Ph.D. DISSERTATION

Low-Complexity Decoding Schemes for LDPC Codes Using Efficient Unreliable Path Search

비신뢰 경로 검색 기법을 이용한
저밀도 패리티 체크 부호를 위한
저복잡도 복호 기법 연구

BY

Pilwoong Yang

AUGUST 2017

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Low-Complexity Decoding Schemes for LDPC Codes Using Efficient Unreliable Path Search

비신뢰 경로 검색 기법을 이용한
저밀도 패리티 체크 부호를 위한
저복잡도 복호 기법 연구

BY

Pilwoong Yang

AUGUST 2017

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Low-Complexity Decoding Schemes for LDPC Codes Using Efficient Unreliable Path Search

비신뢰 경로 검색 기법을 이용한
저밀도 패리티 체크 부호를 위한
저복잡도 복호 기법 연구

지도교수 노 종 선
이 논문을 공학박사 학위논문으로 제출함

2017년 8월

서울대학교 대학원

전기 컴퓨터 공학부

양 필 웅

양필웅의 공학박사 학위 논문을 인준함

2017년 8월

위 원 장: _____
부위원장: _____
위 원: _____
위 원: _____
위 원: _____

Abstract

This dissertation contains the following contributions on the low-complexity decoding schemes of LDPC codes.

- Two-stage decoding scheme for LDPC codes
  - A new stopping criterion for LDPC codes
  - A new decoding scheme for LDPC codes with unreliable path search
- Parallel unreliable path search algorithm
- Analysis of two-stage decoding schemes
  - Validity and complexity analysis

First, a new two-stage decoding scheme for low-density parity check (LDPC) codes to lower the error-floor is proposed. The proposed decoding scheme consists of the conventional belief propagation (BP) decoding algorithm as the first-stage decoding and the re-decodings with manipulated log-likelihood ratios (LLRs) of variable nodes as the second-stage decoding. In the first-stage decoding, an early stopping criterion is proposed for early detection of decoding failure and the candidate set of the variable nodes is determined, which can be partly included in the small trapping sets. In the second-stage decoding, the scores of the variable nodes in the candidate set are computed by the proposed unreliable path search algorithm and the variable nodes are sorted in ascending order by their scores for the re-decoding trials. Each re-decoding

trial is performed by BP decoding algorithm with manipulated LLR of a selected variable node in the candidate set one at a time with the second early stopping criterion.

Secondly, the parallel unreliable path search algorithm is proposed for practical application to the proposed unreliable path search algorithm. In order to reduce the decoding delay and computational complexity, an efficient method for the search algorithm based on the parallel message-passing algorithm in the LDPC decoding is proposed. The parallel unreliable path search algorithm significantly reduces the additional complexity without extra hardware requirements.

Finally, the validity and the complexity analysis of the proposed unreliable path search algorithm is presented. The proposed algorithm effectively finds the variable nodes in small trapping sets much more faster than the previous random selection method. Also, it is verified that the additional complexity of the parallel unreliable path search algorithm is less than that of one iteration of iterative decoders.

keywords: Belief propagation (BP) algorithm, error-floor, low-density parity-check (LDPC) codes, trapping set, two-stage decoding scheme.

student number: 2012-30215

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The digital communication systems can be summarized as three functional units: source coding, channel coding, and (de)modulation, which are illustrated in Figure 1.1 [1], [2]. In the transmitter, an analog or a digital source is converted into a sequence of binary digits by the source encoder. The sequence of binary digits is referred as information bits which include very little redundancy. Next, the channel coding is used to recover the information bits from the received sequence with channel impairments such as noise, interference, and fading caused by the communication channel. The channel encoder adds redundant bits to the information bits, which enables that error-correcting codes [3]-[9] guarantee a reliable transmission. Finally, the modulator transforms the binary codewords into the signal waveforms. The process to recover the original source is carried out in reverse order at the receiver.

In this dissertation, the channel coding is mainly focused on among the three es-

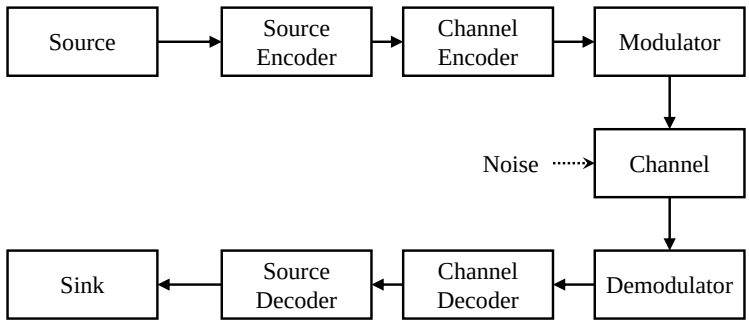

Figure 1.1: Block diagram of a digital communication system.

sential elements of digital communication systems. The channel coding is essential for the reliable data transmission because power and bandwidth are limited in the practical communication system and errors occur in the communication channel. The history of error-correcting codes as channel codes goes way back to 1948.

In 1948, Claude Shannon proves that for any noisy channel, there always exists an error-correcting code that achieves the desired error probability at any information transmission rate as long as it is less than the channel capacity [10]. Unfortunately, Shannon did not report how to find such capacity-approaching error-correcting codes. Since Shannon's work, a great deal of effort has been devoted to find good error-correcting codes and practically implementable encoding and decoding schemes have been also devised.

Error-correcting codes developed for decades can be classified into two categories. The first category is based on algebraic background and is concerned with block codes, which includes Hamming codes, Reed-Muller codes, Bose-Chaudhuri-Hocquenghem (BCH) codes, Reed-Solomon (RS) codes, Golay codes, and Goppa codes. The second category has more probabilistic characteristics and is concerned with tree codes, which

can be expressed by a tree and decoded by algorithms for searching the tree. The convolutional and turbo codes are included in this category.

Low-density parity-check (LDPC) codes were originally invented by Gallager [11] in 1963. LDPC codes show good error-correcting performance which closely approaches the Shannon's theoretical limit in the various channels. However, since the probabilistic iterative decoding scheme requires high computational complexity, it was known that implementing LDPC decoder by the probabilistic iterative decoding scheme was infeasible in the 1960s. Moreover, the concatenated RS codes and convolutional codes [12] were regarded as suitable error-correcting codes for communication systems that require very low error rate. Therefore, LDPC codes and their variants have been forgotten until turbo codes [13] were introduced by Berrou, Glavieux, and Thitimajshima in 1993. The appearance of turbo codes can be seen as the beginning of modern coding theory. With the success of turbo codes, another class of error-correcting codes showing similar characteristics based on iterative decoding has been vigorously investigated. As a result, in 1996, LDPC codes were rediscovered by MacKay and Neal [14], [15]. Since then, LDPC codes have been the main research topic in the channel coding area because they show the capacity-approaching performance with feasible computational complexity.

In 1981, Tanner introduced bipartite graphs to describe linear codes such as LDPC codes and explained the sum-product algorithm on the bipartite graph [16]. The Tanner graph is a basis on understanding graph-based error-correcting codes which can approach the channel capacity through the iterative decoding algorithms. Since the rediscovery of LDPC codes, there have been many new developments for LDPC codes. Several effective decoding schemes of LDPC codes based on the sum-product algo-

rithm have been proposed and many tools for the theoretical analysis of LDPC codes have been developed. Under the iterative message-passing decoding algorithms, LDPC codes show an interesting noise threshold effect [17]; if a noise level of any channel is lower than the noise threshold, bit error probability can converge to zero as the code length goes to infinity. These results were first presented by Gallager for regular LDPC codes over the binary symmetric channel (BSC). Luby, Mitzenmacher, Shokrollahi, and Spielman [18] showed that the noise threshold effect is also valid for irregular LDPC codes and then they constructed some irregular LDPC codes which have very good error-correcting performance closely approaching the theoretical limit on the binary erasure channel (BEC). Finally, these ideas were generalized by Richardson and Urbanke to the belief propagation (BP) decoding algorithm [19] which can be applied to large class of binary input channels including the additive white Gaussian noise (AWGN) channel. They also proposed a numerical tool, called *density evolution*, to analyze the performance of the BP decoding algorithm. In [20], Chung proposed the Gaussian approximation of the density evolution to efficiently analyze the behavior of message-passing decoding algorithm.

Since the recent development of hardware performance, LDPC codes have been one of dominant error-correcting codes for high-speed communication systems or data storage systems. Especially, LDPC codes were adopted in many standards such as IEEE 802.16e [21] for mobile wireless metropolitan area network (WMAN), IEEE 802.11n [22] for wireless local area network (WLAN), IEEE 802.3an [23] for 10GBase-T Ethernet, ITU-T G.hn/G.9960 [24] for home networking over power lines, phone lines, and coaxial cable, and ETSI DVB-S2/C2/T2 [25]–[27] for European digital video broadcasting. Furthermore, in storage systems, high-rate LDPC codes with ex-

cellent error-correcting capability have been researched. It is known that LDPC codes are also applied for recent solid state drive (SSD) storage systems.

The efficient implementation of LDPC codes has been a popular issue due to the great error-correcting capability. Although the great error-correcting performance comes from the BP decoding, it requires a large number of iterations to recover the reliable information and the high computational complexity is followed. Thus, it is very difficult to recover the information data in real-time in a wireless communication system using the conventional BP decoder. In order to reduce the computational complexity caused by unnecessary iterations, various stopping criteria have been proposed to stop the iterative decoder before it reaches the pre-determined maximum iteration number. By using a stopping criterion, the average number of iterations can be substantially reduced especially high signal-to-noise ratio (SNR) without any practical loss of performance.

Stopping criteria for iterative decoder need to be simple and efficient. The efficiency of a stopping criterion can be measured by low false alarm rate and low miss detection rate as well as the reduced number of iterations. For a stopping criterion based on the convergence of mean magnitude (CMM) [28], it efficiently reduces the average required iteration number of the BP decoder, but the additional computational complexity to calculate the mean magnitude at each iteration makes it hard to implement. On the other hand, the stopping criterion in [29] tracks the number of unsatisfied check nodes at each iteration, which is a simple method compared to the observation of CMM. However, it only stops the iterative decoder when the number of unsatisfied check nodes stays the same for several iterations, which cannot cover the case of oscillating and unstable error patterns of the BP decoder.

Although LDPC codes show great error-correcting capability, it is well known that the LDPC codes suffer from the error-floor phenomenon in the high SNR region, which makes it difficult to use LDPC codes for some applications that require significantly low error rate [17]. The error-floor of the LDPC codes is caused by small trapping sets and small minimum Hamming weights of the codewords [15]. There have been many efforts to solve this error-floor problem for LDPC codes. Some of the works are to carefully construct LDPC codes to avoid small trapping sets that cause error-floor, but it cannot be applied to any existing LDPC codes. On the other hand, the BP decoder can be modified with the pre-examined information of the small trapping sets of the LDPC codes, but it is not always possible to investigate the dominant trapping sets of the LDPC codes before decoding process.

1.2 Overview of Dissertation

This dissertation is organized as follows. Chapter 2 briefly reviews LDPC codes. In Chapter 2.1, the basic concepts of LDPC codes are explained such as Tanner graph representation of LDPC codes. The decoding of LDPC codes is illustrated in Section 2.2 which includes message passing decoder and log-likelihood ratio (LLR) representation of BP decoder. In Section 2.3, the density evolution of LDPC codes and its Gaussian approximation are briefly described. The block-type LDPC codes are introduced in Section 2.4.

In Chapter 3, a new two-stage decoding scheme with unreliable path search is proposed. The proposed scheme is based on the early stopping criteria and re-decoding

scheme by unreliable path search algorithm. Section 3.1 describes overall process of the proposed decoding scheme. Since the proposed decoding scheme consists of two stages, the procedure of each stage is presented in Sections 3.2 and 3.3, respectively. In Subsection 3.3.1, unreliable path search algorithm is introduced to find the variable nodes which may be included in small trapping sets. Some of the variable nodes found in the first-stage decoding are set to be the candidates. With the information of unreliable variable nodes, the LLR manipulation of the selected variable node and re-decoding process are described in Subsection 3.3.2. A simple and effective stopping criterion for the second-stage decoding is also proposed.

In Chapter 4, a low-complexity algorithm for unreliable path search algorithm is designed, namely parallel unreliable path search algorithm. By the parallel unreliable path search algorithm, the scoring of variable nodes in the candidate set can be performed simultaneously in the manner of message-passing algorithm of LDPC codes. Section 4.1 describes the procedure of the parallel unreliable path search algorithm and how the proposed algorithm can replace the unreliable path search algorithm in Subsection 3.3.1 with the reduced decoding delay and computational complexity. The scoring of variable nodes by parallel unreliable path search algorithm is also described in Section 4.2.

In Chapter 5, the proposed unreliable path search algorithm is analyzed. The validity of the unreliable path search algorithm is described in Section 5.1, which shows the proposed algorithm helps to select the unreliable variable nodes more quickly than random selection. In Section 5.2, the complexity of the proposed decoding scheme is analyzed. The overall complexity of the proposed decoding scheme is mainly based on the unreliable path search algorithm, but it is shown that the algorithm only requires

less additional computational complexity than that of the one iteration for an iterative decoder. The performance of the proposed decoding scheme is verified via numerical analysis in Chapter 6.

Chapter 2

Overview of LDPC Codes

In this chapter, some preliminaries of LDPC codes, and their decoding and analysis are introduced. First, the basic concepts of LDPC codes are described and background of LDPC codes such as decoding algorithm and density evolution are provided. Finally, a block-type LDPC code is introduced.

2.1 Basic Concepts

An LDPC code is a linear block code defined by a sparse parity-check matrix H. Let d_v and d_c be the number of 1's in each column and row of the parity-check matrix, respectively.

If d_v and d_c are fixed in the matrix, it is then called a (d_v,d_c) regular LDPC code and otherwise, irregular LDPC codes. Let n be the length of the binary codeword x satisfying $Hx^T = 0^T$. Then, the LDPC code can be expressed as a bipartite graph with n variable nodes and $m := \frac{(nd_v)}{d_c}$ check nodes. Each variable node, i.e., one column

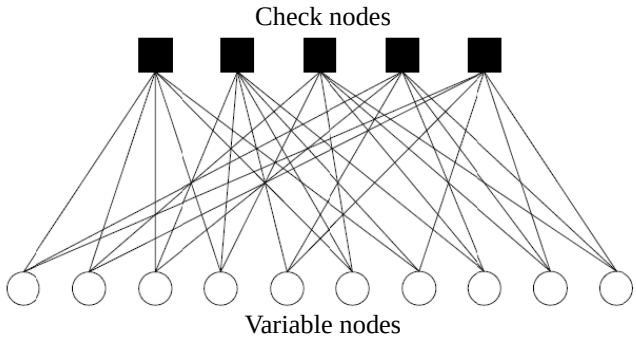

Figure 2.1: A (3,6) regular LDPC code of length 10.

of H, represents one bit of the codeword while each check node, i.e., one row of H, represents one parity-check equation. Edges are defined to be the connections between variable nodes and check nodes, that is, non-zero entries in the parity-check matrix. In general, the parity-check matrices of LDPC codes do not have full rank and thus their exact code rate cannot be defined using the size of H. Assuming that a parity-check matrix has full rank, the rate of LDPC code can be expressed as $r := \frac{(n-m)}{n} = 1 - \frac{d_v}{d_c}$.

LDPC codes can be represented by the graph, called the Tanner graph. Figure. 2.1 shows an example of (3,6) regular LDPC code of length 10, where circles stand for the variable nodes and squares stand for the check nodes. Lines which connect circles to squares represent edges of the code. The parity-check matrix corresponding to the Tanner graph in Figure. 2.1 is given as

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{2.1}$$

Next, some definitions in the Tanner graph are introduced. In a graph, a $path$ is a simple graph whose vertices can be ordered such that two vertices are adjacent if and only if they are consecutive in the list. A $cycle$ is a graph with an equal number of vertices and edges whose vertices can be placed around a circle such that two vertices are adjacent if and only if they appear consecutively along the circle. A $girth$ of a graph is the length of the shortest cycle which contains no repeated edges, provided such a cycle exists.

2.2 Decoding of LDPC Codes

In this section, decoding of LDPC codes is described. When the LDPC codes are decoded, iterative decoder with message-passing algorithm can be used. There are many decoding algorithms of LDPC codes such as sum-product algorithm, min-sum algorithm, bit-flipping algorithm, and their variations. In cycle-free case, sum-product algorithm guarantees optimal performance, which is widely used despite its high computational complexity. In this section, only sum-product algorithm will be introduced as a representative decoding algorithm of LDPC codes. In sum-product algorithm, the output messages of all variable nodes and check nodes are given as a function of all previous input messages except for the incoming message to that node. The decoding algorithm proceeds iteratively.

The sum-product algorithm is described in detail as follows. At the initial step, each variable node has an associated channel output message, u_0. At each following step, messages are interchanged iteratively between variable nodes and check nodes

along the edges in the Tanner graph. First, at iteration 0, each variable node sends a message u_0 to its neighboring check nodes. At iteration 1 or later, each check node calculates its outgoing messages using input messages from variable nodes at the previous iteration, and sends them to its neighboring variable nodes. After receiving all messages from check nodes, each variable nodes also calculates its outgoing messages using input messages from check nodes at the current iteration and channel output message v_0, and passes them back to its neighboring check nodes. In this manner, iteration proceeds until parity-check equation $Hx^T = 0^T$ is satisfied or current iteration reaches the pre-determined maximum number of iterations. Decoding failure will be declared if parity check equation $Hx^T = 0^T$ does not hold until current iteration reaches the maximum number of iterations.

One of the most important requirements in message-passing algorithm is that an outgoing message to a node VN_i (or CN_j) along an adjacent edge e should not have dependency on incoming message from a node VN_i (or CN_j). In massage-passing decoding of LDPC codes, hence, an incoming message along an edge is excluded in calculating the outgoing message along the corresponding edge. This can be understood in the same reason as the case of turbo decoding algorithm that uses extrinsic information only.

Let v be a message from a variable node to a check node. Log-likelihood ratio (LLR) is used generally in calculating messages at each node; i.e., $v = \log \frac{p(y|x=1)}{p(y|x=-1)}$ is used as the outgoing message of a variable node, where $x \in \{+1, -1\}$ is the input bit value into the channel and y denotes total information available on the corresponding node up to the current iteration. Similarly, the output message of a check node u can be defined in the same manner.

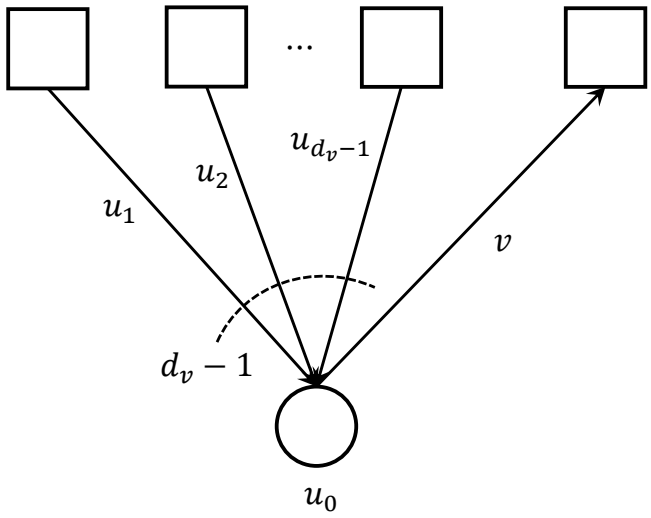

Figure 2.2: Message flow through a variable node.

Under the sum-product algorithm, v is the sum of all incoming messages in form of LLR given by

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

where u_0 is the channel output message, u_i's, $i = 1, 2, ..., d_v - 1$ are the incoming messages from the neighboring check nodes except for the check node having sent a message along the corresponding edge. Figure. 2.2 depicts message flows in the variable node processing.

Message update rule for check nodes can be obtained by transforming the parity-check equation to the equation of LLR value. Detail derivations are shown in [17]. From this, we have

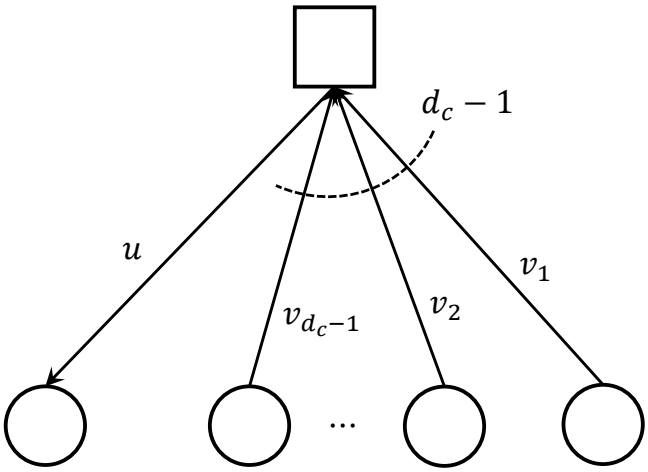

Figure 2.3: Message flow through a check node.

$$\tanh \frac{u}{2} = \prod_{j=1}^{d_c-1} \tanh \frac{v_j}{2}$$

or equivalently,

$$u = \ln \frac{1 + \prod_{j=1}^{d_c-1} \tanh \frac{v_j}{2}}{1 - \prod_{j=1}^{d_c-1} \tanh \frac{v_j}{2}}$$

where v_j's, $j = 1, 2, ..., d_c - 1$ are the incoming messages from $d_c - 1$ neighboring variable nodes of a check node. Figure. 2.3 depicts message flows in check node processing.

2.3 Analysis of LDPC Codes

2.3.1 Density Evolution

In [17], Richardson et al. analyzed decoding algorithms of LDPC codes using density evolution which is one of important tools analyzing LDPC codes together with EXIT chart. It is shown that the error probability of each bit of transmitted codeword tends to go to zero under various message-passing decoding algorithms as code length and iteration go to infinity. Among density evolutions of many algorithms, that of sum-product algorithm will be unfolded in this section in short. In this dissertation, we only consider (d_v, d_c) regular LDPC codes. Let P_{v_i} be the probability density function of v_i which represents a variable node message, and P_{u_j} be the probability density function of u_j which represents a check node message. Since the density of the sum of independent random variables is convolution of each density, the density of a variable node message v is given as

$$P_v = P_{u_0} \otimes P_{u_1} \otimes \ldots \otimes P_{u_{d_v-1}}$$

where operator $\otimes$ denotes convolution. Let $\mathcal{F}$ denote Fourier transform, and thus the above equation is written as

$$P_v = \mathcal{F}^{-1}\left(\mathcal{F}(P_{u_0})\mathcal{F}(P_{u_1})\ldots\mathcal{F}(P_{u_{d_v-1}})\right) = \mathcal{F}^{-1}\left(\prod_{i=0}^{d_v-1}\mathcal{F}(P_{u_i})\right).$$

Let us consider the evolution of densities of the messages at the check nodes. For convenient derivations, probability density function of LLR, $\log\frac{p_0}{p_1}$, can be represented using the following ordered pair:

$$\big(\text{lg sgn}(p_0 - p_1), -\log|p_0 - p_1|\big) \in GF(2) \times [0, \infty)$$

where

$$\text{lg sgn } q = \begin{cases} 1 & \text{if } q < 0 \\ 0 & \text{if } q > 0\,. \end{cases}$$

Note that the above equation essentially decomposes $p_0 - p_1$ into its sign and (log) magnitude. The advantage of this representation arises from the fact that, under this representation, the check node message map in the given space $GF(2) \times [0, \infty)$ is simply addition.

Given a density P of LLR, we can find equivalent density $\widetilde{P}$ over $GF(2) \times [0, \infty)$ by making the appropriate change of measure. Let $y := -\log\tanh(\frac{m}{2})$, and then we find the symmetrical relation $m := -\log\tanh(\frac{y}{2})$.

Let $\widetilde{P}^0(y)$, $y \in [0, \infty)$, be defined by $\widetilde{P}^0(y) = \widetilde{P}(0, y)$, and let $\widetilde{P}^1(y)$, $y \in [0, \infty)$, be defined by $\widetilde{P}^1(y) = \widetilde{P}(1, y)$. By differentiating, we obtain

$$\widetilde{P}^0(y) = \frac{1}{\sinh(y)} P\left(-\log\tanh\frac{y}{2}\right)$$

and

$$\widetilde{P}^1(y) = \frac{1}{\sinh(y)} P\left(\log\tanh\frac{y}{2}\right).$$

Similarly, given $\widetilde{P}$, for $m > 0$, we have

$$P(m) = \frac{1}{\sinh(m)} \widetilde{P}^0\left(-\log\tanh\frac{m}{2}\right)$$

and for $m < 0$,

$$P(m) = \frac{1}{\sinh(-m)} \widetilde{P}^1 \left(- \log \tanh \frac{-m}{2} \right).$$

Let $\widetilde{Q}$ denote the probability density of $\sum_{i=1}^{k} \widetilde{m}_i$, where $\widetilde{m}_i \in GF(2) \times [0, \infty)$ corresponds to a LLR of m_i distributed according to P_i. We then have

$$\mathcal{F}(\widetilde{Q})(s,0) = \prod_{i=1}^{k} \left(\hat{\tilde{P}}_i^0(s) + \hat{\tilde{P}}_i^1(s) \right)$$

$$\mathcal{F}(\widetilde{Q})(s,1) = \prod_{i=1}^{k} \left(\hat{\tilde{P}}_i^0(s) - \hat{\tilde{P}}_i^1(s) \right)$$

where $\hat{\tilde{P}}_i^0$ and $\hat{\tilde{P}}_i^1$ are the Laplace transform of $\tilde{P}_i^0$ and $\tilde{P}_i^0$, respectively.

Finally, let us consider the notation of iteration and use the following relations,

$$\hat{\tilde{Q}}^{(l),0} - \hat{\tilde{Q}}^{(l),1} = \left(\hat{\tilde{P}}^{(l-1),0} - \hat{\tilde{P}}^{(l-1),1} \right)^{d_c - 1}$$

$$\hat{\tilde{Q}}^{(l),0} + \hat{\tilde{Q}}^{(l),1} = \left(\hat{\tilde{P}}^{(l-1),0} + \hat{\tilde{P}}^{(l-1),1} \right)^{d_c - 1}$$

we have a recurrence relation

$$\mathcal{F}\left(P^{(l+1)} \right) = \mathcal{F}\left(P^{(0)} \right) \mathcal{F}\left(Q^{(l)} \right)^{d_v - 1}.$$

It can be shown that using the above relation that the error probability of each bit of transmitted codeword tends to go to zero as l goes to infinity.

2.3.2 Mean Evolution

Density evolution with Gaussian approximation or mean evolution is based on approximating the probability densities of messages as Gaussian random variables or Gaussian mixtures [20]. Since this is easier to analyze and computationally faster than the density evolution, it is useful method for investigating the behavior of the message-passing decoding algorithm. In this section, we will only consider (d_v, d_c) regular LDPC codes. Let m_u and m_v be the means of u and v, respectively. By taking expectation at both sides of $v = \sum_{i=0}^{d_v-1} u_i$, we have

$$m_v^{(l)} = m_{u_0} + (d_v - 1)m_u^{(l-1)}.$$

The updated mean $m_u^{(l)}$ at the l-th iteration can be obtained by taking expectation at both sides of $\tanh \frac{u}{2} = \prod_{j=1}^{d_c-1} \tanh \frac{v_j}{2}$, we have

$$E\left[\tanh \frac{u^{(l)}}{2}\right] = E\left[\tanh \frac{v^{(l)}}{2}\right]^{d_c-1}$$

where

$$E\left[\tanh \frac{u}{2}\right] = \frac{1}{\sqrt{4\pi m_u}} \int_R \tanh \frac{u}{2} e^{-\frac{(u-m_u)^2}{4m_u}} du.$$

Let $\phi(x)$ be the function defined by

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_R \tanh \frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du & \text{if } x > 0 \\ 1 & \text{if } x = 0 \\ 0 & \text{if } x = \infty \, . \end{cases}$$

By combining the above equations, a recursive equation for m_u can be derived as

$$m_u^{(l)} = \phi^{-1}\left(1 - \left[1 - \phi\left(m_{u_0} + (d_v - 1)m_u^{(l-1)}\right)\right]^{d_c - 1}\right).$$

2.4 Quasi-Cyclic LDPC Codes

Let $\mathcal{C}$ be a binary LDPC code whose parity-check matrix $\mathbf{H}$ is a $J \times L$ array of $z \times z$ circulants or zero matrices as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{0,0} & \mathbf{H}_{0,1} & \cdots & \mathbf{H}_{0,L-1} \\ \mathbf{H}_{1,0} & \mathbf{H}_{1,1} & \cdots & \mathbf{H}_{1,L-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{J-1,0} & \mathbf{H}_{J-1,1} & \cdots & \mathbf{H}_{J-1,L-1} \end{bmatrix}$$

where a *circulant* $\mathbf{H}_{j,l}$ is defined as a matrix whose each row is a cyclic shift of the row above it. Such an LDPC code is called *quasi-cyclic*, QC LDPC code, because applying circular shifts to the length-z subblocks of a codeword gives another codeword. Let $M \times N$ be the size of $\mathbf{H}$. Then $M = Jz$, $N = Lz$, and the designed code rate $R = 1 - M/N = 1 - J/L$.

The *weight* of a circulant $\mathbf{H}_{j,l}$ is defined as the number of nonzero elements in the first column and denoted by $\mathrm{wt}(\mathbf{H}_{j,l})$. A circulant of weight 1 is called *circulant permutation matrix*. A *multi-weight circulant* is defined as a circulant of weight larger than 1. A circulant is entirely described by the positions of nonzero elements in the first column. Let i, $0 \leq i \leq z - 1$, be the index of the $(i + 1)$-st element in the first column. Then, the *shift value*(s) of a circulant is defined as the index (indices) of the nonzero element(s) in the first column. Note that a shift value takes the value from 0 to $z - 1$ and ∞ is used as a shift value of a zero matrix $H_{i,j}$.

QC LDPC codes can be fully represented by binary polynomials as in [30]. This polynomial representation is based on the isomorphism between $z \times z$ binary circulants and the polynomial ring $\mathbb{F}_2[x]/(x^z + 1)$. The *polynomial parity-check matrix* $\mathbf{H}(x)$ of $\mathcal{C}$ is defined as

$$\mathbf{H}(x) = \begin{bmatrix} h_{0,0}(x) & h_{0,1}(x) & \cdots & h_{0,L-1}(x) \\ h_{1,0}(x) & h_{1,1}(x) & \cdots & h_{1,L-1}(x) \\ \vdots & \vdots & \ddots & \vdots \\ h_{J-1,0}(x) & h_{J-1,1}(x) & \cdots & h_{J-1,L-1}(x) \end{bmatrix}$$

where $h_{j,l}(x) = \sum_{i=0}^{z-1} h_{j,l,i} x^i \in \mathbb{F}_2[x]/(x^z + 1)$ and $h_{j,l,i}$ is the element with the index i in the first column of $\mathbf{H}_{j,l}$. We can see that the number of nonzero terms in $h_{j,l}(x)$, which is denoted by $\mathrm{wt}(h_{j,l}(x))$, is equal to $\mathrm{wt}(\mathbf{H}_{j,l})$ and the degrees of all nonzero terms in $h_{j,l}(x)$ are equivalent to the shift values of $\mathbf{H}_{j,l}$.

The *protograph* [31] of a QC LDPC code $\mathcal{C}$ is a bipartite graph whose incidence matrix is $\mathbf{P} = [p_{j,l}]$, where $p_{j,l} = \mathrm{wt}(\mathbf{H}_{j,l})$. There are two kinds of nodes in the protograph, where horizontal (check) nodes correspond to rows in $\mathbf{P}$ and vertical (variable) nodes correspond to columns in $\mathbf{P}$. The Tanner graph of $\mathcal{C}$ is constructed by copying the protograph z times and cyclically permuting the same z edges. Such copy-and-permute operation is called *lifting* and the length of a subblock z is also called the *lift size* of $\mathcal{C}$. If $p_{j,l} \geq 2$, there are multiple edges between the horizontal node with index j and the vertical node with index l in the protograph, which equivalently means that $\mathbf{H}_{j,l}$ is a multi-weight circulant. A shift value is assigned to each edge in the protograph so that an edge is lifted by using the cyclic permutation with the assigned shift value to generate $\mathcal{C}$.

A cycle in a QC LDPC code can be considered as a sequence of the corresponding

$p \times p$ permutation matrices. Thus, a cycle of length $2i$ in the conventional QC LDPC code can be expressed as the following sequence

$$(j_0, l_0); (j_1, l_1); \ldots; (j_k, l_k); \ldots; (j_{i-1}, l_{i-1}); (j_0, l_0)$$

where (j_k, l_k) stands for the j_k-th row and l_k-th column block $\mathbf{H}_{j_k,l_k}$ and semicolon between (j_k, l_k) and (j_{k+1}, l_{k+1}) can be considered as the block (j_{k+1}, l_k). Fossorier [32] showed that the necessary and sufficient condition for the existence of the cycle of length $2i$ is

$$\sum_{k=0}^{i-1} (p_{j_k,l_k} - p_{j_{k+1},l_k}) = 0 \mod p$$

where $j_i = j_0$, $j_k \neq j_{k+1}$, and $l_k \neq l_{k+1}$.

2.5 Error-Floor and Trapping Sets

Although the iterative decoding algorithms such as the BP decoding algorithm are known to calculate the exact maximum a posteriori (MAP) bit probability calculation for LDPC codes, the BP decoding algorithm is not optimal in practice because there are many cycles in the practically used LDPC codes. The existence of cycles in Tanner graph of the LDPC codes degrades the error-correcting performance especially in high signal-to-noise ratio (SNR) region. For practical LDPC codes, the bit error ratio (BER) performance rapidly decreases at first, but there is a point after which the BER curve does not fall as quickly as before, which is called the error-floor region of LDPC codes.

It is well known that for LDPC codes, trapping sets are the main reason of the error-floors. Trapping sets are harmful structures caused by various factors, which are

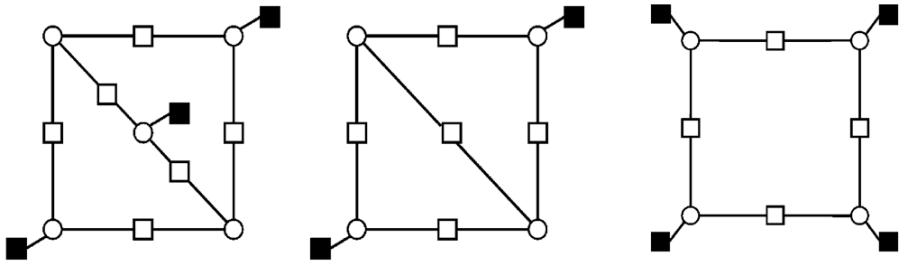

Figure 2.4: (5,3), (4,2), (4,4) trapping sets.

code structure, decoding algorithm, and channel noise configuration. In the case that a noise pattern impairs some received values of the variable nodes in a trapping set, the variable nodes have a large value of reliabilities which are estimated to the wrong direction so that some of the check nodes in the trapping set are mis-satisfied. If all messages converge to some values in this way, stable error event occurs. Otherwise, all reliabilities may not converge to the correct value because the messages cannot increase over the fixed maximum value due to numerical precision problem. In this case, the wrong information propagates from the trapping set to other variable node, and the messages of maximum and minimum values are mixed together at variable nodes on boundaries of the trapping set. This illustrates the behavior of the oscillating error event. Figure 2.4 shows some example of small trapping sets, where a denotes the number of variable nodes and b denotes the number of unsatisfied check nodes in (a, b) trapping set.

Chapter 3

A New Two-Stage Decoding Scheme with Unreliable Path Search

Low-density parity-check (LDPC) codes have been one of the most popular error correcting codes due to their capacity-approaching performance with low-complexity iterative decoding algorithm [14], [15]. From wireless communication systems to data storage systems, LDPC codes have been widely applied and adopted as industrial standards. However, LDPC codes suffer from the error-floor phenomenon in the high signal-to-noise ratio (SNR) region, which makes it difficult to apply LDPC codes to some applications that require significantly low error rate [17]. In the additive white Gaussian noise (AWGN) channel, it is well known that the error-floor of the LDPC codes is caused by small trapping sets and small minimum Hamming weights of the codewords [15].

There have been lots of researches to lower the error-floor of the LDPC codes. Some of the researches have been done to carefully construct LDPC codes to avoid small trapping sets [33]–[36] but it cannot be applied to any existing LDPC codes.

Based on the knowledge of dominant trapping sets in target LDPC codes, some researches mitigate the error-floor by modifying iterative decoder [37], [38]. However, it is not always possible to investigate the trapping sets of LDPC codes before decoding process.

Among the researches which do not require the prior information of dominant trapping sets, two-stage decoding schemes have been one of the popular decoding procedures to deal with the error-floor [39]–[42]. Two-stage decoding schemes usually consist of two decoding steps, the conventional belief propagation (BP) decoding as the first-stage decoding and the re-decodings with manipulated log-likelihood ratios (LLRs) of variable nodes as the second-stage decoding. Some of these schemes are called multi-stage decoding schemes because they use multiple decoding steps in the second-stage decoding but they can also be categorized into the two-stage decoding schemes. Figure 3.1 describes the flow chart of two-stage decoding schemes.

By examining the iterative decoding behavior of codewords in the first-stage decoding, some variable nodes which can be possibly included in trapping sets are selected to be treated in the second-stage decoding. In the perturbation method in [39], the additional Gaussian noise is added to LLR values of the selected variable nodes, which generates nonlinear dynamics to the iterative decoder. However, the variance of the additional Gaussian noise should be carefully chosen. The decoding algorithms in [40] and [29] *invert* the sign of LLR of the selected variable nodes. The backtracking scheme in [40] requires excessive computational complexity especially when it is applied to high-rate LDPC codes. The two-stage decoding scheme in [29] selects types I and II variable nodes to be manipulated in the second-stage decoding but the performance improvement is limited because the selection method of type II variable

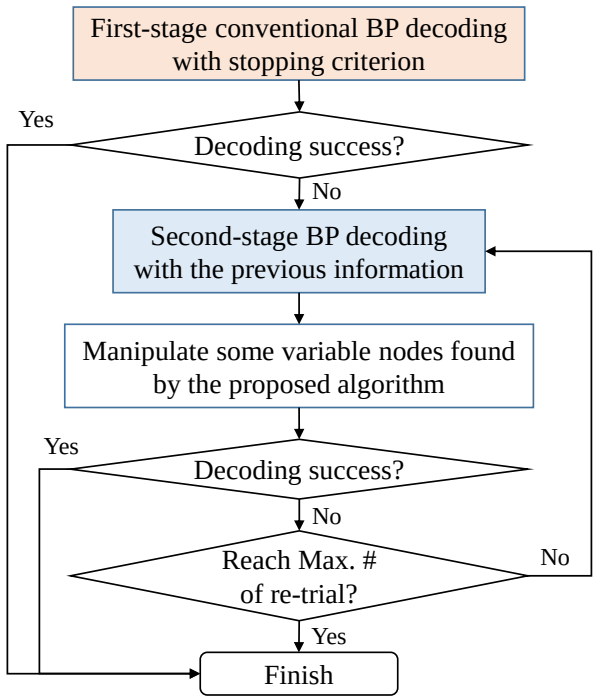

Figure 3.1: Flow chart of two-stage decoding schemes.

nodes does not cover the case of false convergence. In [41], *erasing* of LLR values is performed to the topmost N variable nodes with a large number of sign changes or sign differences. However, determining a proper value of N is quite sensitive to decoding performance for the given code parameters. For a fixed-point implementation, multi-step quantization scheme is proposed to lower the error-floor [42], which is based on successive re-quantization and re-decoding of the input blocks to avoid harmful trapping sets while keeping the same number of quantization bits.

In this chapter, a new two-stage decoding scheme is proposed for LDPC codes to improve the decoding performance in the error-floor region compared to other two-stage decoding schemes. In the first-stage decoding of the proposed decoding scheme,

the conventional BP decoding is used with the proposed early stopping criterion, and the candidate set of the variable nodes is determined. The proposed first early stopping criterion helps to detect the decoding failure at an early time as well as track the small trapping sets which cause the decoding failure. In the second-stage decoding, an unreliable path search algorithm is proposed to assign scores to all variable nodes in the candidate set. Instead of randomly choosing a variable node in the candidate set as in [40], we select the variable nodes in the candidate set sorted by their scores in ascending order. By modifying the LLR of each variable node in the sorted candidate set one at a time, each re-decoding is performed with the other proposed early stopping criterion.

This chapter is organized as follows. Section 3.1 describes the overview of the proposed two-stage decoding scheme with unreliable path search and the detailed descriptions of the first and second stage decoding schemes are presented in Section 3.2 and 3.3, respectively.

3.1 Overview of The Proposed Two-Stage Decoding Scheme

The proposed two-stage decoding scheme is described as follows:

(First-stage decoding)

i) Decode a received codeword by the conventional BP algorithm with the first early stopping criterion up to the maximum number of iterations.

ii) If the decoding is successful, go to end. Otherwise, find a candidate set of variable nodes possibly included in the parts of the small trapping sets and go to the

second-stage decoding.

(Second-stage decoding)

iii) Calculate the scores of the variable nodes in the candidate set by the unreliable path search algorithm and sort them in ascending order by their scores.

iv) Manipulate the LLR value of the first variable node with the lowest score in the sorted candidate set.

v) Re-decode the manipulated codeword by the conventional BP algorithm with the second early stopping criterion.

vi) If the re-decoding is successful, go to end. Otherwise, remove the selected variable node from the candidate set.

vii) If the maximum number of allowed re-decoding trials is reached, go to end. Otherwise, go to step iv).

3.2 First-Stage Decoding with the First Early Stopping Criterion

Even though two-stage decoding schemes work well for LDPC codes in the error-floor region, the decoding schemes may need excessive unnecessary iterations which lead to the unacceptable decoding delay and computational complexity. The unnecessary iterations are caused by the decoding failure in the first-stage conventional BP decoding as well as the failed re-decoding trials in the second-stage decoding. In this section, we focus on an early stopping criterion for the first-stage conventional BP decoding to

reduce the number of iterations. The proposed early stopping criterion is expected to have two purposes, that is, early detection of decoding failure and identification of the unsatisfied check nodes possibly included in the small trapping sets or their mixtures, which will be utilized in the second-stage decoding.

On the error-floor region, the iterative decoders of LDPC codes show three types of behavior for erroneous codewords:

1. Unstable error event: the number of errors randomly changes as iteration progresses as in Figure 3.2.

2. Stable error event: the iterative decoder falls into some trapping sets, and the number of errors stays the same as iteration progresses as in Figure 3.3.

3. Oscillating error event: the iterative decoder falls into some trapping sets, and the number of errors periodically changes as iteration progresses as in Figure 3.4.

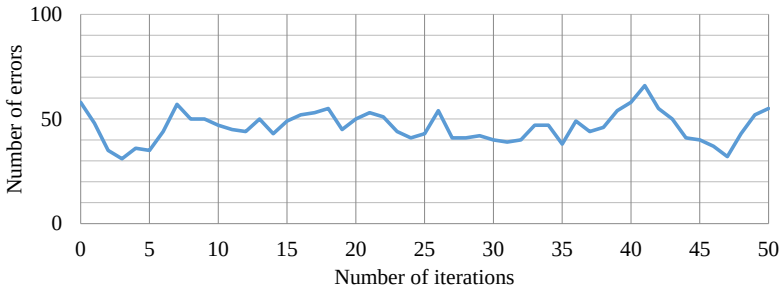

Figure 3.2: Unstable error pattern.

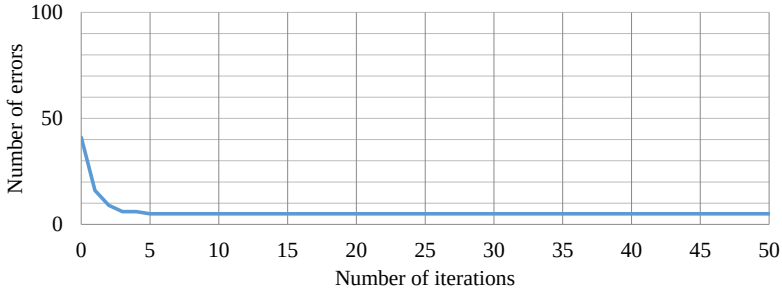

Figure 3.3: Stable error pattern.

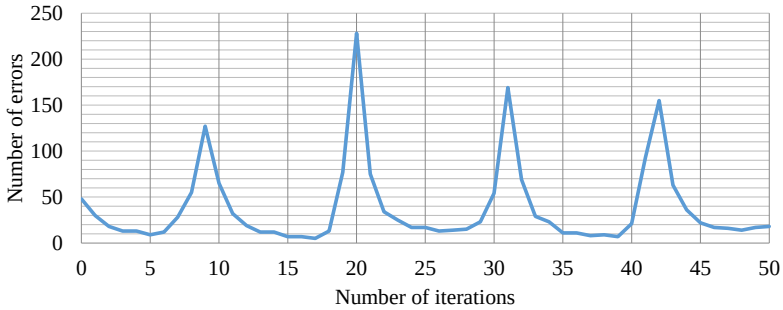

Figure 3.4: Oscillating error pattern.

In practice, the iterative decoders of LDPC codes do not guarantee the convergence of the received sequence to the correct codeword because many cycles exist in LDPC codes with finite lengths. The uncorrectable codewords are known to be related to the trapping sets especially on the error-floor region. From observations, oscillating error events are dominant in the error-floor region, which is known to be caused by numerical precision problems. [43]

Various early stopping criteria [28], [29] have been proposed for the early detection of decoding failure. In [28], Li *et al.* observes the variations of mean magnitude of the LLR values of variable nodes and used them to introduce a criterion for early stopping. However, it requires lots of computation for the mean magnitude at each iteration. Zhang *et al.* [29] track the number of unsatisfied check nodes along with iterations and declare the decoding failure when the number of unsatisfied check nodes remains unchanged for a few iterations. Although tracking the number of unsatisfied check nodes is simpler than computing the mean magnitude, the criterion cannot cover the cases of oscillating and unstable error patterns of the LDPC decoder [37]. On the other hand, Kang *et al.* [40] tries to find the smallest set of unsatisfied check nodes which can be included in the parts of the small trapping sets but the predetermined maximum number of iterations is always used. Shin *et al.* [41] also used the maximum number of allowed iterations in the first-stage conventional BP decoding for counting sign changes and differences, which leads to unnecessary iterations.

The proposed first early stopping criterion is designed for the previously mentioned two purposes. Basically, we track the number of unsatisfied check nodes, stop the decoding and determine the set of unsatisfied check nodes when the number of unsatisfied check nodes reaches the first local minimum. That is, the proposed first early

Table 3.1: The validity of the first early stopping criterion in the error floor region.

	False alarm rate		Miss detection rate		Average number of iterations required before early stopping	
	[29]	Proposed	[29]	Proposed	[29]	Proposed
802.16e R=1/2	0.47%	0.62%	11.54%	1.00%	9.46	7.59
802.16e R=3/4	0.02%	0.03%	23.94%	2.61%	10.11	5.60
QC LDPC R=10/11	0.00%	0.00%	49.14%	0.01%	12.72	4.24

stopping criterion stops the first-stage conventional BP decoding when

$$\left|\mathcal{C}^{(i)}\right| \leq \left|\mathcal{C}^{(i+1)}\right| \leq \left|\mathcal{C}^{(i+2)}\right| \leq \cdots \leq \left|\mathcal{C}^{(i+(\tau-1))}\right|, \quad \left|\mathcal{C}^{(i)}\right| \leq \gamma \tag{3.1}$$

where $\mathcal{C}^{(i)}$ is the set of unsatisfied check nodes at iteration i and γ is a predetermined number. The proposed first early stopping criterion declares a decoding failure when the number of unsatisfied check nodes is nondecreasing for τ iterations and the smallest number of unsatisfied check nodes is smaller than or equal to the predetermined number γ. Not only γ serves as a threshold, but it also reduces the possibility of false alarm, where the false alarm means that the stopping criterion is activated when a correctable codeword by the BP decoding is received. The proposed first early stopping criterion also reduces the miss detection rate, which is the probability that the stopping criterion is not activated when an uncorrectable codeword by the BP decoding is received.

Table 3.1 shows the validity of the proposed first early stopping criterion compared to the stopping criterion in [29] in the error-floor region of the various LDPC codes,

where τ is set to 3 as in [29]. Clearly, lower values of both false alarm and miss detection rates are desirable. When an uncorrectable codeword is detected by the early stopping criterion, the smaller number of iterations required before early stopping is also desirable. The false alarm rate of the proposed early stopping criterion is slightly higher than that in [29] because we also declare the decoding failure when the number of unsatisfied check nodes increases as well as it remains the same. However, the effect on the decoding performance by the false alarm rate increase is negligible. Since the proposed early stopping criterion can catch the oscillating error patterns, the miss detection rate of the proposed algorithm is much lower than that in [29], which results in the decrease of the average number of iterations in the first-stage decoding.

Besides the early and accurate detections of uncorrectable codewords, the proposed first early stopping criterion also has an important role to identify the unsatisfied check nodes possibly included in the small trapping sets, which will be utilized in the second-stage decoding. As in [40], we aim to find the unsatisfied check nodes when the number of unsatisfied check nodes becomes the global minimum for the entire iterations, but it requires a large number of iterations and decoding delay. From the proposed early stopping criterion, $\mathcal{C}^{(i)}$ satisfying (3.1) is regarded as the set of unsatisfied check nodes whose size is the first local minimum, denoted by $\mathcal{C}_0$. Figures. 3.5, 3.6, and 3.7 show that the distribution of $|\mathcal{C}_0|$ for the proposed early stopping criterion is quite similar to that for the global minimum in [40]. The small difference of distributions occurs when $\mathcal{C}_0$ is not in a small trapping set but in the mixture of small trapping sets, which can be correctly handled by the second-stage decoding.

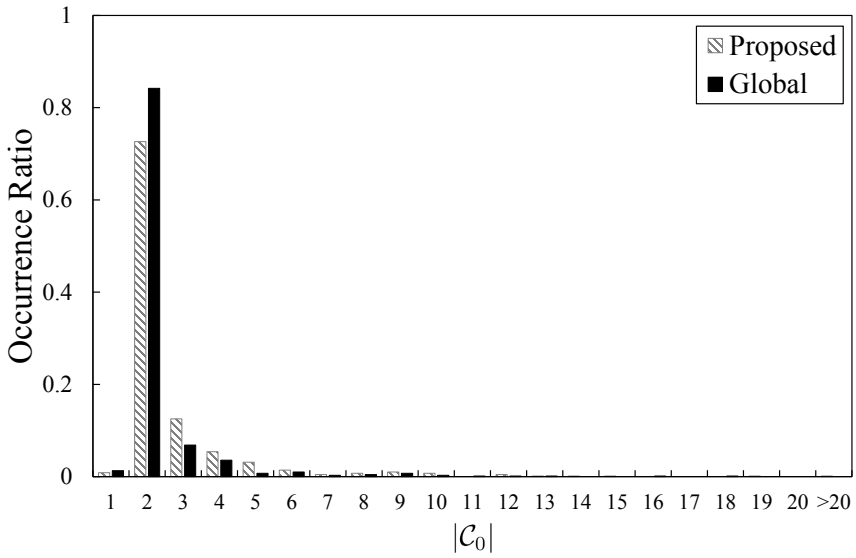

Figure 3.5: Distributions of $|\mathcal{C}_0|$ compared to the global minimum number of unsatisfied check nodes for IEEE 802.16e with $R = 1/2$.

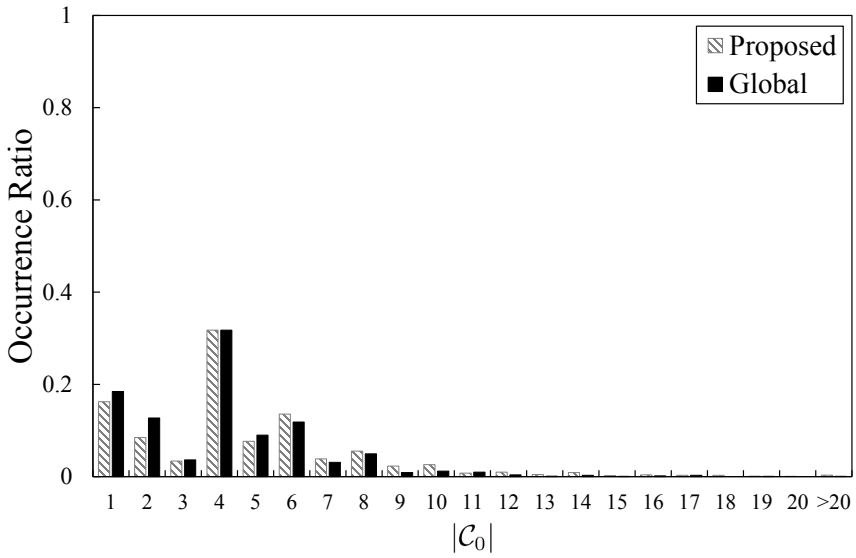

Figure 3.6: Distributions of $|\mathcal{C}_0|$ compared to the global minimum number of unsatisfied check nodes for IEEE 802.16e with $R = 3/4$.

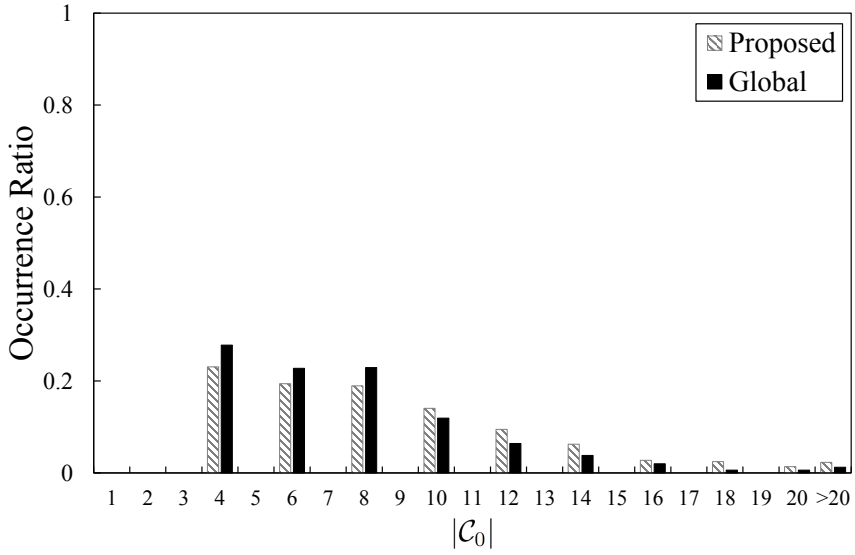

Figure 3.7: Distributions of $|\mathcal{C}_0|$ compared to the global minimum number of unsatisfied check nodes for QC LDPC code with $R = 10/11$.

3.3 Second-Stage Decoding with Unreliable Path Search Algorithm

In this section, we propose a scoring method for the neighboring variable nodes of $\mathcal{C}_0$ by unreliable path search algorithm and the second early stopping criterion for the re-decoding trials in the second-stage decoding. As referred in the previous section, the decoding failures of re-decoding trials in the second-stage decoding cause excessive unnecessary iterations. In the backtracking scheme in [40], after they find the unsatisfied check nodes which may be included in the small trapping sets in the first-stage decoding, all of the variable nodes connected to the unsatisfied check nodes are set to be the candidates for erroneous variable nodes. The backtracking decoding algorithm randomly chooses a variable node in a candidate set to manipulate the initial LLR for each re-decoding trial, which can lead to the unacceptable decoding complexity and delay due to excessive number of iterations. On the other hand, the variable node selection method in [29] chooses the type II variable nodes by investigating their channel-received LLRs, namely channel LLRs, and output LLRs in the first-stage decoding but it is possible that there are no erroneous variable nodes in the selected type II variable nodes. Also, the type II variable nodes do not cover the case that the magnitude of LLR values of the erroneous variable nodes in the trapping set may increase in the wrong way as iteration progresses. In [41], scheme-SC and scheme-SD count the sign changes and sign differences of LLRs of all variable nodes at each iteration, but the predetermined number N, which is how many variable nodes are to be erased, is known to be sensitive to the decoding performance in the error-floor region.

In the second-stage decoding, we reduce the decoding delay and complexity of

re-decoding trials by the unreliable path search algorithm and the second early stopping criterion. The unreliable path search algorithm increases the probability of earlier selection of the erroneous variable nodes in the candidate set and the proposed second early stopping criterion is effective to correctly detect an uncorrectable codeword in the re-decoding trials.

3.3.1 Scoring by Unreliable Path Search Algorithm

For the first early stopping criterion in Section 3.2, we found the set of the unsatisfied check nodes $\mathcal{C}_0$, which is possibly included in the parts of the small trapping sets. Since at least one of the erroneous variable nodes is connected to each unsatisfied check node in the trapping set, we take a set $\mathcal{N}(\mathcal{C}_0)$ of variable nodes connected to $\mathcal{C}_0$ as the candidate set as in [40]. However, instead of randomly choosing a variable node in the candidate set for each re-decoding trial, we propose to make the selecting order of the variable nodes in $\mathcal{N}(\mathcal{C}_0)$ by their assigned scores which are computed by the unreliable path search algorithm as in Algorithm 3.1. Note that the variable node with the lowest score will be firstly selected for its LLR manipulation and re-decoding.

In the proposed unreliable path search algorithm, we focus on the short paths with lower path scores between any of two unsatisfied check nodes in $\mathcal{C}_0$. It is noted that the path distance between any or some of two unsatisfied check nodes in $\mathcal{C}_0$ passing through a trapping set is usually short. Thus, we firstly search for variable nodes in $\mathcal{N}(\mathcal{C}_0)$ which are located in the short paths. Then, the scores of the variable nodes in $\mathcal{N}(\mathcal{C}_0)$ are assigned by the lowest score of the paths including those variable nodes, which are computed as the sum of the absolute values of channel LLRs of the variable nodes in the paths. The path with lower score is highly probable to pass through the

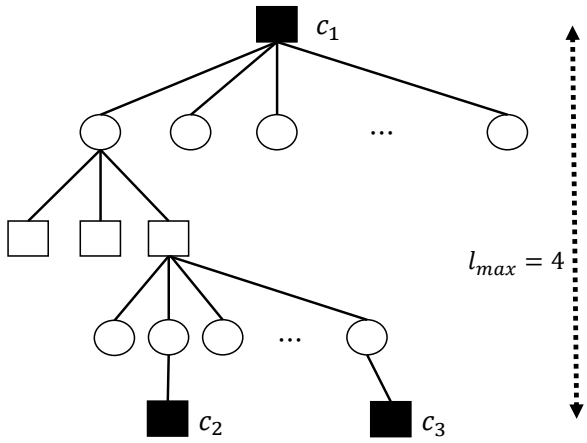

Figure 3.8: Description of unreliable path search by tree spanning.

small trapping set of the LDPC codes.

Let $\mathcal{P} = (c, v_1, c_1, \cdots, v_m, c')$ be a path between two unsatisfied check nodes $\{c, c'\} \in \mathcal{C}_0$. The score of $\mathcal{P}$, $\mathcal{S}(\mathcal{P})$ is defined as

$$\mathcal{S}(\mathcal{P}) = \sum_{i=1}^{m} |r_{v_i}|$$

where m is the number of the variable nodes in the path $\mathcal{P}$ and r_{v_i} is the channel LLR value of the variable node v_i. Then, the scores $\mathcal{S}(v_1)$ and $\mathcal{S}(v_m)$ of the two variable nodes $\{v_1, v_m\}$ in $\mathcal{N}(\mathcal{C}_0)$ are set to the score of the path $\mathcal{S}(\mathcal{P})$. Note that we only assign scores of the variable nodes directly connected to unsatisfied check nodes in $\mathcal{C}_0$. It is possible that some of the variable nodes in $\mathcal{N}(\mathcal{C}_0)$ are included in several different paths. In that case, the lowest score is assigned.

The unreliable path search for all pairs of the unsatisfied check nodes in $\mathcal{C}_0$ continues until we find all paths of length $l \leq l_{max}$, where l_{max} is the predetermined maximum path length. It is obvious that l_{max} is an important factor for the decoding

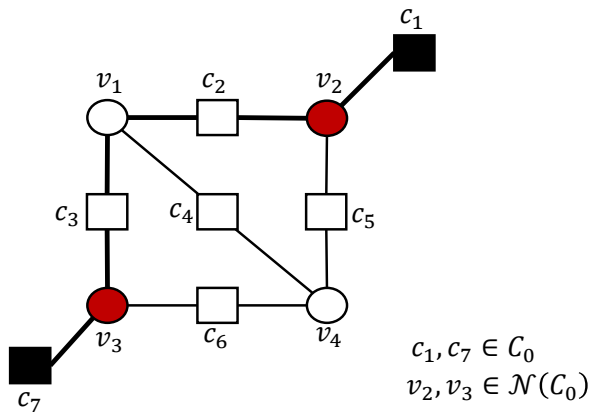

Figure 3.9: Description of unreliable path search between two unsatisfied check nodes in (4,2) trapping set.

complexity of the second-stage decoding. Based on the simulation, we found that six is enough for the LDPC codes of code rates 1/2 and 3/4, and four for LDPC code of code rate 10/11. For practical applications, we perform tree spanning from each of unsatisfied check nodes in $\mathcal{C}_0$ up to depth l_{max} to find all paths starting from each unsatisfied check node in $\mathcal{C}_0$ as shown in Figure 3.8.

Example 3.1 *In Figure. 3.9, let* $\{r_{v_1}, r_{v_2}, r_{v_3}, r_{v_4}\} = \{0.2, -0.5, 0.7, 1.1\}$. *For two unsatisfied check nodes* $c_1, c_7 \in \mathcal{C}_0$, *there are two paths of length 6, that is,* $\mathcal{P}_1 = \{c_1, v_2, c_2, v_1, c_3, v_3, c_7\}$ *and* $\mathcal{P}_2 = \{c_1, v_2, c_5, v_4, c_6, v_3, c_7\}$. *Since* $\mathcal{S}(\mathcal{P}_1) = 1.4$ *and* $\mathcal{S}(\mathcal{P}_2) = 2.3$, $\mathcal{S}(v_2)$ *and* $\mathcal{S}(v_3)$ *are set to 1.4. Note that we only assign scores to* v_2 *and* v_3, *but not* v_1.

We denote a set $\mathbf{V}_{sel}$ as the set of the variable nodes with assigned scores by the unreliable path search algorithm, and S_{max} as the maximum score of the variable nodes in $\mathbf{V}_{sel}$. It is possible that some of the variable nodes in $\mathcal{N}(\mathcal{C}_0)$ are not included

in any path. The scores of these variable nodes in $\mathcal{N}(\mathcal{C}_0)\backslash\mathbf{V}_{sel}$ are assigned by the summation of S_{max} and their channel LLRs, so that they are selected after all variable nodes in $\mathbf{V}_{sel}$ are selected.

Algorithm 3.1 Scoring of $\mathcal{N}(\mathcal{C}_0)$ by unreliable path search algorithm

Input: $r_v, l_{max}, \mathcal{C}_0$

Initialization: $\mathbf{V}_{sel} \leftarrow \emptyset$, $\mathcal{S}(v) = \infty$ for all $v \in \mathcal{N}(\mathcal{C}_0)$, $S_{max} = 0$

for each $c \in \mathcal{C}_0$ **do**
  Build the spanning tree $\mathcal{T}$ from c to depth l_{max}
  Find all $c' \in \mathcal{C}_0$, $c' \neq c$, in $\mathcal{T}$
  for each c', which generates a path $\mathcal{P} = (c, v_1, c_1, \cdots, v_m, c')$ **do**
    $\mathcal{S}(\mathcal{P}) = \sum_{i=1}^{m} |r_{v_i}|$
    for each $v \in \{v_1, v_m\}$ **do**
      $\mathbf{V}_{sel} = \mathbf{V}_{sel} \cup \{v\}$
      if $\mathcal{S}(v) > \mathcal{S}(\mathcal{P})$ **then**
        $\mathcal{S}(v) \leftarrow \mathcal{S}(\mathcal{P})$
      end if
    end for
  end for
end for
for each $v \in \mathcal{N}(\mathcal{C}_0)$ **do**
  if $\mathcal{S}(v) > S_{max}$ **then**
    $S_{max} \leftarrow \mathcal{S}(v)$
  end if
end for
for each $v \in \mathcal{N}(\mathcal{C}_0) \backslash \mathbf{V}_{sel}$ **do**
  $\mathcal{S}(v) \leftarrow S_{max} + |r_v|$
end for

3.3.2 LLR Manipulation and Re-decoding with the Second Early Stopping Criterion

Let $r_{v,out}$ denote the output LLR value of the variable node v at the last iteration of the first-stage decoding. The channel LLRs r_v of all variable nodes and the output LLRs $r_{v,out}$ of the variable nodes in $\mathcal{N}(\mathcal{C}_0)$ are stored for the re-decoding trials. The variable nodes in the candidate set $\mathcal{N}(\mathcal{C}_0)$ are sorted in ascending order by their scores, denoted by $\widetilde{\mathcal{N}}(\mathcal{C}_0)$. In the case of two variable nodes with the same score, the one with lower absolute value of the channel LLR is selected first.

For each re-decoding, the LLR value of the first variable node v in $\widetilde{\mathcal{N}}(\mathcal{C}_0)$ is manipulated as

$$r'_v = -\mathrm{sgn}(r_{v,out}) \cdot \eta$$

where $\mathrm{sgn}(\cdot)$ denotes the sign function, i.e., $\mathrm{sgn}(\cdot) = +1$ or -1, and η denotes the predetermined maximum magnitude of LLR value allowed in the BP decoder. The LLR values of the other variable nodes except v are set to their channel LLRs. Then, the conventional BP decoding with manipulated LLR of the first variable node in $\widetilde{\mathcal{N}}(\mathcal{C}_0)$ is performed for each re-decoding. This manipulation of LLR value may perturb the second-stage decoder to the correct direction and we expect the successful decoding with high probability. Note that when the re-decoding fails, the selected variable node v is removed from the set $\widetilde{\mathcal{N}}(\mathcal{C}_0)$ and the channel LLR of the variable node v is restored in r_v for the next re-decoding trial.

The LLR manipulation and the re-decoding are repeated for the allowed number of re-decoding trials or until the re-decoding is successful. This repetition may cause unacceptable decoding delay and complexity. In order to reduce the average number

of iterations for re-decoding, the second early stopping criterion is proposed, which stops the re-decoding and move to the next re-decoding trial when a sign change of the manipulated LLR value of the selected variable node occurs. Since we oppositely maximize the LLR value of the selected variable node, the sign of the manipulated LLR is hard to change unless the selected variable node is an erroneous variable node, especially in the error-floor region. This second early stopping criterion only requires the information of the previous sign of LLR value of the selected variable node, which makes it simple to implement.

Chapter 4

Parallel Unreliable Path Search Algorithm

Although the unreliable path search algorithm in Algorithm 3.1 works well in the error-floor region, we can further improve the previous unreliable path search algorithm. In Algorithm 3.1, we search for all paths between any two unsatisfied check nodes in $\mathcal{C}_0$ for length $l \leq l_{max}$. For the tree spanning, the additional delay is proportional to $|\mathcal{C}_0|$, which should be improved for some applications. Thus, in order to reduce the decoding delay, we propose an efficient method for the unreliable path search algorithm based on the parallel message-passing algorithm in LDPC decoding, called parallel unreliable path search algorithm.

4.1 Description of Parallel Unreliable Path Search Algorithm

In the unreliable path search algorithm, we search for all paths between any two unsatisfied check nodes in $\mathcal{C}_0$ for length $l \leq l_{max}$. In practice, we select one of the unsatisfied check node c in $\mathcal{C}_0$, and span a tree from the selected unsatisfied check node to the length l_{max}. In the spanned tree, each unsatisfied check nodes c' found in the tree,

where $c' \neq c$, creates a path between the two unsatisfied check nodes c and c'. Thus, we are able to assign scores to the variable nodes which are directly connected to the two unsatisfied check nodes c and c' in the path. Although the tree spanning does not involve any complex calculation, it can still cause the decoding delay. Moreover, since we search for all paths of length less than or equal to l_{max} in Algorithm 3.1, scores of some of the variable nodes are computed more than once by several different paths while the variable nodes only store the minimum score. Note that we assign scores to the two variable nodes connected to the unsatisfied check nodes in the path but the path itself is not needed in the second-stage decoding. Thus, the unreliable path search algorithm can be improved by using the concept of the message-passing algorithm of iterative decoders. The proposed search algorithm, namely parallel unreliable path search algorithm, is described as follows:

i) Each variable node $v \in \mathcal{N}(\mathcal{C}_0)$ sends its own absolute value of channel LLR $|r_v|$ as the score to all connected check nodes except the unsatisfied check node it came from.

ii) Each check node which receives messages from the neighboring variable nodes in the previous step passes the lowest score to all connected variable nodes except the variable node it came from. Since the path that assigns the lowest score to a variable node is only needed, the other paths are discarded.

iii) Each variable node which receives any message from the neighboring check nodes in the previous step checks if it is connected to any of unsatisfied check nodes in $\mathcal{C}_0$. If connected, the score of the variable node v is assigned as the summation of the minimum received score and $|r_v|$. Otherwise, the variable nodes send the

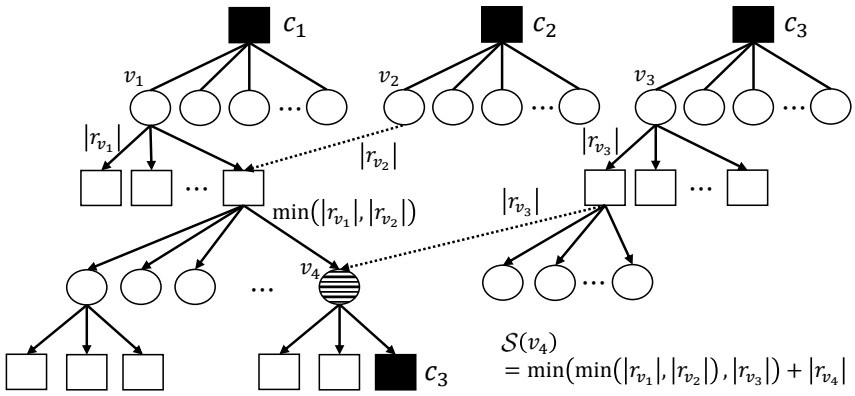

Figure 4.1: Description of parallel unreliable path search algorithm.

score of the summation of its absolute value of the channel LLR and the minimum received score to the neighboring check nodes except the one that came from the check node to be sent.

iv) The same procedure for variable and check nodes continues until l_{max} is reached.

Figure. 4.1 describes the proposed parallel unreliable path search algorithm. An important property for the message propagations of the proposed algorithm is that each message is generated by using the extrinsic information as the message-passing algorithm of LDPC codes. For the update of the check nodes, the outgoing message through an edge is determined by the minimum incoming message except the message coming from the edge. The same procedure is performed for the update of the variable nodes but the outgoing messages from the variable nodes are added by the absolute value of the channel LLR of the variable node. Similarly, the score of a variable node in $\mathcal{N}(\mathcal{C}_0)$ is assigned by the summation of the minimum incoming message and its channel LLR. By using the extrinsic information, the two variable nodes connected

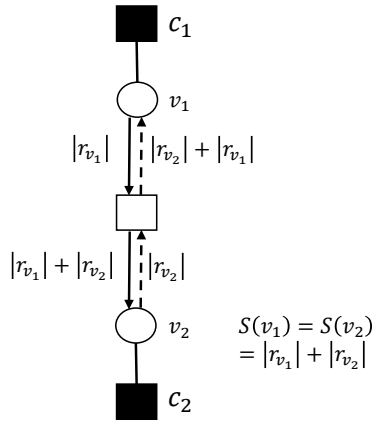

Figure 4.2: Scoring of variable nodes by message-passing with extrinsic information.

to the two unsatisfied check nodes in $\mathcal{C}_0$ can take the same score if the path has the minimum score. Fig 4.2 shows how the two variable nodes in a path are assigned to the same score. Since the message with minimum score generated from one side does not affect the message from the other side, the two variable nodes in a path can have the same score as long as the path has the minimum score.

When the girth g of the target LDPC code is $g \leq l_{max}$, a cycle from an unsatisfied check node to itself can be found in the proposed parallel unreliable path search algorithm. Although the performance degradation by the cycle paths is negligible, we can avoid the cycle paths by tagging the scores with the indices of their starting unsatisfied check nodes. Without any further calculation, the tagged index of the starting unsatisfied check node is compared with that of the unsatisfied check nodes at destination. The score of the variable nodes in $\mathcal{N}(\mathcal{C}_0)$ is assigned when the searched path is not a cycle. Note that the cycle path with lowest score can possibly eliminate the non-cycle path with lowest score. However, the performance degradation is negligible because

the length of closed path is larger than or equal to g and the two variable nodes in $\mathcal{N}(\mathcal{C}_0)$ in the closed path may be re-scored by other paths.

4.2 Scoring by Parallel Unreliable Path Search Algorithm

Parallel unreliable path search algorithm can replace the unreliable path search algorithm in Algorithm 3.1. Note that the scores of the variable nodes $\mathcal{N}(\mathcal{C}_0)\backslash\mathbf{V}_{sel}$ are assigned by the summation of S_{max} and their channel LLRs. The detailed scoring by parallel unreliable path search algorithm is described in Algorithm 4.2.

Algorithm 4.2 Scoring of $\mathcal{N}(\mathcal{C}_0)$ by parallel unreliable path search algorithm

Input: $r_v, l_{max}, \mathcal{C}_0$

Initialization: $\mathbf{V}_{sel} \leftarrow \emptyset, l \leftarrow 2, \mathcal{S}(v) = \infty$ for all $v \in \mathcal{N}(\mathcal{C}_0)$

1: **for** each $c \in \mathcal{C}_0$ **do**

2: $m_{c \to v} \leftarrow 0$ (initialization)

3: **for** each $v \in \mathcal{N}(c)$ **do**

4: **if** v is connected to any $c' \in \mathcal{C}_0, c' \neq c$ **then**

5: $\mathbf{V}_{sel} = \mathbf{V}_{sel} \cup \{v\}$

6: $\mathcal{S}(v) \leftarrow |r_v|$

7: **else**

8: $m_{v \to c', c' \neq c} \leftarrow |r_v|$

9: **end if**

10: **end for**

11: **end for**

12: **while** $l < l_{max}$ **do**

13: **for** each c, receiving any message from $\mathcal{N}(c)$ **do**

14: $m_{c \to v} \leftarrow \min(\left|m_{v' \to c, v' \neq v}\right|$, for any $\left|m_{v' \to c, v' \neq v}\right| \neq 0)$

15: **end for**

Algorithm Scoring of $\mathcal{N}(\mathcal{C}_0)$ by parallel unreliable path search algorithm (cont')

for each v, receiving any message from $\mathcal{N}(v)$ **do**

  if v is connected to any $c \in \mathcal{C}_0$ **then**

    $\mathbf{V}_{sel} = \mathbf{V}_{sel} \cup \{v\}$

    if $\mathcal{S}(v) > \min(\left|m_{c' \to v, c' \neq c}\right|$, for any $\left|m_{c' \to v, c' \neq c}\right| \neq 0) + |r_v|$ **then**

      $\mathcal{S}(v) \leftarrow \min(\left|m_{c' \to v, c' \neq c}\right|$, for any $\left|m_{c' \to v, c' \neq c}\right| \neq 0) + |r_v|$

    end if

  else

    $m_{v \to c} \leftarrow \min(\left|m_{c' \to v, c' \neq c}\right|$, for any $\left|m_{c' \to v, c' \neq c}\right| \neq 0) + |r_v|$

  end if

end for

$l \leftarrow l + 2$

end while

Chapter 5

Analysis of the Unreliable Path Search Algorithm

5.1 Validity of the Unreliable Path Search Algorithm

In Figure. 5.1, 5.2, and 5.3, the validity of the proposed scoring algorithm by the unreliable path search is described. In this context, *hit* is defined as the successful selection of the erroneous variable node in the candidate set $\mathcal{N}(\mathcal{C}_0)$. For three different LDPC codes with different code rates, the IEEE 802.16e irregular LDPC codes with $R = 1/2,\ 3/4$ and a high-rate regular QC LDPC code with $R = 10/11$ are simulated in the error-floor regions. To show that the proposed scoring algorithm selects the erroneous variable nodes faster than the conventional backtracking scheme in [40], more than a thousand unsuccessfully decoded codewords by the first-stage decoder are collected in the error-floor region for each LDPC code. For unsuccessfully decoded codewords, we apply both selection algorithms of variable nodes and check whether an erroneous variable node is correctly selected for each re-decoding trial.

The cumulative *hit* ratio of the vertical lines in Figures. 5.1, 5.2, and 5.3 denotes

the cumulative ratio of the number of unsuccessfully decoded codewords of which at least one erroneous variable node is successfully selected. For the IEEE 802.16e LDPC code with $R = 1/2$, the proposed scoring algorithm successfully selects an erroneous variable node at the first selection for about 40% of the unsuccessfully decoded codewords. Although the cumulative hit ratio of the proposed algorithm turns out to be slightly degraded after 6 selection trials, it is important to select erroneous variable nodes during the first few re-decoding trials because the computational complexity becomes implausible for a larger number of re-decoding trials. In addition, we can see that the proposed selection algorithm significantly outperforms for the LDPC codes of code rates 3/4 and 10/11. The higher the code rate of LDPC code is, the more variable nodes are connected to each check node, which generally makes it difficult to select the erroneous variable nodes in the conventional method because too many variable nodes are kept in the candidate set. Note that from the numerical analysis, the maximum length l_{max} is set to six for the LDPC codes of code rates 1/2 and 3/4, and four for the LDPC code of code rate 10/11.

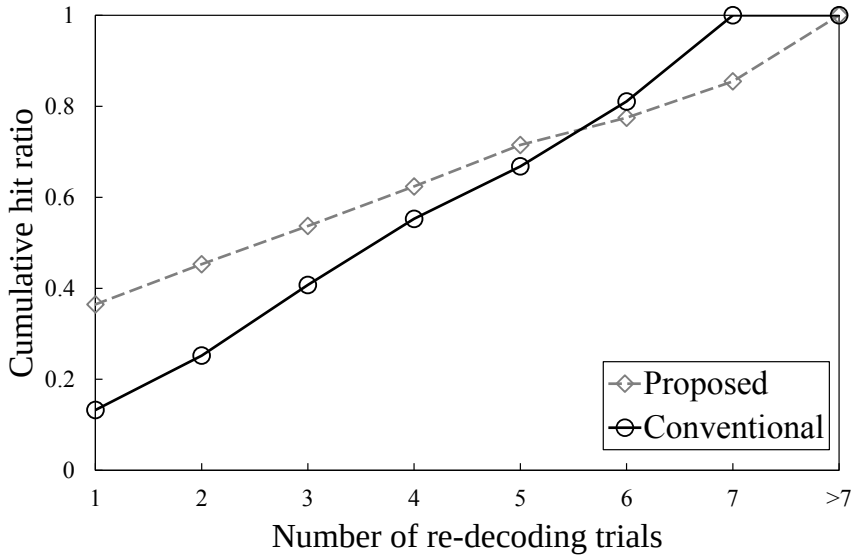

Figure 5.1: The cumulative hit ratio versus the number of re-decoding trials for IEEE 802.16e with $R = 1/2$.

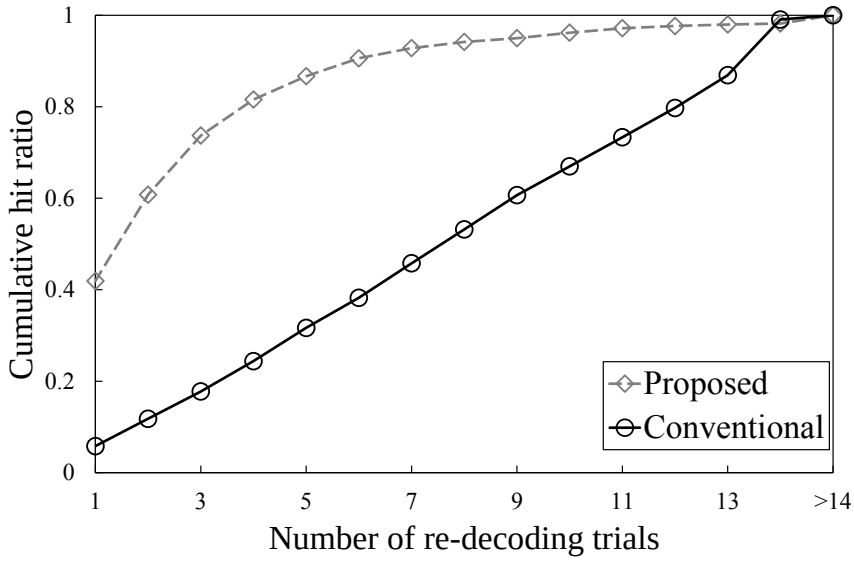

Figure 5.2: The cumulative hit ratio versus the number of re-decoding trials for IEEE 802.16e with $R = 3/4$.

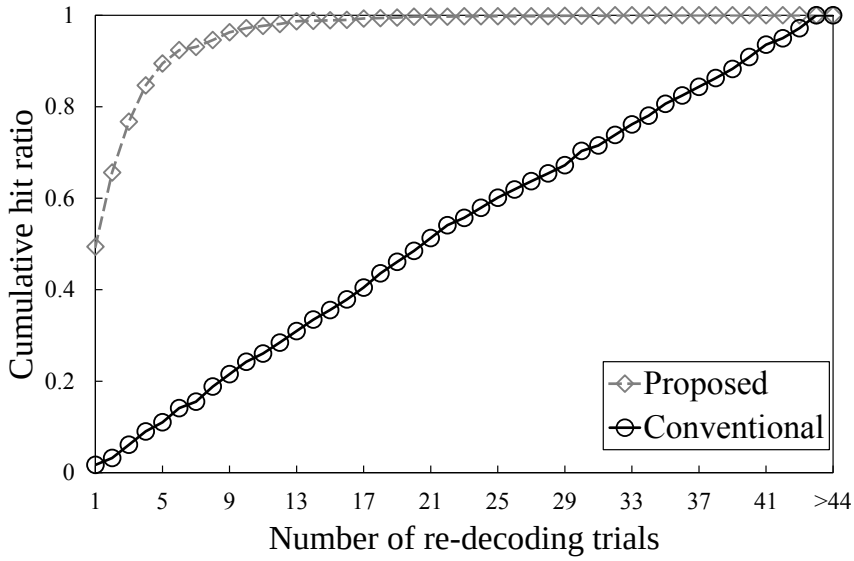

Figure 5.3: The cumulative hit ratio versus the number of re-decoding trials for QC LDPC code with $R = 10/11$.

5.2 Complexity Analysis of the Unreliable Path Search Algorithm

In the proposed decoding scheme, the additional complexity is caused by; i) the observation of the number of unsatisfied check nodes for the first-stage decoding, ii) scoring of the variable nodes by unreliable path search algorithm and finding the candidate with the lowest score at each re-decoding trial for the second-stage decoding. For i), the additional complexity is negligible because it only involves integer comparisons for τ times, where τ is usually three. Also, we can neglect the additional complexity to find the candidate with the lowest score for ii) because $\mathcal{N}(\mathcal{C}_0)$ is usually a small set and the number of re-decoding trials is restricted. Thus, we estimate the additional complexity for scoring of the variable nodes by the unreliable path search algorithm in ii), where the parallel unreliable path search algorithm is proposed for practical usage.

The extra computations by the parallel unreliable path search algorithm include: floating-point comparisons at the selected check nodes, and floating-point comparisons and two more additions at the selected variable nodes. Note that no floating-point multiplications are required in the proposed algorithm. Since the conventional BP decoding involves approximately $6nt$ (where t denotes the column weight of the parity-check matrix) floating-point multiplications per iteration, the additional complexity of the parallel unreliable path search algorithm is much lower than one conventional BP iteration [15]. On the other hand, suppose that the min-sum decoding algorithm is used for iterative decoder. The update of each check node in the min-sum decoding requires the determination of two incoming LLRs with lowest magnitude as well as the signs of the outgoing messages, which is the same in the unreliable path search algorithm with-

Table 5.1: Number of computations for parallel unreliable path search in the error-floor.

	Min-sum decoding iteration	Parallel unreliable path search
R=1/2	23040	4676
R=3/4	25632	15250
R=10/11	57664	22650

out determining the signs of the outgoing messages. The computational complexity of determining two messages with lowest magnitude is known to be $d_c + \lceil \log d_c \rceil - 2$ floating point comparisons [44]. For the update of each variable node, the min-sum decoding algorithm performs $2d_v$ additions (subtractions) while the unreliable path search algorithm requires $d_v + \lceil \log d_v \rceil - 2$ comparisons and two more additions. Considering that the complexity of addition and comparison is nearly the same, the proposed parallel unreliable path search algorithm requires less computational complexity compared to the min-sum algorithm. Since no floating-point computation is required for initialization ($l = 1, 2$), the additional complexity of the unreliable path search algorithm is restricted to $\frac{l_{max}}{2} - 1$ iterations of the min-sum decoding algorithm. Further, the check and variable node updates are only performed for selected nodes at each step in the proposed algorithm.

Table 5.1 shows the average number of computations (which include addition, subtraction, and comparison) for the parallel unreliable path search algorithm in the error-floor region, as well as those for each iteration of min-sum decoding. Although the maximum path length l_{max} is set to six for the rate $R = 1/2$ and $R = 3/4$, the required computations for the parallel unreliable path search are less than those of one min-sum decoding iteration because not all variable and check nodes are involved for calculation of the proposed algorithm. Thus, the proposed decoding scheme can be

performed with a marginal computational overhead.

Chapter 6

Simulation Results

Numerical analysis is carried out for the various two-stage decoding schemes of three different LDPC codes: the LDPC codes of code rates 1/2 and 3/4 with $n = 2304$ in the IEEE 802.16e standard, and regular QC LDPC code of code rate 10/11 with $n = 4664$. Three different decoding schemes are simulated for comparison with the proposed scheme; i) Conventional (single stage) BP decoding, ii) the backtracking decoding in [40], and iii) the multi-stage decoding in [41]. For the proposed decoding scheme, the parallel unreliable path search algorithm is applied without tagging for the second-stage decoding. Although IEEE 802.16e LDPC codes of code rates 1/2 and 3/4 have girth $g = 6$ and l_{max} is set to six, the performance degradation by existence of cycle paths is negligible. BP decoding with double precision is considered for all decoding schemes and the maximum number of iterations for the first-stage decoding is set to 50. For the second-stage decoding, the maximum number of iterations for re-decoding trials is set to 20 for the backtracking and the proposed decoding scheme, and 100 for the multi-stage decoding in [41]. The maximum number of re-decoding

Table 6.1: Average number of iterations for two-stage decodings in the error-floor.

	Backtracking decoding in [40]	Multi-stage decoding in [41]	Proposed decoding
R=1/2	144.54	59.28	68.49
R=3/4	194.17	58.52	47.93
R=10/11	245.47	60.02	40.85

trials for the scheme in [40] and the proposed scheme is set to 20.

Figures. 6.1, 6.2, and 6.3 show the FER performances of the four different decoding schemes in the error-floor region and the average numbers of iterations of two-stage decoding schemes are listed in Table 6.1. Note that the average number of iterations is calculated only when the two-stage decoding scheme is successful. In Figure. 6.1, the multi-stage decoding in [41] shows performance improvement with marginal additional iterations, but it was hard to decide the parameter N, which is how many variable nodes are to be erased, before decoding process. The FER performances of the backtracking decoding scheme and the proposed decoding scheme show similar results but the average number of iterations has been significantly reduced due to the first and the second early stopping criteria as in Table 6.1. For the LDPC codes of code rates 3/4 and 10/11 in Figure. 6.2 and Figure. 6.3, the simulation results show that the proposed decoding scheme outperforms the other decoding schemes. In Figure. 6.2, the backtracking decoding in [40] still outperforms multi-stage decoding in [41] but the performance improvement of the backtracking decoding is limited for the LDPC codes of code rate 10/11 in Figure. 6.3 due to the limitation of the number of re-decoding trials. Although the proposed decoding scheme is also based on re-decoding, the unreliable path search algorithm efficiently finds the erroneous variable nodes in

small trapping sets, which results in the best FER performance.

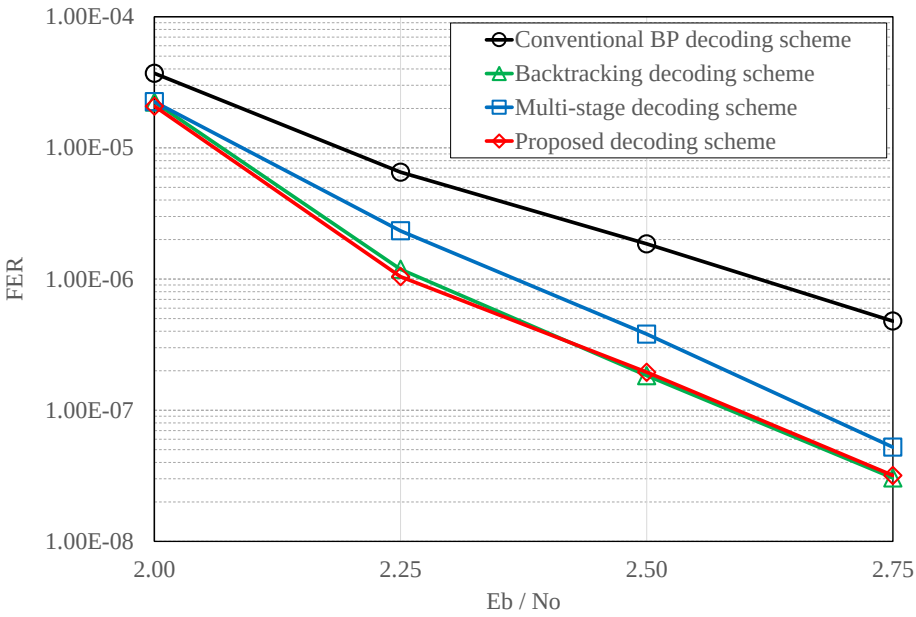

Figure 6.1: FER comparison of the proposed two-stage decoding scheme with the conventional schemes for IEEE 802.16e with $R = 1/2$.

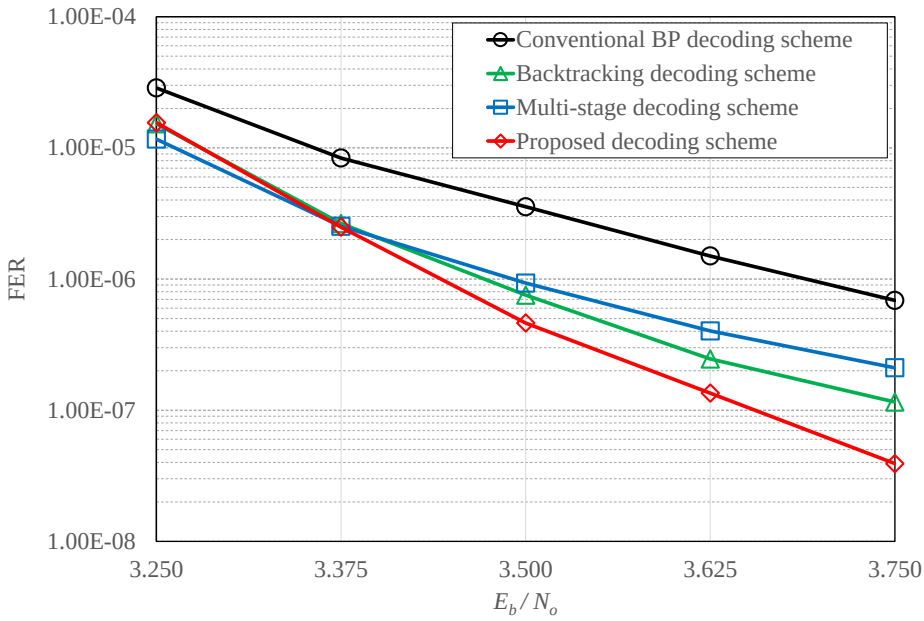

Figure 6.2: FER comparison of the proposed two-stage decoding scheme with the conventional schemes for IEEE 802.16e with $R = 3/4$.

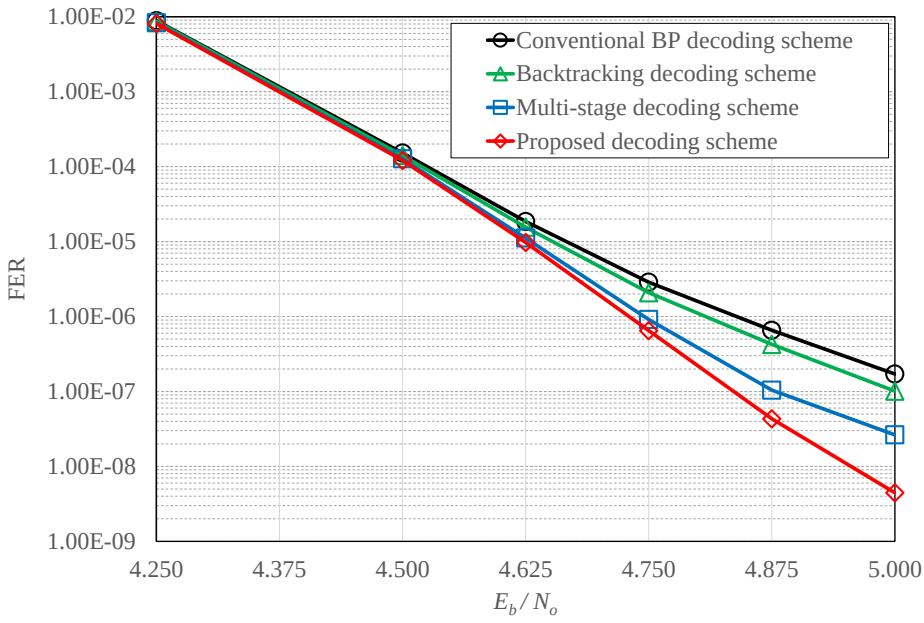

Figure 6.3: FER comparison of the proposed two-stage decoding scheme with the conventional schemes for QC LDPC code with $R = 10/11$.

Chapter 7

Conclusions

In this dissertation, research on the low-complexity decoding schemes and their applications are presented.

In Chapter 2, LDPC codes are briefly overviewed. Basic concepts, decoding, and analysis of the LDPC codes are presented. The block type QC LDPC codes are also introduced

In Chapter 3, a new two-stage decoding scheme with unreliable path search is proposed. The proposed scheme is based on the early stopping criteria and re-decoding scheme by unreliable path search algorithm. Section 3.1 describes overall process of the proposed decoding scheme. Since the proposed decoding scheme consists of two stages, the procedure of each stage is presented in Sections 3.2 and 3.3, respectively. In Section 3.3.1, unreliable path search algorithm is introduced to find the variable nodes which may be included in small trapping sets. Some of the variable nodes found in the first-stage decoding are set to be the candidates. With the information of unreliable variable nodes, the LLR manipulation of the selected variable node and re-decoding

process are described in Section 3.3.2. A simple and effective stopping criterion for the second-stage decoding is also proposed.

In Chapter 4, a low-complexity algorithm for unreliable path search algorithm is designed, namely parallel unreliable path search algorithm. By the parallel unreliable path search algorithm, the scoring of variable nodes in the candidate set can be performed simultaneously in the manner of message-passing decoder of LDPC codes. The procedure of the parallel unreliable path search algorithm is described in Section 4.1 and the scoring of variable nodes by parallel unreliable path search algorithm is also described in Section 4.2.

In Chapter 5, the proposed unreliable path search algorithm is analyzed. The validity of the unreliable path search algorithm is described in Section 5.1, which shows the proposed algorithm helps to select the unreliable variable nodes more quickly than random selection. In Section 5.2, the complexity of the proposed decoding scheme is analyzed. The overall complexity of the proposed decoding scheme is mainly based on the unreliable path search algorithm, but it is shown that the algorithm only requires less additional computational complexity than that of the one iteration for an iterative decoder. The performance of the proposed decoding scheme is verified via simulation in Chapter 6.

Bibliography

[1] S. Haykin, *Communication Systems*, 4th Ed. NY, NY, USA: John Wiley & Sons, Inc., 2001.

[2] J. G. Proakis and M. Salehi, *Digital Communications*, 5th Ed. NY, NY, USA: MacGraw-Hill, 2008.

[3] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[4] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ, USA: Prentice Hall, 1995.

[5] R. E. Blahut, *Algebraic Codes for Data Transmission*. NY, NY, USA: Cambridge University Press, 2003.

[6] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. NY, NY, USA: Cambridge University Press, 2003.

[7] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. NY, NY, USA: Cambridge University Press, 2003.

[8] S. Lin and D. J. Costello, Jr., *Error Control Coding*, 2nd Ed. Upper Saddle River, NJ, USA: Prentice Hall, 2004.

[9] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*. NY, NY, USA: Cambridge University Press, 2008.

[10] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, Jul., 1948.

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

[12] G. D. Forney, Jr., *Concatenated Codes*. Cambridge, MA, USA: MIT Press, 1966.

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

[14] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645-1646, Aug. 1996.

[15] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 399-431, Mar. 1999.

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

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

[18] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585-595, Feb. 2001.

[19] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann, 1988.

[20] S.-Y. Chung, *On the construction of some capacity-approaching coding schemes*. Ph.D. dissertation, MIT, Cambridge, MA, USA: MIT Press, 2000.

[21] IEEE, "IEEE standard for local and metropolitan area networks. Part 16: Air interface for broadband wireless access systems," *IEEE Std 802.16-2009*, May 2009.

[22] IEEE, "IEEE standard for information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Amendment 5: Enhancements for higher throughput," *IEEE Std 802.11n-2009*, Oct. 2009.

[23] IEEE, "IEEE standard for information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. Amend-

ment 1: Physical layer and management parameters for 10 Gb/s operation, type 10GBASE-T," *IEEE Std 802.3an-2006*, Sep. 2006.

[24] ITU-T, "Unified high-speed wire-line based home networking transceivers foundation," *Recommendation ITU-T G.9960*, Oct. 2009.

[25] ETSI, "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)," *EN 302 307 v1.2.1*, Aug. 2009.

[26] ETSI, "Digital video broadcasting (DVB); Frame structure channel coding and modulation for a second generation digital transmission system for cable systems (DVB-C2)," *EN 302 769 v1.2.1*, Apr. 2011.

[27] ETSI, "Digital video broadcasting (DVB); Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2)," *EN 302 755 v1.2.1*, Feb. 2011.

[28] 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.

[29] X. Zhang and S. Chen, "A two-stage decoding algorithm to lower the error-floors for LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 4, pp. 517–520, Apr. 2015.

[30] R. Smarandache and P. O. Vontobel, "Quasi-cyclic LDPC codes: Influence of proto- and Tanner-graph structure on minimum Hamming distance upper bounds," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 585-607, Feb. 2012.

[31] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protograph," *IPN Progress Report 42-154, JPL*, Aug. 2003.

[32] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788-1793, Aug. 2004.

[33] S. J. Johnson and S. R. Weller, "Constraining LDPC degree distributions for improved error floor performance," *IEEE Commun. Lett.*, vol. 10, no. 2, pp. 103–105, Feb. 2006.

[34] Z. He, P. Fortier, and S. Roy, "A class of irregular LDPC codes with low error floor and low encoding complexity," *IEEE Commun. Lett.*, vol. 10, no. 5, pp. 372–374, May 2006.

[35] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, "Design of finite-length irregular protograph codes with low error floors over the binary-input AWGN channel using cyclic liftings," *IEEE Trans Commun.*, vol. 60, no. 4, pp. 902–907, Apr. 2012.

[36] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, "Lowering the error floor of LDPC codes using cyclic liftings," *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2213–2224, Apr. 2011.

[37] S. Lander and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. Int. Conf. Wireless Netw., Commun. Mobile Comput.*, Jun. 2005, pp. 630–635.

[38] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans Commun.*, vol. 58, no. 6, pp. 1663–1673, Jun. 2009.

[39] X. Dongliang and L. Jianhua, "A perturbation method for decoding LDPC concatenated with CRC," in *Proc. Int. Conf. Wireless Commun. Netw. Conf.*, Mar. 2007, pp. 668–672.

[40] J. Kang, Q. Huang, S. Lin, and K. A. Ghaffar, "An iterative decoding algorithm with backtracking to lower the error-floors of LDPC codes," *IEEE Trans. Commun.*, vol. 59, no. 1, pp. 64–73, Jan. 2011.

[41] B. Shin, H. Park, J.-S. No, and H. Chung, "Multi-stage decoding scheme with post-processing for LDPC codes to lower the error floors," *IEICE Trans. Commun.*, vol. E94-B, no. 8, pp. 2375–2377, Aug. 2011.

[42] S. Tolouei and A. H. Banihashemi, "Lowering the error floor of LDPC codes using multi-step quantization," *IEEE Commun. Lett.*, vol. 18, no. 1, pp. 86–89, Jan. 2014.

[43] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright, ''Investigation of error floors of structured low-density parity-check codes by hardware emulation," in *Proc. of IEEE GLOBECOM*, Nov. 2006.

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

초 록

본 논문은 저밀도 패리티 체크 부호를 위한 저복잡도 복호 기법에 대한 것이며, 다음의 연구결과를 포함한다.

- 저밀도 패리티 체크 부호를 위한 이단계 복호 기법
  - 저밀도 패리티 체크 부호를 위한 새로운 중단 기법 제안
  - 비신뢰 경로 검색 기법에 의한 저밀도 패리티 체크 부호를 위한 새로운 복호 기법 제안
- 평행 비신뢰 경로 검색 기법
- 이단계 복호 기법의 분석
  - 이단계 복호 기법의 유효성과 복잡도 분석

본 논문에서는 저밀도 패리티 체크 부호의 오류 마루 현상을 줄이기 위한 새로운 이단계 복호 기법을 제안 한다. 제안된 복호 기법은 기존의 일반적인 신뢰 전파 복호 알고리즘을 일단계로, 변형된 변수노드의 우도비(LLR)를 이용한 재복호 단계를 이단계로 수행한다.

일단계복호에서는 기존 신뢰 전파 복호 알고리즘의 실패를 빠르게 검출하고 작은 트래핑 집합에 포함될 가능성이 있는 변수 노드를 검색할 수 있는 새로운 중단 기법이 제안 되었다. 이단계복호에서는 일단계복호에서 검색된 변수 노드들의 신뢰도를 확인하기 위해 제안된 비신뢰 경로 검색 기법을 이용하여 변수 노드들의 신뢰값을 계산한다. 비신뢰 경로 검색 기법은 오류 마루 현상의 원인인 작은 트래핑 집합에 포함될 가능성이 있는 변수 노드들을 우선적으로 검색하여 해당 변수 노드에 낮은 신뢰값을 할당한다. 이렇게 계산된 변수 노드들의 신뢰값에 따라 선택된 변수 노드의 우도비 값을 변형하여 차례로 재 복호를 시도함으로써 높은 확률로 복호에 성공한다.

제안된 비신뢰 경로 검색 기법을 더욱 낮은 복잡도로 구현하기 위하여 평행 비신뢰 경로 검색 기법이 또한 제안 되었다. LDPC부호의 메시지 전달 알고리즘을 기반으로 한 평행 비신뢰 경로 검색 기법은 추가적인 하드웨어 구현 없이 비신뢰 경로 검색 기법을 수행함으로써 효과적으로 복잡도와 지연시간을 낮출 수 있다.

마지막으로 제안된 이단계 복호 기법의 유효성과 복잡도가 분석 되었다. 비신뢰 경로 검색 기법은 효과적으로 작은 트래핑 집합에 포함된 변수 노드에 낮은 신뢰값을 할당함을 확인하였으며, 평행 비신뢰 경로 검색 기법은 낮은 추가 복잡도로도 성능 손실 없이 높은 복호 성능을 보임을 확인하였다.

주요어: 신뢰 전파 알고리즘, 오류 마루, 저밀도 패리티 체크 부호, 트래핑 집합, 이단계 복호 기법

학번: 2012-30215