite fields," *Research Report, International Computer Science Institute, Berkeley, California*, 1996.

[124] N. Sourlas, "Spin-glass models as error-correcting codes," *Nature*, vol. 338, pp. 693–695, 1989.

[125] P. Rujan, "Finite temperature error-correcting codes," *Physical Review Letters*, vol. 70, pp. 2968–2971, 1993.

[126] H. Nishimori, "Optimum decoding temperature for error-correcting codes," *Journal of the Physical Society of Japan*, vol. 62, pp. 2973–2975, 1993.

[127] M. L. Belongie, "Spin galsses and error-correcting codes," *TMO Progress Report 42-118, Jet Propulsion Laboratory, Pasadena, CA*, pp. 26–36, Aug. 1994.

[128] J. S. Yedidia, "An idiosyncratic journey beyond mean field theory," *Research Report TR-2000-27, Mitsubishi Electric Research Laboratories*, 2000.

[129] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Research Report TR-2001-22, Mitsubishi Electric Research Laboratories*, 2001.

[130] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Bethe free energy, Kikuchi approximations, and belief propagation algorithms," *Research Report TR-2001-16, Mitsubishi Electric Research Laboratories*, 2001.

[131] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Characterization of belief propagation and its generalizations," *Research Report TR-2001-15, Mitsubishi Electric Research Laboratories*, 2001.

[132] P. J. Cameron, *Combinatorics: Topics, Techniques, Algorithms.* Cambridge University Press, 1994.

[133] G. Poltyrev, "Bounds on the decoding error probability of binary linear codes via their spectra," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1284–1292, July 1994.

[134] H. Herzberg and G. Poltyrev, "The error probability of $M$-ary PSK block coded modulation schemes," *IEEE Trans. Communications*, vol. 44, pp. 427–433, Apr. 1996.

[135] I. Sason and S. Shamai (Shitz), "Improved upper bounds on the ensemble performance of ML decoded low-density parity-check codes," *IEEE Commun. Lett.*, vol. 4, pp. 89–91, Mar. 2000.

[136] J. Hagenauer, M. Moerz, and E. Offer, "Analog turbo-networks in VLSI: The next step in turbo decoding and equalization," in *Proc. 2nd Intl. Symp. on Turbo Codes and Related Topics, Brest, France*, Sept. 2000.

[137] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog VLSI," *IEEE Trans. Inform. Theory*, vol. 47, pp. 837–843, Feb. 2001.

[138] L. Ping and W. Leung, "Decoding low density parity check codes with finite quantization bits," *IEEE Commun. Lett.*, vol. 4, pp. 62–64, Feb. 2000.

[139] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low density parity check codes based on belief propagation," *IEEE Trans. Communications*, vol. 47, pp. 673–680, May 1999.

[140] T. Mittelholzer, A. Dholakia, and E. Elftheriou, "Reduced-complexity decoding of low density parity check codes for generalized partial response channels," *IEEE Trans. Magnetics*, vol. 37, pp. 721–728, Mar. 2001.

[141] V. Sorokine, F. Kschischang, and S. Pasupathy, "Gallager codes for CDMA applications – part II: Implementations, complexity, and system capacity," *IEEE Trans. Communications*, vol. 48, pp. 1818–1828, Nov. 2000.

[142] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity-check codes," *IEEE Trans. Communications*, vol. 50, pp. 406–414, Mar. 2002.

[143] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, pp. 208–210, May 2002.

[144] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.

[145] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[146] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Intl. Conf. Commun. (ICC)*, pp. 1009–1013, June 1995.

[147] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM), San Antonio, Texas, USA*, Nov. 2001.

[148] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *IEE Elect. Lett.*, vol. 34, pp. 1577–1578, Aug. 1998.

[149] X. Wei and A. N. Akansu, "Density evolution for low-density parity-check codes under Max-Log-MAP decoding," *IEE Elect. Lett.*, vol. 37, pp. 1125–1126, Aug. 2001.

[150] X.-Y. Hu and T. Mittelholzer, "An ordered-statistics-based approximation of the sum-product algorithm," in *Proc. Intl. Telecommun. Symp. (ITS), Brazil*, 2002.

[151] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching.* Addison-Wesley, 1973.

[152] S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.

[153] L. Perez, J. Seghers, and D. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1698–1709, Nov. 1996.

[154] S. t. Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Communications*, vol. 49, pp. 1727–1737, Oct. 2001.

[155] W. E. Ryan, "Performance of high rate turbo codes on a PR4-equalized magnetic recording channels," in *Proc. IEEE Intl. Conf. Commun. (ICC), Atlanta, GA*, pp. 947–951, June 1998.

[156] W. E. Ryan, L. L. McPheters, and S. W. McLaughlin, "Combined turbo coding and turbo equalization for PR4-equalized Lorentzian channels," in *Proc. Conf. Info. Sci. Sys., Princeton, NJ.*, pp. 489–493, Mar. 1998.

[157] C. Heegard, "Turbo coding for magnetic recording," in *Proc. IEEE Inform. Theory Workshop, San Diego, CA*, pp. 18–19, Feb. 1998.

[158] J. L. Fan, A. Friedmann, E. Kurtas, and S. W. McLaughlin, "Low density parity check codes for magnetic recording," in *Proc. 37th Annual Allerton Conf. Communication, Control, and Computing*, pp. 1314–1323, 1999.

[159] T. Souvignier, A. Friedmann, M. Oberg, P. H. Siegel, R. E. Swanson, and J. K. Wolf, "Turbo decoding for PR4: parallel versus serial concatenation," in *Proc. IEEE Intl. Conf. Commun. (ICC), Vancouver, Canada*, pp. 1638–1642, June 1999.

[160] T. Souvignier, M. öberg, P. H. Siegel, R. E. Swanson, and J. K. Wolf, "Turbo decoding for partial response channels," *IEEE Trans. Communications*, vol. 48, pp. 1297–1308, Aug. 2000.

[161] H. Song, R. M. Todd, and J. R. Cruz, "Low density parity check codes for magnetic recording channels," *IEEE Trans. Magnetics*, pp. 2183–2186, Sept. 2000.

[162] A. Dholakia, E. Eleftheriou, and T. Mittelholzer, "On iterative decoding for magnetic recording channels," in *Proc. 2nd Intl. Symp. on Turbo Codes and Related Topics, Brest, France*, pp. 219–225, Sept. 2000.

[163] A. Dholakia, E. Eleftheriou, and T. Mittelholzer, "Iterative detection/decoding techniques for the magnetic recording channel," in *Dig. 11th Annu. Magnetic Recording Conf. (TMRC 2000), Santa Clara, CA*, Aug. 2000.

[164] D. Hoesli and E. Svensson, "Low-density parity-check codes for magnetic recording," diploma thesis, no. 7103, Signal and Information Processing Lab., ETH Zurich, Switzerland, Mar. 2000.

[165] H. Song, R. M. Todd, and J. R. Cruz, "Applications of low-density parity-check codes to magnetic recording channels," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 918–923, May 2001.

[166] B. Vasic, "Structured iteratively decodable codes based on Steiner systems their application in magnetic recording," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM), San Antonio, Texas, USA*, Nov. 2001.

[167] N. Varnica and A. Kavčić, "Optimized LDPC codes for partial response channels," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[168] B. Vasic, A. Kuznetsov, and E. Kurtas, "Lattice low-density parity-check codes and their application in partial response systems," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[169] B. M. Kurkoski, P. H. Siegel, and J. K. Wolf, "Joint message-passing decoding of LDPC codes and parital-response channels," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1410–1422, June 2002.

[170] J. Garcia-Frias, "Decoding of low-density parity-check codes over finite state binary Markov channels," in *Proc. IEEE Intl. Symp. Inform. Theory, Washington, DC*, p. 72, June 2001.

[171] B. Frey, F. Kschischang, and P. Gulak, "Concurrent turbo decoding," in *Proc. IEEE Intl. Symp. Inform. Theory, Ulm, Germany*, July 1997.

[172] J. Sun, O. Y. Takeshita, and M. P. Fitz, "A highly parallel decoder for turbo codes," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[173] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE J. Solid-State Circuits*, vol. 37, pp. 404–412, Mar. 2002.

[174] E. Yeo, P. Pakzad, B. Nikolić, and V. Anantharam, "VLSI architectures for iterative decoders in magnetic recording channels," *IEEE Trans. Magnetics*, vol. 37, pp. 748–755, Mar. 2001.

[175] S. Yoon and Y. Bar-Ness, "A parallel MAP algorithm for low latency turbo decoding," *IEEE Commun. Lett.*, vol. 7, pp. 288–290, July 2002.

[176] C. Berrou and S. Vaton, "Computing the minimum distance of linear codes by the error impulse method," in *Proc. IEEE Intl. Symp. Inform. Theory, Lausanne, Switzerland*, July 2002.

[177] C. Berrou, S. Vaton, M. Jézéquel, and C. Douillard, "Computing the minimum distance of linear codes by the error impulse method," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM), Taiwan (to appear)*, Nov. 2002.

[178] E. R. Berlekamp, "The technology of error correction codes," *Proc. IEEE*, vol. 68, pp. 564–593, May 1980.

# Biography

Xiaoyu Hu was born in a tiny beautiful village in China in 1970. After attending middle and high school in Sichuang province, in 1987 he joined East China Institute of Technology (ECIT) at Nanjing to study electrical engineering, with major in automation. He graduated with a Bachelor and a Master degree in automation from ECIT in 1991 and 1993, respectively. After a couple of years as a teaching/research assistant with ECIT and Southeast University in the same city, where he was involved in the design and implementation of digital signal processing algorithms for low-bit-rate speech codec, IS-95 mobile station receiver, and digital TV receiver, he came to Switzerland in 1998 and participated in the doctoral school of the department of Computer and Communication Systems at the Swiss Federal Institute of Technology Lausanne (EPFL). From 1999 to 2002 he held a Pre-Doc position at IBM Research, Zurich Research Laboratory, working on signal processing and coding issues of xDSL and magnetic recording channels. He received a post-graduate certificate in 1999 and his Ph. D in 2002 from EPFL.