2015

New Identification and Decoding Techniques for Low-Density Parity-Check Codes

Tian Xia
Louisiana State University and Agricultural and Mechanical College, tiaxian@gmail.com

NEW IDENTIFICATION AND DECODING TECHNIQUES FOR LOW-DENSITY PARITY-CHECK CODES

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The School of Electrical Engineering and Computer Sciences

by
Tian Xia
B.S., University of Electronic Science and Technology of China, 2008
M.S., University of Electronic Science and Technology of China, 2011
M.S., Louisiana State University, 2013
May 2015

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Dr. Hsiao-Chun Wu. This work cannot be fulfilled without his kind and precious guidance. Dr. Wu's profound knowledge and constant encouragement inspire and motivate me to pursue the challenging but interesting questions encountered in this work. His academic serious and respectable personality will surely have a lasting impact for my future career.

I also would like to thank my committee members Dr. Xuebin Liang, Dr. Xin Li, Dr. Supratik Mukhopadhyay, and Dr. Frank Tsai for their invaluable time and constructive suggestions to improve this work. I would like to thank the division of the electrical and computer engineering as well for building a great learning environment during my study.

Moreover, I would like to thank my former group members Dr. Yonas G. Debessu and Ms. Hongting Zhang. They generously share their experience and knowledge not only in directions of research topics but also in details of daily life. They also helped me a lot by leaving me useful books and driving me to buy groceries, just to name a few.

I am also grateful to the Graduate School of Louisiana State University for offering me the distinguished Dissertation Year Fellowship. Part of this work was developed under this financial assistance.

Finally, I would like to say thanks to my parents who raised me in their unconditional love. Their endless support keeps me focused on my research and lets me continue to chase my dreams. Their patience and diligence are absolutely reflected in every aspect of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Error-correction coding schemes are indispensable for high-capacity high data-rate communication systems nowadays. Among various channel coding schemes, low-density parity-check (LDPC) codes introduced by pioneer Robert G. Gallager are prominent due to the capacity-approaching and superior error-correcting properties. There is no hard constraint on the code rate of LDPC codes. Consequently, it is ideal to incorporate LDPC codes with various code rate and codeword length in the adaptive modulation and coding (AMC) systems which change the encoder and the modulator adaptively to improve the system throughput. In conventional AMC systems, a dedicated control channel is assigned to coordinate the encoder/decoder changes. A questions then rises: if the AMC system still works when such a control channel is absent. This work gives positive answer to this question by investigating various scenarios consisting of different modulation schemes, such as quadrature-amplitude modulation (QAM), frequency-shift keying (FSK), and different channels, such as additive white Gaussian noise (AWGN) channels and fading channels.

On the other hand, LDPC decoding is usually carried out by iterative belief-propagation (BP) algorithms. As LDPC codes become prevalent in advanced communication and storage systems, low-complexity LDPC decoding algorithms are favored in practical applications. In the conventional BP decoding algorithm, the stopping criterion is to check if all the parities are satisfied. This single rule may not be able to identify the undecodable blocks, as a result, the decoding time and power consumption are wasted for executing unnecessary iterations. In this work, we propose a new stopping criterion to identify the undecodable blocks in the

early stage of the iterative decoding process. Furthermore, in the conventional BP decoding algorithm, the variable (check) nodes are updated in parallel. It is known that the number of iterations can be reduced by the serial scheduling algorithm. The informed dynamic scheduling (IDS) algorithms were proposed in the existing literatures to further reduce the number of iterations. However, the computational complexity involved in finding the update node in the existing IDS algorithms would not be neglected. In this work, we propose a new efficient IDS scheme which can provide better performance-complexity trade-off compared to the existing IDS ones.

In addition, the iterative decoding threshold, which is used for differentiating which LDPC code is better, is investigated in this work. A family of LDPC codes, called LDPC convolutional codes, has drawn a lot of attentions from researchers in recent years due to the threshold saturation phenomenon. The IDT for an LDPC convolutional code may be computationally demanding when the termination length goes to thousand or even approaches infinity, especially for AWGN channels. In this work, we propose a fast IDT estimation algorithm which can greatly reduce the complexity of the IDT calculation for LDPC convolutional codes with arbitrary large termination length (including infinity). By utilizing our new IDT estimation algorithm, the IDTs for LDPC convolutional codes with arbitrary large termination length (including infinity) can be quickly obtained.

1. INTRODUCTION

In this chapter, we give a brief introduction of low-density parity-check (LDPC) codes and the conventional iterative belief-propagation algorithms used for LDPC decoding. The following chapters of this work are developed upon these fundamental concepts. For much wider and deeper details on LDPC codes, the reader is referred to [2–4] and the references therein.

1.1 LDPC Codes

LDPC codes were introduced by Robert G. Gallager in 1960s [3]. An LDPC code is defined by a *sparse* parity-check matrix (PCM). The sparsity implies that the number of non-zero entries in the PCM increase linearly rather than quadratically with respect to the codeword length. Denote a sparse PCM by $\mathbf{H}$ with dimension $m \times n$ ($m < n$). An LDPC is defined by $\mathbf{H}$ if and only if each codeword, denoted by $\mathbf{c}$ with dimension $n \times 1$, satisfies

$$\mathbf{Hc} = \mathbf{0}, \tag{1.1}$$

where $\mathbf{0}$ is all-zero vector with dimension $m \times 1$. The corresponding code rate $R \geq 1 - m/n$, where the equality hold when all the rows in $\mathbf{H}$ are independent. If all the non-zero entries in $\mathbf{H}$ are 1, then $\mathbf{H}$ defines a binary LDPC code; if all the non-zero entries in $\mathbf{H}$ are from finite field with order q ($q > 2$) , denoted by $\mathbb{GF}(q)$, then $\mathbf{H}$ defines a nonbinary LDPC code over $\mathbb{GF}(q)$.

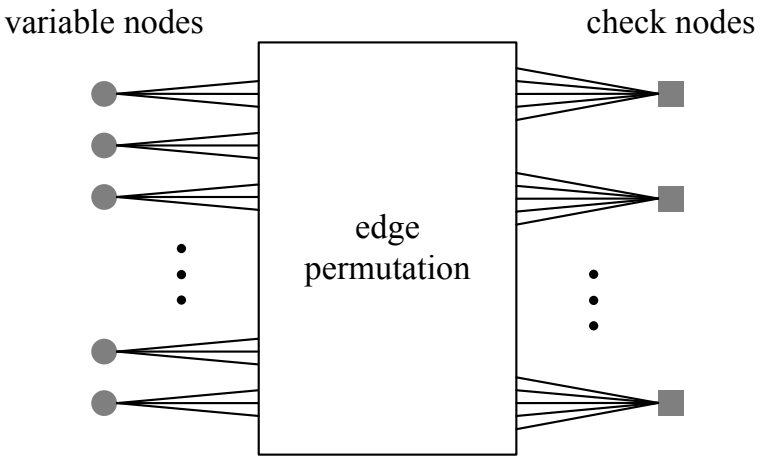

Figure 1.1: An example of a Tanner graph representation for a regular LDPC code.

LDPC codes are one of the graph codes. Specifically, an LDPC code's PCM $\mathbf{H}$ can also be represented by a bipartite graph, which is also called a Tanner graph [5]. In the corresponding Tanner graph, the j^{th} column of $\mathbf{H}$ is represented by a variable node j, the i^{th} row of $\mathbf{H}$ is represented by a check node i, and there is an edge between a variable node j and a check node i if the entry is non-zeros in the i^{th} row and j^{th} column of $\mathbf{H}$. It is inevitable to have cycles in the corresponding Tanner graph when constructing LDPC codes [2]. The minimum length of any cycles in a Tanner graph is called *girth*. Usually, LDPC codes are constructed carefully to avoid cycles with length 4 (the girth is thus at least 6), since short cycles are unfavorable for the iterative LDPC decoding algorithms and impairs the bit-error rate (BER) performance.

In a Tanner graph, if every variable node has degree d_l and every check node has degree d_r, the corresponding LDPC code is called *regular* ; otherwise, it is *irregular*. An example of a Tanner graph representation for a regular LDPC code is depicted in Figure 1.1. Note that there is an edge permutation operation in Figure 1.1 to permute edge connections between variable nodes and check nodes. Given all possible edge permutation instances, an LDPC

ensemble is then formed. It is of interest to investigate an LDPC ensemble rather than a particular instance due to the concentration property when codeword length n grows [6].

An LDPC ensemble is characterized by a *degree distribution pair* [7]. Give a specific degree distribution pair, the *iterative decoding threshold* (IDT) can then be determined by the *density evolution* technique [6] or the *extrinsic information transfer* (EXIT) chart analysis [8]. The IDT indicates the best possible performance of an LDPC code under iterative decoding and can then be utilized for LDPC code design. Usually, carefully designed irregular LDPC codes have better iterative decoding thresholds than regular LDPC codes.

The encoding procedure of an LDPC code is usually not straightforward. A efficient encoding scheme was proposed in [9] for general LDPC codes. For practical applications, it is favorable to employ quasi-cyclic LDPC codes whose PCM is constructed by concatenating circulant sub-matrix [10]. The constraint can greatly simplify the encoding process and the corresponding circuit design [11, 12].

It is worth mentioning that in recent years LDPC convolutional codes, also called *spatially-coupled LDPC codes*, have drawn a lot of attentions from both academia and industry [13]. A remarkable phenomenon, called *threshold saturation*, is observed for terminated LDPC convolutional codes when the termination length goes large [14]. In detail, the iterative decoding threshold (IDT) of an LDPC convolutional code can approach the maximum *a posteriori* (MAP) decoding threshold as the termination length increases.

1.2 Iterative BP Decoding

The superior error-correction performance of LDPC codes is offered by the iterative belief-propagation (BP) decoding algorithms. When a Tanner graph of an LDPC code has no

cycles, the BP decoding is optimal and can be accomplished in one iteration. As mentioned above, to construct LDPC codes to be good in finite lengths, cycles are inevitable but short cycles of length 4 should be eliminated. Consequently, the BP algorithms has to be carried out iteratively for decoding LDPC codes, and in general, the iterative BP decoding is not optimal anymore. Here, we illustrate the iterative BP decoding procedure for binary LDPC codes. For nonbinary LDPC decoding, the reader is referred to [15–18].

The iterative BP decoding process can be described over the Tanner graph. Each variable node (each column of the PCM $\mathbf{H}$) is considered as a repetition code, and each check node (each row of $\mathbf{H}$) is considered as a single parity-check code. The soft extrinsic information messages, presented by the probabilities or log-likelihood ratios (LLR) which infer the beliefs of the received symbols being 0 or 1, are propagated between the variable nodes and the check nodes. Thus, the name belief propagation comes.

Consider the binary phase-shift keying modulation and additive white Gaussian noise (AWGN) channels. Denote the received symbol by r_j, $j = 1, 2, \ldots, n$. Denoted the extrinsic information in LLR from the variable node j to the check node i by $\alpha_{i,j}$. Denoted the extrinsic information in LLR from the check node i to the variable node j by $\beta_{i,j}$. Denote the LLR of *a posteriori* probability (APP) for the variable node j by ρ_j. The standard iterative BP algorithm for LDPC decoding can thus be described as follows [4].

Step 1 Initialization: The LLR input to the LDPC decoder can be represented by

$$\mu_j = \frac{2ar_j}{\sigma^2}, \quad j = 1, 2, \ldots, n, \tag{1.2}$$

where a is the signal amplitude and σ^2 is the noise variance. For every edge connecting the variable node j to the check node i (every non-zero entry in PCM $\mathbf{H}$ in row i and column

j), initialize $\alpha_{i,j}$ by

$$\alpha_{i,j} = \mu_j. \tag{1.3}$$

Step 2 Check-node processing: At the check-node side, calculate $\beta_{i,j}$ using the incoming messages $\alpha_{i,j}$ by

$$\beta_{i,j} = \prod_{j' \in \mathcal{V}_i \backslash j} \operatorname{sign}(\alpha_{i,j'}) \, \phi\left(\sum_{j' \in \mathcal{V}_i \backslash j} \phi\Big(|\alpha_{i,j'}|\Big) \right), \tag{1.4}$$

where $\mathcal{V}_i \backslash j$ is the set of the variable nodes connected to the check node i except the variable node j, and the function $\phi(x)$ is expressed by

$$\phi(x) \stackrel{\text{def}}{=} \log\left(\frac{1+e^{-x}}{1-e^{-x}} \right), \quad x \geq 0. \tag{1.5}$$

Step 3 Variable-node processing: At the variable-node side, calculate $\alpha_{i,j}$ using the incoming messages $\beta_{i,j}$ by

$$\alpha_{i,j} = \mu_j + \sum_{i' \in \mathcal{C}_j \backslash i} \beta_{i',j}, \tag{1.6}$$

where $\mathcal{C}_j \backslash i$ is the set of the check nodes connected to the variable node j except the check node i.

Step 4 APP: Calculate the LLR of APP, ρ_j, which can be expressed by

$$\rho_j = \mu_j + \sum_{i \in \mathcal{C}_j} \beta_{i,j}, \quad j = 1, 2, \ldots, n, \tag{1.7}$$

where $\mathcal{C}_j$ is the set of the check nodes connected to the variable node j.

Step 5 Stopping rule: Perform hard decision on ρ_j to obtain the codeword estimation $\hat{\mathbf{c}}$. Carry out the syndrome check using Eq. (1.1). If all the parity check equations are satisfied, that is, $\mathbf{H}\hat{\mathbf{c}} = \mathbf{0}$, terminate the algorithm and output estimated codeword $\hat{\mathbf{c}}$. Otherwise, go back to Step 2 until the maximum iteration number, denoted by N_{iter}, is reached.

The aforementioned iterative BP decoding is called the *sum-product* algorithm in the logarithm domain [4]. The complexity burden lies at the check-node processing in the Step 2. Take a closer look at the function $\phi(x)$ defined by Eq.(1.5). It can be observed that the smallest $|\alpha_{i,j'}|$ dominates the sum in Eq.(1.4) [4]. That is,

$$\begin{aligned} \phi\left(\sum_{j'\in\mathcal{V}_i\backslash j}\phi\Big(|\alpha_{i,j'}|\Big)\right) &\approx \phi\left(\phi\Big(\min_{j'\in\mathcal{V}_i\backslash j}|\alpha_{i,j'}|\Big)\right) \\ &= \min_{j'\in\mathcal{V}_i\backslash j}|\alpha_{i,j'}|. \end{aligned} \tag{1.8}$$

Thus, replacing Eq.(1.4) by

$$\beta_{i,j} = \prod_{j'\in\mathcal{V}_i\backslash j}\operatorname{sign}\big(\alpha_{i,j'}\big)\min_{j'\in\mathcal{V}_i\backslash j}|\alpha_{i,j'}|, \tag{1.9}$$

we obtain the so called *min-sum* algorithm. Although the expensive calculation on $\phi(x)$ is avoided in the min-sum algorithm, there is certain BER performance degradation incurred by the approximation in Eq. (1.8) [4].

1.3 Motivation and Objectives

LDPC codes have been successfully adopted in various standards, such as the DVB-S2 (digital televisions) [19], the IEEE 802.11 WLAN (Wi-Fi) [20], 10 Gigabit Ethernet [21], etc. Research interests and applications of LDPC codes can also be found in advanced optical communications and modern data-storage systems [22, 23].

In the aforementioned standards, LDPC codes are defined by various code rates and codeword lengths. Consequently, the transceivers therein can employ different LDPC encoders/decoders according to the channel qualities. This is the so-called *adaptive modulation and coding* technique. Usually, there is a dedicated control channel to facilitate the changes

of encoders/decoders between transceivers, which complicates the transceiver design and impairs the spectral efficiency. To avoid such a control channel, blind LDPC encoder identification schemes are proposed in this work so that the receiver can blindly identify LDPC codes from a predefined LDPC encoder candidate set. We investigate various modulation schemes and channel models and assume that the receivers have no knowledge of the channel state information.

Furthermore, there are possibilities to reduce the computational complexity (number of iterations) of the standard iterative BP decoding algorithm described in Chapter 1.2 in following ways. Note that the conventional stopping rule (Step 5 in Chapter 1.2) cannot recognize undecodable blocks. As a result, when an undecodable block is experienced in the BP decoding, all available iterations will be exhausted and no legitimate codeword will be generated. To save the decoding time and the power consumption when an undecodable block is experienced, in this work, we devise a new stopping criterion for BP decoding, which can identify undecodable blocks and terminate the BP decoding process in an early stage. Moreover, the parallel scheduling method in the standard iterative BP decoding algorithm could be replaced by serial scheduling schemes. It is known that serial scheduling schemes can reduce the number of iterations and converge fast compared to the parallel (flooding) scheme. In this work, we propose an efficient dynamic scheduling scheme for BP decoding, which can further reduce the number of iterations compared to the existing dynamic scheduling algorithms.

In addition, the IDTs of LDPC convolutional codes with large termination lengths are computationally demanding to be determined, especially for the additive white Gaussian noise (AWGN) channel. Instead of using the existing protograph-based extrinsic informa-

tion transfer (PEXIT) algorithm to determine the IDTs for protograph-based LDPC convolutional codes, in this work, we propose a PEXIT-fast algorithm based on our new analysis and proofs on the monotonic properties involved in the PEXIT analysis of LDPC convolutional codes. The computational complexity can thus be greatly reduced for determining the IDTs of LDPC convolutional codes with arbitrary large termination lengths which include infinity.

The rest of this work is organized as follows. In Chapter 2 and Chapter 3, the blind identification schemes for LDPC codes are developed for different modulation formats and channels. Joint blind frame synchronization and LDPC encoder identification is also addressed in Chapter 3. In Chapter 4, two fast BP decoding algorithms are proposed to reduce the computational complexity (number of iterations). One is a new stopping criterion, and the other one is an efficient dynamic serial scheduling for LDPC decoding. In Chapter 5, we propose an efficiently IDT estimate algorithm for LDPC convolutional codes, which is useful especially for large termination length.

2. BLIND IDENTIFICATION OF LDPC CODES FOR AWGN CHANNELS

Adaptive modulation and coding (AMC) technologies exploit the channel state information (CSI) to improve the data rate (throughput) or enhance the bit-error-rate performance, especially in time-varying fading channels [24]. Based on the feedback CSI, the AMC transmitter dynamically selects an appropriate combination of modulator and channel encoder from the predefined candidate pool [25–28]. Instead of employing a dedicated *control channel* to update the changes in the modulation/demodulation and coding/decoding schemes in conventional AMC transceivers, people proposed *blind encoder identification* techniques in [29–34] and *blind modulation classification* schemes in [35–37] recently to boost the spectral efficiency and remove the corresponding control mechanisms (thus simplify the transceiver design) by using advanced signal processing methods.

It is known that the *redundancy* introduced in the existing coding schemes offers potentials for the receiver to blindly identify the unknown encoder adopted by an AMC transmitter. In [29, 30], the *space-time redundancy* of the received signal samples was exploited to distinguish the underlying coding schemes for flat- and frequency-selective fading channels, respectively. In [31], the receiver utilized the *parity-check* constraints to identify the original encoder. In [32, 33], the *blind* encoder identification schemes were developed for binary and nonbinary low-density parity-check (LDPC) codes over the additive white Gaussian noise (AWGN) channel, respectively.

In this chapter, we extend our previous work in [32] to blindly identify binary LDPC codes for M-quadrature amplitude modulation (M-QAM) signals over the additive white

Gaussian noise (AWGN) channel. The main contributions of this work are highlighted as follows. First, since the transmitted symbols change from BPSK modulation to M-QAM modulation, an unknown phase offset is introduced. The expectation-maximization (EM) algorithm is thus developed accordingly for estimating the unknown parameters, namely *signal amplitude*, *noise variance*, and *phase offset*. Second, the *a posteriori* probabilities (APPs) of the received signal symbols have to be transformed to the corresponding coded bits for facilitating the syndrome APP of binary LDPC codes subject to the mapping of M-QAM. This new framework involving the two aforementioned attributes enables our proposed blind binary LDPC encoder identification scheme to work reliably in the AMC systems where both modulation type and coding scheme change dynamically with respect to the channel state.

The rest of this chapter is organized as follows. The basic AMC transceiver system is introduced in Chapter 2.1. The blind LDPC encoder identification method for M-QAM signals and the associated EM algorithm are presented in Chapter 2.2. Monte Carlo simulation results are demonstrated in Chapter 2.3 to illustrate the effectiveness of our proposed new scheme.

2.1 System Model

In this section, we introduce the basic AMC system model for the transceivers involving a binary LDPC encoder and an M-QAM modulator. At the transmitter, original information bits are grouped into blocks, each of which consists of k consecutive bits, say $\mathbf{b}_\nu$, where ν is the *block index*. This block of information bits are passed to the binary LDPC encoder θ to generate a corresponding block of *codeword* or *coded bits*, say $\mathbf{c}_\nu^\theta$ with codeword length

n, where θ denotes a particular type of binary LDPC encoder. Obviously the corresponding code rate is $R = k/n$. Then, the codeword $\mathbf{c}_\nu^\theta$ is modulated by the M-QAM such that L ($L = \log_2 M$) consecutive coded bits form one M-QAM symbol. The corresponding block of modulated symbols to $\mathbf{c}_\nu^\theta$ are denoted by $\mathbf{s}_\nu^\theta$ with length $N = n/L$.

It is assume that the timing, frequency, and frame synchronizations are properly undertaken at the receiver frontend [38–40]. Thus, the received baseband signal symbols are also collected in blocks, say $\mathbf{r}_\nu$. We propose to feed $\mathbf{r}_\nu$ to our *blind encoder identification scheme* to identify θ, the unknown binary LDPC encoder adopted in the transmitter. Once the encoder type is identified by our proposed scheme as $\hat{\theta}_\nu$ where the subscript ν indicates that it is estimated from the ν^{th} block of received signal symbols, then the appropriate LDPC decoder can be employed to construct the information symbol estimates $\hat{\mathbf{b}}_\nu$. As our blind binary LDPC encoder identification scheme can rely on a single codeword block, the block index ν can be omitted for notational convenience in the rest of this chapter.

To establish the signal model, each element of one block of received baseband signal symbols, $\mathbf{r} \stackrel{\text{def}}{=} [r_1, r_2, \ldots, r_j, \ldots, r_N]^T$, can be expressed as

$$r_j = a\, e^{\imath\varphi} s_j^\theta + w_j, \quad j = 1, 2, \ldots, N, \tag{2.1}$$

where a is the unknown signal amplitude, $\imath \stackrel{\text{def}}{=} \sqrt{-1}$, φ is the unknown phase offset, s_j^θ is the M-QAM symbol generated from the encoder θ, and w_j is the zero-mean complex AWGN with independent real and imaginary parts both having the variance σ^2. Consequently, the *energy per information bit to the noise power spectrum density ratio* E_b/N_0 is given by

$$\frac{E_b}{N_0} = \frac{a^2}{2\sigma^2 LR}. \tag{2.2}$$

In practice, the AMC transceivers would not change their modulators and encoders arbitrarily but have a predefined modulator/encoder candidate set. In this chapter, we assume that a predetermined LDPC encoder candidate set, say Θ, which contains multiple encoder candidates, is known to both transmitter and receiver. We also assume that the encoders in Θ are different from each other so that the parity-check matrices of any two encoders do not share identical row(s). In the next section, we will present our scheme to blindly identify the binary LDPC encoder $\theta \in \Theta$ for M-QAM signals.

2.2 Blind LDPC Encoder Identification

Note that the unknown parameters, namely signal amplitude a, noise variance σ^2, and phase offset φ need to be estimated first in our blind binary LDPC encoder identification scheme (see [32]). According to the system model formulated by Eq. (2.1), we propose to adopt the EM algorithm to estimate all of them [41].

2.2.1 EM Estimation

When a maximum-likelihood estimation (MLE) problem is complicated, it is favorable to adopt the EM algorithm to find the optimal solution due to its monotonicity [41]. The received signal samples formulated by Eq. (2.1) comply with the Gaussian mixture model which the EM algorithm is built upon.

In the EM framework, the *missing data* are the transmitted symbols $\mathbf{s} = [s_1, s_2, \ldots, s_N]$. The *complete data* are denoted by $\mathbf{z} = [\mathbf{r}; \mathbf{s}]$. Let $\mathcal{C}$ denote the M-QAM constellation set where $x_m \in \mathcal{C}$ $(m = 1, 2, \ldots, M)$ represents the m^{th} constellation point. Each x_m corresponds to a *mode* in the Gaussian mixture. Here we assume that s_j is randomly picked from x_m

and therefore the probability weight of the m^{th} mode is $1/M$. The unknown parameter set is $\boldsymbol{\lambda} = [a, \sigma^2, \varphi]$. According to Eq. (2.1), the *conditional expected log-likelihood function* $Q(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)})$, where $\boldsymbol{\lambda}^{(t)}$ is the EM estimate in the t^{th} iteration, can thus be formulated as (see [41])

$$\begin{aligned} Q\left(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)}\right) &= \sum_{j=1}^{N} \mathbb{E}_{s_j|r_j,\boldsymbol{\lambda}^{(t)}}\Big[\log P\left\{r_j, s_j|\boldsymbol{\lambda}\right\}\Big] \\ &= \sum_{j=1}^{N}\sum_{m=1}^{M} \delta_{j,m}^{(t)} \log\left(\frac{1}{M}\phi_m\left(r_j|s_j,\boldsymbol{\lambda}\right)\right) \\ &= \sum_{j=1}^{N}\sum_{m=1}^{M} \delta_{j,m}^{(t)}\left(C - \frac{|r_j|^2 - a^2|x_m|^2}{2\sigma^2} + \frac{a\Re\{r_j^* e^{\imath\varphi} x_m\}}{2\sigma^2}\right), \end{aligned} \tag{2.3}$$

where

$$\delta_{j,m}^{(t)} \stackrel{\text{def}}{=} P\left\{s_j = x_m|r_j,\boldsymbol{\lambda}^{(t)}\right\}, \tag{2.4}$$

$$\phi_m(r_j|s_j,\boldsymbol{\lambda}) \stackrel{\text{def}}{=} \frac{1}{2\pi\sigma^2}\,\exp\left(-\frac{|r_j - a\,e^{\imath\varphi}\,x_m|^2}{2\sigma^2}\right), \tag{2.5}$$

and

$$C = \log\left(\frac{1}{2\pi\sigma^2 M}\right). \tag{2.6}$$

At the E-step, according to Eq. (2.3), only $\delta_{j,m}^{(t)}$ needs to be updated such that

$$\delta_{j,m}^{(t)} = \frac{\phi_m\left(r_j|s_j,\boldsymbol{\lambda}^{(t)}\right)}{\sum_{m=1}^{M}\phi_m\left(r_j|s_j,\boldsymbol{\lambda}^{(t)}\right)}. \tag{2.7}$$

At the M-step, one needs to solve

$$\boldsymbol{\lambda}^{(t+1)} = \max_{\boldsymbol{\lambda}} Q(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)}). \tag{2.8}$$

The phase offset needs to be updated such that

$$\varphi^{(t+1)} = \arg\max_{\varphi}\left\{\sum_{j=1}^{N}\sum_{m=1}^{M}\delta_{j,m}^{(t)}\Re\left\{r_j^* e^{\imath\varphi} x_m\right\}\right\}. \tag{2.9}$$

According to [42], Eq. (2.9) leads to

$$\varphi^{(t+1)} = -\angle \left\{ \sum_{j=1}^{N} \sum_{m=1}^{M} \delta_{j,m}^{(t)} \Re\{r_j^* \, x_m\} \right\}. \tag{2.10}$$

By setting the partial derivatives of $Q(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)})$ with respect to a and σ^2 to zero respectively and using the phase offset estimator given by Eq. (2.10), one can obtain the optimal updates for the signal amplitude and the noise variance as follows:

$$a^{(t+1)} = \frac{\sum_{j=1}^{N} \sum_{m=1}^{M} \delta_{j,m}^{(t)} \, \Re\left\{ r_j^* \, e^{\imath\varphi^{(t+1)}} x_m \right\}}{\sum_{j=1}^{N} \sum_{m=1}^{M} \delta_{j,m}^{(t)} |x_m|^2}, \tag{2.11}$$

$$\sigma^{2(t+1)} = \frac{1}{2N} \sum_{j=1}^{N} \sum_{m=1}^{M} \delta_{j,m}^{(t)} |r_j - a^{(t+1)} e^{\imath\varphi^{(t+1)}} x_m|^2. \tag{2.12}$$

By the end of each iteration, to check if the EM algorithm converges, the log-likelihood also needs to be updated as

$$f^{(t+1)} = \frac{1}{N} \sum_{j=1}^{N} \log \left(\frac{1}{M} \sum_{m=1}^{M} \phi_m \left(r_j | s_j, \boldsymbol{\lambda}^{(t+1)} \right) \right). \tag{2.13}$$

The EM algorithm continues to iterate its E-step and M-step alternately if $|f^{(t+1)} - f^{(t)}| \geq \epsilon$, where ϵ is a predefined threshold; it stops when either $|f^{(t+1)} - f^{(t)}| < \epsilon$ or the maximum iteration number is reached, and outputs the ultimate estimates $\hat{\boldsymbol{\lambda}} = [\hat{a}, \hat{\sigma}^2, \hat{\varphi}]$.

2.2.2 APPs of Coded Bits

Note that in order to identify the binary LDPC codes, the *a posteriori* probabilities (APPs) of coded bits need to be carried out to facilitate the log-likelihood ratios (LLRs). This calculation is straightforward for BPSK signals [32]. Nevertheless, more rigors are required for M-QAM signals. Recall that there is a one-to-one mapping between each M-QAM constellation point and L consecutive bits within a binary LDPC codeword. Therefore,

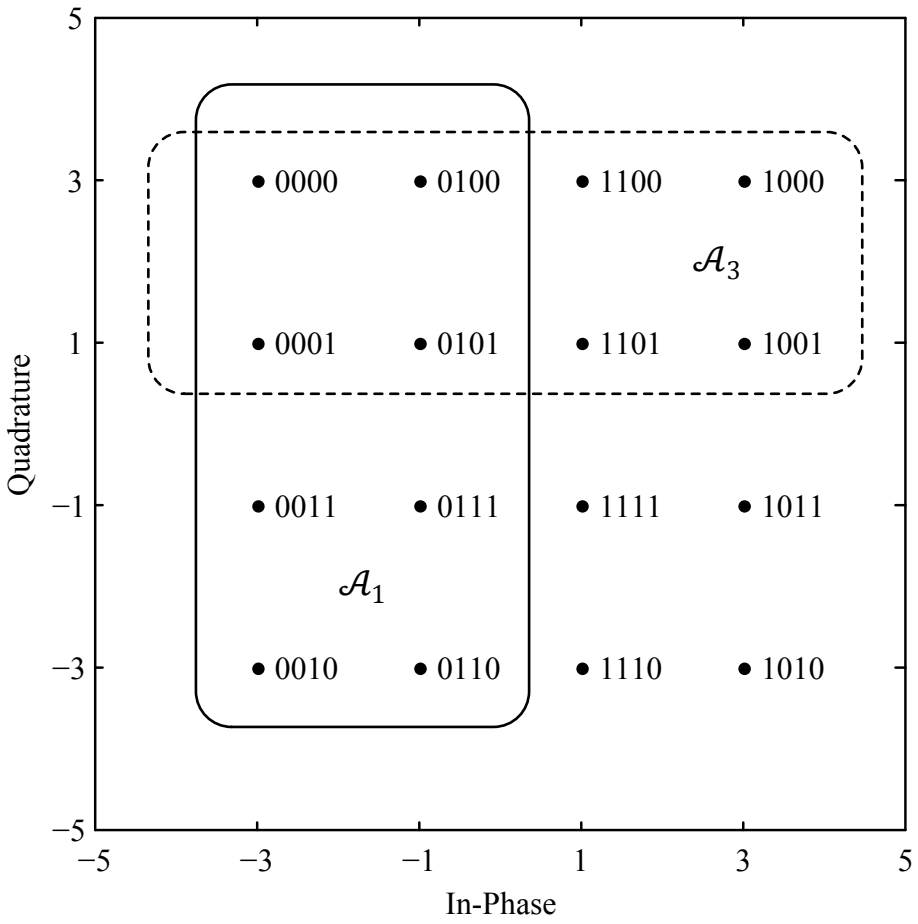

Figure 2.1: The illustration of the one-to-one mapping for a typical 16-QAM constellation together with two sets $\mathcal{A}_1$ and $\mathcal{A}_3$.

each received symbol r_j corresponds to the L consecutive bits denoted by $c_{j,1}, c_{j,2}, \ldots, c_{j,L}$. For each bit $c_{j,l}$, $l = 1, 2, \ldots L$, denote $\mathcal{A}_l \subseteq \mathcal{C}$ such that the constellation points $x_m \in \mathcal{A}_l$ all result in the mapping $c_{j,l} = 0$. The APP of the coded bit $c_{j,l}$ can therefore be obtained from the APP of the transmitted symbol s_j as expressed by

$$P\{c_{j,l} = 0|r_j\} = \sum_{x_m \in \mathcal{A}_l} P\{s_j = x_m|r_j\}, \tag{2.14}$$

and

$$P\{c_{j,l} = 1|r_j\} = 1 - P\{c_{j,l} = 0|r_j\}, \tag{2.15}$$

where $P\{s_j = x_m|r_j\}$ can be carried out according to Eq. (2.4) by plugging in $\hat{a}$, $\hat{\sigma}^2$, and $\hat{\varphi}$ resulting from the EM algorithm which is discussed in Section 2.2.1.

Figure 2.1 depicts the one-to-one mapping for a typical 16-QAM constellation. The constellation points contained in the set $\mathcal{A}_1$ are circled by the solid line while the constellation points contained in $\mathcal{A}_3$ are circled by the dashed line. For clarity, the constellation subsets $\mathcal{A}_2$ and $\mathcal{A}_4$ are not illustrated therein. It is obvious that each set $\mathcal{A}_l$, $l = 1, 2, \ldots, \log_2(M)$, contains $M/2$, i.e., half of the constellation points.

2.2.3 LDPC Encoder Identification

According to Eqs. (2.14) and (2.15), we can calculate the corresponding LLR as given by

$$\mathcal{L}(c_{j,l}|r_j) = \log \frac{P\{c_{j,l} = 0|r_j\}}{P\{c_{j,l} = 1|r_j\}}. \tag{2.16}$$

Let $\mathbf{g}$ denote a row vector of the LLRs of APPs such that $\mathbf{g} \stackrel{\text{def}}{=} [\mathcal{L}(c_{1,1}|r_1), \mathcal{L}(c_{1,2}|r_1), \ldots, \mathcal{L}(c_{1,L}|r_1), \mathcal{L}(c_{2,1}|r_2), \ldots, \mathcal{L}(c_{2,L}|r_2), \ldots, \mathcal{L}(c_{N,L}|r_N)]$ with length $n = NL$.

For each encoder $\theta' \in \Theta$, denote its $q \times n$ parity-check matrix by $\mathbf{H}^{\theta'}$. Denote the i^{th} row of $\mathbf{H}^{\theta'}$ by $\mathbf{h}_i^{\theta'}$, $i = 1, 2, \ldots, q$. Denote $\mathbf{g}_i \stackrel{\text{def}}{=} [g_{i,1}, g_{i,2}, \ldots, g_{N_i}]$ the sub-vector of $\mathbf{g}$ by retaining the elements in $\mathbf{g}$ which coincide with the positions of the non-zero elements of $\mathbf{h}_i^{\theta'}$, where N_i is the total number of the non-zero elements of $\mathbf{h}_i^{\theta'}$. According to [32], the LLR of syndrome APP for $\mathbf{h}_i^{\theta'}$ can then be expressed as

$$\begin{aligned}
\gamma_i^{\theta'} &\stackrel{\text{def}}{=} \boxplus_{\tau=1}^{N_i} g_{i,\tau} \\
&\stackrel{\text{def}}{=} g_{i,1} \boxplus g_{i,2} \boxplus \cdots \boxplus g_{i,N_i} \\
&= 2\tanh^{-1}\left(\prod_{\tau=1}^{N_i} \tanh\left(g_{i,\tau}/2\right)\right),
\end{aligned} \tag{2.17}$$

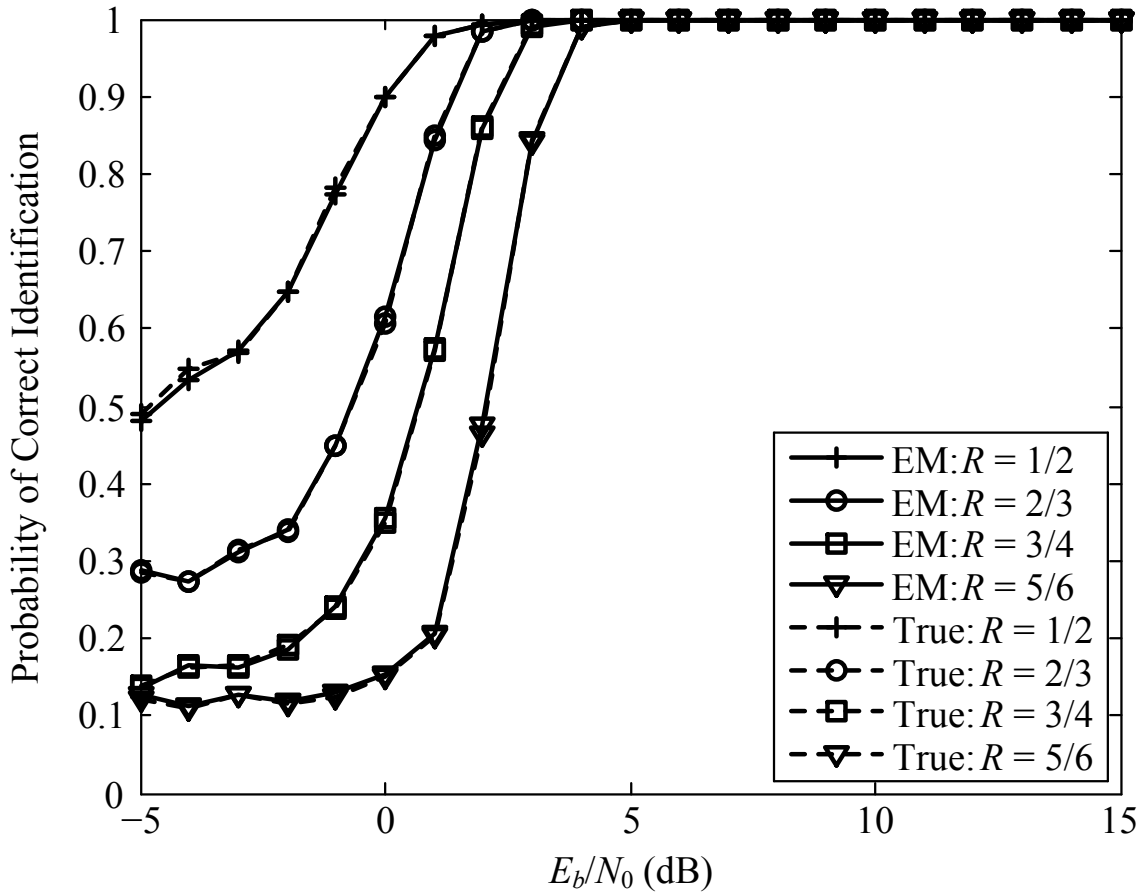

Figure 2.2: The probabilities of correct identification P_c with respect to E_b/N_0 for 4-QAM signals.

where $\boxplus$ is the *box-plus* operation defined in [32]. The average LLR of syndrome APP subject to the encoder candidate θ' is thus given by

$$\Gamma^{\theta'} \stackrel{\text{def}}{=} \frac{1}{q}\sum_{i=1}^{q}\gamma_i^{\theta'}. \tag{2.18}$$

Consequently, according to Eqs. (2.17) and (2.18), the underlying LDPC encoder can be identified as

$$\hat{\theta} = \arg\max_{\theta'\in\Theta} \Gamma^{\theta'}, \tag{2.19}$$

where Θ is the predefined encoder candidate set.

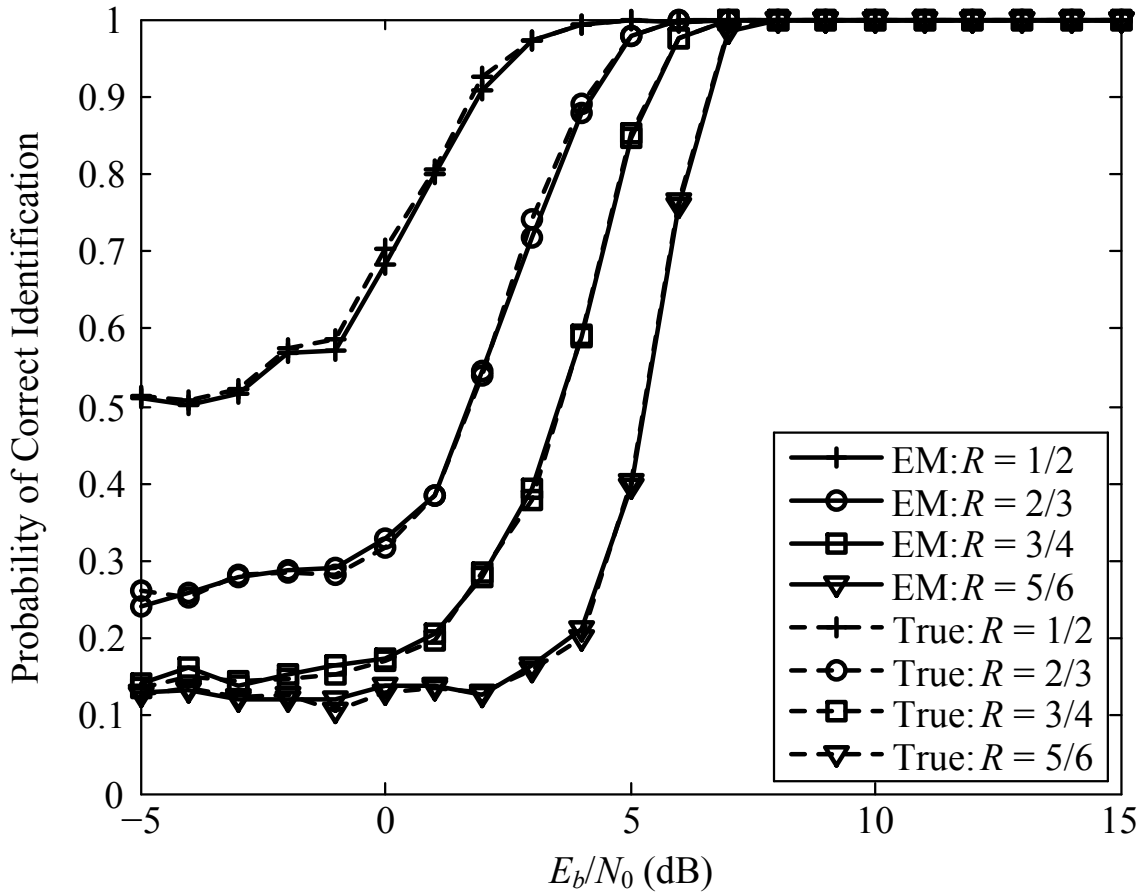

Figure 2.3: The probabilities of correct identification P_c with respect to E_b/N_0 for 16-QAM signals.

2.3 Simulation

The performances of our proposed blind LDPC-encoder identification scheme for M-QAM signals are evaluated by computer simulations. The performance metric we choose is the *probability of correct identification*, which is the probability that the receiver can correctly identify the types of the LDPC encoders adopted by the transmitter, i.e., $P_c = P\{\hat{\theta} = \theta\}$. The binary LDPC codes with length $n = 648$ and four different code-rates $R = 1/2$, $2/3$, $3/4$, and $5/6$ defined in the IEEE 802.11-2012 standard constitute the encoder candidate set Θ here [20]. For each particular modulation order M, one thousand Monte Carlo trials are taken for each encoder to be the actual one adopted by the transmitter. In each trial, one

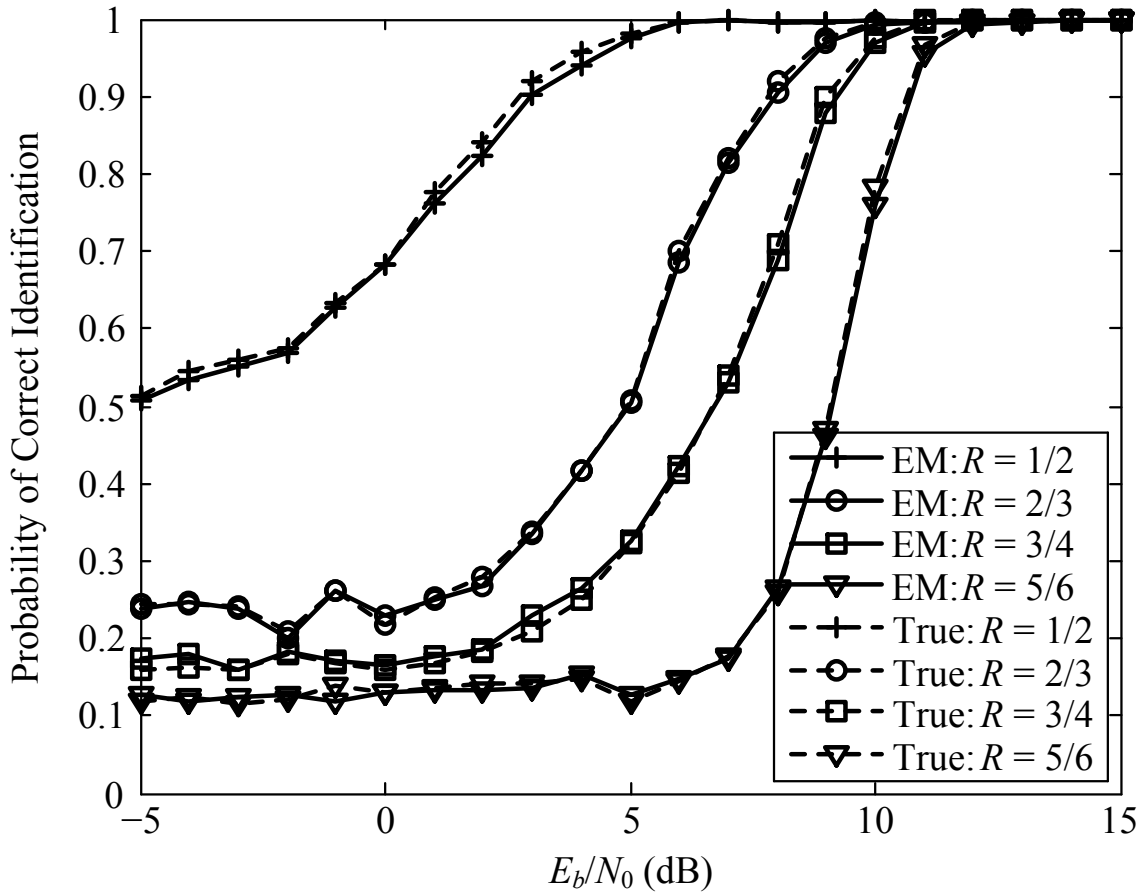

Figure 2.4: The probabilities of correct identification P_c with respect to E_b/N_0 for 64-QAM signals.

codeword block consisting of random information bits is generated, and the phase offset is randomly chosen within $(-\pi/4, \pi/4)$ to avoid the phase ambiguity inherent in any square QAM constellation (see [25, 43]). For the EM algorithm, we use the M_2M_4 method in [44] to establish the initial estimates $a^{(0)}$ and $\sigma^{2(0)}$. The phase offset is initialized as $\varphi^{(0)} = 1/4 \angle \left\{ -\sum_{j=1}^{N} r_j^4 \right\}$ according to [45]. In addition, the initial log-likelihood $f^{(0)}$ is set as an arbitrary number, say 1. The threshold ϵ is set as 10^{-3} and the maximum iteration number is set as 100.

Figures 2.2–2.4 delineate the probabilities of correct identification P_c with respect to E_b/N_0 for M-QAM signals when M is 4, 16, and 64, respectively. It can be discovered

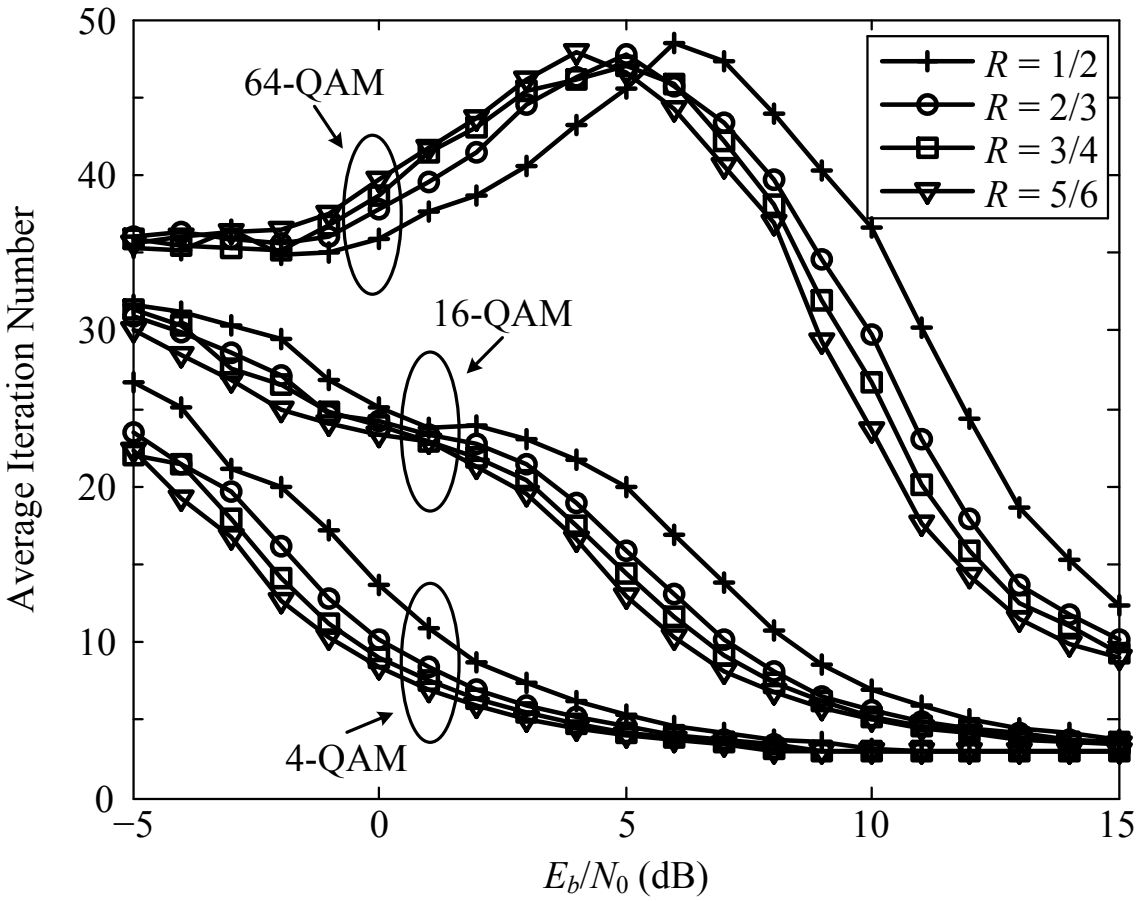

Figure 2.5: The average iteration numbers with respect to E_b/N_0 for 4-QAM, 16-QAM, and 64-QAM signals.

that P_c can achieve 100% for all the four codes when $E_b/N_0 > 5$ dB for 4-QAM signals, $E_b/N_0 > 9$ dB for 16-QAM signals, and $E_b/N_0 > 13$ dB for 64-QAM signals. Moreover, the probabilities of correct identification P_c using the estimated signal amplitude $\hat{a}$, noise variance $\hat{\sigma}^2$ and phase offset $\hat{\varphi}$ from the EM algorithm (denoted by "EM" in the figures) are compared to those using the true values of a, σ^2, and φ (denoted by "True" in the figures). The simulation results show that the differences between these two cases are quite negligible.

Note that as the modulation order M goes larger, the block length N of the received symbols becomes shorter since the binary LDPC codeword length n remains the same. As a result, the EM algorithm requires more iterations to converge for higher modulation orders.

Hence, it is interesting to investigate the *average iteration number* (AIN) of the EM algorithm for different M. Figure 2.5 demonstrates the AINs with respect to E_b/N_0 for M-QAM signals when M is 4, 16, and 64. It exhibits that the AIN required for 64-QAM signals is much larger than that required for 4-QAM signals as expected.

2.4 Summary

In this chapter, we propose a novel blind binary LDPC encoder identification technique for arbitrary M-QAM signals. The EM algorithm is also devised to estimate the unknown signal amplitude, noise variance, and phase offset. Monte Carlo simulation results illustrate the effectiveness of our blind binary LDPC encoder identification scheme. Besides, the average iteration number the EM algorithm needs to converge will be proportional to the modulation order M under the same channel condition. Our proposed new LDPC encoder identification mechanism can be a very promising solution for the next generation wireless adaptive modulation and coding transceivers.

3. BLIND IDENTIFICATION OF LDPC CODES FOR FADING CHANNELS

In this chapter, we would like to address blind LDPC encoder identification problems in fading channels. In Chapter 3.1, how to blindly identify LDPC codes for time-varying fading channels is discussed. In Chapter 3.2, a joint blind frame synchronization and LDPC encoder identification scheme is proposed for multipath fading channels. In order to keep our identification scheme blind, the channel state information which is hard to be estimate accurately for fading channels is not required anymore in our proposed identifications schemes by proper approximation techniques in this chapter.

3.1 Blind LDPC Encoder Identification for Time-Varying Fading Channels

The blind LDPC encoder identification schemes discussed in Chapter 2 in [31–33] cannot be directly applied to time-varying fading channels, which are often used as a practical scenario for modern wireless communication systems. Therefore, we would like to address the blind LDPC encoder identification for time-varying flat-fading channels in this section.

Specifically, our blind LDPC encoder identification scheme does not require the receiver to have any knowledge of the channel station information (CSI), i.e., *symbol energy*, *noise variance*, *fading amplitude*, and *phase offset*. Instead of trying to blindly estimate these parameters, our proposed new scheme resorts to the following techniques to avoid dealing with the CSI directly. To ignore the phase offset, an orthogonal modulation, namely M-ary frequency-shift-keying (FSK) is adopted so that the non-coherent detection can be carried out. The fading amplitude, assumed to be Rayleigh distributed, can also be aver-

aged out analytically and hence it does not need to be estimated. To make our identification scheme "blind" to the signal energy and the noise variance, we propose to use the "max-log" and "min-sum" approximations to calculate the log-likelihood ratio (LLR) of the syndrome *a posteriori* probability (APP), which is the key metric for identifying different encoders (see [32, 33] for details).

3.1.1 System Model

In this section, we introduce the basic transceiver system model in the baseband for our focused problem. At the transmitter, k successive information bits are grouped and passed through a particular binary (n, k) LDPC encoder (labeled by θ), which generates a codeword $\mathbf{c}^\theta$ with length n. Then, D blocks of codewords are interleaved by the interleaver with the interleaving depth D. The interleaved stream is modulated by the M-ary FSK modulator to generate the orthogonal signal. After it travels over a time-varying flat-fading channel, the t^{th} received signal sample can be represented by

$$\mathbf{r}_t = a_t\, e^{j\phi_t}\, \mathbf{s}_t + \mathbf{w}_t, \qquad t = 1, 2, \ldots, nD, \tag{3.1}$$

where $j \stackrel{\text{def}}{=} \sqrt{-1}$, $a_t\, e^{j\phi_t}$ is the complex fading coefficient that both real and imaginary parts are zero-mean Gaussian variables with the same variance $1/2$, and $\mathbf{s}_t$ is the M-ary FSK symbol in vector form. Specifically, a_t is the Rayleigh distributed fading amplitude and ϕ is the phase offset uniformly distributed over $[0, 2\pi]$. The complex fading coefficients $a_t\, e^{j\phi_t}$ are generated by Jakes' model [46]. In essence, the l^{th} M-ary FSK symbol $\mathbf{e}_l$ can be represented by an M-dimensional vector whose entries are all 0 except that the l^{th} entry should be $\sqrt{E_s}$ instead where E_s is the *symbol energy*. Usually, M is a radix-2 number and

therefore every $\log_2(M)$ bits resulting from the interleaver generates one M-ary FSK symbol $\mathbf{s}_t$. In addition, $\mathbf{w}_t$ denotes the M-dimensional complex AWGN vector such that the real and imaginary parts of each complex entry are statistically independent with zero mean and the same variance σ^2. Note that one should write $\mathbf{s}_t = \mathbf{s}_t^\theta$ and θ here specifies a particular LDPC encoder used by the transmitter but unknown to the receiver. How to blindly identify θ will be discussed in Section 3.1.2. Without loss of generality, we neglect the superscript θ for notational convenience throughout this section. The *energy per information-bit to noise power-spectrum-density ratio*, E_b/N_0, can thus be represented as

$$\frac{E_b}{N_0} = \frac{E_s}{2\sigma^2 \, R \log_2(M)}, \tag{3.2}$$

where $R = k/n$ is the *code rate*.

According to the system model given by Eq. (3.1), we can derive the APP and the corresponding LLR as follows. For notional simplicity, henceforth we will omit the index t dictated in Eq. (3.1) without causing any further ambiguity. Given a, ϕ, and $\mathbf{s} = \mathbf{e}_l$, the channel transition probability $p(\mathbf{r}|\mathbf{s}, a, \phi)$ can then be expressed by

$$\begin{aligned} p(\mathbf{r}|\mathbf{e}_l, a, \phi) &= \left(\frac{1}{2\pi\sigma^2}\right)^M \exp\left(-\frac{1}{2\sigma^2}\left|\mathbf{r} - ae^{j\phi}\mathbf{e}_l\right|^2\right) \\ &= C_1 \exp\left(-\frac{E_s a^2}{2\sigma^2} + \frac{\sqrt{E_s}a}{\sigma^2}\Re\left\{r_l e^{-j\phi}\right\}\right), \end{aligned} \tag{3.3}$$

where

$$C_1 \stackrel{\text{def}}{=} \left(\frac{1}{2\pi\sigma^2}\right)^M \exp\left(-\frac{1}{2\sigma^2}\sum_{m=1}^{M}\left|r_m\right|^2\right), \tag{3.4}$$

and r_m, r_l denote the m^{th} and l^{th} entries of $\mathbf{r}$, respectively. It is obvious that C_1 is independent of l and is not related to a and ϕ.

Since the receiver has no knowledge of the phase offset ϕ and the fading amplitude a and there is no pilot available for estimating these parameters (as we consider the *blind* scenario), these two unknown variables need to be "averaged" out. First, a non-coherent detection is carried out by averaging over ϕ using Eq. (3.3), which can be expressed by

$$\begin{aligned} p(\mathbf{r}|\mathbf{e}_l, a) &= \frac{1}{2\pi}\int_0^{2\pi} p(\mathbf{r}|\mathbf{e}_l, a, \phi)\, d\phi \\ &= C_1 \exp\left(-\frac{E_s}{2\sigma^2}a^2\right) I_0\left(\frac{\sqrt{E_s}}{\sigma^2}a|r_l|\right), \end{aligned} \tag{3.5}$$

where $I_0(\,)$ is the zero-order modified Bessel function of the first kind. Then, by averaging over a, which is Rayleigh-distributed, using Eq. (3.5), according to [47], we have

$$\begin{aligned} p(\mathbf{r}|\mathbf{e}_l) &= \int_0^{\infty} p(a)p(\mathbf{r}|\mathbf{e}_l, a)\, da \\ &= C_1 \int_0^{\infty} 2a \exp\left[-\left(\frac{E_s}{2\sigma^2}+1\right)a^2\right] \times I_0\left(\frac{\sqrt{E_s}}{\sigma^2}a|r_l|\right)\, da \\ &= \frac{C_1}{\left(\frac{E_s}{2\sigma^2}+1\right)} \exp\left[\frac{E_s|r_l|^2}{4\sigma^2\left(\frac{E_s}{2}+\sigma^2\right)}\right]. \end{aligned} \tag{3.6}$$

For the input of the binary LDPC decoder, the M-ary FSK demodulator's "soft output" should be the probability for each information bit rather than that for each modulated symbol as shown in Eq. (3.6). Therefore, a *"symbol-to-bit" probability mapping* needs to be carried out. Moreover, the LDPC decoding algorithm is usually performed in the logarithm domain for the numerical precision reason. Thus, the bit probabilities will be further converted to the LLRs prior to decoding. Denote $\mathcal{A}_\mu$ the set of modulation indices l such that the μ^{th} bit of $\mathbf{e}_l$ is 0, and denote $\mathcal{A}_\mu^c$ the set of indices l such that the μ^{th} bit of $\mathbf{e}_l$ is 1 instead, for $\mu = 1, 2, \ldots, \log_2(M)$. For a time instant t, the received signal sample $\mathbf{r}$ contains $\log_2(M)$ coded bits $c_\mu, \mu = 1, 2, \ldots, \log_2(M)$. Assume that the coded bits have equal probabilities to

be 0 or 1. The corresponding LLR $\mathcal{L}(c_\mu|\mathbf{r})$ can thus be expressed as

$$\begin{aligned}\mathcal{L}(c_\mu|\mathbf{r}) &= \log\left(\frac{\sum\limits_{l\in\mathcal{A}_\mu} p(\mathbf{r}|\mathbf{e}_l)}{\sum\limits_{l\in\mathcal{A}_\mu^c} p(\mathbf{r}|\mathbf{e}_l)}\right) \\ &= \log\left(\frac{\sum\limits_{l\in\mathcal{A}_\mu} \exp\left[\frac{E_s|r_l|^2}{4\sigma^2\left(\frac{E_s}{2}+\sigma^2\right)}\right]}{\sum\limits_{l\in\mathcal{A}_\mu^c} \exp\left[\frac{E_s|r_l|^2}{4\sigma^2\left(\frac{E_s}{2}+\sigma^2\right)}\right]}\right). \end{aligned} \tag{3.7}$$

The LLRs given by Eq. (3.7) are obtained and then the deinterleaving operation is performed. These "deinterleaved" LLRs can be sent to our new blind encoder identification scheme for identifying the unknown encoder θ finally.

3.1.2 Blind LDPC Encoder Identification

In this section, we present our proposed *blind LDPC encoder identification* scheme for the system model involving FSK modulated signals and the time-varying flat-fading channel manifested in Section 3.1.1. Note that the encoder θ cannot be arbitrary and it should be drawn from a predefined candidate set Θ which is known to both transmitter and receiver.

For LDPC codes, each encoder θ is specified by its associated $(n-k)$-by-n *parity-check matrix* $\mathbf{H}^\theta$. Each row of $\mathbf{H}^\theta$, denoted by $\mathbf{h}_i^\theta$, $i = 1, 2, \ldots, n-k$, manifests the corresponding parity-check constraint. Without loss of generality, it is assumed that the parity-check matrix $\mathbf{H}^\theta$ has full rank, that is, all rows of $\mathbf{H}^\theta$ are linearly independent of each other. Since only the codeword generated from encoder θ can satisfy the syndrome check of $\mathbf{H}^\theta$, i.e., $\mathbf{H}^\theta \mathbf{c}^{\theta^T} = \mathbf{0}$ ($\mathbf{0}$ is an $(n-k)$-by-1 all-zero vector), we can investigate the likelihood of the received signal block satisfying the parity check for each encoder candidate, and then identify the unknown encoder θ in the sense of maximum likelihood. Such likelihood is usually represented by

the LLR of the syndrome APP (see [32, 33]), which can be obtained from LLRs given by Eq. (3.7).

Note that in Eq. (3.7), the CSI, namely the symbol energy E_s and the noise variance σ^2, are needed. Because the receiver has no knowledge of these parameters, our proposed blind encoder identification scheme can depend on neither E_s nor σ^2, and therefore we propose to adopt "max-log" and "min-sum" approximations as follows.

First, based on the max-log approximation, Eq. (3.7) can be modified as

$$\begin{aligned}\mathcal{L}(c_\mu|\mathbf{r}) &\approx \max_{l\in\mathcal{A}_\mu}\left\{\frac{E_s|r_l|^2}{2E_s\sigma^2+4\sigma^4}\right\}-\max_{l\in\mathcal{A}_\mu^c}\left\{\frac{E_s|r_l|^2}{2E_s\sigma^2+4\sigma^4}\right\}\\ &= C_2\left[\max_{l\in\mathcal{A}_\mu}\left\{|r_l|^2\right\}-\max_{l\in\mathcal{A}_\mu^c}\left\{|r_l|^2\right\}\right], \end{aligned} \tag{3.8}$$

where

$$C_2 \stackrel{\text{def}}{=} \frac{E_s}{2E_s\sigma^2+4\sigma^4}. \tag{3.9}$$

Thus, according to Eq. (3.9), E_s and σ^2 are inherently included in the new parameter C_2. Note that when the FSK modulation order M is 2, the max-log approximation becomes equality since there remains only one term in the max operation.

Denote $\mathbf{H}^{\theta'}$ the $(n-k)\times n$ parity-check matrix of the encoder candidate θ'. The locations of the non-zero elements in the i^{th} row of $\mathbf{H}^{\theta'}$ are denoted by a vector $\mathbf{z}_i \stackrel{\text{def}}{=} [z_{i_1}, z_{i_2}, \ldots, z_{i_{N_i}}]^T$, where N_i is the total number of the non-zero elements in the i^{th} row of $\mathbf{H}^{\theta'}$. Denote $\mathbf{R} \stackrel{\text{def}}{=} [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\nu]$, where $\nu \stackrel{\text{def}}{=} n/\log_2(M)$ such that $\mathbf{R}$ results from one LDPC codeword block $\mathbf{c} = [c_1, c_2, \ldots, c_n]$. Recall that in the min-sum algorithm [48], the box-plus operation in the check node can be approximated by selecting the incoming information which has the minimum absolute value among those calculated from all connected variable nodes. Thus,

by adopting the min-sum algorithm (see [40,48]), the LLR of syndrome APP for the i^{th} check node, denoted by $\gamma_i^{\theta'}$, can be expressed as

$$\gamma_i^{\theta'} \approx \left[\prod_{d=1}^{N_i} \operatorname{sign}\Big[\mathcal{L}\big(c_{z_{i_d}}|\mathbf{R}\big)\Big]\right] \times \min_{z_{i_d}} \Big|\mathcal{L}\big(c_{z_{i_d}}|\mathbf{R}\big)\Big|. \tag{3.10}$$

Since the coefficient C_2 involving E_s and σ^2 poses no effect on the sign and min operations used in Eq. (3.10), it can be simply dropped from Eq. (3.8). Therefore, after these two approximations in Eqs. (3.8) and (3.10), our proposed blind identification scheme does not depend on the CSI anymore.

The calculation of the LLRs of the syndrome APP, according to Eq. (3.10), is essentially undertaken at the *check* nodes involved in the message passing (MP) decoding algorithm. Note that all incoming information from the *variable* nodes must be used to compute the syndrome APP, while only extrinsic information are used at the check nodes in the MP decoding algorithm.

Having obtained the LLRs of the syndrome APP, we are ready to identify the unknown encoder θ. Obviously, the encoder can be identified if it has the highest percentage of the satisfied syndrome checks over all candidates. It yields

$$\hat{\theta} = \arg\max_{\theta' \in \Theta} \left\{ \frac{1}{n-k} \sum_{i=1}^{n-k} (\gamma_i^{\theta'})^+ \right\}, \tag{3.11}$$

where

$$(\gamma_i^{\theta'})^+ \stackrel{\text{def}}{=} \begin{cases} 1, & \gamma_i^{\theta'} > 0 \\ 0, & \gamma_i^{\theta'} < 0 \end{cases}. \tag{3.12}$$

Eq. (3.12) can be considered as a "*hard*" decision. If $\gamma_i^{\theta'} > 0$, the i^{th} parity check relation is more likely to be satisfied, then Eq. (3.12) will mark the i^{th} parity check "satisfied"; otherwise, if $\gamma_i^{\theta'} < 0$, Eq. (3.12) indicates that the parity check is failed.

In analogy to the difference between the hard decision and the soft decision used in decoders, a soft decision can also be carried out to identify θ, i.e.,

$$\hat{\theta} = \arg\max_{\theta' \in \Theta} \left\{ \frac{1}{n-k} \sum_{i=1}^{n-k} \gamma_i^{\theta'} \right\}, \tag{3.13}$$

where the argument of $\arg\max\{\ \}$ is the *average LLR of the syndrome APP* according to [32, 40]. Note that different encoders θ may have different combinations of n and k, and therefore the normalization factor $1/(n-k)$ is necessary in both Eq. (3.11) and Eq. (3.13). When a parity-check matrix of some encoder has more (independent) rows than others, it implies that more parity-check constraints are available and thus better identification performance could be expected. As a matter of fact, this normalization factor serves to facilitate a fair comparison among different encoder candidates so that the impact of the variations in the total number of parity-check constraints would be mitigated.

Since the unknown encoder θ is identified by examining the likelihood of a codeword (or multiple codewords) satisfying all the parity-check relations manifested by the parity-check matrix, a crucial assumption has to be made for Θ that the parity-check matrices of any two encoders share no common rows (no identical parity-check relations). This assumption is usually valid for LDPC codes. On the other hand, the encoders in Θ can have the same length and/or the same code rate. They can even be drawn from the same ensemble.

3.1.3 Simulation

The performance of our proposed new blind LDPC encoder identification scheme for the transmitted signals subject to orthogonal modulations traveling through time-varying flat-fading channels is evaluated via Monte Carlo simulations in this section. The performance

metric we choose is the *probability of correct identification*, P_c, which is the probability that the receiver can correctly identify the unknown LDPC encoder, i.e., $P_c \stackrel{\text{def}}{=} P_r(\hat{\theta} = \theta)$. The LDPC parity-check matrices with codeword length $n = 1944$ specified in the IEEE 802.11-2012 standard [20] are adopted for our simulations. Thus, there are four encoder candidates with code rates $R = 1/2$, $R = 2/3$, $R = 3/4$, and $R = 5/6$ in the candidate set Θ. The interleaver depth is 50 so that 50 codewords are interleaved before passed into the M-ary FSK modulator. The complex fading coefficients described in Eq. (3.1) are generated by Jakes' model [46], in which the maximum Doppler shift is denoted by f_D, the symbol period is denoted by T_s, and the normalized Doppler rate is denoted by $f_D T_s$. One thousand Monte Carlo trials are carried out to obtain the average performance for each simulation setting.

Figure 3.1 depicts the probabilities of correct identification P_c with respect to E_b/N_0 for the four aforementioned LDPC encoders using the hard decision given by Eq. (3.11) and the soft decision given by Eq. (3.13), respectively. The binary FSK (BFSK) modulation is used and the normalized Doppler rate $f_D T_s$ is 0.001 for this figure. It is shown that the probabilities of correct identification P_c for all encoders approach 100% when $E_b/N_0 \geq 15$ dB. The lower the code rate (the more the parity-check bits), the better the identification performance. Moreover, the average LLR of the syndrome APP (Eq. (3.13)) offers better identification than the percentage of the satisfied syndrome checks (Eq. (3.11)) in the high E_b/N_0 region. This phenomenon coincides with the well-known concept that soft-decision based methods are superior to hard-decision based schemes. Nevertheless, the performance gap between these two methods narrows down as the code rate increases.

The probabilities of correct identification P_c with respect to E_b/N_0 using Eq. (3.13) are investigated in Figure 3.2 for different FSK modulation orders and different normalized

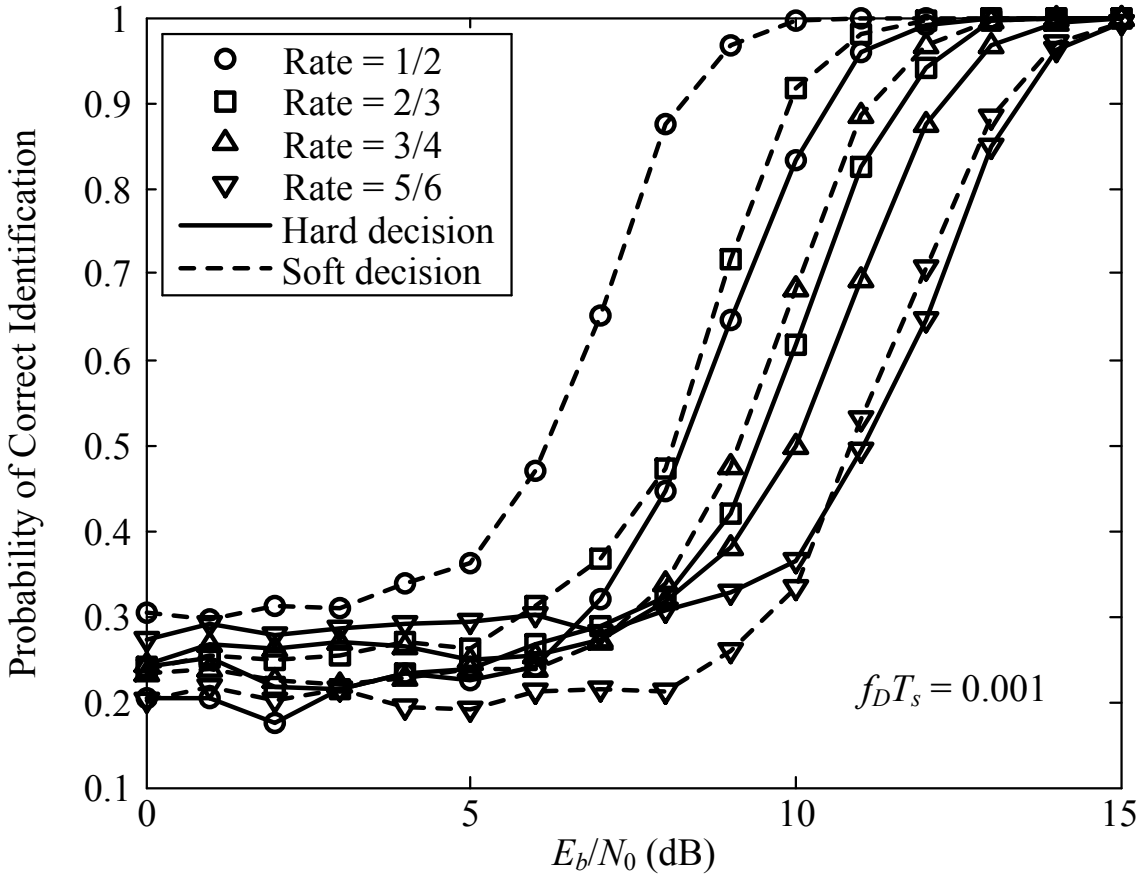

Figure 3.1: The probabilities of correct identification P_c with respect to E_b/N_0 for the four LDPC encoder candidates using Eq. (3.11) ("hard decision") and Eq. (3.13) ("soft decision"), respectively. BFSK modulator is used, and the normalized Doppler rate is $f_D T_s = 0.001$.

Doppler rates. For clarity, only rate 1/2 code's identification performances are presented. It is shown that as the FSK modulation order M increases, the identification performance improves. Moreover, the normalized Doppler rates varying from 0.001 (slow fading) to 0.05 (fast fading) have little impact on the performance of our proposed blind scheme. Similar results can be observed for other encoder candidates.

3.1.4 Summary

In this section, we propose a novel blind LDPC encoder identification scheme for time-varying flat-fading channels when orthogonal modulations such as M-ary FSK are used. The

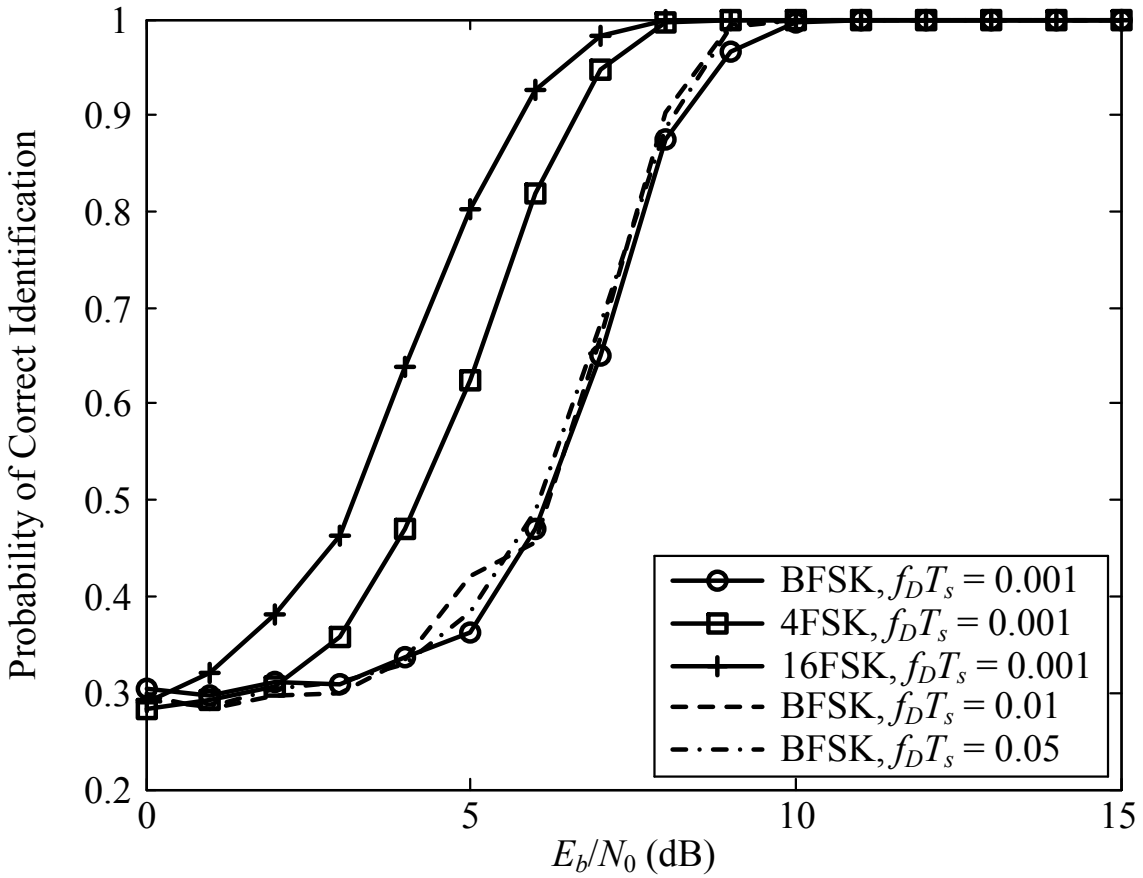

Figure 3.2: The probabilities of correct identification P_c with respect to E_b/N_0 using Eq. (3.13) for different FSK modulation orders and different normalized Doppler rates.

proposed blind scheme is devised in a convenient way that all channel state information, namely the phase offset, the fading coefficient, the symbol energy, and the noise variance, are not required. The performances of our proposed LDPC encoder identification scheme through Monte Carlo simulations demonstrate that this method is very robust against both slow and fast time-varying flat-fading channels.

3.2 Joint Blind Frame Synchronization and Encoder Identification

For the aforementioned blind LDPC encoder identification schemes in Chapter 2 and Chapter 3.1, one common underlying assumption is that the frame synchronization is per-

fectly accomplished beforehand. However, in a "practical" blind scenario, this assumption is unrealistic. Fortunately, various blind techniques were proposed to address timing synchronization, carrier frequency offset or phase offset estimation [39, 49, 50]. Among these schemes, the blind synchronization methods for LDPC-coded systems in [39, 50] are based on the log-likelihood ratios (LLRs) of the syndrome, which are related to the essential metric, the average LLR of syndrome *a posteriori* probability (APP) in our recently proposed blind encoder identification schemes [33, 51].

Therefore, it would be quite interesting to investigate a "practical" blind transceiver structure addressing both blind frame synchronization and blind encoder identification. In this chapter, we would like to explore the joint blind frame synchronization and blind encoder identification of binary LDPC codes for binary phase-shift keying (BPSK) signals over multi-path fading channels. We propose to use average LLR as the unifying metric for this new joint blind scheme. Furthermore, we propose a two-stage search method with a search step-size q by taking advantage of the quasi-cyclic property of the parity-check matrix. Such a new method can mitigate the cumbersome computational burden brought by the blind frame synchronization problem. Our proposed new joint blind scheme is then evaluated by the probability of correct identification in various multi-path channel scenarios.

The rest of this section is organized as follows. The signal model is introduced in Section 3.2.1. The joint blind frame synchronization and LDPC encoder identification scheme is presented in Section 3.2.2. The new two-stage search algorithm is presented in Section 3.2.3 to reduce the complexity of blind frame synchronization. Monte Carlo simulation results are demonstrated in Section 3.2.4 to evaluate the effectiveness of our proposed new scheme.

3.2.1 Signal Model

In this section, we introduce the basic binary LDPC-coded system. At the transmitter, original information bits are grouped into blocks, each of which consists of k consecutive bits. Each block of information bits is passed to the LDPC encoder θ to generate a corresponding block of *codeword*, say $\mathbf{c}^\theta$ with length n, where θ denotes a particular type of LDPC encoder. The corresponding code rate is thus $R = k/n$. Then, the codeword $\mathbf{c}^\theta$ should be modulated by BPSK modulator and the corresponding block of modulated symbols is denoted by $\mathbf{s}^\theta$. The transmitted pass-band signals travel through the multipath channel and arrive at the receiver. Each sample of the received baseband signals, $r(j)$, can be expressed as

$$r(j) = \sum_{l=1}^{L} a_l \, s^\theta(j - \tau_l) + w(j), \tag{3.14}$$

where L is the number of the paths, a_l is the unknown channel fading coefficient for the l^{th} signal path, $s^\theta(j)$ is the modulated BPSK signal generated from the encoder θ, τ_l is the time delay for the l^{th} signal path, and $w(j)$ is the zero-mean additive white Gaussian noise (AWGN) with the variance σ^2. Without loss of generality, it is assumed that $a_{l_1} \geq a_{l_2}$ and $\tau_{l_1} \leq \tau_{l_2}$ for $l_1 < l_2$. That is, the shorter path the signal travels, the larger the signal strength one expects.

According to Eq. (3.14), the *signal-to-interference ratio* (SIR) is given by

$$\text{SIR} \stackrel{\text{def}}{=} \frac{a_1^2}{\sum_{l=2}^{L} a_l^2}, \tag{3.15}$$

and the signal energy per bit (bit energy E_b) to noise power spectrum density (N_0) ratio is defined as

$$\frac{E_b}{N_0} \stackrel{\text{def}}{=} \frac{a_1^2}{R\,\sigma^2}. \tag{3.16}$$

In practice, the AMC transceivers usually select the modulation/encoder schemes only over a predefined candidate set. In this chapter, we assume that a predetermined LDPC encoder candidate set, say Θ, which contains multiple encoder candidates, is known to both transmitter and receiver beforehand. We also assume that the encoders in Θ are different from each other by that the parity-check matrices of any two encoders do not have identical row(s). This assumption is valid for existing AMC schemes. It is further assumed that the delay for the first signal path (with the shortest time delay) is within a codeword length, that is, $\tau_1 \in [0, n-1]$ according to [39]. In the next section, we will present a joint blind frame synchronization and blind encoder identification method using the average LLR of syndrome APP.

3.2.2 New Joint Blind Scheme

First consider blind frame synchronization. The probability of having a verified parity-check equation (the syndrome is 0) when the timing synchronization is achieved is greater than the probability of making the same statement true when it is out of synchronization according to [39]. Then consider blind encoder identification. The average LLR of syndrome APP when the true encoder is picked is larger than those average LLRs when incorrect encoders are picked from the candidate set instead according to [33, 51]. Therefore, when the receiver needs to blindly identify the encoder θ and to blindly estimate the time delay τ_1 altogether from the received signals given by Eq. (3.14), it is expected that the average LLR of syndrome APP attains its maximum when the underlying signal block is synchronized and meanwhile the true encoder is identified. In this section, we will design a new unified framework for joint blind frame synchronization and blind LDPC encoder identification.

The proposed joint scheme will be based on the same metric, namely the average LLR of syndrome APP.

Denote $\mathbf{H}^{\theta'}$ the $m \times n$ parity-check matrix of the encoder candidate θ'. The locations of the non-zero elements in the i^{th} row of $\mathbf{H}^{\theta'}$ are denoted by a vector $\mathbf{z}_i = [z_{i_1}, z_{i_2}, \ldots, z_{i_{N_i}}]^T$, where N_i is the total number of the non-zero elements in the i^{th} row of $\mathbf{H}^{\theta'}$. Denote $\mathbf{r}_t^{\theta} \stackrel{\text{def}}{=} [r(t), r(t+1), \ldots, r(t+n-1)]^T$ the received signal vector starting from the time instant t subject to the encoder θ used by the transmitter. The range of t ($t = 0, 1, \ldots, n-1$) is determined by the time delay of the first path, τ_1. According to [48,51], the LLR of syndrome APP for the i^{th} parity-check equation ($i = 1, 2, \ldots, m$) of $\mathbf{H}^{\theta'}$ when the sliding window for collecting received signal samples starts at t can be written as follows:

$$\begin{aligned}
\gamma_{t,i}^{\theta'} &= \ln \frac{1 + \prod_{d=1}^{N_i} \tanh\left(\mathcal{L}\Big(r(t+z_{i_d})|\, c(z_{i_d})\Big)\Big/2\right)}{1 - \prod_{d=1}^{N_i} \tanh\left(\mathcal{L}\Big(r(t+z_{i_d})|\, c(z_{i_d})\Big)\Big/2\right)} \\
&= 2\tanh^{-1}\left[\prod_{d=1}^{N_i} \tanh\left(\mathcal{L}\Big(r(t+z_{i_d})|\, c(z_{i_d})\Big)\Big/2\right)\right] \\
&\approx \left[\prod_{d=1}^{N_i} \operatorname{sign}\Big(\mathcal{L}\Big(r(t+z_{i_d})|\, c(z_{i_d})\Big)\Big)\right] \times \min_d \left|\mathcal{L}\Big(r(t+z_{i_d})|\, c(z_{i_d})\Big)\right|. \qquad (3.17)
\end{aligned}$$

When the SIR is much larger than 1, the effect of "*fading interferences*" a_l ($l = 2, \ldots, L$) can be neglected. Thus, according to [51], the LLR can be approximated as

$$\mathcal{L}\big(r(j)|c(j)\big) \approx \frac{2a_1 r(j)}{\sigma^2}. \qquad (3.18)$$

It is obvious that $\mathcal{L}\big(r(j)|c(j)\big)$ in Eq. (3.18) can be simplified as $r(j)$ when sign and min operations are taken in Eq. (3.17). Consequently, it is not required to estimate the fading coefficient a_1 and the noise variance σ^2.

Following [39,51], the LLR of syndrome APP, $\gamma_{t,i}^{\theta'}$, is expected to be a positive value when the true encoder is picked and the sliding window aligns with the time delay of the first path, that is, $\theta' = \theta$ and $t = \tau_1$ for all $i = 1, 2, \ldots, m$. On the other hand, if $\theta' \neq \theta$ or $t \neq \tau_1$, the parity-check equations do not necessarily hold. As a result, individual LLRs $\gamma_{t,i}^{\theta'}$ may be sometimes positive and sometimes negative and thus they exhibit fluctuations around zero. Therefore, for each θ' and t, we can average $\gamma_{t,i}^{\theta'}$ over all i and the maximum average value should correspond to the true encoder θ and the correct time delay τ_1. The average LLR for the received signal block $\mathbf{r}_t^{\theta}$ subject to the encoder candidate θ' is given by

$$\Gamma_t^{\theta'} \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \gamma_{t,i}^{\theta'}. \tag{3.19}$$

Consequently, according to Eqs. (3.17) and (3.19), the underlying LDPC encoder and the time delay of the first path for the received signals can be identified by

$$\Lambda \stackrel{\text{def}}{=} [\hat{\theta}, \hat{\tau}] = \underset{\theta' \in \Theta, t \in \Delta}{\arg\max}\ \Gamma_t^{\theta'}, \tag{3.20}$$

where $\Delta \stackrel{\text{def}}{=} \{0, 1, 2, \ldots, n-1\}$. One can see that it is necessary to search for every possible encoder candidate $\theta' \in \Theta$ and every possible time delay $t \in \Delta$ for the joint blind scheme.

3.2.3 Computational Complexity Reduction

According to Eq. (3.20), the complexity of our proposed joint blind encoder identification and blind frame synchronization scheme depends on the dimension of the entire search space, $|\Theta| \times n$. Usually, the encoder candidate set Θ just includes a few elements; however, the codeword length n is as large as hundreds or even thousands. For instance, twelve high-throughput LDPC codes ($|\Theta| = 12$) are specified in the IEEE 802.11-2012 standard [20] with codeword lengths n equal to 648, 1296, and 1944. Therefore, when $n \gg |\Theta|$, the big majority

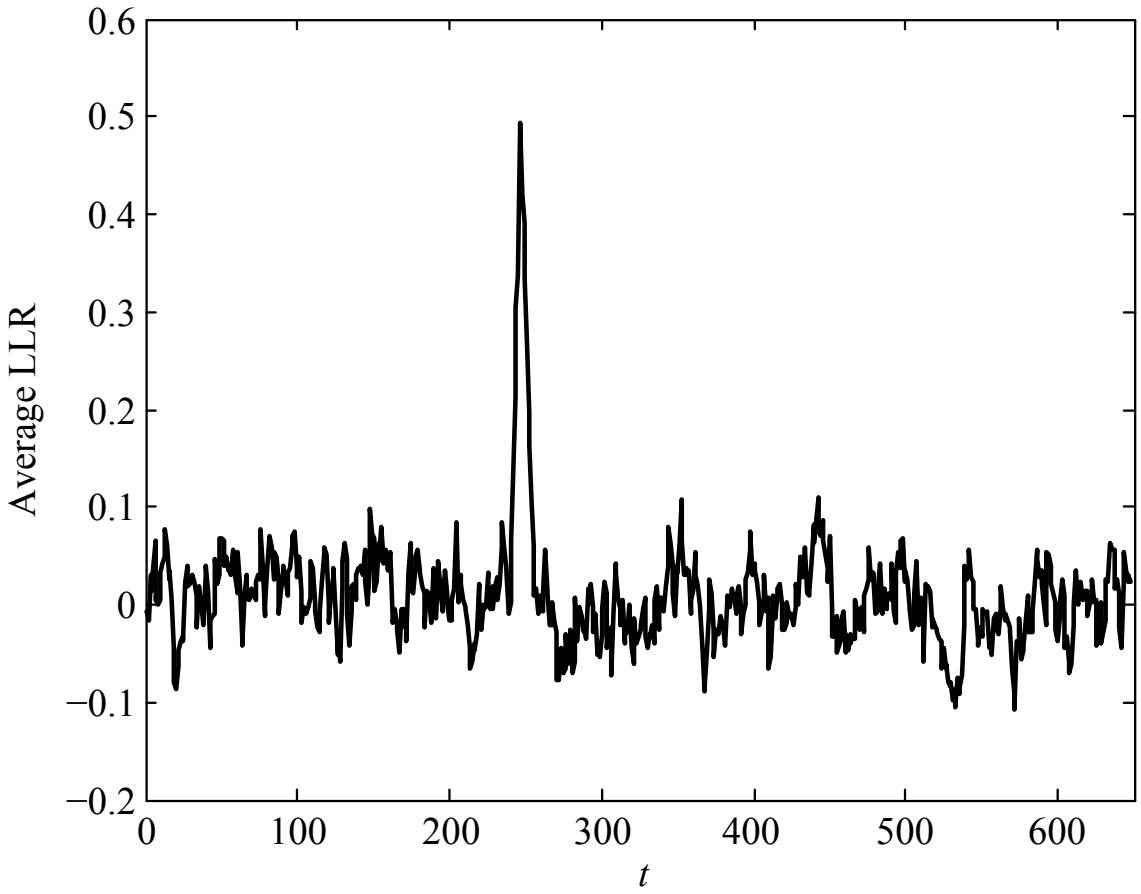

Figure 3.3: The average LLR $\Gamma_t^{\theta'}$ versus the sliding window's starting time point t for the rate 1/2 LDPC encoder ($\theta' = \theta$).

of the complexity burden of our joint scheme arises from blind frame synchronization. To combat this computational bottleneck, in this section, we propose a *two-stage search method* to reduce the search scope for time delays.

We observe that when $\theta' = \theta$, the average LLR $\Gamma_t^{\theta'}$ may show an abrupt peak around $t = \tau_1$ for quasi-cyclic LDPC codes (see [20, 52] for quasi-cyclic LDPC codes). For instance, as illustrated in Figure 3.3, the average LLR $\Gamma_t^{\theta'}$ is depicted with respect to the sliding window's starting time t for the LDPC code with codeword length 648 and the code rate 1/2 specified in the IEEE 802.11-2012 standard [20]. The explanation of this phenomenon is that the quasi-cyclic property makes $\gamma_{t,i}^{\theta'}$ repeat a lot of times between consecutive time

points t's; hence the average LLR $\Gamma_t^{\theta'}$ would not decrease drastically when t lies within the neighborhood of the true time delay τ_1 (a few time points away from it). Thanks to this neat quasi-cyclic property, we do not need to search the time delay sample by sample from $t = 0$ to $t = n - 1$. Instead, the search grid spacing can be a few samples apart to save a lot of computational complexity. This "*coarse search*" will be facilitated at Stage One. The coarse search will narrow our general time-delay search down to the neighborhood of the true value. Then, at Stage Two, we carry out the fine search with the one-sample resolution over the neighborhood of the spotted time delay from the coarse search. Through these two stages, we can find the correct time delay which corresponds to the maximum of $\Gamma_t^{\theta'}$ at the second search stage.

In summary, our proposed two-stage time-delay search algorithm is detailed as follows:

Step 1) Choose a proper search step-size q; for each encoder candidate $\theta' \in \Theta$, do Step 2 to Step 5;

Step 2) First-Stage Search (*coarse search*): slide the window by q samples each time from the starting point $t = 0$, that is, $t \in \{0, q, 2q, 3q, \ldots\}$ $(t \leq n-1)$; compute the corresponding average LLRs $\Gamma_t^{\theta'}$ accordingly;

Step 3) Find the time position $t_1(\theta')$ where $\Gamma_{t_1(\theta')}^{\theta'}$ attains the maximum among $\Gamma_t^{\theta'}, t \in \{0, q, 2q, 3q, \ldots\}$ $(t \leq n-1)$;

Step 4) Second-Stage Search (*fine search*): slide the window sample by sample only in the range of $b_1 \leq t \leq b_2$, where $b_1 = \max\{0, t_1 - q + 1\}$ and $b_2 = \min\{t_1 + q - 1, n - 1\}$; compute the corresponding average LLRs $\Gamma_t^{\theta'}$ accordingly;

Step 5) Find the time position $t_2(\theta')$ and $\Gamma_{t_2(\theta')}^{\theta'}$ which attains the maximum among $\Gamma_t^{\theta'}, t \in \{b_1, b_1 + 1, b_1 + 2, \ldots, b_2\}$;

Step 6) Pick $\hat{\theta}$ for which $\Gamma^{\hat{\theta}}_{t_2(\hat{\theta})}$ is the maximum of $\Gamma^{\theta'}_{t_2(\theta')}$ among $\theta' \in \Theta$. The corresponding time delay $t_2(\hat{\theta})$ is chosen as $\hat{\tau}_1$.

Now we can calculate the total number of search points required by the above-mentioned two-stage search method. In the first stage, the total number of search points is $\lceil n/q \rceil$ where $\lceil a \rceil$ gives a nearest integer which is not less than a. In the second stage, the total number of search points is upper-bounded by $2q-1$. Hence, compared to the exhaustive search, the total number of search points required by our proposed two-stage algorithm is tremendously decreased from n to at most $\lceil n/q \rceil + 2 \times q - 1$. Theoretically speaking, it is better to choose q ($q \leq \sqrt{n/2}$) as large as possible so that the computational complexity can be greatly reduced. However, in practice, the search step-size q has to be smaller than the peak width of the average LLR for a particular encoder so that the global maximum of the average LLR can be found. The *peak width* of the average LLR $\Gamma^{\theta'}_t$ is the neighborhood of τ_1 ($\Gamma^{\theta'}_{\tau_1}$ attains the maximum), denoted by $\mathcal{N}_{\tau_1} \stackrel{\text{def}}{=} \{\tau_1 - \delta_1, \tau_1 - \delta_1 + 1, \ldots, \tau_1, \ldots, \tau_1 + \delta_2 - 1, \tau_1 + \delta_2\}$, where δ_1 and δ_2 are non-negative integers to be chosen as large as possible, such that

$$\Gamma^{\theta'}_t > \Gamma^{\theta'}_{t'}, \quad \forall\, t \in \mathcal{N}_{\tau_1}, \forall\, t' \notin \mathcal{N}_{\tau_1}. \tag{3.21}$$

For a proper search step-size q ($q \ll n$), our proposed two-stage search method can decrease the dimension of the search scope for t by almost q times .

3.2.4 Simulation

The performance of our proposed new joint blind LDPC-encoder identification and blind frame synchronization scheme is evaluated by Monte Carlo computer simulations in this section. The performance metric we choose is the *probability of correct identification*, P_c, which is

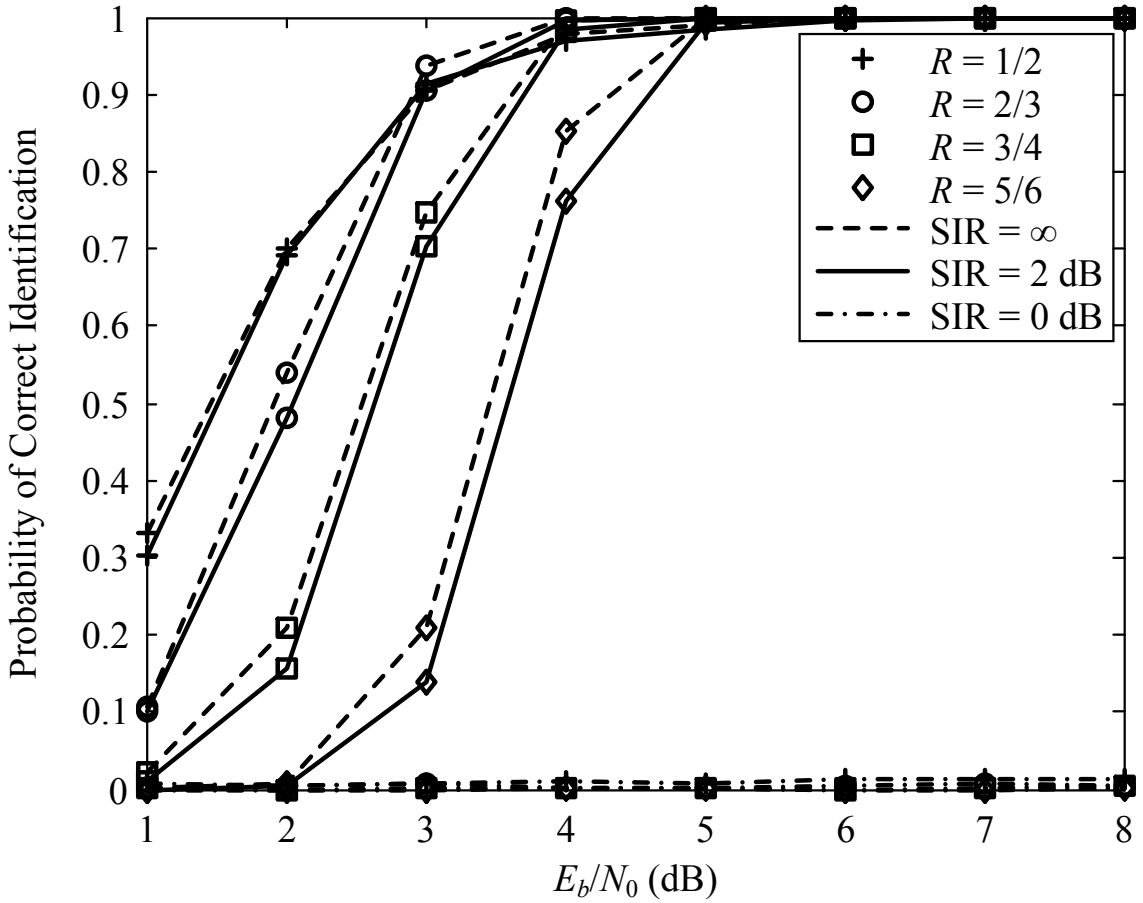

Figure 3.4: The probabilities of correct identification P_c with respect to E_b/N_0 for different SIR values when $L = 3$ (three channel paths).

the probability that the receiver can correctly identify the LDPC encoder the transmitter actually adopts and estimate the true time delay of the first path, i.e., $P_c \stackrel{\text{def}}{=} P_r\{\hat{\theta} = \theta, \hat{\tau} = \tau_1\}$. The LDPC parity-check matrices with codeword length $n = 648$ specified in the IEEE 802.11-2012 standard [20] are adopted for our simulations. Hence, there are four encoder candidates to form Θ. Different channel scenarios are randomly created by different numbers of channel paths as well as different SIR values. One thousand Monte Carlo experiments are carried out to obtain the average performance for each particular simulation set-up.

Figure 3.4 depicts the probability of correct identification with respect to E_b/N_0 for different SIR values. When SIR approaches either infinity (the multipath channel becomes the

AWGN channel in this case) or 2 dB, the probability of correct identification can reach up to 100% as E_b/N_0 is larger than 7 dB. In other words, our proposed joint blind scheme can correctly identify the encoder and estimate the true time delay at the same time. Moreover, it is shown that the performance difference between SIR $= \infty$ dB and SIR $= 2$ dB is very small. On the contrary, when SIR decreases to 0 dB, the probability of correct identification cannot be any significantly better than $P_c = 0\%$ for all encoders no matter how large E_b/N_0 is. It is also observed that the lower the code rate of the encoder, the better the detection performance (similar to the phenomenon presented in our previous chapter [51]). This is because the lower-rate encoder has more parity-check bits which lead to a more reliable average LLR.

The exceptions can be found for SIR $= 2$ or ∞ dB, where the probability of correct identification of the rate 1/2 encoder becomes smaller than that of the rate 2/3 encoder when E_b/N_0 is larger than 3 dB. This eccentric phenomenon is due to the peak widths of the average LLRs $\Gamma_t^{\theta'}$ as shown in Figure 3.5. Each sub-figure of Figure 3.5 depicts the average LLR $\Gamma_t^{\theta'}$ versus t for $\theta' = \theta$. The number of channel paths is $L = 3$ and the SIR is set to be 5 dB. For the rate 1/2 encoder, the peak width covers 11 samples within $t \in [144, 154]$; for the rate 2/3 encoder, the peak width covers 5 samples in the interval $t \in [165, 169]$; for the rate 3/4 encoder, the peak width covers 4 samples in the interval $t \in [94, 99]$; for the rate 5/6 encoder, the peak width covers only 3 samples in the interval $t \in [244, 246]$. It suggests that the average LLR of the lower rate encoder has a larger peak width. When the noise level is high, the more ambiguity (error) would be induced for time-delay estimation by the larger peak width. Indeed, the time-delay estimate is usually just a sample apart from the true time delay when this phenomenon is observed during our simulations.

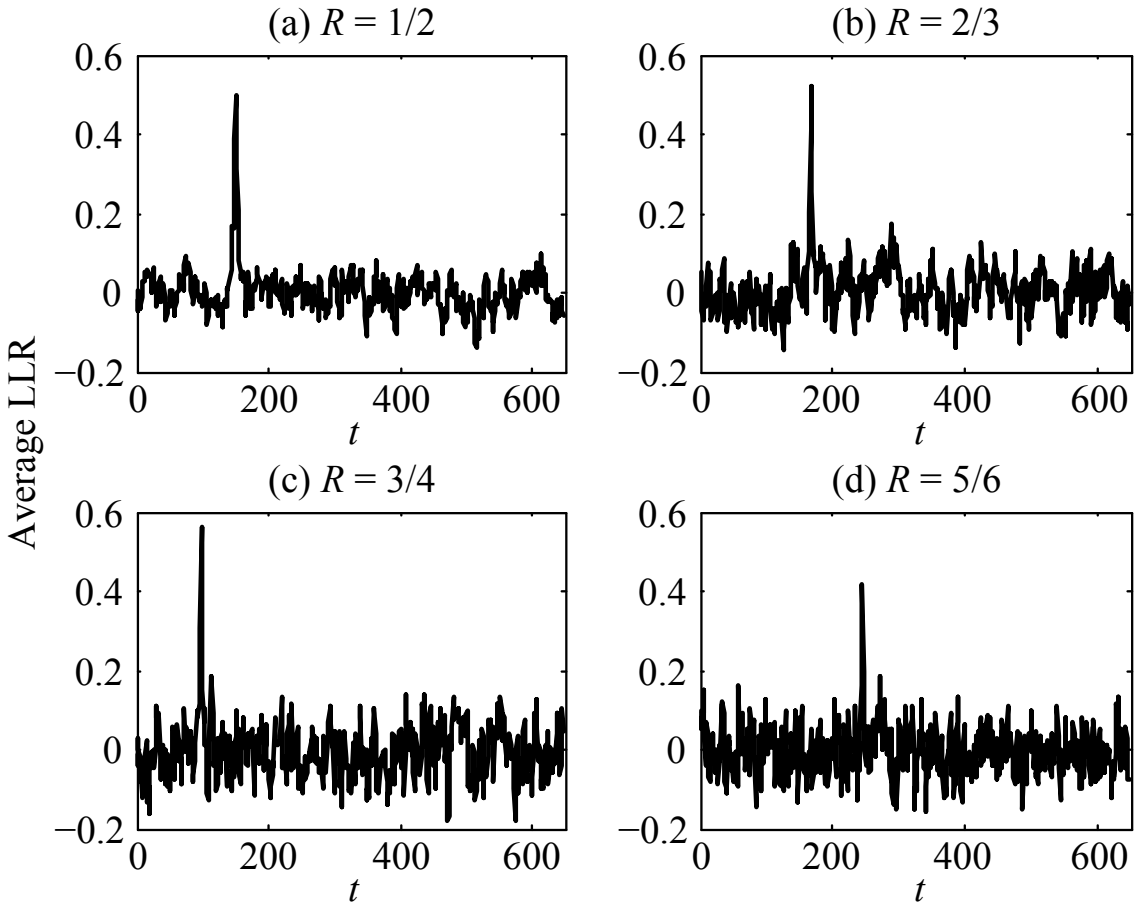

Figure 3.5: The average LLR $\Gamma_t^{\theta'}$ with respect to the sliding window's starting time point t for each encoder $\theta' = \theta$.

On the other hand, the peak width of the average LLR is also related to the proper value of the search step-size for each LDPC encoder, which is a key parameter involved in our proposed two-stage search scheme. Recall that the peak width of the average LLR is due to the quasi-cyclic property of the parity-check matrix. The reason that the peak width decreases as the code rate increases as demonstrated by Figure 3.5 is that the two consecutive rows of the parity-check matrix for the higher rate encoder are much more different due to a larger *row weight* of its parity-check matrix. The search step-size cannot be chosen larger than the peak width; otherwise, there may be some risk that no time point would be chosen from the peak area in the first stage so the global maximum of average LLR cannot be

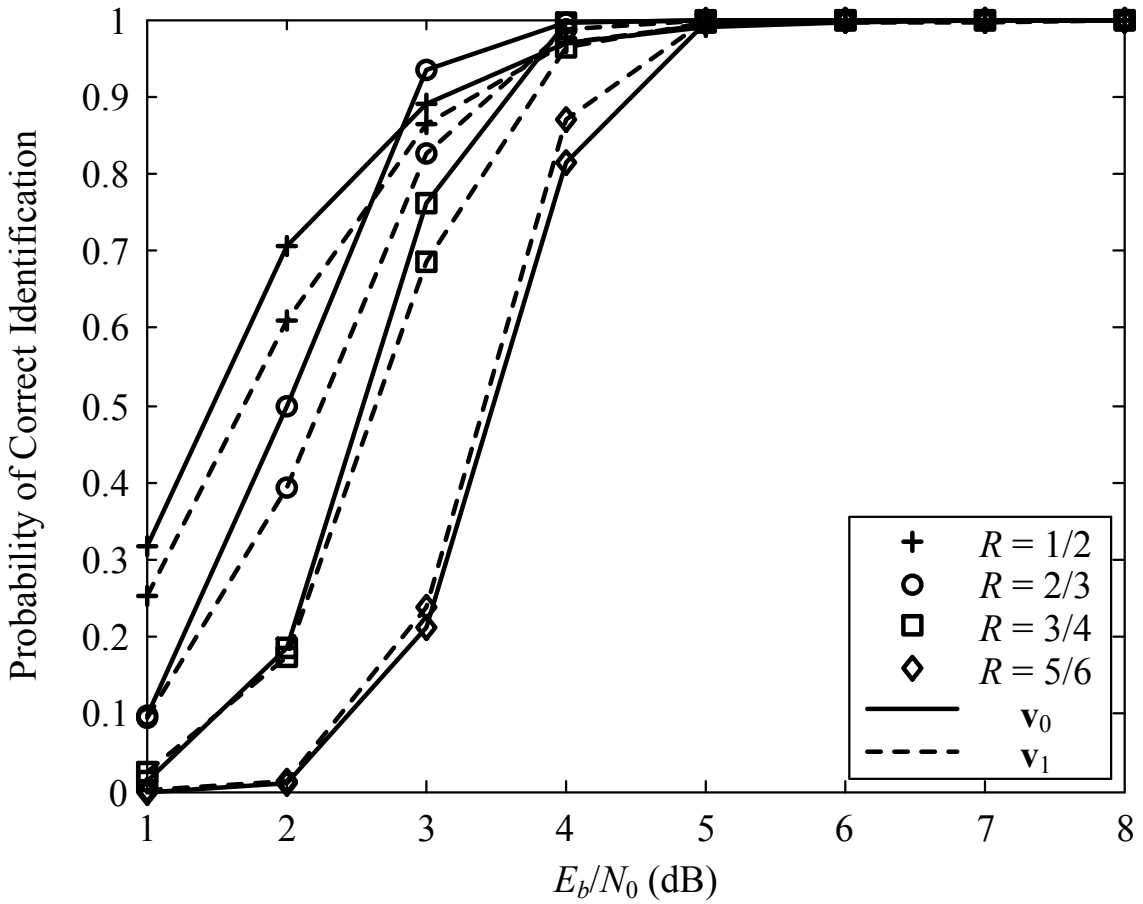

Figure 3.6: The probabilities of correct identification P_c with respect to E_b/N_0 for the search step-size scenarios $\mathbf{v}_0$ and $\mathbf{v}_1$ when $L = 3$ (three channel paths) and SIR = 5 dB.

spotted during the second stage.

Figure 3.6 and Figure 3.7 illustrate the probabilities of detection for three different search step-size scenarios denoted by $\mathbf{v}_0 = [1, 1, 1, 1]^T$ (one-stage search only, namely the conventional sample-by-sample search), $\mathbf{v}_1 = [5, 3, 2, 1]^T$, and $\mathbf{v}_2 = [10, 6, 4, 2]^T$. The four elements of $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_0$ correspond to the search step-size for the rate 1/2, 2/3, 3/4, and 5/6 encoders, respectively. The lower rate encoder should correspond to a larger search step-size because it leads to a broader peak width as illustrated by Figure 3.5. One can observe from Figures 3.6 and 3.7 that the detection performances for the search step-size scenario $\mathbf{v}_1$ almost stay the same as those for the scenario $\mathbf{v}_0$. However, the detection performances for the

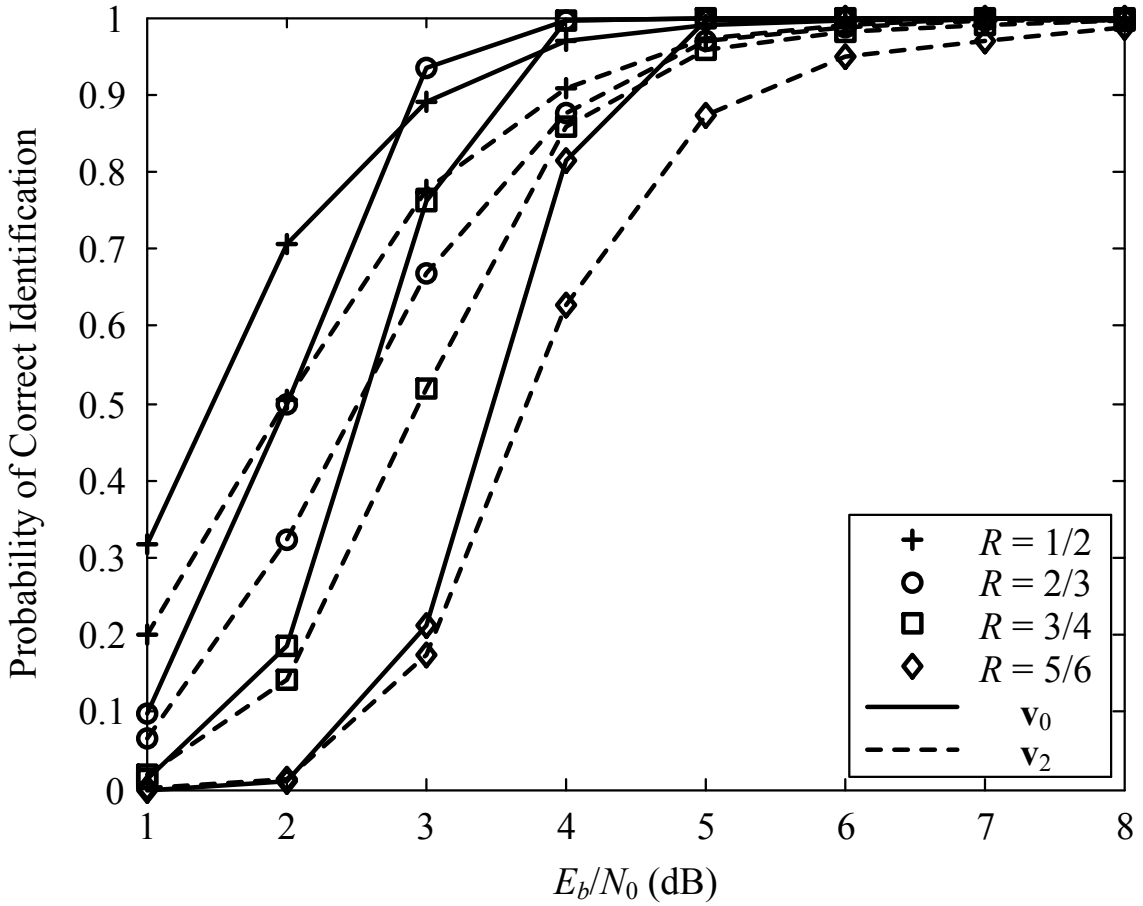

Figure 3.7: The probabilities of correct identification P_c with respect to E_b/N_0 for the search step-size scenarios $\mathbf{v}_0$ and $\mathbf{v}_2$ when $L = 3$ (three channel paths) and SIR = 5 dB.

search step-size scenario $\mathbf{v}_2$ demonstrate obvious degradations compared to the conventional scenario $\mathbf{v}_0$.

Finally, the probabilities of detection are also investigated for two different numbers of channel paths ($L = 3$ and $L = 5$) in Figure 3.8. It is shown that the number of channel paths poses little impact on the detection performance when the SIR is fixed.

3.2.5 Summary

In this section, we propose an innovative joint scheme for blind LDPC encoder identification and blind frame synchronization in a unified framework. The encoder is blindly identified

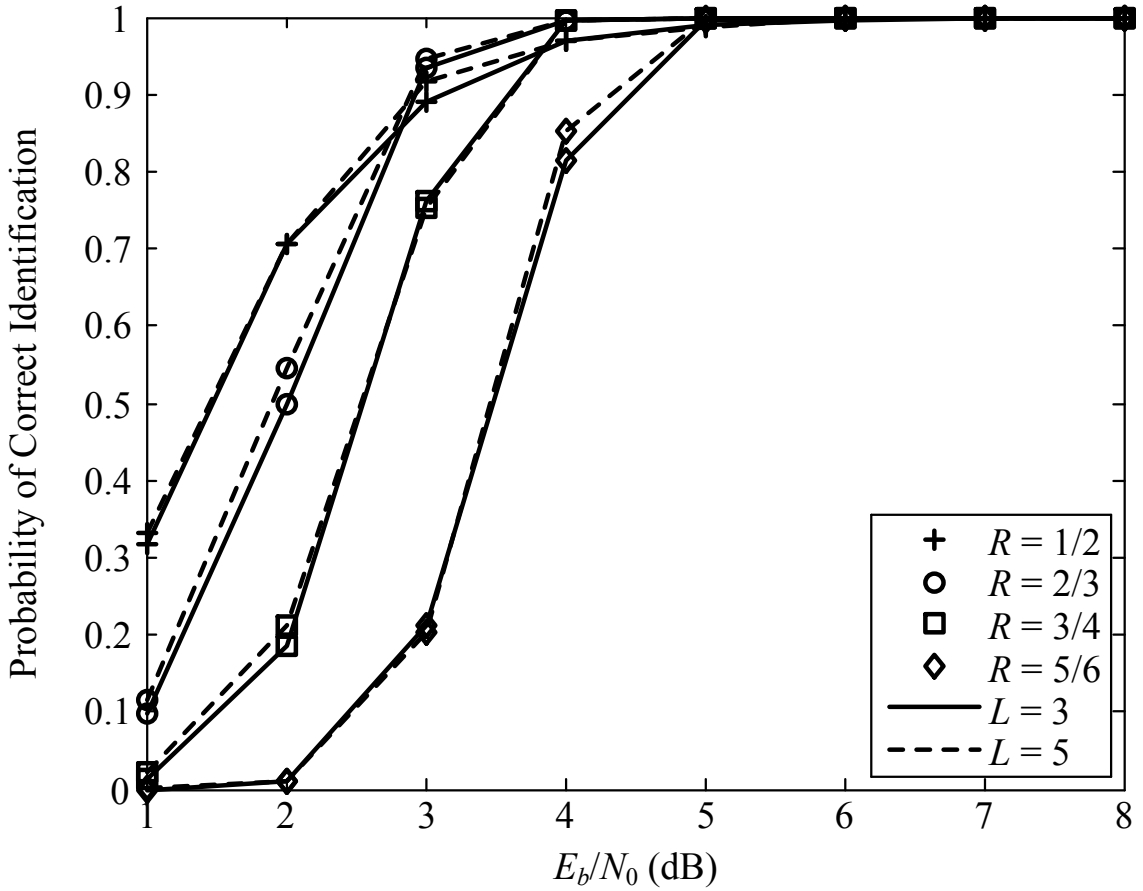

Figure 3.8: The probabilities of correct identification P_c with respect to E_b/N_0 for $L = 3$ and $L = 5$ when SIR = 5 dB. The conventional one-stage sample-by-sample search is used here.

and the time delay is blindly estimated both by maximizing the average log-likelihood ratio. In order to mitigate the computational complexity of blind frame synchronization, we propose a new two-stage search algorithm with a search step-size q by taking advantage of the quasi-cyclic property of the parity-check matrices so that the complexity of our proposed scheme can be greatly reduced by almost q times. The effectiveness of our proposed new joint blind scheme is evaluated by the probability of detection performances. Numerous Monte Carlo simulations are carried out for various SIRs and multipath channel scenarios. Besides, how to choose the proper search step-sizes is also studied via empirical information.

4. FAST LDPC DECODING ALGORITHMS

In this chapter, we will shift our focus from blind identification schemes on LDPC codes to LDPC decoding techniques. In Chapter 4.1, we propose a new stopping criterion which can be useful when undecodable blocks are experienced frequently. In Chapter 4.2, we propose an efficient informed dynamic scheduling (IDS) algorithm for LDPC decoding, which can provide more freedom in the performance-complexity trade-off compared to existing IDS algorithms. Both schemes discussed in this chapter are to reduce the number of iterations during the iterative decoding process. The computational complexities involved in both schemes are very little compared to the existing algorithms.

4.1 A New Stopping Criterion

The outstanding LDPC decoding performance relying on the conventional *belief propagation* (BP) algorithms would often induce high computational complexity, especially for nonbinary LDPC codes. The computational bottleneck lies in the *check node processing* in the iterative BP algorithms [15, 53]. To combat this complexity problem, the *extended min-sum* (EMS) algorithm was proposed lately where the messages of length q were truncated by choosing the n_m largest ones ($n_m \ll q$) [17, 54]. In [55, 56], new sophisticated decoding algorithms were proposed to adaptively truncate the message length over iterations to further reduce the decoding complexity.

In addition to the message length, the decoding complexity of BP algorithms for LDPC codes also depends on the required iteration number. Usually, a maximum iteration number

is preset. The BP algorithms terminate when either the estimated codeword satisfies all parity checks or the maximum iteration number is reached. This conventional stopping criterion is not efficient since it is observed that for the undecodable blocks, the BP algorithms always run up to the maximum iteration limit and cannot generate the correct codeword. Therefore, it would be better to terminate the BP algorithms early so as to avoid the unnecessary computational time and reduce the extra power consumption when undecodable blocks are experienced. To deal with this problem, a stopping criterion was proposed before to decode binary LDPC codes using the *variable node reliability* for identifying undecodable codeword blocks in [57]. In [58], the iterative decoding process would be terminated when a *soft-word cycle* appears. In [59], an BP algorithm would skip further iterations if the mean magnitude of the log-likelihood ratios converges. In [60], an *error frame* is identified for both binary and nonbinary LDPC codes when the *check-sum* stays the same value over several consecutive iterations.

However, the aforementioned previous stopping criteria either require a lot of memory storage space [58] or involve several parameters which are difficult to optimize systematically in practice [57, 59]. Moreover, it cannot be justified that undecodable blocks always demonstrate the constant check-sum patterns. To address these issues, in this chapter, we propose a robust new stopping criterion for both binary and nonbinary LDPC decoders, which involves only one parameter and requires little computational burden. Heuristically, the maximum *a posteriori* probability (APP) for each variable node would increase when an BP algorithm is executed in progress across iterations [55]. Consequently, we propose to use the total *a posteriori* probability (APP) as the essential metric to check if an BP algorithm has a tendency to converge to a correct solution. Specifically, we propose a T-tolerance

criterion such that the iterative decoding process terminates when the total APP decreases exactly for $T+1$ times in aggregate during the execution of an BP algorithm.

The rest of this section is organized as follows. The LDPC decoding complexity in terms of iteration number is analyzed in Chapter 4.1.1. Based on this analysis, we devise a new T-tolerance stopping criterion for BP algorithms in Chapter 4.1.2. Monte Carlo simulation results are shown in Chapter 4.1.4 to demonstrate the trade-off between the error performance and the complexity of our proposed stopping criterion for both binary and nonbinary LDPC codes.

4.1.1 Undecodable Blocks

To reduce the overall complexity of an BP algorithm for decoding LDPC codes, not only it is important to lower the computational complexity for each iteration, but also it is desirable to make the required iteration number as small as possible. Generally speaking, the lower E_b/N_0 is given, the more iterations may be required. In this section, we carry out Monte Carlo simulations to illustrate how many iterations are actually taken for a given E_b/N_0 when a certain LDPC codeword is transmitted. We take a $(147, 108)$ LDPC code over $\mathbb{GF}(64)$, which is arbitrarily generated using the finite field method according to [61], for illustration. The maximum iteration number is set to be 50. One thousand Monte Carlo trials are undertaken for each of four E_b/N_0 values. In each trial, the codeword is modulated by 64-quadrature amplitude modulation (64-QAM) and transmitted over the additive white Gaussian noise (AWGN) channel. The received signal is decoded using q-ary sum-product algorithm (QSPA) subject to the maximum iteration number.

Table 4.1: Proportions of decodable and undecodable blocks

E_b/N_0 (dB)	8	9	10	11
Decodable (correct)	0.168	0.837	0.993	1
Decodable (wrong)	0.017	0.033	0.003	0
Undecodable	0.815	0.130	0.004	0

According to Table 4.1, the estimated codewords can be classified into three groups: (*i*) codewords which are correctly restored, (*ii*) codewords which satisfy the parity checks but are not the true ones, and (*iii*) codewords which are undecodable and thus exhaust all possible iterations but the syndrome still contains nonzero element(s). For the blocks belonging to the first two aforementioned groups, we say that the blocks are *decodable* since QSPA succeeds in generating a valid codeword. It is explicit that for low E_b/N_0, undecodable blocks account for a large portion, while for high E_b/N_0 undecodable blocks are rare.

To investigate how many iterations are taken for decodable blocks, we depict the *cumulative density function* (CDF) of the iteration numbers subject to different E_b/N_0 values in Figure 4.1. One can observe that when E_b/N_0 is 9 dB, more than 95% of decodable blocks need at most 20 iterations for QSPA to converge; when E_b/N_0 is enlarged to 11 dB, all decodable blocks need at most just 5 iterations for QSPA to converge. On the other hand, the maximum iteration number should be set to 48 when E_b/N_0 is 9 dB for all decodable blocks, and it should be reduced to 33 when E_b/N_0 increases to 10 dB.

Since it is hard to estimate a precise upper bound of the required iteration number for an arbitrary E_b/N_0, the maximum iteration number is usually set to be rather large in BP algorithms. However, the larger the maximum iteration number, the less efficient the BP algorithm when undecodable blocks are encountered, especially in low E_b/N_0 conditions. To

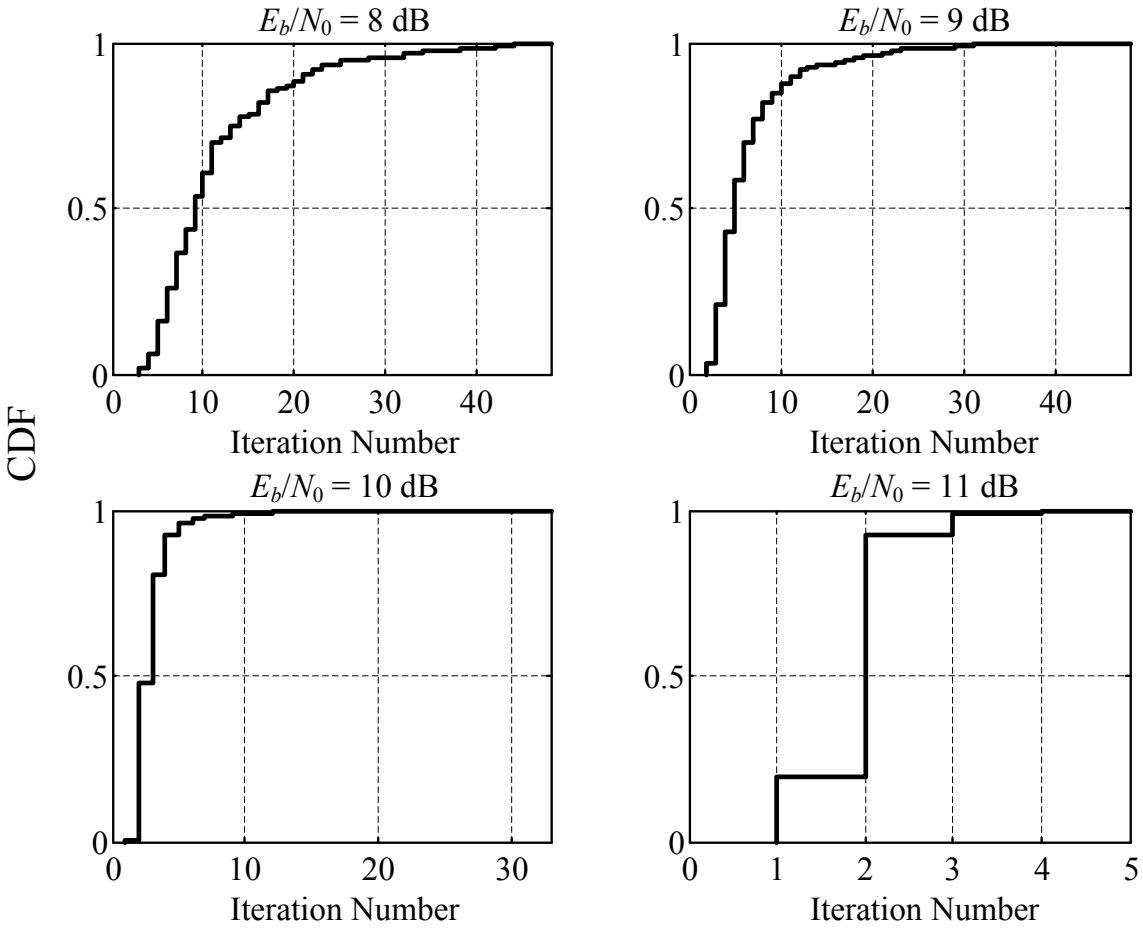

Figure 4.1: The cumulative density functions of the iteration numbers required for decodable blocks subject to different E_b/N_0 values.

tackle this difficulty, in the next section, we will introduce a new stopping criterion for BP algorithms.

4.1.2 Robust T-Tolerance Stopping Criterion

For *undecodable* blocks which constitute a great portion of codeword blocks in the low signal-to-noise ratio (SNR) conditions, no matter how large the maximum iteration number is chosen, the BP algorithm will always use up all iterations but fail to converge to the correct codeword. Indeed, the conventional stopping criterion of BP algorithms does not involve any *convergence check* during the iterative decoding process and therefore lacks the ability to

identify the undecodable blocks. In this section, we propose a novel robust stopping criterion to monitor the evolution of the *total APP* such that the BP iterative decoding process can be terminated at an early stage when dealing with an undecodable block.

4.1.2.1 Total A Posteriori Probability

For BP algorithms in the probability domain, at the end of the t^{th} iteration, the estimated codeword symbols are obtained as

$$\hat{c}_j = \arg\max \left\{ \mathbf{p}_j^{(t)} \right\}, \quad j = 1, 2, \ldots, n \tag{4.1}$$

where $\mathbf{p}_j^{(t)}$ is the $q \times 1$ APP vector whose elements sum to 1, q is the order of the Galois field, and n is the codeword length. Refer to [15] for the detailed formulation of the APP vectors. If an BP algorithm, say QSPA, can converge to the correct codeword, the distribution of APP vectors $\mathbf{p}_j$ would concentrate more and more around the j^{th} correct symbol as the iteration number increases according to [55] since the iterative decoding process can be deemed an *SNR amplifier* [62]. On the other hand, if the BP algorithm has difficulty to converge, it indicates that some estimated symbols either change from iteration to iteration or converge to the incorrect results. Based on this fact, to distinguish whether the iterative decoder is *progressive* or *stagnant*, we propose to use the measure of *total APP*, denoted by $P^{(t)}$, as given by

$$P^{(t)} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{j=1}^{n} \max \left\{ \mathbf{p}_j^{(t)} \right\}. \tag{4.2}$$

Figure 4.2 depicts the evolution of the total APP $P^{(t)}$ with respect to the iteration number t when E_b/N_0 is fixed at 9 dB for the nonbinary LDPC code $(147, 108)$ over $\mathbb{GF}(64)$ using QSPA. Note that four individual trials drawn from the simulations stated in Chapter 4.1.1

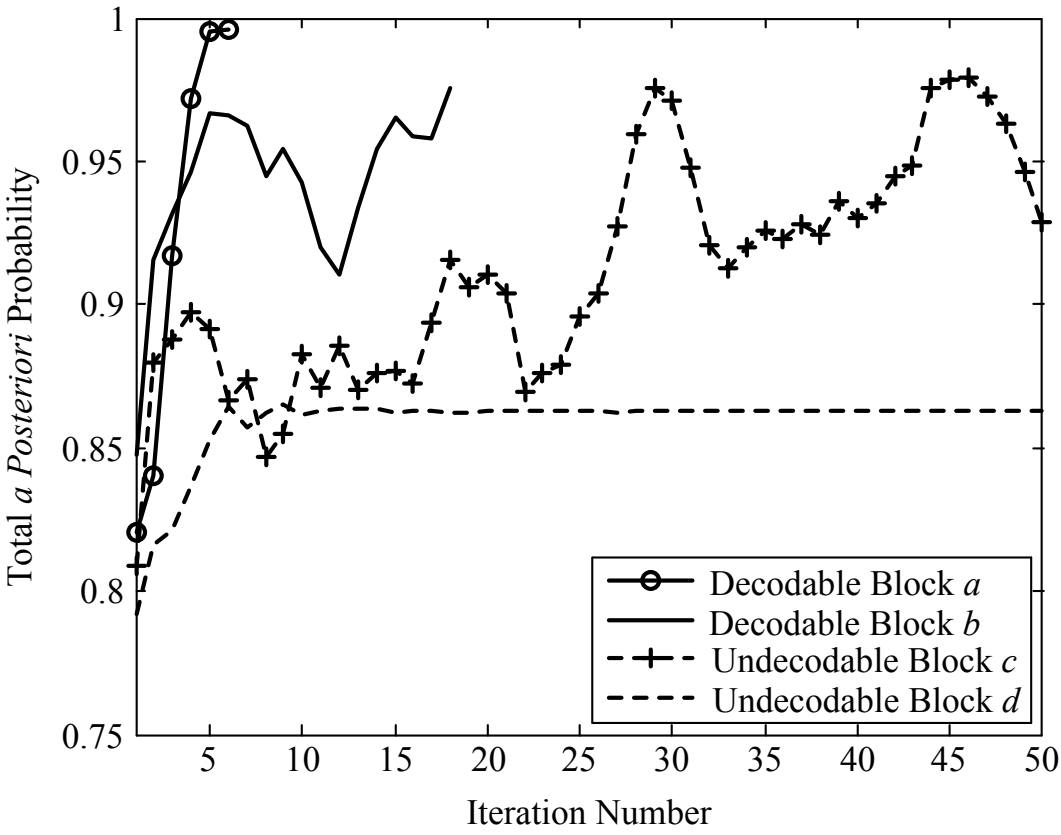

Figure 4.2: The evolution of the total APP $P^{(t)}$ with respect to the iteration number t for $E_b/N_0 = 9$ dB.

are evinced here as examples: two decodable blocks a and b and two undecodable blocks c and d. Specifically, both block a and block b are correctly recovered by using six and eighteen iterations of QSPA, respectively. On the other hand, for both undecodable blocks c and d, QSPA runs out of fifty iterations but cannot restore any valid codeword.

4.1.2.2 New T-Tolerance Criterion

According to Figure 4.2, the number of the cumulative times that the total APP decreases from one iteration to next during the LDPC decoding process for undecodable blocks is usually much larger than that for decodable blocks. Based on this key observation, we devise a new *T-tolerance stopping criterion* for BP algorithms where T-tolerance suggests

that an BP algorithm can only tolerate at most T times of total APP decreases and will terminate as long as the number of the cumulate times of such total APP decreases exceeds T.

In detail, the T-tolerance stopping criterion can be carried out as follows:

(1) Initialization: Set the tolerance time T, and reset a counter $C = 0$;

(2) At the end of the t^{th} iteration, compute the total APP $P^{(t)}$ according to Eq. (4.2);

(3) Compare $P^{(t)}$ with the previous value $P^{(t-1)}$; if $P^{(t)} < P^{(t-1)}$, a *decrease* occurs and C is incremented by 1;

(4) Once $C > T$, the BP algorithm terminates; otherwise, continue to the next iteration $(t + 1)$ and go back to Step (2).

Obviously, our proposed T-tolerance criterion only requires to record two scalars, namely $P^{(t)}$ and C. The computation of $P^{(t)}$ is also handy since the "max" operation in Eq. (4.2) is already available when the BP algorithm iteratively estimates the codeword in the probability domain. Hence, only a simple arithmetic average is required as the additional computational burden. For the BP algorithms executed in the logarithm domain, the log-likelihood ratios need to be converted to the APPs (see [55]) to accommodate our proposed new stopping criterion. Note that our proposed new T-tolerance criterion will be incorporated into the standard syndrome-check stopping criterion for early termination of BP algorithms. Thus, the iterative decoding process will terminate when either there are T times of total APP decreases or the maximum iteration number is reached.

4.1.3 Complexity Comparison

The complexity comparison between the BP decoding using our proposed T-tolerance stopping criterion and using the conventional syndrome check stopping rule is investigated as follows. As the T-tolerance stopping criterion includes the syndrome check process, the computational complexity of the T-tolerance stopping criterion actually increases for each iteration of the BP decoding process. The extra computational complexity arises from calculating the total APP expressed by Eq. (4.2), which is $\mathcal{O}(qn)$. If the messages are represented by log-likelihood ratios (LLRs) rather than probabilities, then a conversion from LLRs to probabilities is needed, the complexity of which is also $\mathcal{O}(qn)$. The extra memory requirement for the T-tolerance stopping criterion is negligible, only two scalars, the total APP and the counter, need to be stored. The complexity of comparison between $P^{(t)}$ and $P^{(t-1)}$ can also be neglected. Note that the computational complexity of the check-node processing in the BP decoding algorithms is $\mathcal{O}(q \log q \; N_{\text{edge}})$, where N_{edge} is the total number of edges (non-zero entries of the parity-check matrix). Thus, the extra computational complexity of the T-tolerance stopping criterion is rather small compared to the that of the conventional BP decoding algorithm, which is on the order of $\frac{n}{\log q \; N_{\text{edge}}}$ of the conventional BP decoding algorithm.

On the other hand, the number of iterations used by the T-tolerance stopping criterion is no more than that used by the conventional BP decoding algorithm. When undecodable blocks are frequently encountered, it is shown by simulations in the next section that the number of iterations can be greatly reduced by adopting the T-tolerance stopping criterion compared to that of the conventional BP decoding algorithm. Therefore, it is favorable

to adopt the T-tolerance stopping criterion when undecodable blocks are experienced very often.

4.1.4 Simulation

The performance of a popular BP algorithm (QSPA) using our proposed new T-tolerance criterion is investigated for both binary and nonbinary LDPC codes via numerous Monte Carlo simulations. A binary $(648, 324)$ LDPC code from the IEEE 802.11-2012 standard [20] and a nonbinary $(147, 108)$ LDPC code over $\mathbb{GF}(64)$ generated according to [61] are used for illustration. The maximum iteration number is defaulted to 50. The binary $(648, 324)$ LDPC codewords are modulated by binary phase-shift keying (BPSK) and the nonbinary $(147, 108)$ LDPC codewords are modulated by 64-QAM. All codewords are transmitted over the AWGN channel.

Figure 4.3 depicts the *frame error rate* (FER) performance of binary $(648, 324)$ LDPC code versus E_b/N_0 for different T values compared to the performance of the conventional *sum-product algorithm* (labeled as "SPA" in the figure). It can be observed that the larger the value of T, the better the FER performance. When T is too small, such as $T = 0, 1$, the T-tolerance criterion is more likely to terminate the algorithm too early before the decodable block can be recovered. However, the FER performance of SPA using the T-tolerance criterion approaches the performance of the conventional scheme very quickly as T increases. It is shown that the FER performance degradation due to the 5-tolerance criterion ($T = 5$) is almost within 0.1 dB compared to the conventional SPA in the low E_b/N_0 conditions.

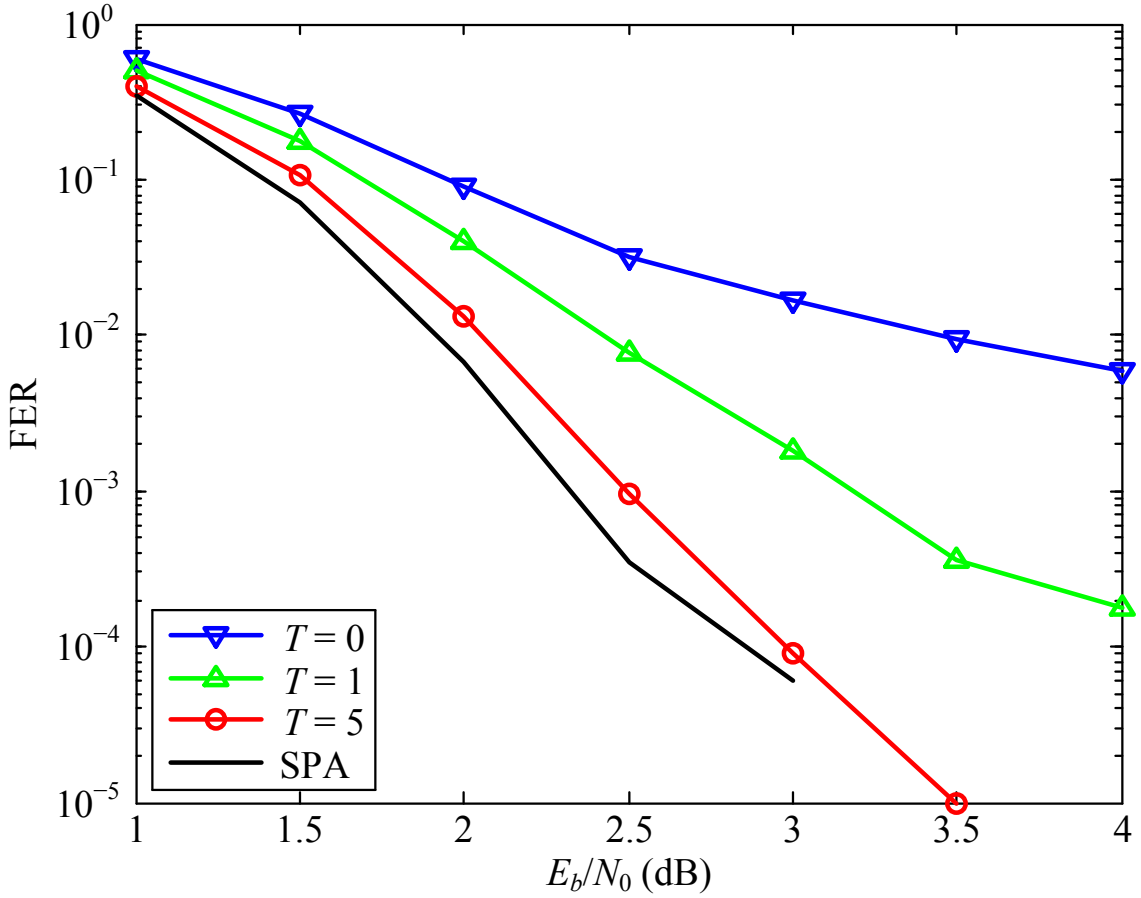

Figure 4.3: The frame error rate of the binary LDPC code $(648, 324)$ versus E_b/N_0 for different T values.

Since the proposed stopping criterion is to reduce the iteration numbers (thus reducing the complexity), Figure 4.4 is drawn to show the *average iteration number* (AIN) versus E_b/N_0 for different T values compared to that of the conventional SPA. It can be observed that the smaller the value of T, the less the value of AIN. The AIN gap between the SPA using the 5-tolerance criterion and the conventional SPA scheme is large in the low E_b/N_0 scenarios.

Figure 4.5 and Figure 4.6 also depict the FER performances and the AINs of nonbinary $(147, 108)$ LDPC code versus E_b/N_0 for different T values compared to the performances of the conventional q-ary sum-product algorithm (labeled as "QSPA" in the figures), re-

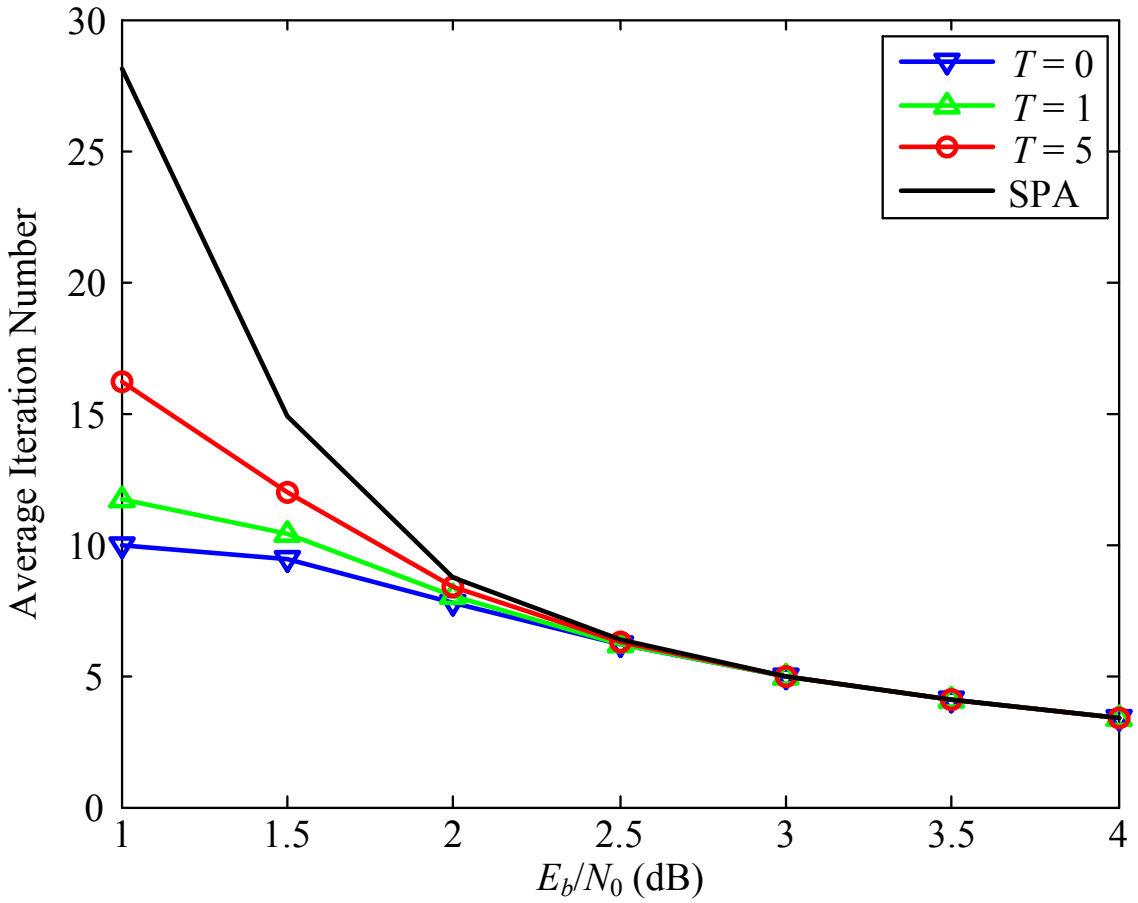

Figure 4.4: The average iteration number of the binary LDPC code $(648, 324)$ with respect to E_b/N_0 for different T values.

spectively. Compared to the results for the binary code in Figures 4.3 and 4.4, the FER performances of the nonbinary code for $T = 0, 1$ are much closer to the performance of the conventional scheme, and the AIN gap between the QSPA using the T-tolerance criterion and the conventional scheme is yet larger in the low E_b/N_0 scenarios.

The FER-complexity trade-off can simply be maneuvered by adjusting T. Our simulation results suggest that it is favorable to adopt the T-tolerance criterion in the low E_b/N_0 situations where undecodable blocks are often found.

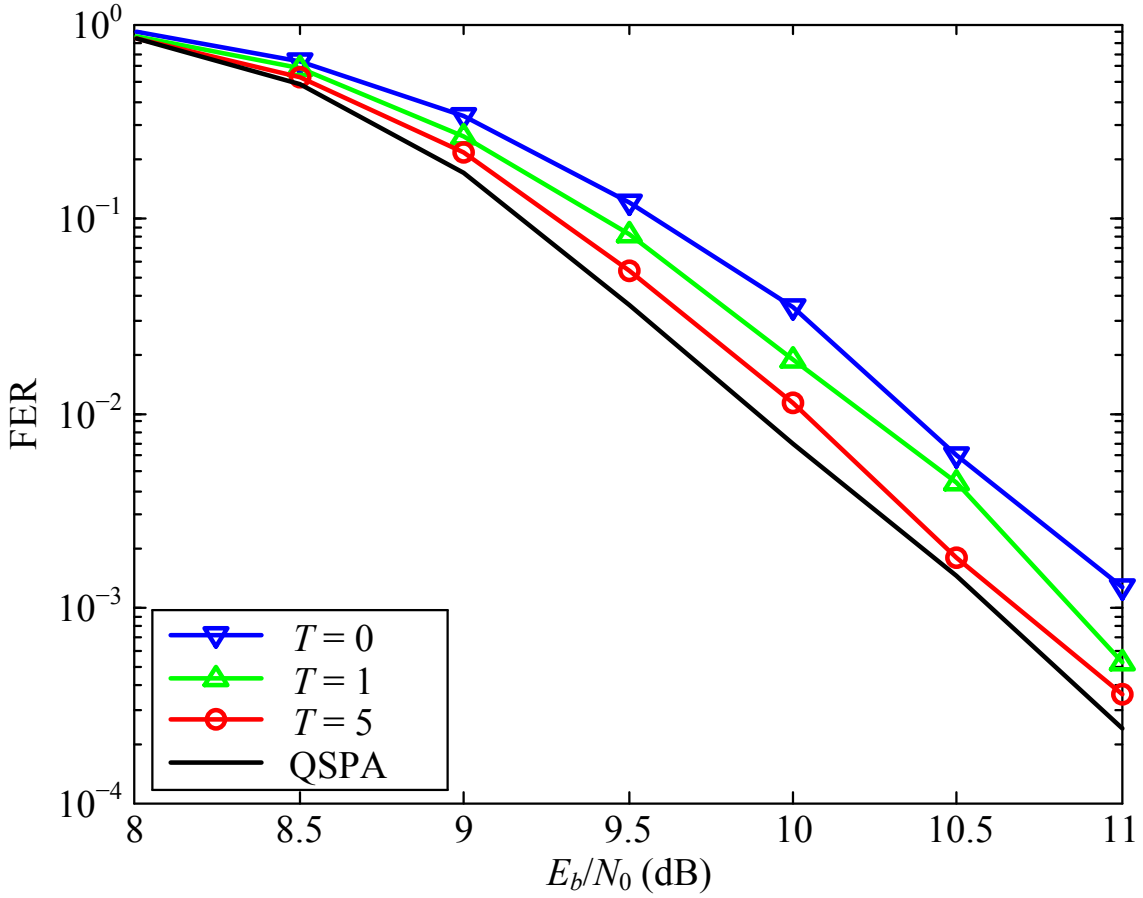

Figure 4.5: The frame error rate of the nonbinary LDPC code $(147, 108)$ over $\mathbb{GF}(64)$ versus E_b/N_0 for different T values.

4.1.5 Summary

In this section, we propose a novel T-tolerance stopping criterion for decoding LDPC codes. It allows the total *a posteriori* probability to decrease iteration by iteration at most T times when a message passing algorithm is executed. Monte Carlo simulation results demonstrate that the 5-tolerance criterion ($T = 5$) can greatly reduce the average iteration number (complexity) in the low E_b/N_0 conditions while maintaining the frame-error-rate performance degradation within 0.1 dB. It is desirable to adopt our proposed new stopping criterion when undecodable blocks are frequently encountered during the LDPC decoding process.

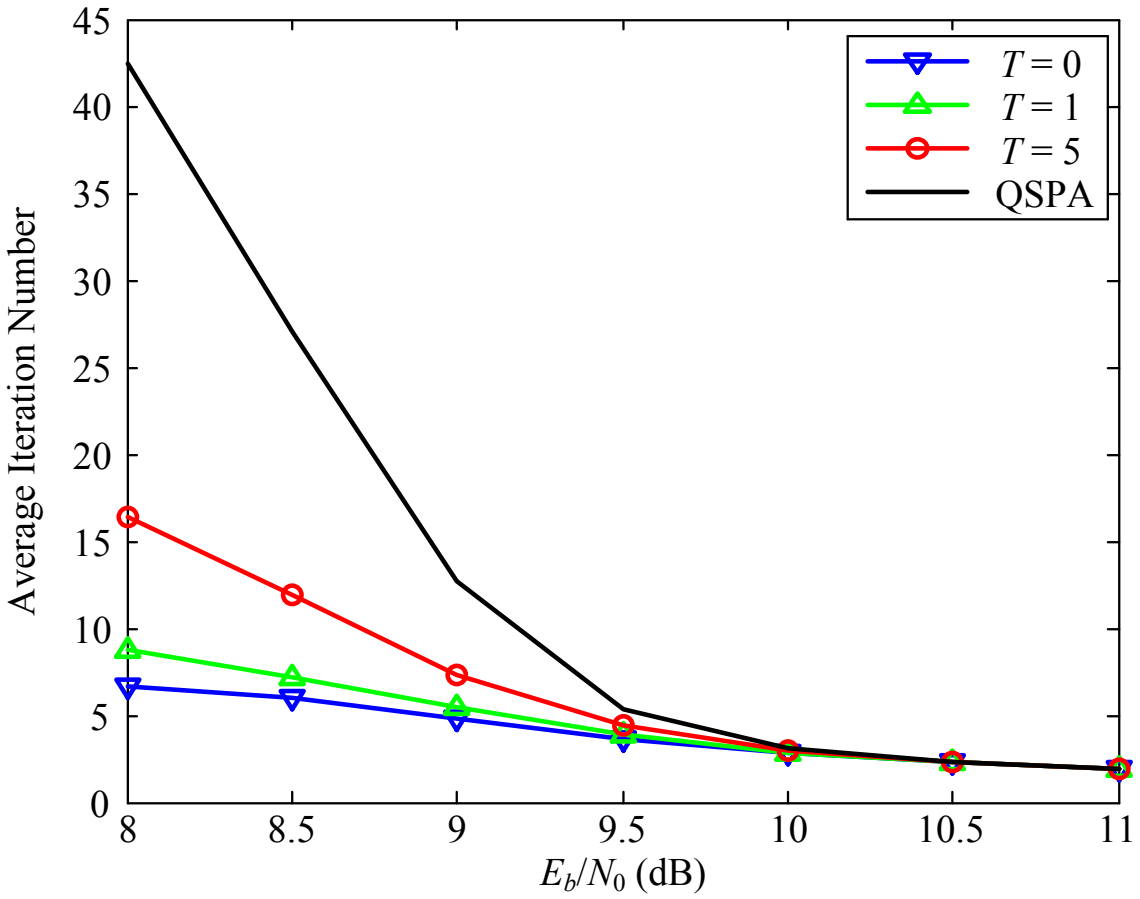

Figure 4.6: The average iteration number of the nonbinary LDPC code $(147, 108)$ over $\mathbb{GF}(64)$ with respect to E_b/N_0 for different T values.

4.2 An Efficient APP-based Dynamic Scheduling

The LDPC decoding procedure is usually carried out using the belief-propagation (BP) algorithms as discussed in Chapter 1.2. In practice, an LDPC code is of finite length and the Tanner graph representation of its parity-check matrix (PCM) inevitably has cycles [2]. As a result, the BP algorithm for LDPC decoding is an iterative process. The conventional BP decoding is carried out in a flooding fashion. That is, in one iteration, all the variable (check) nodes are updated in parallel, then all the check (variable) nodes are updated in parallel. On the contrary, serial BP decoding schemes can be carried out by sequentially

updating the variable nodes [63] or the check nodes [64]. The serial BP decoding in which the check nodes are updated in a fixed order is also called layered BP (LBP) decoding [65].

The advantage of the serial scheduling is that the useful extrinsic information can be used in the same iteration. Compared to the flooding BP (FBP) decoding, it is demonstrated by analysis and simulation that fixed-order serial scheduling schemes can reduce the required number of iterations by half and thus converge twice faster [65,66]. To make fair comparison between different LDPC decoding scheduling algorithms, we need to define the meaning of *iterations* and *sub-iterations*. For FBP decoding, one iteration consists of updating all variable and check nodes once, and there is no sub-iterations. For check-node (variable-node) based serial scheduling algorithms, one iteration is counted when the number of check-node (variable-node) updates is equal to the total number of check (variable) nodes, and each check-node (variable-node) update is called a sub-iteration. Note that in serial scheduling schemes, there is no restriction that every check (variable) node needs to be updated and updated once in each iteration. Some check (variable) nodes may be updated as many times and some other check (variable) nodes may not even have opportunities to be updated.

An example for illustrating the importance of scheduling is given by Figure 4.7 which is drawn from [2]. It can be seen that three erasures are recovered correctly by the FBP decoding using three iterations. One might ask how many iterations are required for a serial scheduling decoding algorithm to correct these three erasures. The answer is only one iteration. The order of the erasure being corrected by the FBP decoding algorithm actually suggests the sequential order for the serial scheduling decoding. Specifically, we first update the bottom check node (same as iteration 1), then update the top check node (same as iteration 2), and finally update the middle check node (same as iteration 3).

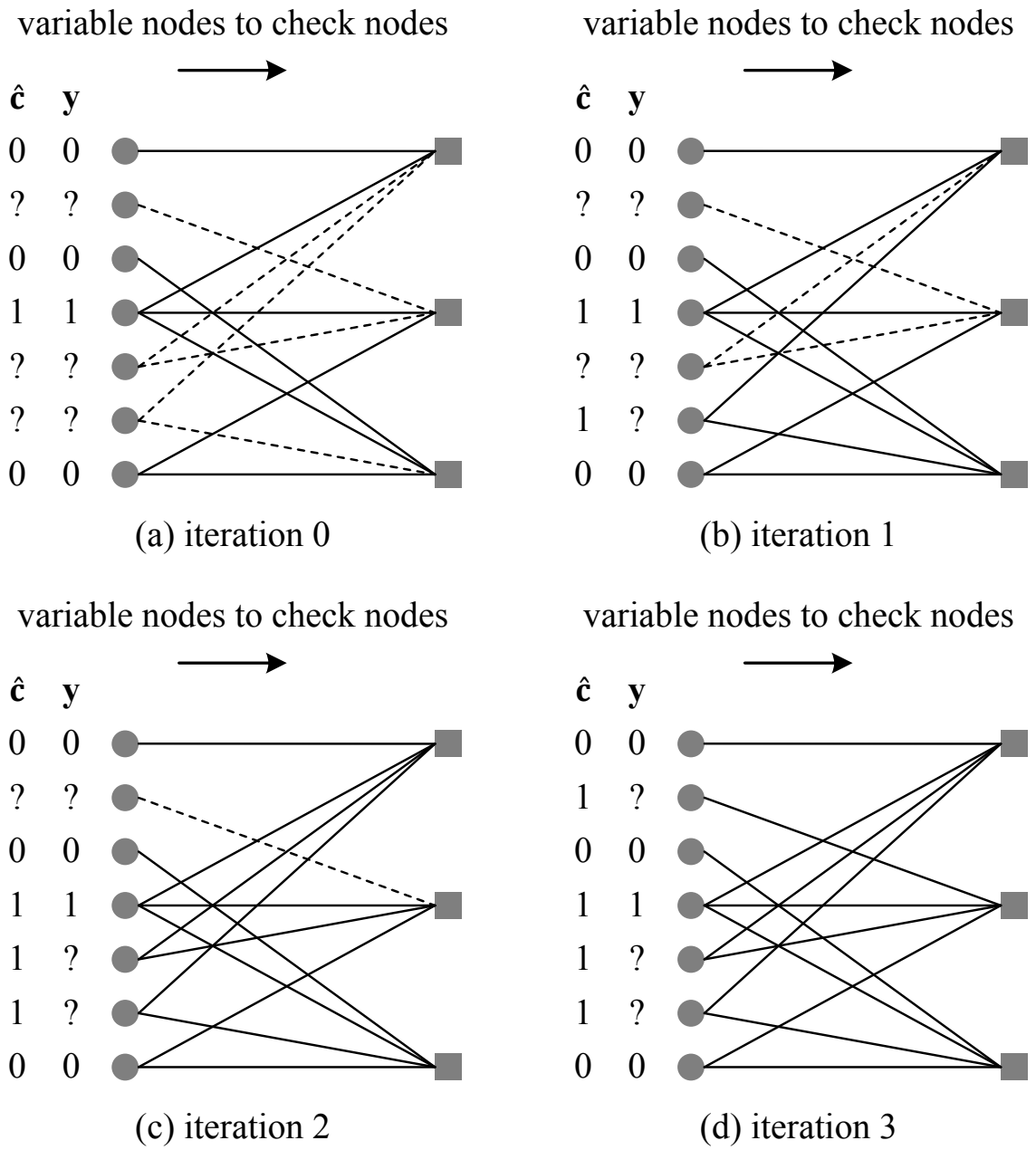

Figure 4.7: The illustration of the flooding BP decoding to correct erasures using three iterations [2]. The channel is binary erasure channel (BEC). The received signal is denoted by $\mathbf{y}$, and the estimated codeword is denoted by $\hat{\mathbf{c}}$. The solid line represents messages of 0 or 1, and the dashed line represents the messages of erasure after each iteration.

The optimal scheduling order for serial scheduling soon becomes intricate when we consider LDPC codes with codeword length over thousands and other channels, such as additive white Gaussian noise (AWGN) channels. The number of possible serial scheduling exponen-

tially increases with respect to the number of check (variable) nodes or even the number of edges (the non-zero entries in the corresponding PCM). The optimal serial scheduling algorithm for LDPC decoding is not discovered yet. In fact, it is still an open question whether such an optimal serial scheduling algorithm exists.

In the rest of this section, we first reviews several existing state-of-art serial scheduling algorithm in Chapter 4.2.1 which also includes a variable-node-wise RBP (VNWRBP). In Chapter 4.2.2, we propose an efficient serial scheduling algorithm, called *a posteriori* probability (APP) based residual BP (APPRBP). The simulation results are shown in Chapter 4.2.3 to demonstrate the effectiveness of our proposed APPRBP algorithm.

4.2.1 Existing Serial Scheduling Algorithms

Serial scheduling LDPC decoding algorithms appeared in early 2000s soon after the renaissance of LDPC codes [64, 67]. In [63], a serial BP decoding scheme was proposed to update the variable nodes sequentially. In [64, 65], it was suggested to serially update the check nodes. In these serial scheduling algorithms, the variable (check) nodes updating order is fixed for each iteration. It was demonstrated in these algorithms that the serial scheduling for LDPC decoding can converge twice faster compared to the conventional FBP decoding algorithm [65, 66]. The serial scheduling can also improve the bit-error-rate (BER) performance when the maximum number of iteration, denoted by N_{iter}, is limited. For the rest of this chapter, we choose the LBP algorithm [65] as the representative for these fixed-order serial algorithms.

Later, check-node based informed dynamic scheduling (IDS) algorithms, namely residual BP (RBP) and node-wise RBP (NWRBP), were proposed in [68, 69] in which the updating

order of check nodes is determined by the maximum residual. The residual is defined by the absolute difference between the message before and after the update. Consequently, the check nodes updating order is no long fixed but varies for each iteration in the IDS algorithms. The simulation results in [68, 69] suggested that the NWRBP algorithm not only converges faster than the LBP algorithm but also provides superior BER performance compared to the FBP algorithm. The reduction in number of iterations comes from the greedy nature of the NWRBP algorithm by updating the check node having the maximum residual. The reason for BER performance enhancement is that the NWRBP algorithm has capabilities to break the trapping sets when serially updating the check nodes.

Several other IDS algorithms were then proposed. In [70], an efficient dynamic schedule for layered LBP decoding of LDPC codes was presented. Note that the check nodes connecting to a variable node with maximum relative message residual are chosen to be updated. In [71], an IDS strategy, called informed variable-to-check RBP (IVCRBP) was proposed which utilizes not only the instability of the variable node (the hard decisions before and after one update are different) but also the residual of the variable-to-check message to locate the message to be updated first. In [72], a new dynamic selection strategy was proposed by adding the unsatisfied syndrome checks into consideration. An oscillation processing method was also proposed in [72] to suppress the oscillating behaviors of variable nodes.

One common feature of the aforementioned IDS algorithms is that after each sub-iteration, the stopping rule will be carried out. Note than the conventional stopping rule, namely the syndrome check, is not computationally free and would become a burden if carried out in such high frequency.

There are also some other serial scheduling algorithms in the existing literatures. Probabilistic scheduling based on the girth or reliability were proposed in [73–75], in which the variable or check nodes will be updated in certain probabilities determined by their corresponding girths or reliabilities. A maximum mutual information increase (M^2I^2)-based algorithm was proposed recently to find the fixed edge-based order [76]. The edge-based updating order is determined by running the protograph-based extrinsic information transfer (PEXIT) algorithm and finding the maximum mutual information increase. High order prediction was also proposed in [76] to determine the update sequence in more than one step. The edge updating order for an LDPC ensemble can be predetermined in the M^2I^2 algorithm. Thus the complexity is actually the same as the LBP algorithm. The simulation results in [76] demonstrated that the M^2I^2 algorithm is superior than the LBP algorithm in terms of convergence and the BER performance. However, the authors did not compare the M^2I^2 algorithm to the NWRBP algorithm in [76].

To facilitate the illustration of our proposed new scheduling algorithm in Chapter 4.2.2, here we present the variable-node-wise RBP (VNWRBP) algorithm which is the counterpart of the NWRBP algorithm in [69] but not presented in the existing literatures. Denote the message (extrinsic information in log-likelihood ratio) from the j^{th} variable node to the i^{th} check node before and after the update by $\alpha_{i,j}$ and $\alpha'_{i,j}$. Then the residual for each edge can be expressed by

$$\xi_{i,j} = |\alpha'_{i,j} - \alpha_{i,j}|. \tag{4.3}$$

Consequently, each variable node j can be assigned a metric ξ_j^* which is the maximum residual of $\xi_{i,j}$ among all check nodes connected to the variable node j. Denote the set of

Input: The initial LLR values μ_j and the corresponding PCM $\mathbf{H}$
Output: The estimated codeword $\hat{\mathbf{c}}$
1: Initialize $\alpha_{i,j} = \mu_j$, $\beta_{i,j} = 0$, and $\xi_{i,j} = |\mu_j|$
2: **for** $k = 1$ to n **do**
3: Find j^* according to Eq. (4.4) and (4.5)
4: **for** every check node $i \in \mathcal{C}_{j^*}$ **do**
5: Generate and propagate α_{i,j^*}
6: Reset $\xi^*_{j^*} = 0$
7: **for** every variable node $j \in \mathcal{V}_i \backslash j^*$ **do**
8: Generate and propagate $\beta_{i,j}$
9: Update $\xi_{i,j}$
10: Generate $\hat{\mathbf{c}}$ and carry out the stopping criterion
11: **if** the stopping criterion is not satisfied **then**
12: Go back to Line 2
13: **else**
14: **return** $\hat{\mathbf{c}}$

Figure 4.8: The VNWRBP algorithm.

neighboring check nodes of a variable node j by $\mathcal{C}_j$ and the set of neighboring variable nodes of a check node i by $\mathcal{V}_i$. The metric ξ^*_j can then be expressed by

$$\xi^*_j = \max_{i \in \mathcal{C}_j} \{\xi_{i,j}\}. \tag{4.4}$$

A variable node j^* will be picked up for update if

$$j^* = \arg\max_j \{\xi^*_j\}, \quad j \in \{1, 2, \ldots, n\}. \tag{4.5}$$

where n is the total number of variable nodes, i.e., the codeword length. According to Eq. (4.4) and (4.5), the variable node will be updated if it has the maximum residual $\xi_{i,j}$ among all (i, j) pairs.

Denote the initial log-likelihood ratio (LLR) for the variable node j by μ_j. The corresponding PCM is denoted by $\mathbf{H}$ with dimension $m \times n$. Denote the message (extrinsic information in LLR) from the i^{th} check node to the j^{th} variable node by $\beta_{i,j}$. The pseudocode of the VNWRBP algorithm can be illustrated in Figure 4.8.

Note that the stopping criterion is carried out after updating variable nodes n times in the VNWRBP algorithm. Recall that the in existing IDS algorithms, the stopping criterion is triggered after each variable (check) node update. Thus, for the same number of iterations, the number of executing the stopping criterion in the VNWRBP algorithm is the same as that of the FBP algorithm, but the number of executing the stopping criterion in the existing IDS algorithms is around m times (check-node based) or n times more (variable node based). Now we are ready to propose a new IDS algorithm for LDPC decoding in Chapter 4.2.2.

4.2.2 The APPRBP Algorithm

Although the IDS algorithms, for instance, the NWRBP scheduling algorithm, can further reduce the number of iterations compared to that of the LBP scheduling algorithm, their additional computational complexities cannot be neglected, which arise from the calculation of the residuals and the often checking operations on the stopping criterion. Since the check-node processing is more complex than the variable-node processing in the LDPC iterative decoding algorithms [4], the check-node based RBP scheduling algorithms usually involve higher computational complexity than the variable-node based RBP scheduling algorithms.

To address the aforementioned drawbacks inherent in the existing IDS algorithms, in this section, we devise a novel efficient variable-node based IDS algorithm called the "*a posteriori* probability RBP (APPRBP) scheduling algorithm". A threshold parameter δ is introduced in the APPRBP scheduling algorithm to determine if a variable node needs to be updated or can remain as it is. The BER performance and the number of iterations (complexity) of the APPRBP scheduling algorithm can thus be maneuvered by adjusting δ.

Denote the LLR of APP for the variable node j before and after the update by ρ_j and ${\rho'}_j$, respectively. The *APP residual* for the variable node j, denoted by ξ_j, can thus be expressed by

$$\xi_j \stackrel{\text{def}}{=} \left|{\rho'}_j - \rho_j\right|. \tag{4.6}$$

The pseudocode of our proposed APPRBP scheduling algorithm is illustrated in Figure 4.9. The threshold parameter δ is included as one input. The number of iterations executed by the APPRBP scheduling algorithm is recorded as the output N_{used}. In Line 1, a counter C is initialized to 0, which is used to record the number of variable-node updates. In Line 2, reset a queue Q to be empty. In Line 4, the APP residuals ξ_j are duplicated to η_j, $\forall j$ so that the following manipulations on η_j will not change ξ_j. In Line 5, η_j is reset to 0 for the variable nodes whose indices j are in the queue Q. By doing this, it can avoid the possibility that some variable nodes are updated more often than others. In Line 6, the variable node j^* is chosen to be updated, which has the maximum of η_j among all variable nodes. In Line 7, the threshold δ is compared with the current maximum APP residual η_{j^*}. If $\eta_{j^*} < \delta$, then the for-loop indexed by k will break as shown by Line 8. The intuition behind this threshold-controlled termination is as follows. As the index k of the for-loop increments, η_{j^*} becomes smaller and smaller due to the confinement of the queue Q in Line 5. Thus, at some point, η_{j^*} is very small so that there is no significant change even after the update of variable node j^*. Hence, no further increment in k is required and the for-loop terminates thereby.

On the other hand, if $\xi_{j^*} > \delta$, we place the index j^* into the queue Q as stated by Line 9. Lines 10 to 13 are the same as Lines 4, 5, 7, and 8 of the VNWRBP scheduling algorithm

Input: The initial LLR values μ_j, $\forall j$, the corresponding PCM $\mathbf{H}$, and the threshold parameter δ

Output: The estimated codeword $\hat{\mathbf{c}}$, and the number of iterations executed N_{used}

1: Initialize $\alpha_{i,j} = \mu_j$, $\beta_{i,j} = 0$, $\xi_j = |\mu_j|$, for all i, j, and a counter $C = 0$
2: Reset an empty queue Q
3: **for** $k = 1$ to n **do**
4: Duplicate ξ_j to η_j, for $j = 1, 2, \ldots, n$
5: Set $\eta_j = 0$, for $j \in Q$
6: Find $j^* = \arg\max_j \{\eta_j\}$, $j \in \{1, 2, \ldots, n\}$
7: **if** $\eta_{j^*} < \delta$ **then**
8: **break**
9: Append j^* into Q
10: **for** every check node $i \in \mathcal{C}_{j^*}$ **do**
11: Generate and propagate α_{i,j^*} (see [69] for details)
12: **for** every variable node $j \in \mathcal{V}_i \backslash j^*$ **do**
13: Generate and propagate $\beta_{i,j}$ (see [69] for details)
14: Update ξ_j according to Eq. (4.6)
15: Increment the counter by $C = C + k$
16: Generate $\hat{\mathbf{c}}$ using the APP-based hard-decision and then check the stopping criterion according to [4]
17: **if** the stopping criterion is not satisfied **then**
18: Go back to Line 2
19: **else**
20: **return** $\hat{\mathbf{c}}$ and $N_{\text{used}} = C/n$

Figure 4.9: The APPRBP scheduling algorithm.

described in Figure 4.8. In Line 14, ξ_j is updated according to Eq. (4.6).

After the k-for-loop finishes (it runs through $k = n$ or terminates for some $k = k_0$ at Line 8), in Line 15, the counter C will be incremented by k, which indicates the number of variable-node updates. In Line 16, the hard decision on the LLR of APP will be performed to generate the estimated codeword $\hat{\mathbf{c}}$ and then a stopping criterion is checked if $\hat{\mathbf{c}}$ is a legitimate (hopefully correct) codeword [4]. If the stopping criterion is not satisfied, the APPRBP scheduling algorithm will go back to Line 2 where the queue Q will be reset to empty again. Otherwise, the estimated codeword $\hat{\mathbf{c}}$ and the number of iterations $N_{\text{used}} = C/n$ will be returned as outputs in Line 20. Note that the number of variable-node updates C

is normalized by the total number of variable nodes n to obtain N_{used} according to the definition of *iteration* stated in Section 4.2.

The key merits of our proposed new APPRBP scheduling algorithm can be summarized as follows. The APP residual is calculated for each variable node rather than each edge in our scheme. Thus, the number of residuals is greatly reduced and the computational complexity for finding the maximum residual is reduced thereby. The calculation of the APP residual ξ_j is straightforward, which is the absolute difference between the messages from the check node i to the variable node j before and after the update, i.e., $\left|\beta_{i,j} - \beta'_{i,j}\right|$. This calculation is valid for there is no cycle of length 4 in the corresponding Tanner graph of the PCM $\mathbf{H}$, which is usually the case when an LDPC code is constructed [77].

It can be perceived that the APP residual calculation in the APPRBP scheduling algorithm involves the least computational complexity over all IDS algorithms. The threshold δ can also be adjusted to further reduce the number of iterations and thus speed up the convergence. Note that when the threshold δ is set to 0, the condition stated by Line 7 in the APPRBP scheduling algorithm will never be true. An appropriate value of δ which leads to remarkable reduction in the number of iterations will be pursued via computer simulations in the next section.

4.2.3 Simulation

Computer simulations are carried out to demonstrate the effectiveness of our proposed APPRBP algorithm described in Chapter 4.2.2. The performances of the conventional FBP algorithm descried in Chapter 1.2, the LBP algorithm [65], and the NWRBP algorithm [69] are also investigated for comparison. Since the VNWRBP algorithm (variable-node based)

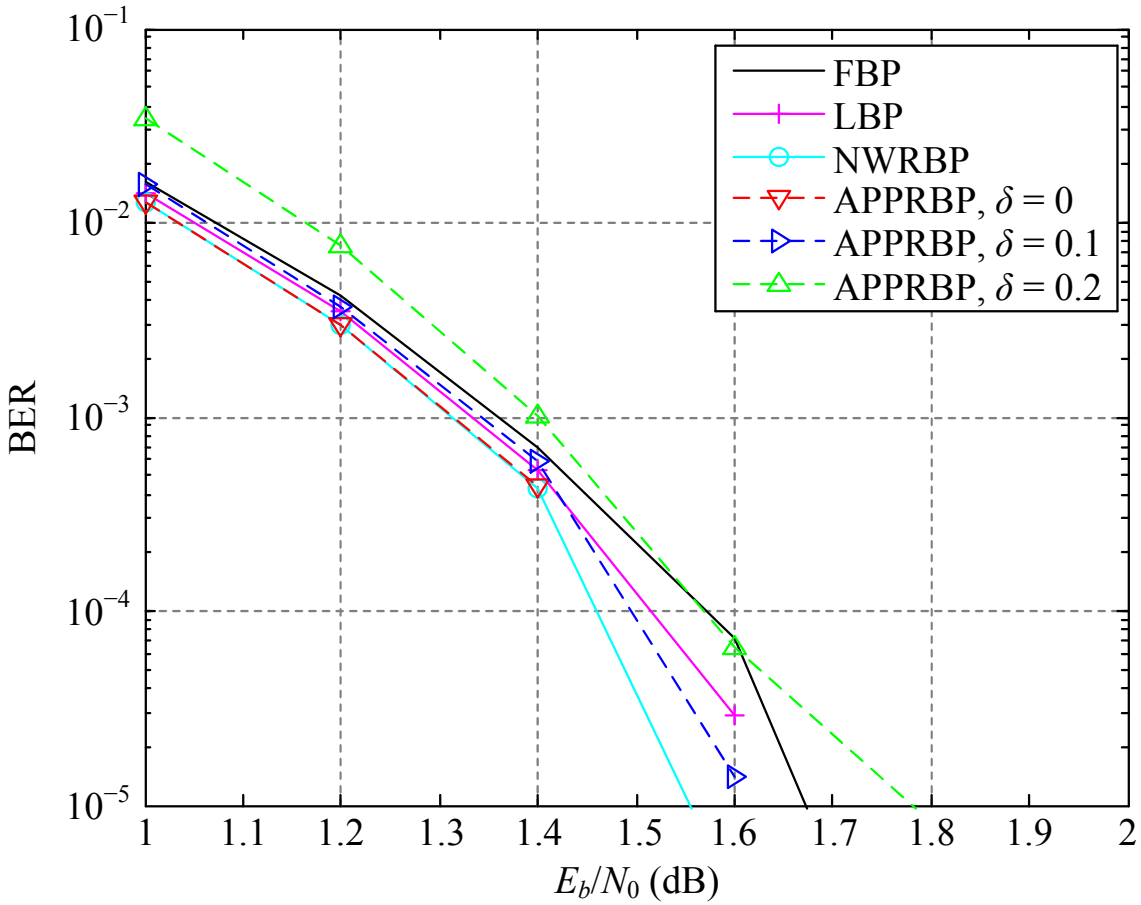

Figure 4.10: The BER performances of the APPRBP algorithm using different threshold δ. The BER performances of the FBP algorithm, the LBP algorithm, and the NWRBP algorithm are also depicted for comparison.

is the counterpart of the NWRBP algorithm (check-node based), these two algorithms have almost the same performance according to our simulations. Thus, for clarity of illustration, the performance of the VNWRBP algorithm is not shown in the simulation results. The LDPC code defined in the IEEE 802.11 standard [20] with codeword length 1944 and code rate 1/2 is taken for the simulations. The maximum iteration number N_{iter} is set to 50 for all examined scheduling algorithms.

Figure 4.10 depicts the BER performances of the APPRBP algorithm using different threshold δ. When $\delta = 0$ (the condition in Line 6 in the APPRBP algorithm described in

Figure 4.9 will never be satisfied and Line 7 will never be carried out), the BER performance of the APPRBP algorithm is almost the same as that of the NWRBP algorithm. When $\delta = 0.1$, the BER performance of the APPRBP algorithm is very close to that of the LBP algorithm, When $\delta = 0.2$, the BER performance of the APPRBP algorithm is a little worse than that of the FBP algorithm. It is clear that the bigger the threshold δ, the worse the BER performance of the APPRBP algorithm. The reason is that when δ is larger, the condition in Line 6 will be more likely satisfied. Therefore, the APPRBP may be terminated before converging to the correct codeword for large δ. However, it is worth mentioning that the BER degradation is not too much (within 0.05 dB) when δ is increased from 0 to 0.1.

Figure 4.11 illustrates the average iteration numbers (AIN) of the APPRBP algorithm using different threshold δ. When $\delta = 0$, the AIN curve of the APPRBP algorithm is between that of the LBP algorithm and the NWRBP algorithm. When $\delta = 0.1$, the AIN curve of the APPRBP algorithm is below that of the NWRBP algorithm in the low E_b/N_0 region ($E_b/N_0 < 1.4$ dB). When $\delta = 0.2$, the AIN of the APPRBP is further reduced for $E_b/N_0 < 0.4$ dB. On the other hand, when $E_b/N_0 > 1.4$ dB, the AINs of the APPRBP using $\delta = 0.1$ and $\delta = 0.2$ are almost the same as that of the NWRBP algorithm. When $E_b/N_0 = 1$ dB, the APPRBP using $\delta = 0.1$ reduces the AIN of the NWRBP by almost half. This remarkable AIN reduction phenomenon is not observed in the existing advanced IDS algorithms, considering that the BER performance of the APPRBP is almost the same as that of the LBP algorithm and is very close to that of the NWRBP.

According to Figure 4.10 and 4.11, there is apparently more freedom in the APPRBP algorithm to leverage the BER performance and computational complexity (indicated by the AIN) trade-off by adjusting threshold δ.

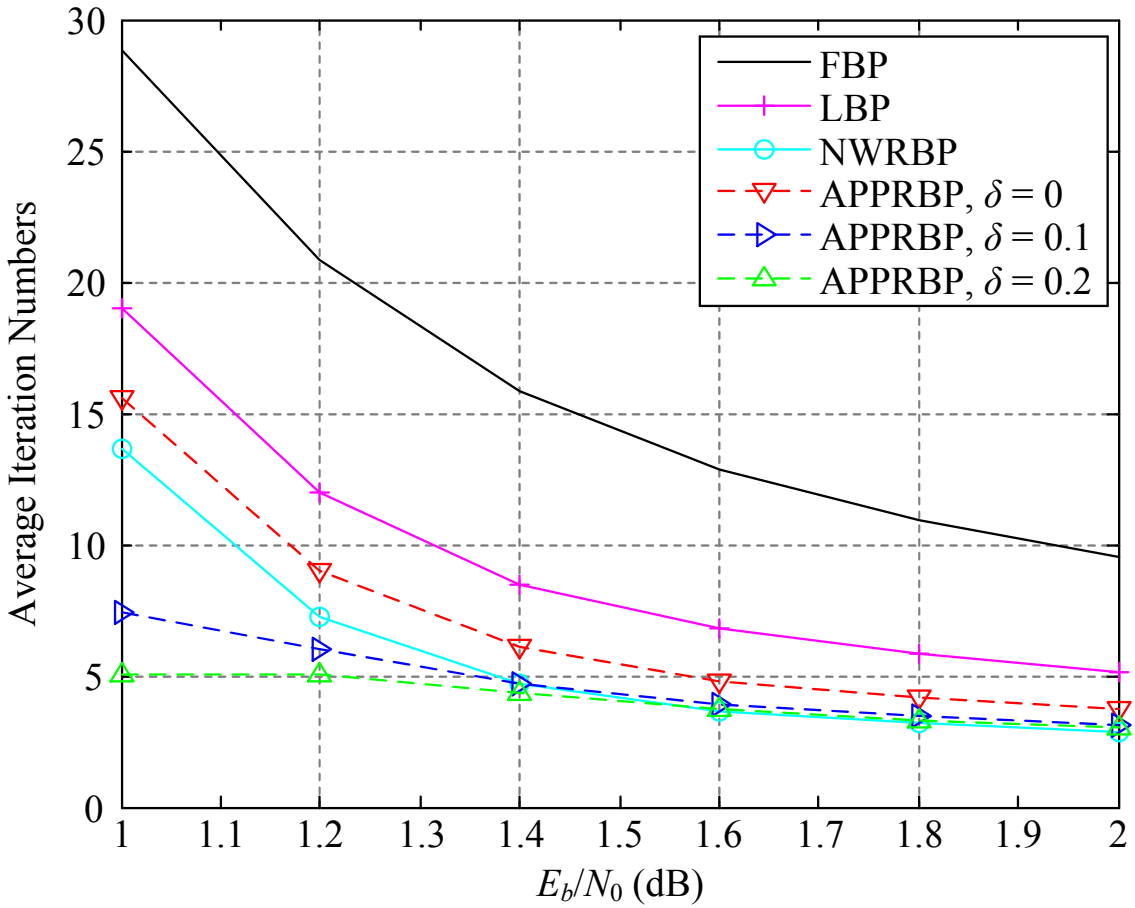

Figure 4.11: The AIN of the APPRBP algorithm using different threshold δ. The AIN performances of the FBP algorithm, the LBP algorithm, and the NWRBP algorithm are also depicted for comparison.

4.2.4 Summary

In this section, a new efficient informed dynamic scheduling (IDS) algorithm is proposed for LDPC (low-density parity-check) decoders, called *a posteriori* probability residual belief propagation (APPRBP) scheduling algorithm. A threshold parameter is incorporated in our proposed algorithm to maneuver the performance-complexity trade-off. The computational complexity burden involved in the residual calculation of the proposed APPRBP scheduling algorithm is reduced compared to those in other existing IDS algorithms. The remarkable complexity reduction in terms of the average iteration numbers shown by the simulation

results demonstrates that our proposed APPRBP scheduling algorithm is capable of providing excellent flexibility for maneuvering the performance-complexity trade-off compared to existing scheduling schemes for LDPC decoding.

5. FAST ITERATIVE DECODING THRESHOLD ESTIMATION

Low-density parity-check convolutional codes (LDPC-CCs), also called *spatially-coupled* codes, have drawn a lot of research interest in recent years [13, 78]. One common way to construct LDPC-CCs is based on *protographs* [79, 80]. LDPC-CCs can be of infinite length [13] or be *terminated* [14]. For terminated LDPC-CCs, a significant observation is the *threshold saturation* phenomenon that makes the iterative decoding threshold of LDPC-CCs asymptotically approach the maximum *a posteriori* (MAP) decoding threshold [14, 80]. Recently proposed *window decoding* techniques [81, 82] also made LDPC-CCs favorable in practical applications subject to stringent memory and latency requirements.

Given a channel, the *iterative decoding threshold* (IDT) is one of the key parameters associated with an LDPC code ensemble and related to the belief-propagation (BP) algorithms [1]. For instance, let us consider the additive white Gaussian noise (AWGN) channel where the link quality can be expressed by the signal-energy-per-bit-to-noise-power-spectral-density ratio, denoted by E_b/N_0. For some LDPC-CC ensemble, if E_b/N_0 is greater than the corresponding IDT, then *reliable communications* can be achieved using BP decoding algorithms as the codeword length approaches infinity. On the other hand, if E_b/N_0 is lower than the IDT, no matter how many iterations the BP decoding algorithms take, the bit error probability is bounded above zero (the error probability can never be zero). Thus, how good an LDPC code ensemble is can be evaluated by examining the gap between its IDT and the *Shannon limit*, and there exists ceaseless effort for researchers to design *good* LDPC codes with IDTs *as close to the Shannon limit as possible*.

One of the IDT calculation techniques is *density evolution* (DE) (see [83]), which tracks the evolution of the probability densities during the BP decoding process. For a given channel, the IDT is obtained when the bit error probability approaches zero. To avoid the potential complexity of calculating the probability densities in DE, an alternative to compute the IDT is utilizing the *extrinsic information transfer (EXIT) chart* (see [8, 84]), by which one can investigate the *mutual information* (MI) between the extrinsic information at the variable (check) nodes and the transmitted bits/symbols. The IDT is found from the EXIT chart when the MI curve for check nodes partially overlaps with the MI curve for variable nodes. Recently, a protograph-based EXIT (PEXIT) analysis was proposed for protograph-based LDPC codes [85], which calculated the MI for each *edge* (each non-zero entry of the protograph base matrix), and could be capable of dealing with protograph base matrices of multi-edge type, punctured type, etc.

While the IDT calculation for conventional LDPC block code ensembles can be carried out given the *degree distribution pair* [84], averaging over degree distributions cannot be directly applied to LDPC-CCs due to the *spatially-coupled structure* [14]. Instead, to determine the IDT for given LDPC-CCs using either DE or EXIT analysis, the calculation has to be performed for each individual variable and check node. As a result, it is inevitable to go through the BP iteration process for determining the IDTs for LDPC-CCs. For LDPC-CCs with large termination length L, given the AWGN channel, it can be time-consuming to estimate the IDT, denoted by η, due to the exorbitant number of decoding iterations and high storage memory requirement.

For the rest of this chapter, the IDT actually means "the IDT for the AWGN channel". For simplicity, the AWGN channel is omitted in the IDT expression. To mitigate

the computational complexity involved in the calculation of the IDT for protograph-based LDPC-CCs [1], in this chapter, we propose a novel *PEXIT-fast algorithm*. Based on the fact that the *termination effect* propagates from both ends to the center of the *Tanner graph* for LDPC-CCs, we propose to monitor the MI of the *a posteriori* probability for the first variable node. If this MI value can approach one, the investigated (E_b/N_0) value is then an upper bound of the IDT; on the other hand, if this MI value converges to a value less than one, the evaluated (E_b/N_0) value is instead a lower bound of the IDT. It is thus unnecessary to get through the whole evolution process and the corresponding computational complexity can be greatly reduced. We also develop the asymptotic analysis when the termination length L of an LDPC-CC goes to infinity. This leads to an efficient way to determine the IDTs of LDPC-CCs for an arbitrarily large L, which includes the case of $L \to \infty$.

5.1 Preliminaries

In this section, we first provide a brief introduction on LDPC-CCs constructed by protographs and then establish the basics of the PEXIT analysis.

5.1.1 LDPC Convolutional Codes (LDPC-CCs)

A protograph-based LDPC-CC is constructed from a *protograph*. For simplicity of illustration, in this chapter, we assume that there are no multiple edges between any two nodes in a protograph. Besides, we only consider *terminated LDPC-CCs* with termination length L; however, the asymptotic analysis of L will also be addressed. A terminated LDPC-CC

can be derived upon a *base matrix*, denoted by $\mathbf{B}_{[0,L-1]}$, which can be expressed as

$$\mathbf{B}_{[0,\,L-1]} \stackrel{\text{def}}{=} \overbrace{\begin{bmatrix} \mathbf{B}_0 & & & & \\ \mathbf{B}_1 & \mathbf{B}_0 & & & \\ \vdots & \mathbf{B}_1 & \ddots & & \\ \mathbf{B}_{m_s} & \vdots & \ddots & \mathbf{B}_0 & \\ & \mathbf{B}_{m_s} & \ddots & \mathbf{B}_1 & \\ & & \ddots & \vdots & \\ & & & \mathbf{B}_{m_s} & \end{bmatrix}}^{L}, \tag{5.1}$$

where m_s is called the *syndrome former memory*, and the sub-matrices $\mathbf{B}_\mu$, $\mu = 0, 1, \ldots, m_s$, have dimension $b \times c, b \leq c$, in which all entries are equal to one [81].

We call $\mathbf{B}_{[0,L-1]}$ a (J, K, L) *base matrix* if $\mathbf{B}_{[0,L-1]}$ has exactly J nonzero entries in each column and K nonzero entries in each row (except that the top and bottom rows of $\mathbf{B}_{[0,L-1]}$ have less weights due to the termination effect of the corresponding protograph). A (J, K, L, M) LDPC-CC can then be constructed from $\mathbf{B}_{[0,L-1]}$ by changing every nonzero entry in $\mathbf{B}_{[0,L-1]}$ into an arbitrary permutation matrix with dimension M (M is also called the *graph lifting factor*). All possible permutation combinations consist of the (J, K, L, M) *LDPC-CC ensemble*. Generally speaking, the code rate of the protograph-based LDPC-CC can be determined by its (J, K, L) base matrix and has nothing to do with M.

It is known that the IDT of an LDPC-CC ensemble can be improved (get closer to the Shannon limit) by increasing the termination length L. Meanwhile, the rate loss incurred by termination is mitigated to 0 when L goes to infinity. When determining the IDT of an protograph-based LDPC-CC, it is assumed that the graph lifting factor M approaches

infinity. How to use the PEXIT analysis to determine the IDTs for LDPC-CCs with finite L is discussed as follows.

5.1.2 PEXIT Analysis

In the EXIT analysis for the AWGN channel, by modeling the extrinsic information propagated over each edge connecting a check node and a variable node as a Gaussian variable with mean $\sigma^2/2$ and variance σ^2, one can express the MI between the extrinsic information and the transmitted bits as (see [8])

$$\mathcal{J}(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{\exp\left\{ -\frac{(y-\sigma^2/2)^2}{2\sigma^2} \right\}}{\sqrt{2\pi}\sigma} \log_2\left(1 + e^{-y}\right)\, dy. \tag{5.2}$$

The curve-fitting functions are given in [84] to calculate $\mathcal{J}(\sigma)$ and its inverse $\mathcal{J}^{-1}(\sigma)$ efficiently.

The EXIT analysis is usually implemented by averaging the degree distributions from the edge perspective [84]. Thus, in each iteration of the EXIT analysis, there is only one MI value between the extrinsic messages from variable (check) nodes to check (variable) nodes and the transmitted random bits. However, to deal with the protograph-based LDPC codes which involve base matrices of multi-edge type, punctured type, etc., the EXIT analysis is extended to the PEXIT analysis which is carried out for each edge [85]. Apparently, the downside of the PEXIT analysis is that it consumes more storage memory and leads to higher computational complexity since the MI for each edge has to be calculated and stored during the iterative BP process.

Nevertheless, due to the spatially-coupled structure imposed by an LDPC-CC, its iterative decoding threshold can not be evaluated anymore by taking average of the variable/check-

node degree distributions from the edge perspective. Instead, the EXIT analysis has to be carried out for each edge exactly as the PEXIT analysis does. A protograph-based LDPC-CC's IDT can be determined by running the PEXIT algorithm on its (J, K, L) protograph base matrix. Here, to simplify the discussion, it is assumed that the (J, K, L) base matrix has the form given by Eq. (5.1).

In the l^{th} iteration of the PEXIT algorithm, denote the mutual information between the extrinsic information from the j^{th} variable node to the i^{th} check node and the coded bit c_j by $x_{i,j}^{(l)}$, the mutual information between the extrinsic information from the i^{th} check node to the j^{th} variable node and the coded bit c_j by $y_{i,j}^{(l)}$, and the mutual information between the LLR (log-likelihood ratio) of APP (*a posteriori* probability) and c_j by $z_j^{(l)}$. The details of the PEXIT algorithm for determining the IDT of an protograph-based LDPC-CC, given its (J, K, L) base matrix, can be described as follows.

Input: E_b/N_0, a (J, K, L) base matrix with dimension $m \times n$.

Step 1: Initialize $x_{i,j}^{(l)} = 0$, $y_{i,j}^{(l)} = 0$ for all edges (i, j), and $z_j^{(l)} = 0$ for all j, $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$. Set the maximum iteration number N_{iter}, and the iteration number is reset to $l = 0$. Calculate the initial variance of the channel output to the decoder, denoted by σ_{ch}^2, as

$$\sigma_{\text{ch}}^2 = 8R\left(\frac{E_b}{N_0}\right), \tag{5.3}$$

where $R \stackrel{\text{def}}{=} 1 - m/n$ is the code rate of the LDPC-CC having the (J, K, L) base matrix.

Step 2: Increment l by 1. Update $x_{i,j}^{(l)}$ for each edge at variable nodes:

$$x_{i,j}^{(l)} = \mathcal{J}\left(\sqrt{\sum_{k \in \mathcal{C}_j \backslash i} \left[\mathcal{J}^{-1}\left(y_{k,j}^{(l-1)}\right)\right]^2 + \sigma_{\text{ch}}^2}\right), \tag{5.4}$$

where $\mathcal{C}_j$ is the set of the neighboring check nodes of the j^{th} variable node, and $\mathcal{C}_j \backslash i$ excludes the i^{th} check node from $\mathcal{C}_j$.

Step 3: Update $y_{i,j}^{(l)}$ for each edge at check nodes:

$$y_{i,j}^{(l)} = 1 - \mathcal{J}\left(\sqrt{\sum_{k \in \mathcal{V}_i \backslash j} \left[\mathcal{J}^{-1}\left(1 - x_{i,k}^{(l)}\right)\right]^2}\right), \tag{5.5}$$

where $\mathcal{V}_i$ is the set of the neighboring variable nodes of the i^{th} check node, and $\mathcal{V}_i \backslash j$ excludes the j^{th} variable node from $\mathcal{V}_i$.

Step 4: Update $z_j^{(l)}$ for all variable nodes:

$$z_j^{(l)} = \mathcal{J}\left(\sqrt{\sum_{k \in \mathcal{C}_j} \left[\mathcal{J}^{-1}\left(y_{k,j}^{(l)}\right)\right]^2 + \sigma_{\text{ch}}^2}\right). \tag{5.6}$$

Step 5: If $l < N_{\text{iter}}$ and there is any j such that $z_j^{(l)} < 1$, go back to Step 2. If $z_j^{(l)} = 1, \forall j$, terminate the PEXIT algorithm and declare "The input E_b/N_0 is larger than the IDT. Thus, an upper bound of IDT, $\eta' = E_b/N_0$, is found." Otherwise, all N_{iter} iterations are used up but there is some j such that $z_j^{(l)} < 1$; this indicates "The input E_b/N_0 is lower than the IDT and a lower bound of IDT, $\eta'' = E_b/N_0$, is found instead."

Run the PEXIT algorithm above in a search routine for a sequence of E_b/N_0 values. Then one can find a closest adjacent pair of E_b/N_0 values (η'', η') in between the IDT η lies, i.e., $\eta \in [\eta'', \eta']$. For a given accuracy requirement θ, if the difference between η'' and η' is smaller than θ, i.e., $\eta' - \eta'' < \theta$, we assign η' as the estimated IDT, denoted by $\hat{\eta}$, i.e., $\hat{\eta} = \eta'$; otherwise, the search routine continues. Therefore, the smaller the value of θ, the more accurate the IDT estimate $\hat{\eta}$ one can obtain.

When the termination length L is very large, it may require an enormous number of iterations for the PEXIT algorithm to find the IDT of the given LDPC-CC. This is because

the maximum iteration number of the PEXIT algorithm, N_{iter}, has to be set very large for large L values to guarantee that all $z_j^{(l)}$ can converge to 1 if the input E_b/N_0 is greater than the IDT, and on the other hand an E_b/N_0 value less than the IDT during the search routine will consume all N_{iter} iterations. The memory requirement of the PEXIT algorithm may also be very demanding for very large L values. On the other hand, as L goes to infinity, the PEXIT algorithm cannot be employed anymore. However, according to the threshold saturation phenomenon, the terminated LDPC-CC specified by the (J, K, L) base matrix has the best possible IDT when $L \to \infty$, where the values of J and K are fixed. Consequently, it would be very favorable to design a new algorithm to estimate this "best possible IDT" for the theoretical study of the best scenario when the termination length L approaches infinity.

5.2 Monotonicity Analysis and the PEXIT-fast Algorithm

In this section, we propose a new "PEXIT-fast algorithm" which can address all aforementioned issues challenging the existing PEXIT algorithm as discussed in Section 5.1.2. According to [2], we first establish several crucial *monotonic properties* of the functions involved in the PEXIT algorithm when an LDPC-CC is evaluated. Based on these monotonic properties, we derive two corollaries for efficiently facilitate a pair of upper and lower bounds of the IDT. According to these two corollaries, our *PEXIT-fast algorithm* can terminate the algorithm in a very early stage by only monitoring the MI of APP for the first variable node rather than all variable nodes. That is, the number of required iterations (the computational complexity) can be significantly reduced.

When the input E_b/N_0 happens to be the IDT η, Eq. (5.3) facilitates the corresponding variance threshold, denoted by $\bar{\sigma}^2_{\text{ch}}$, that is,

$$\bar{\sigma}^2_{\text{ch}} = 8\,R\,\eta. \tag{5.7}$$

Note that both R and η vary with respect to L. Thus, according to Eq. (5.7), we can write

$$\bar{\sigma}^2_{\text{ch}}(L) = 8\,R(L)\,\eta(L). \tag{5.8}$$

Obviously, from Eq. (5.8), one can find that the IDT $\eta(L)$ can be easily calculated when $\bar{\sigma}^2_{\text{ch}}(L)$ is available. Therefore, we would like to establish the asymptotic analysis of $\bar{\sigma}^2_{\text{ch}}(L)$ as L goes to infinity and investigate the convergence of $\bar{\sigma}^2_{\text{ch}}(L)$ as $L \to \infty$. Assume that $\bar{\sigma}^2_{\text{ch}}(L)$ converges as $L > L_0$ for a moderate value L_0 (the simulation results presented later in this section demonstrate that $L_0 = 100$ is a good choice). According to Eq. (5.8), the estimate of the IDT $\eta(L)$ of an LDPC-CC with $L > L_0$ can be obtained as

$$\hat{\eta}(L) \stackrel{\text{def}}{=} \frac{R(L_0)\,\hat{\eta}(L_0)}{R(L)}, \quad \text{for all } \; L > L_0. \tag{5.9}$$

Note that the LDPC-CC's code rate $R(L)$ can be easily computed for an arbitrary L given the (J, K, L) base matrix defined by Eq. (5.1). Thus, our new PEXIT-fast algorithm will first determine L_0 (=100), then calculate $\bar{\sigma}^2_{\text{ch}}(L_0)$, and compute the IDT estimates according to Eq. (5.9) for all $L > L_0$ quickly.

To facilitate the detailed analysis of the PEXIT algorithm for an LDPC-CC, we rewrite Eq. (5.4) and Eq. (5.5) in vector forms. Denote the number of edge of the (J, K, L) base matrix by N. Then in the l^{th} iteration, there are two sets of MI vectors with dimension N: one consists of the MI values from the variable nodes to the check nodes, denoted by $\mathbf{x}^{(l)}$, and the other consists of the MI values from the check nodes to the variable nodes, denoted

by $\mathbf{y}^{(l)}$. Thus, Eq. (5.4) and Eq. (5.5) can be expressed abstractly as

$$\mathbf{x}^{(l)} \stackrel{\text{def}}{=} \mathcal{F}\big(\mathbf{y}^{(l-1)},\ \sigma_{\text{ch}}^2\big), \tag{5.10}$$

where $\mathcal{F} : [0,1]^N \times [0,\infty) \to [0,1]^N$ represents the evaluation (functional mapping) of Eq. (5.4) and

$$\mathbf{y}^{(l)} \stackrel{\text{def}}{=} \mathcal{G}\big(\mathbf{x}^{(l)}\big), \tag{5.11}$$

where $\mathcal{G} : [0,1]^N \to [0,1]^N$ represents the evaluation (functional mapping) of Eq. (5.5). Denote the N-dimensional all-zero and all-one vectors by $\mathbf{0}$ and $\mathbf{1}$, respectively. As a matter of fact, we have (i) $\mathcal{F}(\mathbf{0},0) = \mathbf{0}$, $\mathcal{F}(\mathbf{1},\sigma_{\text{ch}}^2) = \mathbf{1}$, and $\mathcal{F}(\mathbf{x},\infty) = \mathbf{1}$; (ii) $\mathcal{G}(\mathbf{0}) = \mathbf{0}$ and $\mathcal{G}(\mathbf{1}) = \mathbf{1}$.

Substitute Eq. (5.11) into Eq. (5.10). For notational simplification, we further let $\epsilon = \sigma_{\text{ch}}^2$. Then, we have the following recursion:

$$\mathbf{x}^{(l+1)} = \mathcal{F}\Big(\mathcal{G}\big(\mathbf{x}^{(l)}\big), \epsilon\Big), \quad l = 0, 1, \ldots. \tag{5.12}$$

where l specifies the iteration number during the execution of the PEXIT algorithm.

The IDT of an LDPC-CC depends on the monotonicity properties of Eq. (5.12). Before establishing the analysis, we present the following definitions.

Definition 1. *Given two arbitrary N-dimensional real-valued vectors $\mathbf{v} = (v_1, v_2, \ldots, v_k, \ldots, v_N)$ and $\mathbf{w} = (w_1, w_2, \ldots, w_k, \ldots, w_N)$, we say $\mathbf{v} \leqq \mathbf{w}$ if and only if $v_k \leq w_k$, $\forall k$.*

Definition 2. *A real-valued multivariate function $g(v_1, v_2, \ldots, v_N)$ can be expressed as a function with an N-dimensional variable vector $\mathbf{v} \in \mathcal{R}^N$ such that $g(v_1, v_2, \ldots, v_N) = g(\mathbf{v})$. For any two vectors $\mathbf{v}' \in \mathcal{R}^N$ and $\mathbf{v}'' \in \mathcal{R}^N$, if $\mathbf{v}' \leqq \mathbf{v}''$, we have $g(\mathbf{v}') \leq g(\mathbf{v}'')$. Thus, we say $g(v_1, v_2, \ldots, v_N)$ is a "monotonically increasing" function.*

Definition 3. *A collection of P real-valued multivariate functions $g_1(\mathbf{v}), g_2(\mathbf{v}), \ldots, g_P(\mathbf{v})$ can be expressed as $G(\mathbf{v}) \stackrel{def}{=} [g_1(\mathbf{v}), g_2(\mathbf{v}), \ldots, g_P(\mathbf{v})]^T$ where $\mathbf{v} \in \mathcal{R}^N$. If all element functions $g_p(\mathbf{v})$, $p = 1, 2, \ldots, P$, are monotonically increasing, we say $G(\mathbf{v})$ is monotonically increasing. In other words, given a monotonically increasing functional collection $G(\mathbf{v})$, we have $G(\mathbf{v}') \leqq G(\mathbf{v}'')$ for all $\mathbf{v}' \in \mathcal{R}^N$, $\mathbf{v}'' \in \mathcal{R}^N$, and $\mathbf{v}' \leqq \mathbf{v}''$.*

Based on Definitions 1-3, we can investigate the *monotonicity* of Eq. (5.12) with respect to $\mathbf{x}$ and ϵ. Let's just check one iteration (for any particular iteration l) with the following lemma.

Lemma 1. *The function $\mathcal{F}\big(\mathcal{G}(\mathbf{x}), \epsilon\big)$ is monotonically increasing with respect to both $\mathbf{x}$ and ϵ.*

Proof. It is easy to justify that the functions $\mathcal{J}(\cdot, \cdot)$ and $\mathcal{J}^{-1}(\cdot)$ in Eqs.(5.4)-(5.5) are both monotonically increasing and $\mathcal{J}(\cdot, \cdot)$ is bounded within $[0, 1]$. Therefore, $\mathcal{F}(\cdot, \cdot)$ and $\mathcal{G}(\cdot)$ are both monotonically increasing with respect to their arguments. Specifically, for $\mathbf{0} \leqq \mathbf{x}_1 \leqq \mathbf{x}_2 \leqq \mathbf{1}$, $\mathcal{F}\big(\mathcal{G}(\mathbf{x}_1), \epsilon\big) \leqq \mathcal{F}\big(\mathcal{G}(\mathbf{x}_2), \epsilon\big)$; for $0 \leq \epsilon_1 \leq \epsilon_2$, $\mathcal{F}\big(\mathcal{G}(\mathbf{x}), \epsilon_1\big) \leqq \mathcal{F}\big(\mathcal{G}(\mathbf{x}), \epsilon_2\big)$. □

Next, we will provide the following lemma for the monotonicity of $\mathbf{x}^{(l)}$ in Eq. (5.12) over the iteration l.

Lemma 2. *Let $\mathbf{0} \leqq \mathbf{x}^{(0)} \leqq \mathbf{1}$. Then $\mathbf{x}^{(l)}$ is a monotonic sequence, for $l = 0, 1, \ldots$, and will converge to a fixed point as l goes to infinity.*

Proof. If $\mathbf{x} = \mathbf{0}$ and $\epsilon = 0$, then $\mathcal{F}\big(\mathcal{G}(\mathbf{0}), 0\big) = \mathbf{0}$. Thus, for $\epsilon = 0$, $\mathbf{x} = \mathbf{0}$ is a fixed point. If $\mathbf{x} = \mathbf{1}$ and ϵ can be arbitrary, $\mathcal{F}\big(\mathcal{G}(\mathbf{1}), \epsilon\big) = \mathbf{1}$. Thus, $\mathbf{x} = \mathbf{1}$ is a fixed point in this regard. If $\epsilon = \infty$, then $\mathbf{x}^{(l+1)} = \mathcal{F}\big(\mathcal{G}(\mathbf{x}^{(l)}), \infty\big) = \mathbf{1}$, for $l = 0, 1, 2, \ldots$. Thus, after one iteration, $\mathbf{x} = \mathbf{1}$ becomes the corresponding fixed point.

If $\mathbf{0} \leqq \mathbf{x}$, for some iteration l_0, we have $\mathbf{x}^{(l_0-1)} \leqq \mathbf{x}^{(l_0)}$. According to the monotonicity given by Lemma 1, $\mathbf{x}^{(l_0)} = \mathcal{F}\big(\mathcal{G}(\mathbf{x}^{(l_0-1)}), \epsilon\big) \leqq \mathcal{F}\big(\mathcal{G}(\mathbf{x}^{(l_0)}), \epsilon\big) = \mathbf{x}^{(l_0+1)}$. Since $\mathbf{x}^{(l)} \in [0, 1]$, $\mathbf{x}^{(l)}$ will converge to some fixed point, denoted by $\mathbf{x}^{\infty}$, when the iteration number l approaches infinity. □

In the protograph represented by the (J, K, L) base matrix, the termination length L can be seen as L *positions* [80]. According to Eq. (5.1), there are c variable nodes for each position. It can be seen that the variable nodes at the same position are *isomorphic* to each other since relabeling them leads to the identical protograph. From the base matrix perspective, relabeling the variable nodes at the same position is equivalent to permuting the corresponding columns of the base matrix, which results in the same base matrix. Consequently, the variable nodes at the same position have the same MI of APP. For illustrational simplicity, the positions of the variable (check) nodes are indexed in the following way according to the termination length L. If L is even, the positions of variable nodes are indexed as $t \in \{-L/2, -L/2+1, \ldots, -1, 1, \ldots, L/2\}$. If L is odd, the positions of variable nodes are indexed as $t \in \{-(L-1)/2, -(L-1)/2+1, \ldots, -1, 0, 1, \ldots, (L-1)/2\}$ instead.

Proposition 1. *Assume that at iteration 0,* $\mathbf{x}^{(0)} = \mathbf{0}$ *is initialized. For two variable nodes* j_1 *and* j_2 *where* j_1 *is at the position* t_1 *and* j_2 *is at the position* t_2*, if* $|t_1| \geq |t_2|$*, then* $z_{j_1}^{(l)} \geq z_{j_2}^{(l)}$ *for the iteration number* $l = 1, 2, \ldots$.

Proof. At iteration 1, $\mathbf{x}^{(1)} = \mathcal{J}(\sigma_{\text{ch}})$ since $\mathbf{x}^{(0)} = \mathbf{0}$ and $\mathcal{G}(\mathbf{x}) = \mathbf{0}$. Given two variable nodes j_1 and j_2, for all check nodes $i \in \mathcal{C}_{j_1}$, place all the corresponding check node degree d_i in a vector $\mathbf{d}_{j_1}$ of a nondecreasing order; for all check nodes $i' \in \mathcal{C}_{j_2}$, place all the corresponding check node degree $d_{i'}$ in a vector $\mathbf{d}_{j_2}$ of a nondecreasing order. Since all variable nodes for

an LDPC-CC have the same degree distribution J, $\mathbf{d}_{j_1}$ and $\mathbf{d}_{j_2}$ have the same dimension J. Then, if $|t_1| \geq |t_2|$, we have $\mathbf{d}_{j_1} \leqq \mathbf{d}_{j_2}$ because the check nodes connected to j_1 are closer to the boundary of the corresponding protograph and the check node degree is non-increasing from the middle positions to the boundary positions of the protograph.

Let $i_k, k = 1, 2, \ldots, J$ be the k^{th} check node which is connected to the variable node j_1 and it has the degree given by the k^{th} element of $\mathbf{d}_{j_1}$. Let i'_k be the k^{th} check node which is connected to the variable node j_2 and it has the degree given by the k^{th} element of $\mathbf{d}_{j_2}$. Then, according to Eq. (5.5), $y^{(l)}_{i_k,j_1} \geq y^{(l)}_{i'_k,j_2}$ for $i_k \in \mathcal{C}_{j_1}, i'_k \in \mathcal{C}_{j_2}$. Thus, according to Eq. (5.6), we have

$$\begin{aligned} z^{(l)}_{j_1} &= \mathcal{J}\left(\sqrt{\sum_{k=1}^{J}\left[\mathcal{J}^{-1}\left(y^{(l)}_{i_k,j_1}\right)\right]^2 + \sigma^2_{\text{ch}}}\right) \\ &\geq \mathcal{J}\left(\sqrt{\sum_{k=1}^{J}\left[\mathcal{J}^{-1}\left(y^{(l)}_{i'_k,j_2}\right)\right]^2 + \sigma^2_{\text{ch}}}\right) \\ &= z^{(l)}_{j_2}. \end{aligned}$$

□

The phenomenon manifested by Proposition 1 that the MI of APP is larger for the variable nodes at the boundary positions was also observed in [1]. However, the proof was not provided in [1]. It is well known that the termination of an LDPC-CC makes the variable nodes at the boundary positions of the protograph "*stronger*" (see [1]) and the termination effect is *propagated* from both boundary positions to the middle positions of the protograph for a terminated LDPC-CC.

Remark: According to Proposition 1, at the iteration l, $z^{(l)}_1 \geq z^{(l)}_j, \forall j$. Thus, when the currently investigated E_b/N_0 is appropriate ($E_b/N_0 > \eta$), it is the MI of the first variable $z^{(l)}_1$

that achieves 1 ahead of all other variable nodes during the iterative process of the PEXIT analysis. Thus, we can have the following corollary which can be used for identifying the lower bound of the IDT in the PEXIT analysis without exhausting all N_{iter} iterations.

Corollary 1. *If the MI of the LLR of APP cannot approach 1 for the first variable node, then the MI of the LLR of APP cannot reach 1 for any other variable node no matter how many iterations are undertaken.*

Revisit Eqs. (5.8) and (5.9). It is crucial to analyze the asymptotic behavior of the variance threshold $\bar{\sigma}^2_{\text{ch}}(L)$ with respect to L in order to calculate $\hat{\eta}(L)$ for a large termination length L. The pertinent analysis is facilitated by the following lemma.

Lemma 3 (Monotonicity of $\bar{\sigma}^2_{\text{ch}}(L)$)**.** *Given an LDPC-CC having the (J, K, L) base matrix, the threshold $\bar{\sigma}^2_{\text{ch}}(L)$ is monotonically increasing with respect to L.*

Proof. We prove this lemma by contradiction. Suppose that $L_1 = L_2 + 1$ and $\bar{\sigma}^2_{\text{ch}}(L_1) < \bar{\sigma}^2_{\text{ch}}(L_2)$. Since $\bar{\sigma}^2_{\text{ch}}(L_1)$ can make $z_j^{(l)} = 1, \forall j$ when l is sufficiently large, according to Proposition 1, at some iteration l_0, $z_j^{(l)}$ first become 1 for all j belonging to the very left boundary position. Because of the isomorphism of the variable nodes at the same position, the MI values of APP for all variable nodes at the very left boundary position also attain 1. Note that once the variable node j's MI of APP attains 1 (the corresponding coded symbols can be correctly decoded), this variable node has no further influence on the PEXIT recursion. Thus, the variable nodes at the very left boundary position can be deleted in the PEXIT algorithm. The remaining protograph becomes the protograph constituted by the (J, K, L_2) base matrix with $L_1 = L_2 + 1$. Since $\bar{\sigma}^2_{\text{ch}}(L_1)$ can make all $z_j^{(l)}$ achieve 1, it is greater than or equal to

the IDT, i.e., $\bar{\sigma}^2_{\mathrm{ch}}(L_2)$, of the LDPC-CC having the (J, K, L_2) base matrix. This contradicts with the assumption. Finally, by induction, for $L' \geq L''$, we have $\bar{\sigma}^2_{\mathrm{ch}}(L') \geq \bar{\sigma}^2_{\mathrm{ch}}(L'')$. □

Lemma 3 leads to the following corollary which can be used to identify the upper bound of the IDT in the PEXIT analysis without requiring $z_j^{(l)} = 1, \forall j$.

Corollary 2. *If the MI of the LLR of APP can reach 1 for the first variable node, i.e., $z_1^{(l)} = 1$, then by running a sufficiently large number of iterations furthermore, the MI values of the LLR of APP for all other variable nodes will also reach 1, i.e., $z_j^{(l)} = 1$, for $j = 2, \ldots, n$.*

As L goes to infinity, the LDPC-CC with termination length L or $L + 1$ can be seen as the same code. According to Lemma 3, it implies that $\bar{\sigma}^2_{\mathrm{ch}}(L)$ converges as L gets large. Suppose that there exists a value L_0 such that for $L \geq L_0$, the threshold $\bar{\sigma}^2_{\mathrm{ch}}(L)$ converges. Then, we can utilize Eq. (5.9) to easily derive the IDTs of the LDPC-CCs for any arbitrary termination length $L > L_0$ since the code rate $R(L)$ is available given the (J, K, L) values. Thus, as long as L_0 is not a huge number, we can first identify it and then all IDTs for $L > L_0$ (no matter how large L is) can be computed. Figure 5.1 illustrates the trend of $\bar{\sigma}^2_{\mathrm{ch}}(L)$ with respect to L for several typical J, K values. It is shown that $\bar{\sigma}^2_{\mathrm{ch}}(L)$ increases rapidly when L increases from 10 to 20. The convergence behavior of $\bar{\sigma}^2_{\mathrm{ch}}(L)$ appears substantial as L exceeds 50. Generally speaking, it is thus quite handy to choose $L_0 = 100$ and calculate the IDT for $L \geq 100$ using $\bar{\sigma}^2_{\mathrm{ch}}(100)$. Obviously, one does not need to carry out the PEXIT algorithm to compute $\bar{\sigma}^2_{\mathrm{ch}}(L)$ for large L since its convergence is fortunately *fast*. This fact greatly simplifies the IDT calculation for the LDPC-CCs having arbitrary large L, which includes the asymptotic case for $L \to \infty$.

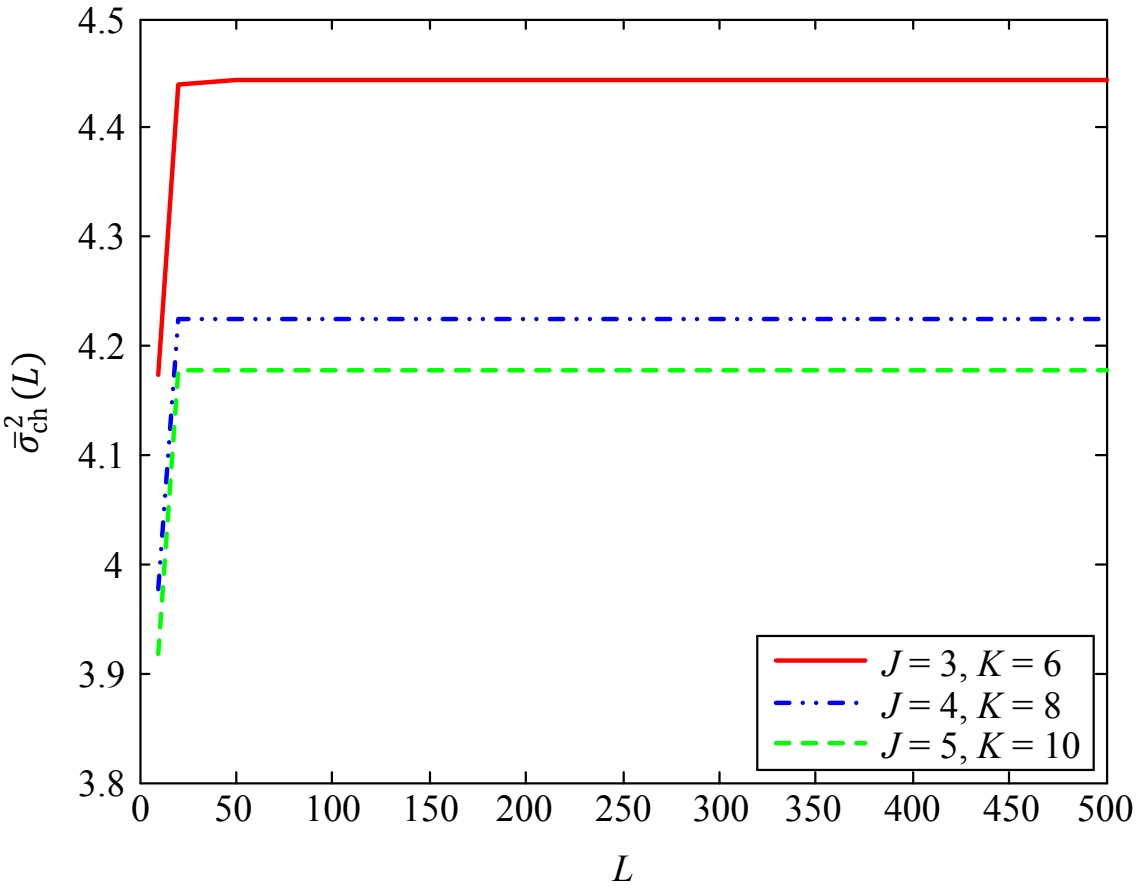

Figure 5.1: The variance threshold $\bar{\sigma}^2_{\mathrm{ch}}(L)$ with respect to the termination length L for some typical LDPC-CCs with three different (J, K) combinations.

5.2.1 Our Proposed PEXIT-Fast Algorithm

By taking advantage of Corollary 1, Corollary 2, and the aforementioned asymptotic analysis of $\bar{\sigma}^2_{\mathrm{ch}}(L)$, we propose a new *PEXIT-fast algorithm* which can greatly reduce the complexity for obtaining the IDT estimate of an LDPC-CC with an arbitrary termination length L over AWGN channels. The pseudocode of our proposed PEXIT-fast algorithm is presented by Figure 5.2. The specifications of the input variables are given as follows. For the (J, K, L) base matrix, it is assumed that we are interested in $L \geq 10$ because LDPC-CCs become favorable in practice when L is large. According to Figure 5.1, L_0 is set to 100. The initial E_b/N_0 search range is specified by $[\eta'', \eta']$. An initial lower bound η'' can be set as

Input: A (J, K, L) base matrix with dimension $m \times n$, L_0, η'', η', θ, δ.
Output: The IDT estimate $\hat{\eta}(L)$.

1: **if** $L \leq L_0$ **then**
2: **if** $\eta' - \eta'' < \theta$ **then**
3: **return** $\hat{\eta}(L) = \eta'$
4: $E_b/N_0 = (\eta' - \eta'')/2$
5: Do Step 1 of the PEXIT algorithm in Section 5.1.2
6: Initialize $z_1^{(0)} = 0$
7: Do Steps 2 to 4 of the PEXIT algorithm in Section 5.1.2
8: Store $z_1^{(l)}$
9: **if** $z_1^{(l)} = 1$ **then**
10: $\eta' = E_b/N_0$; go to Line 2
11: **else if** $z_1^{(l)} < 1$ and $z_1^{(l)} - z_1^{(l-1)} < \delta$ **then**
12: $\eta'' = E_b/N_0$; go to Line 2
13: **else**
14: Go to Line 7
15: **else**
16: Go to Line 2 using the (J, K, L_0) base matrix instead to obtain $\hat{\eta}(L_0)$. Note that L should be replaced by L_0 here for Lines 2-14
17: **return** $\hat{\eta}(L) = \hat{\eta}(L_0)R(L_0)/R(L)$

Figure 5.2: Our proposed PEXIT-fast algorithm.

the *ultimate Shannon limit*, -1.59 dB. An initial upper bound η' can then be obtained by running the PEXIT algorithm using the $(J, K, L' = 10)$ base matrix several times; each time the input E_b/N_0 is incremented by 1 dB starting from $E_b/N_0 = \eta''$. The accuracy of the IDT estimate is specified by θ. In this chapter, we set $\theta = 10^{-3}$ dB. Whether the MI of APP for the first variable node $z_1^{(l)}$ converges or not is specified by δ such that convergence occurs when $z_1^{(l)} - z_1^{(l-1)} \leq \delta$. Here δ is set to be 10^{-12}. The aforementioned specifications of these input variables may be changed for other circumstances if necessary.

As illustrated by Figure 5.2, in Lines 5 and 7, Steps 1-4 of the PEXIT algorithm stated in Section 5.1.2 are carried out. In Line 6, the first variable node's MI of APP $z_1^{(0)}$ is initialized as 0. In Line 8, at the iteration l, store the first variable node's MI of APP $z_1^{(l)}$ which will be needed for comparison with the result from the previous iteration. In Lines 9-14, a new

stopping criterion is facilitated to determine, in a fast manner, if the currently investigated E_b/N_0 is the upper or lower bound of the IDT.

Note that for large L ($L > L_0$), the PEXIT-fast algorithm does not need to be executed directly for L as specified by Lines 16 and 17 (impossible to be directly executed anyway for $L \to \infty$). Instead, it calculates the threshold $\bar{\sigma}^2_{\text{ch}}(L_0)$ using Eq. (5.8) by running the routine specified by Lines 2-14 to compute $\hat{\eta}(L_0)$. Then, $\bar{\sigma}^2_{\text{ch}}(L_0)$ is used to calculate the estimated IDT $\hat{\eta}$ for arbitrary $L > L_0$ using Eq. (5.9). As a matter of fact, one only needs to calculate $\bar{\sigma}^2_{\text{ch}}(L_0)$ once to obtain various IDT estimates for all $L > L_0$.

Our proposed new PEXIT-fast algorithm does not carry out the entire iterative process which is otherwise needed by the conventional PEXIT analysis described in Section 5.1.2. Our scheme also provides a computationally-efficient way to determine the IDT of an LDPC-CC for an arbitrary large termination length L, which includes the case of $L \to \infty$. Thus, the computational complexity for calculating the IDTs of LDPC-CCs over AWGN channels can be significantly reduced.

5.2.2 Complexity Analysis

The advantage of our proposed PEXIT-fast algorithm is manifested by the "early termination" mechanism described in Section 5.2.1 and by that one may simply use a "small" termination length L_0 to obtain all IDT estimates for $L > L_0$. For a given L, the computational complexities of the existing PEXIT algorithm and our proposed PEXIT-fast algorithm can be represented by the *total number of iterations* applied to obtain the IDT estimate $\hat{\eta}$, denoted by $\Delta(L)$ and $\Lambda(L)$, respectively. The computational complexity comparison can then be carried out by evaluating the trends of $\Delta(L)$ and $\Lambda(L)$ with respect to L.

In the search procedure for the IDT of a given LDPC-CC with termination length L, there will be a sequence of E_b/N_0 values to be examined (see Line 4 of Figure 5.2) for determining the final IDT estimates. For the PEXIT algorithm, denote the examined sequence of E_b/N_0 by $[\alpha_1, \alpha_2, \ldots, \alpha_{D_1}]$. For the PEXIT-fast algorithm, denote the examined sequence of E_b/N_0 by $[\beta_1, \beta_2, \ldots, \beta_{D_2}]$. The lengths of these two sequence, say D_1 and D_2, are determined by the precision requirement θ for the IDT estimates. Assume that the conventional PEXIT algorithm and our proposed PEXIT-fast algorithm both use the same precision parameter θ and the same binary section search procedure described in Figure 5.2. For each α_μ, $\mu = 1, 2, \ldots, D_1$, denote the number of iterations undertaken in the conventional PEXIT algorithm by $\Delta_\mu(L)$. For each β_ν, $\nu = 1, 2, \ldots, D_2$, denote the number of iterations undertaken in our proposed PEXIT-fast algorithm by $\Lambda_\nu(L)$. The total number of iterations used by the PEXIT algorithm, $\Delta(L)$, can thus be expressed by

$$\Delta(L) = \sum_{\mu=1}^{D_1} \Delta_\mu(L). \tag{5.13}$$

On the other hand, according to Figure 5.2, the total number of iterations used by the PEXIT-fast algorithm, $\Lambda(L)$, can be expressed by

$$\Lambda(L) = \begin{cases} \sum_{\nu=1}^{D_2} \Lambda_\nu(L), & L \leq L_0 \\ \Lambda(L_0), & L > L_0 \end{cases}. \tag{5.14}$$

According to Eq. (5.13), it is expected that $\Delta(L)$ always increases with L. However, $\Lambda(L)$ becomes a "constant" for $L > L_0$. Thus, the larger the termination length L, the more efficient the PEXIT-fast algorithm for calculating the IDT estimate. Since the PEXIT-fast algorithm always runs up to L_0 (once) for obtaining the IDT estimates for $L > L_0$, the memory storage requirement for the PEXIT-fast algorithm does not increase furthermore

with respect to L for $L > L_0$. On the other hand, the memory storage requirement of the PEXIT algorithm always increases with L.

5.3 Numerical Results

In this section, we demonstrate the numerical results for evaluating our proposed new PEXIT-fast algorithm over various (J, K, L) LDPC-CCs. First, the evolution of mutual information of APP $z_j^{(l)}$ with respect to the iteration number l for different E_b/N_0 values will be illustrated. Then, the IDTs estimated from our proposed PEXIT-fast algorithm are compared to the IDTs given by [1]. Finally, the computational complexity comparison is also demonstrated in terms of total number of iterations for our proposed PEXIT-fast algorithm (details can be referred to Section 5.2.1) and the conventional PEXIT algorithm (details can be referred to Section 5.1.2).

Figure 5.3 illustrates the evolution of mutual information $z_j^{(l)}$ when the conventional PEXIT algorithm is carried out for the LDPC-CC with the $(3, 6, 500)$ base matrix whose corresponding IDT estimate $\hat{\eta}$ is 0.475 dB. Two E_b/N_0 values are examined to exemplify the two cases of evolution. For $E_b/N_0 = 0.55$ dB which is greater than $\hat{\eta}$ (note that $\hat{\eta}$ is quite precise and very close to the true IDT η), the percentage of the variable nodes j with $z_j^{(l)} = 1$ increases *linearly* with the number of iterations l as shown by Figure 5.3. In particular, 15.2% of the variable nodes have attained $z_j^{(l)} = 1$ when 1000 iterations are undertaken. And this percentage is further increased to 32%, 48.4%, and 64.8% as we carry out 2000, 3000, and 4000 iterations, respectively. Actually, 5,912 iterations are needed to make $z_j^{(l)} = 1$ for all variable nodes. Note that as the iteration number l increases, $z_j^{(l)}$ reaches 1 sequentially from both ends (boundary positions) of the variable nodes to the center. On the other hand, for

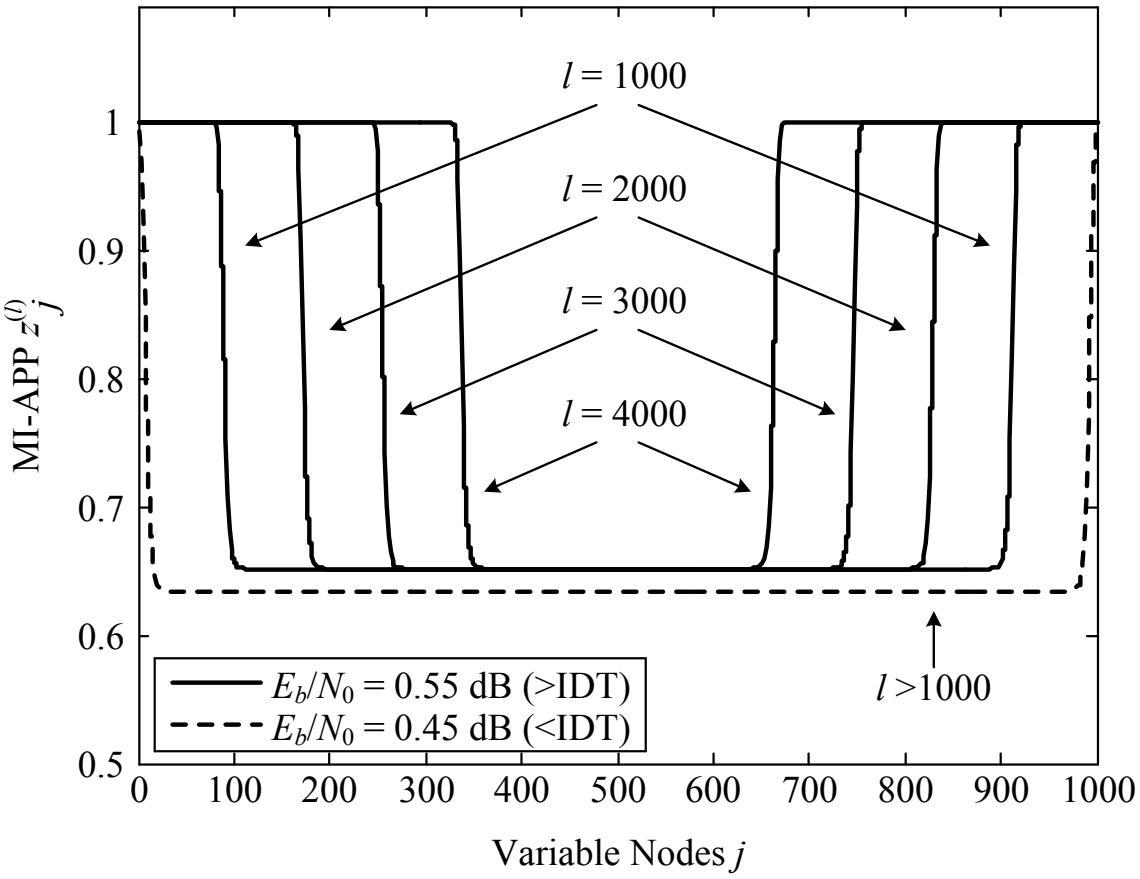

Figure 5.3: The evolution of mutual information of APP $z_j^{(l)}$ for different iteration numbers l and different E_b/N_0 values when the conventional PEXIT algorithm is adopted. The $(3, 6, 500)$ LDPC-CC is used for illustration here.

$E_b/N_0 = 0.45$ dB which is smaller than $\hat{\eta}$, the mutual information $z_j^{(l)}$ for each variable node exhibits the "*premature*" convergence (to some value below 1) at an early stage. Regardless of how many iterations are carried out in the PEXIT algorithm (as shown in Figure 5.3, $l > 1000$), the MI $z_j^{(l)}$ for each variable node stops increasing and none of them can reach 1.

Similar phenomena can be observed for LDPC-CCs with other (J, K, L) combinations. It is also observed that for very large L, when an E_b/N_0 value is greater than but quite close to the IDT, the evolution of the mutual information $z_j^{(l)}$ becomes very slow, and an enormously large iteration number l is required to make $z_j^{(l)}$ reach 1 for every variable node.

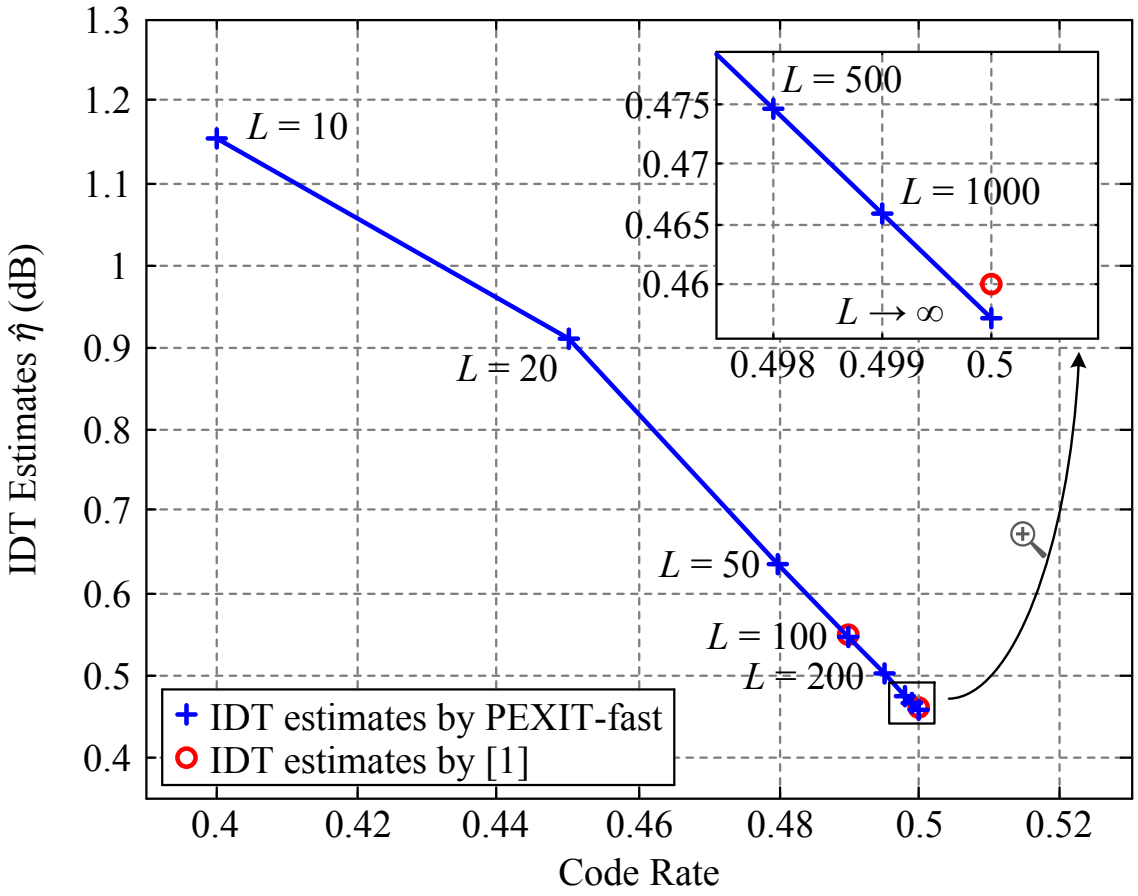

Figure 5.4: The IDT estimates for the $(3, 6, L)$ LDPC-CCs resulting from our proposed PEXIT-fast algorithm and [1], where the termination lengths L range from 20 to infinity.

Figure 5.4 depicts the trend of IDT estimates versus code rates for the $(3, 6, L)$ LDPC-CCs with different termination lengths L. It is shown that with the increase of L, the IDT continues to decrease and the code rate continues to increase. The IDTs estimated by [1] using the *density evolution technique* are also provided here for comparison. The closeness between these two sets of estimated IDTs (by our proposed method and by [1]) demonstrates the effectiveness of our proposed PEXIT-fast algorithm.

The detailed comparison between the IDTs obtained by our proposed PEXIT-fast algorithm and the IDTs presented in [1] is shown in Table 5.1. Note that code rate 0.5 implies $L \to \infty$. The IDT for $L \to \infty$ using our proposed PEXIT-fast algorithm is derived by

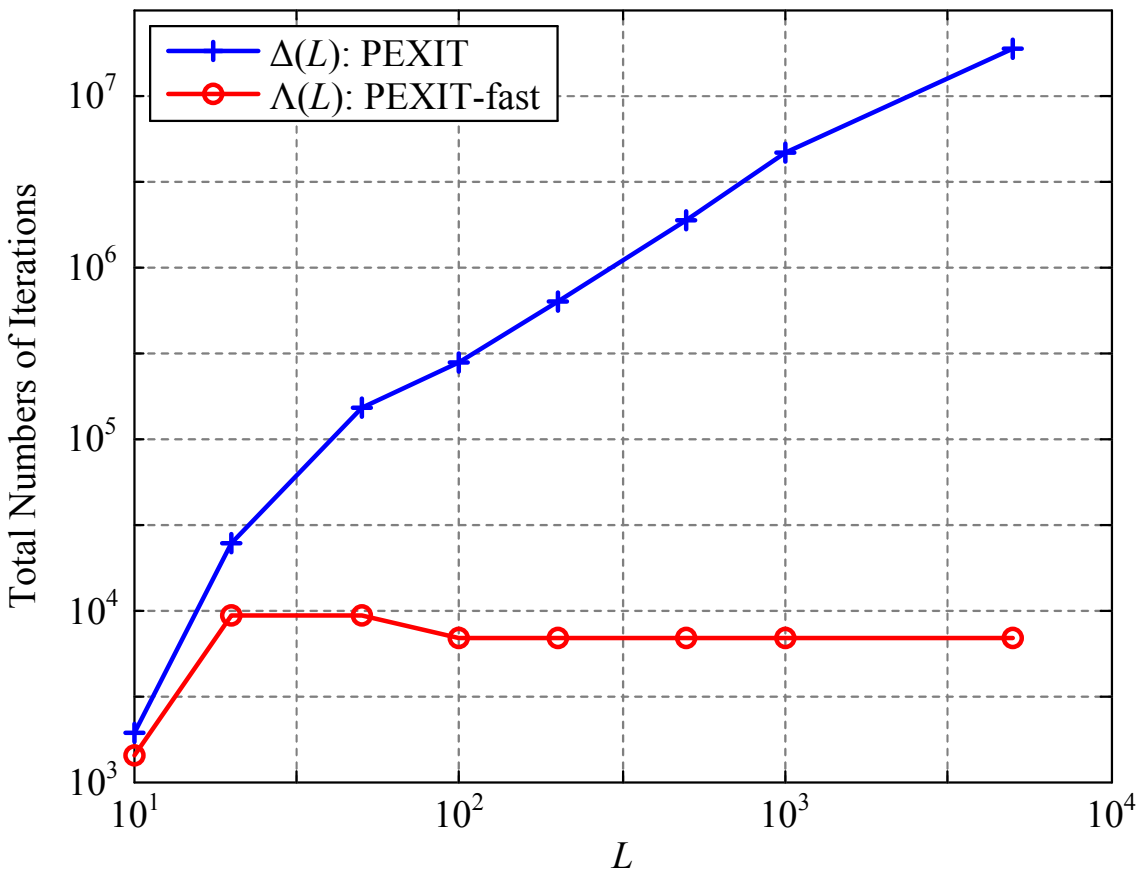

Figure 5.5: The total numbers of iterations undertaken by the conventional PEXIT algorithm and our PEXIT-fast algorithm for calculating the IDTs of the $(3, 6, L)$ LDPC-CCs with the termination lengths L ranging from 10 to 5000.

$\bar{\sigma}^2_{\rm ch}(L_0 = 100)$. It is shown by Table 5.1 that the absolute values of errors between these two sets of IDTs are within 0.025 dB. In Table 5.2, we list the IDTs for various (J, K, L) LDPC-CCs using our PEXIT-fast algorithm.

In Figure 5.5, the total numbers of iterations, namely $\Delta(L)$ and $\Lambda(L)$ given by Eq. (5.13) and Eq. (5.14), respectively, are compared with respect to L. For the conventional PEXIT algorithm, $\Delta(L)$ increases linearly with L. However, for our PEXIT-fast algorithm, $\Lambda(L)$ converges to a constant as $L > L_0 = 100$. It is obvious that for L=50, 500, and 5000, there is a 10-, 100-, and 1000-times complexity reduction by using our proposed PEXIT-fast algorithm, respectively. As a result, our PEXIT-fast algorithm demonstrates the outstanding

Table 5.1: Comparison between IDTs obtained from our PEXIT-fast algorithm and IDTs in [1]

(J, K)	Code Rate	$\hat{\eta}$ ([1])	$\hat{\eta}$ (PEXIT-fast)	Absolute Error
$(3, 6)$	0.49	0.55 dB	0.545 dB	0.005 dB
$(3, 6)$	0.50	0.46 dB	0.458 dB	0.002 dB
$(4, 8)$	0.49	0.35 dB	0.326 dB	0.024 dB
$(4, 8)$	0.50	0.26 dB	0.238 dB	0.022 dB
$(5, 10)$	0.49	0.30 dB	0.278 dB	0.022 dB
$(5, 10)$	0.50	0.21 dB	0.190 dB	0.020 dB

computational-complexity advantage for estimating IDTs.

5.4 Summary

In this chapter, we propose a novel PEXIT-fast algorithm to estimate the iterative decoding thresholds (IDTs) for prevalent low-density parity-check convolutional codes (LDPC-CCs). Our PEXIT-fast algorithm can determine the lower and upper bounds of the IDTs quickly. New theoretical analysis and lemmas are established as the basis of our new IDT estimation scheme. Accordingly, we devise an efficient approach to determine the IDTs for the LDPC-CCs with arbitrarily large termination lengths L. The effectiveness of our PEXIT-fast algorithm is demonstrated by comparing the IDTs obtained by our PEXIT-fast algorithm and the conventional method. The computational-complexity analysis is also presented to demonstrate the significant advantage of our PEXIT-fast algorithm for calculating the IDT estimates, especially when the termination length L becomes large.

Table 5.2: IDT Estimates $\hat{\eta}$ for Various (J, K, L) LDPC-CCs Using Our PEXIT-fast Algorithm

(J, K)	Code Rate					IDT Estimate $\hat{\eta}$ (dB)				
	$L = 20$	$L = 50$	$L = 100$	$L = 500$	$L \to \infty$	$L = 20$	$L = 50$	$L = 100$	$L = 500$	$L \to \infty$
$(3, 6)$	0.4500	0.4800	0.4900	0.4980	0.5000	0.911	0.635	0.545	0.475	0.458
$(3, 9)$	0.6333	0.6533	0.6600	0.6653	0.6666	1.581	1.448	1.404	1.369	1.360
$(3, 12)$	0.7250	0.7400	0.7450	0.7490	0.7500	2.072	1.984	1.955	1.932	1.926
$(3, 15)$	0.7800	0.7920	0.7960	0.7992	0.8000	2.443	2.378	2.356	2.338	2.334
$(3, 18)$	0.8166	0.8266	0.8300	0.8326	0.8333	2.737	2.685	2.667	2.653	2.650
$(4, 8)$	0.4250	0.4700	0.4850	0.4970	0.5000	0.943	0.507	0.370	0.264	0.238
$(4, 12)$	0.6166	0.6466	0.6566	0.6646	0.6666	1.477	1.271	1.205	1.152	1.139
$(4, 16)$	0.7125	0.7350	0.7425	0.7485	0.7500	1.932	1.798	1.754	1.719	1.710
$(4, 20)$	0.7700	0.7880	0.7940	0.7988	0.8000	2.291	2.191	2.158	2.132	2.126
$(4, 24)$	0.8083	0.8233	0.8283	0.8323	0.8333	2.581	2.501	2.475	2.454	2.449
$(5, 10)$	0.4000	0.4600	0.4800	0.4960	0.5000	1.158	0.552	0.367	0.225	0.190
$(5, 15)$	0.6000	0.6400	0.6533	0.6640	0.6666	1.540	1.260	1.170	1.100	1.083
$(5, 20)$	0.7000	0.7300	0.7400	0.7480	0.7500	1.951	1.769	1.710	1.663	1.652
$(5, 25)$	0.7600	0.7840	0.7920	0.7984	0.7800	2.290	2.155	2.111	2.076	2.068
$(5, 30)$	0.8000	0.8200	0.8266	0.8320	0.8333	2.569	2.462	2.427	2.399	2.392

BIBLIOGRAPHY

[1] M. Lentmaier, A. Sridharan, D. Costello, and K. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inform. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

[2] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge University Press, 2008.

[3] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.

[4] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. New York, NY, USA: Cambridge University Press, 2009.

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

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

[7] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[8] S. Ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[9] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 638–656, Feb 2001.

[10] M. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inform. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug 2004.

[11] S. Myung, K. Yang, and J. Kim, "Quasi-cyclic ldpc codes for fast encoding," *Information Theory, IEEE Transactions on*, vol. 51, no. 8, pp. 2894–2901, Aug 2005.

[12] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, no. 1, pp. 71–81, Jan 2006.

[13] A. Jimenez Felstrom and K. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[14] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

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

[16] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. IEEE International Conference on Communications (ICC'2004)*, Paris, France, Jun. 2004, pp. 772–776.

[17] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[18] R. A. Carrasco and M. Johnston, *Non-binary Error Control Coding for Wireless Communication and Data Storage*. West Sussex, United Kingdom: John Wiley & Sons, Ltd, 2008.

[19] *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*, European Telecommunications Standards Institute Std. ETSI EN 302 307 V1.2.1, 2009.

[20] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std. 802.11n-2012, 2012.

[21] *Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T*, IEEE Std. 802.3an-2006, 2006.

[22] A. Leven and L. Schmalen, "Status and recent advances on forward error correction technologies for lightwave systems," *J. Lightwave Technol.*, vol. 32, no. 16, pp. 2735–2750, Aug 2014.

[23] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 2, pp. 429–439, Feb 2011.

[24] A. Goldsmith and S.-G. Chua, "Adaptive coded modulation for fading channels," *IEEE Trans. Commun.*, vol. 46, no. 5, pp. 595–602, May 1998.

[25] X. Huang, H.-C. Wu, and Y. Wu, "Novel pilot-free adaptive modulation for wireless OFDM systems," *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3863–3867, Nov. 2008.

[26] S.-K. Ahn and K. Yang, "Adaptive modulation and coding schemes based on LDPC codes with irregular modulation," *IEEE Trans. Commun.*, vol. 58, no. 9, pp. 2465–2470, Sep. 2010.

[27] A. Sharma and S. De, "Exploiting fading dynamics along with AMC for energy-efficient transmission over fading channels," *IEEE Commun. Lett.*, vol. 15, no. 11, pp. 1218–1220, Nov. 2011.

[28] M. Mazzotti, S. Moretti, and M. Chiani, "Multiuser resource allocation with adaptive modulation and LDPC coding for heterogeneous traffic in OFDMA downlink," *IEEE Trans. Commun.*, vol. 60, no. 10, pp. 2915–2925, Oct. 2012.

[29] V. Choqueuse, M. Marazin, L. Collin, K. Yao, and G. Burel, "Blind recognition of linear space-time block codes: A likelihood-based approach," *IEEE Trans. Signal Processing*, vol. 58, no. 3, pp. 1290–1299, Mar. 2010.

[30] M. Marey, O. A. Dobre, and R. Inkol, "Blind STBC identification for multiple-antenna OFDM systems," *IEEE Trans. Commun.*, vol. 62, no. 5, pp. 1554–1567, May 2014.

[31] R. Moosavi and E. Larsson, "A fast scheme for blind identification of channel codes," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'2011)*, Houston, TX, Dec. 2011, pp. 1–5.

[32] T. Xia and H.-C. Wu, "Novel blind identification of LDPC codes using average LLR of syndrome a posteriori probability," *IEEE Trans. Signal Processing*, vol. 62, no. 3, pp. 632–640, Feb. 2014.

[33] ——, "Blind identification of nonbinary LDPC codes using average LLR of syndrome *a posteriori* probability," *IEEE Commun. Lett.*, vol. 17, no. 7, pp. 1301–1304, Jul. 2013.

[34] Y. Debessu, H.-C. Wu, and H. Jiang, "Novel blind encoder parameter estimation for turbo codes," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 1917–1920, Dec. 2012.

[35] H.-C. Wu, M. Saquib, and Z. Yun, "Novel automatic modulation classification using cumulant features for communications via multipath channels," *IEEE Trans. Wireless Commun.*, vol. 7, no. 8, pp. 3098–3105, Aug. 2008.

[36] F. Hameed, O. Dobre, and D. Popescu, "On the likelihood-based approach to modulation classification," *IEEE Trans. Wireless Commun.*, vol. 8, no. 12, pp. 5884–5892, Dec. 2009.

[37] W. Headley and C. da Silva, "Asynchronous classification of digital amplitude-phase modulated signals in flat-fading channels," *IEEE Trans. Commun.*, vol. 59, no. 1, pp. 7–12, Jan. 2011.

[38] M. Morelli, C.-C. Kuo, and M.-O. Pun, "Synchronization techniques for orthogonal frequency division multiple access (OFDMA): A tutorial review," *Proc. IEEE*, vol. 95, no. 7, pp. 1394–1427, July 2007.

[39] R. Imad, G. Sicot, and S. Houcke, "Blind frame synchronization for error correcting codes having a sparse parity check matrix," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1574–1577, Jun. 2009.

[40] T. Xia and H.-C. Wu, "Joint blind frame synchronization and encoder identification for low-density parity-check codes," *IEEE Commun. Lett.*, vol. 18, no. 2, pp. 352–355, Feb. 2014.

[41] M. R. Gupta and Y. Chen, "Theory and use of the EM algorithm," *Found. Trends Signal Process.*, vol. 4, no. 3, pp. 223–296, Mar. 2011.

[42] W. Gappmair, R. Lopez-Valcarce, and C. Mosquera, "Joint NDA estimation of carrier frequency/phase and SNR for linearly modulated signals," *IEEE Signal Processing Lett.*, vol. 17, no. 5, pp. 517–520, May 2010.

[43] H.-C. Wu, X. Huang, and D. Xu, "Novel semi-blind ICI equalization algorithm for wireless OFDM systems," *IEEE Trans. Broadcast.*, vol. 52, no. 2, pp. 211–218, Jun. 2006.

[44] D. Pauluzzi and N. Beaulieu, "A comparison of SNR estimation techniques for the AWGN channel," *IEEE Trans. Commun.*, vol. 48, no. 10, pp. 1681–1691, Oct. 2000.

[45] E. Serpedin, P. Ciblat, G. Giannakis, and P. Loubaton, "Performance analysis of blind carrier phase estimators for general QAM constellations," *IEEE Trans. Signal Processing*, vol. 49, no. 8, pp. 1816–1823, Aug. 2001.

[46] W. C. Jakes, Ed., *Microwave Mobile Communications*. New York, NY, USA: Wiley-IEEE Press, 1994.

[47] "NIST Digital Library of Mathematical Functions," http://dlmf.nist.gov/, Release 1.0.9 of 2014-08-29. [Online]. Available: http://dlmf.nist.gov/

[48] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[49] H. Bolcskei, "Blind estimation of symbol timing and carrier frequency offset in wireless OFDM systems," *IEEE Trans. Commun.*, vol. 49, no. 6, pp. 988–999, Jun. 2001.

[50] R. Imad, S. Houcke, and M. Ghogho, "Blind estimation of the phase and carrier frequency offsets for LDPC-coded systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, no. 1, pp. 293–572, 2010.

[51] T. Xia and H.-C. Wu, "Novel blind identification of LDPC codes using average LLR of syndrome *a posteriori* probability," in *Proceedings of IEEE International Conference on Intelligent Transport Systems Telecommunications (ITST'2012)*, Taipei, Taiwan, Nov. 2012, pp. 12–16.

[52] S. Song, B. Zhou, S. Lin, and K. Abdel-Ghaffar, "A unified approach to the construction of binary and nonbinary quasi-cyclic LDPC codes based on finite fields," *IEEE Trans. Commun.*, vol. 57, no. 1, pp. 84–93, Jan. 2009.

[53] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[54] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[55] W. Tang, J. Huang, L. Wang, and S. Zhou, "Nonbinary LDPC decoding by min-sum with adaptive message control," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'2011)*, Prague, Czech Republic, May 2011, pp. 3164–3167.

[56] X. Guan and Y. Fei, "Adaptive extended min-sum algorithm for nonbinary LDPC decoding," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'2011)*, Houston, TX, Dec. 2011, pp. 1–6.

[57] F. Kienle and N. Wehn, "Low complexity stopping criterion for LDPC code decoders," in *Proc. IEEE Vehicular Technology Conference (VTC'2005-Spring)*, May 2005, pp. 606–609.

[58] G. Glikiotis and V. Paliouras, "A low-power termination criterion for iterative LDPC code decoders," in *Proc. IEEE Workshop on Signal Processing Systems: Design and Implementation (SiPS'2005)*, Athens, Greece, Nov. 2005, pp. 122–127.

[59] J. Li, X.-H. You, and J. Li, "Early stopping for LDPC decoding: convergence of mean magnitude (CMM)," *IEEE Commun. Lett.*, vol. 10, no. 9, pp. 667–669, Sep. 2006.

[60] G. Han and X. Liu, "A unified early stopping criterion for binary and nonbinary LDPC codes based on check-sum variation patterns," *IEEE Commun. Lett.*, vol. 14, no. 11, pp. 1053–1055, Nov. 2010.

[61] L. Zeng, L. Lan, Y. Tai, S. Song, S. Lin, and K. Abdel-Ghaffar, "Constructions of nonbinary quasi-cyclic LDPC codes: A finite field approach," *IEEE Trans. Commun.*, vol. 56, no. 4, pp. 545–554, Apr. 2008.

[62] D. Costello and G. Forney, Jr., "Channel coding: The road to channel capacity," *Proc. IEEE*, vol. 95, no. 6, pp. 1150–1177, Jun. 2007.

[63] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[64] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "High throughput low-density parity-check decoder architectures," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'01)*, 2001, pp. 3019–3024 vol.5.

[65] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop on Signal Processing Systems (SIPS'2004)*, Oct. 2004, pp. 107–112.

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

[67] J. Zhang and M. Fossorier, "Shuffled belief propagation decoding," in *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers 2002*, vol. 1, Nov 2002, pp. 8–15 vol.1.

[68] A. Vila Casado, M. Griot, and R. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proc. IEEE International Conference on Communications (ICC'07)*, Jun. 2007, pp. 932–937.

[69] ——, "LDPC decoders with informed dynamic scheduling," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3470–3479, Dec. 2010.

[70] G. Han and X. Liu, "An efficient dynamic schedule for layered belief-propagation decoding of LDPC codes," *IEEE Commun. Lett.*, vol. 13, no. 12, pp. 950–952, Dec. 2009.

[71] Y. Gong, X. Liu, W. Yecai, and G. Han, "Effective informed dynamic scheduling for belief propagation decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 59, no. 10, pp. 2683–2691, Oct. 2011.

[72] X. Liu, Y. Zhang, and R. Cui, "Variable-node-based dynamic scheduling strategy for belief-propagation decoding of LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 2, pp. 147–150, Feb. 2015.

[73] Y. Mao and A. Banihashemi, "Decoding low-density parity-check codes with probabilistic scheduling," *IEEE Commun. Lett.*, vol. 5, no. 10, pp. 414–416, Oct. 2001.

[74] M. Beermann, L. Schmalen, and P. Vary, "Improved decoding of binary and non-binary LDPC codes by probabilistic shuffled belief propagation," in *Proc. IEEE International Conference on Communications (ICC'2011)*, Jun. 2011, pp. 1–5.

[75] D. Levin, E. Sharon, and S. Litsyn, "Lazy scheduling for LDPC decoding," *IEEE Commun. Lett.*, vol. 11, no. 1, pp. 70–72, Jan. 2007.

[76] H.-C. Lee and Y.-L. Ueng, "LDPC decoding scheduling for faster convergence and lower error floor," *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3104–3113, Sep. 2014.

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

[78] D. Costello, L. Dolecek, T. Fuja, J. Kliewer, D. Mitchell, and R. Smarandache, "Spatially coupled sparse codes on graphs: theory and practice," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 168–176, July 2014.

[79] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," Jet Propulsion Lab, Pasadena, CA, IPN Progress Rep. 42-154, Aug. 2003.

[80] M. Lentmaier, D. Mitchell, G. Fettweis, and D. Costello, "Asymptotically good LDPC convolutional codes with AWGN channel thresholds close to the shannon limit," in *6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC'2010)*, Sep. 2010, pp. 324–328.

[81] A. Iyengar, M. Papaleo, P. Siegel, J. Wolf, A. Vanelli-Coralli, and G. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inform. Theory*, vol. 58, no. 4, pp. 2303–2320, Apr. 2012.

[82] N. ul Hassan, A. Pusane, M. Lentmaier, G. Fettweis, and D. Costello, "Non-uniform windowed decoding schedules for spatially coupled codes," in *Proc. IEEE Global Communications Conference (GLOBECOM'2013)*, Dec. 2013, pp. 1862–1867.

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

[84] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, Apr. 2004.

[85] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM'2007)*, Nov. 2007, pp. 3250–3254.

VITA

Tian Xia was born in 1987 in a small town of Shanxi province, China. He received his B.S. and M.S. degrees in electrical engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2008 and 2011, respectively. He also received an M.S. degree in electrical engineering from Louisiana State University in 2013. He came to Louisiana State University in 2012 and is currently a Ph.D. candidate in electrical and computer engineering.