CHALMERS
UNIVERSITY OF TECHNOLOGY

Coding for DNA-Based Storage

LDPC Code Optimization

Master's Thesis in Communication Engineering

HAN WANG

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's Thesis

Coding for DNA-Based Storage

LDPC Code Optimization

HAN WANG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Communication Systems Group
Chalmers University of Technology
Gothenburg, Sweden 2021

Coding for DNA-Based Storage
LDPC Code Optimization
HAN WANG

Supervisor: Alexandre Graell i Amat, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
Co-supervisor: Eirik Rosnes, Simula UiB, Bergen, Norway
Examiner: Alexandre Graell i Amat, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden

Master's Thesis EENX60
Department of Electrical Engineering
Communication Systems Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in $\LaTeX$
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Coding for DNA-Based Storage
LDPC Code Optimization
HAN WANG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

DNA-based storage is an advanced technique for solving the increasing demand for data storage by providing high storage capacity and remarkable longevity. Different processes involved in DNA-based storage introduce both synchronization errors (insertions and deletions) and substitution errors. Lots of literature have explored methods for correcting such errors. Among them, Davey and Mackay proposed a so-called watermark code for solving synchronization errors. They also introduced a concatenated coding scheme with a low-density parity-check (LDPC) code as an outer code and a watermark code as an inner code for correcting both synchronization and substitution errors simultaneously. Davey and Mackay considered an off-the-shelf LDPC code in their coding scheme. The LDPC code can be improved by adjusting the degree distribution based on the underlying channel.

In this thesis, we consider a concatenated coding scheme as in Davey and Mackay's paper to correct errors in a DNA-based storage channel. Besides, we calculate achievable information rates for DNA-based storage channels. Furthermore, to find the LDPC code best suitable for DNA-based storage channels and to approach the achievable information rates, an optimization mechanism based on density evolution and extrinsic-information-transfer charts is proposed in this thesis.

Keywords: DNA-based storage, coding theory, LDPC code, achievable information rate, insertion/deletion channel, synchronization error, degree optimization

Acknowledgements

First of all, I would like to express my sincere appreciation to my two supervisors, Alexandre Graell i Amat and Eirik Rosnes, for providing me an opportunity to experience scientific research on this interesting and enjoyable topic, as well as supporting me with patience and encouragement during the thesis.

I would also like to thank Issam Maarouf and Christian Häger for helping me understand the theories and giving me practical guidance about simulation, and to thank Anton Frigård for providing me helpful suggestions and a useful template near the end of the thesis.

Finally, I must express my gratitude to my parents for supporting me financially and spiritually during my years of studying in Sweden.

Han Wang, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

With the development of the Internet, more and more advanced techniques and technologies, such as Internet of Things (IoT) and cloud computing, are continuously being developed. Following the appearance of these advanced techniques, the demand for data storage is increasing exponentially. There will be 175 zettabytes (one zettabyte is equal to a trillion gigabytes) of data in the global datasphere by 2025 [2]. Current data storage techniques are not able to accommodate the increasing data storage demand [3]. Hence, searching for new techniques and technologies is an urgent challenge for data storage. Data storage in deoxyribonucleic acid (DNA) is one technology to accomplish this requirement. Having existed for millions of years, DNA has significant properties for data storage, such as high capacity storage and remarkable longevity [4,5], which indicate that much higher storage density and long durability can be offered by DNA-based storage.

DNA-based storage is the technique of saving data in DNA. The progress of translating digital data into DNA strands is shown in Figure 1.1. Unlike binary data (e.g., consisting of 0s and 1s, such as {...0011101001...}) stored in most digital electronics equipment, DNA strands are built from four different DNA bases (alphabet of A, T, C, G), which also can be referred to as nucleotides. Therefore, the first step is mapping (or encoding) digital data into the alphabet of DNA bases. The constructed alphabet will be used for building actual DNA strands during a process called synthesis, which is a robust biotechnological process for generating DNA sequences without pre-existing templates [4]. After that, the DNA strands are stored in gene pools and to be accessed later. As shown in Figure 1.2, in order to access the data in the gene pool, the first step is DNA sequencing. The exact nucleotides and their order in a DNA molecule is determined at this step [3]. The final step is demapping (or decoding) the determined DNA sequences back to digital data.

In this thesis, the biotechnological processes, including DNA synthesis, DNA storage medium and DNA sequencing are modeled as a special channel, called DNA-based storage channel, as shown in Figure 1.3 [4]. In the first step, data is typically synthesized into many short DNA strands. These DNA stands are replicated thousands of times in the DNA storage medium by a process called polymerase chain reaction (PCR) amplification. Therefore, multiple copies of each DNA strand are read through DNA sequencing. During these processes several errors can be introduced, which can be divided into three categories: substitutions, insertions, and

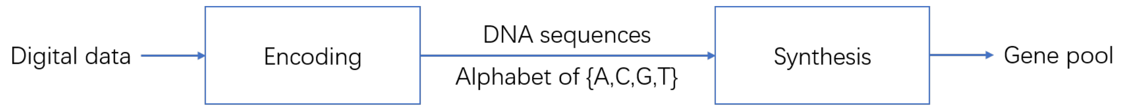


Figure 1.1: The process of translating digital data into DNA strands.

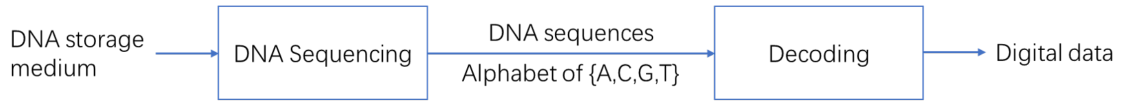


Figure 1.2: The process of extracting digital data from DNA strands.

deletions errors. A substitution error means that one DNA base is swapped by another base (e.g., {...ATCGGTC...} → {...ATCGATC...}), as shown as yellow parts in Figure 1.3. Insertion occurs when a new base is inserted into the original sequence (e.g., {...ATG...} → {...ACTG...}). Similarly, deletions occur when a base is deleted from the original sequence (e.g., {...ACGT...} → {...AGT...}). Insertion errors and deletion errors are called synchronization errors collectively. Thus at the receiver, the accessed data may have different length, as shown in Figure 1.3.

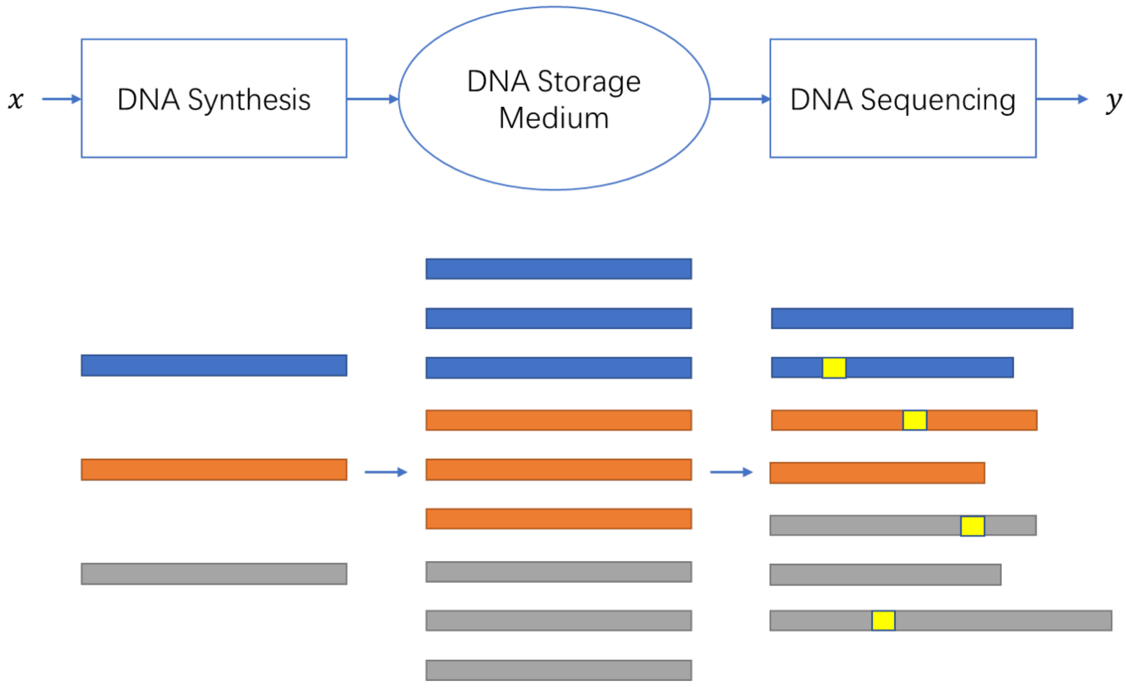


Figure 1.3: DNA-based storage channel.

Several papers propose building error-correction codes for DNA-based storage. Nevertheless, many of them only deal with one or two types of errors. Lenz *et al.* [6] proposed the construction of subset codes, which is a subset of the data set over a finite set $\mathcal{S}$ that contains all input sequences in an unordered fashion. Furthermore, constructions of multi-set codes are also proposed [7, 8]. The difference between

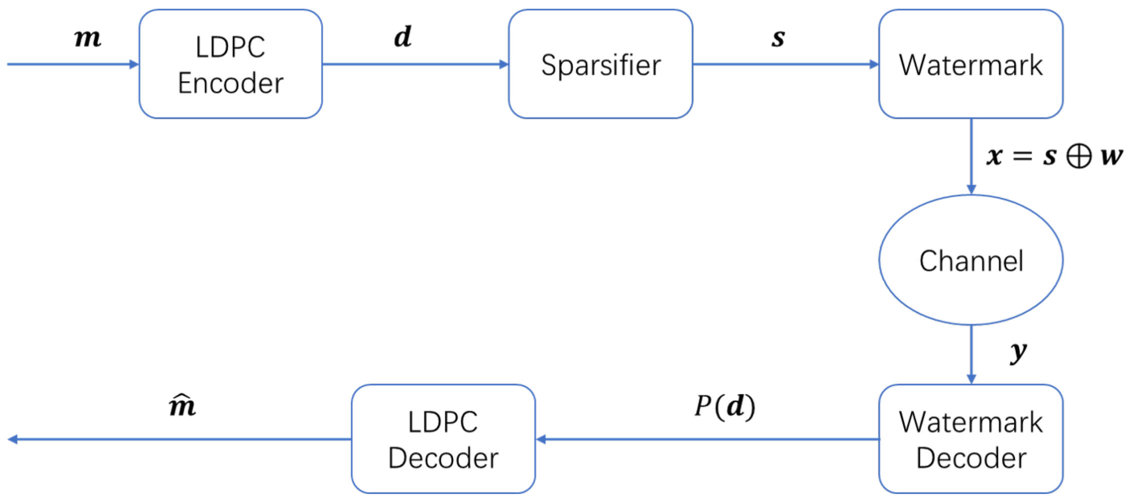


Figure 1.4: Coding scheme proposed in Davey and Mackay's paper [1].

subset codes and multi-set codes is that repetition of elements is allowed in multi-set codes. All these codes are algebraic codes. In addition, some literature also proposed coding schemes using modern codes with markers. Davey and Mackay [1] proposed a coding scheme dealing with substitution, insertion, and deletion errors. They introduced a concatenation of an inner code to deal with insertion and deletion errors and an outer code to solve substitution errors as shown in Figure 1.4. In their coding scheme, the message $\boldsymbol{m}$ is encoded into the LDPC codeword $\boldsymbol{d}$ by the LDPC encoder. Then the LDPC codeword $\boldsymbol{d}$ is sent to a sparsifier to transform each symbol d_i into a sparse binary vector $\boldsymbol{s}$. After that, the sparse vector $\boldsymbol{s}$ is added (modulo 2) with the watermark string $\boldsymbol{w}$, which is known to both transmitter and receiver, to form the sequence $\boldsymbol{x}$ to be transmitted. Then the transmitted bits in $\boldsymbol{x}$ enter into the channel, the received sequence $\boldsymbol{y}$ is obtained on the other side. The received bits in $\boldsymbol{y}$ are decoded successively by the watermark decoder and the LDPC decoder to recover message $\hat{\boldsymbol{m}}$. Some literature attempted to improve Davey and Mackay's code construction. For example, the algorithm utilized in Davey and Mackay's paper referred to as bit-level synchronization (BLS) has been improved by Briffa *et al.* [9], which introduced symbol level synchronization (SLS) for decoding.

1.2 Thesis Objective

This thesis proposes a concatenated coding scheme for DNA-based storage based on Davey and Mackay's coding scheme [1]. The inner code of this coding scheme can be either a convolution code or a watermark code. In contrast, the outer code is an LDPC code. In addition, achievable information rates are calculated for our channel in order to analyze the performance of the coding schemes. Furthermore, an optimization mechanism to optimize the LDPC code based on density evolution (DE) and extrinsic-information-transfer (EXIT) charts is introduced in this thesis. The aim of the optimization is to search for a good LDPC code, which will improve the performance of the coding scheme. The optimization mechanism is implemented and tested by simulations with or without inner-outer iterations.

1.3 Thesis Structure

The structure of the rest of this thesis is as follows. The basic theory of LDPC codes is introduced in Chapter 2. Chapter 3 introduces the channel model and coding for DNA-based storage. Chapter 4 introduces the coding scheme utilized in the thesis and calculates achievable information rates. The LDPC code optimization mechanism is presented in Chapter 5. Numerical results are given in Chapter 6. Finally, the thesis is concluded in Chapter 7.

2
Low-Density Parity-Check Codes

Invented by Gallager in 1960 [10] as a class of linear block codes, low-density parity-check (LDPC) codes are commonly used because of implementable decoders and near-capacity performance [11]. For simplicity, only binary LDPC codes will be considered in this chapter.

2.1 Basic Concepts

An LDPC code can be defined as a linear block code characterized by having an $m \times n$ parity-check matrix $\mathbf{H}$ with column weight g and row weight r. $\mathbf{H}$ is low density when $g \ll m$. An LDPC code can also be described by a graphical representation, called Tanner graph. A Tanner graph is a bipartite graph, whose nodes may be separated into two types, with edges connecting only nodes of different types [11]. In the Tanner graph for LDPC codes, the nodes can be sorted into two types, variable nodes and check nodes, which can are referred to as VNs and CNs, respectively. Corresponding to $\mathbf{H}$, there are m CNs and n VNs in the Tanner graph. Figure 2.1 shows an example of a parity-check matrix $\mathbf{H}$ and its Tanner graph representation. In this Tanner graph, CNs are represented by squares, while VNs are represented by circles. From this figure, it is easy to observe that VN i will be connected to CN j if there is a one in row j and column i in $\mathbf{H}$. If the row and the column weights are both constant, the code is a *regular* LDPC code. For example, the code in Figure 2.1 is a regular LDPC code since the weight is $g = 2$ for all columns (each VN has two outgoing edges), and weight is $r = 4$ for all rows (each CN has four outgoing edges). Otherwise, the code is an *irregular* LDPC code. In irregular LDPC codes, g and r vary between columns and rows (the numbers of connecting edges are different for different CNs and VNs).

The number of edge connections for each VN or CN is defined as the *degree*. In the example above, the degree of each VN is 2, and the degree of each CN is 4. Since in irregular LDPC codes g and r vary for each VN and CN, it is customary to represent the degrees by *degree distribution polynomials*, denoted by $\lambda(X)$ and $\rho(X)$, respectively. In the polynomial $\lambda(X)$ in Equation (2.1), d_v denotes the maximum VN degree and λ_d denotes the fraction of edges connected to a VN of degree d,

$$\lambda\left(X\right) = \sum_{d=1}^{d_v} \lambda_d X^{d-1}. \tag{2.1}$$

Similarly, the polynomial of the CN degree distribution is defined in Equation (2.2),

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

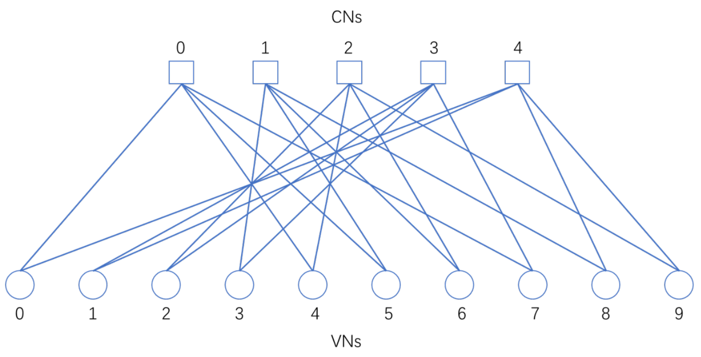


Figure 2.1: An example of a parity-check matrix **H** and the corresponding Tanner graph.

$$\rho\left(X\right) = \sum_{d=1}^{d_c} \rho_d X^{d-1} \tag{2.2}$$

where d_c denotes the maximum CN degree and ρ_d denotes the fraction of edges connected to the check node(s) of degree d. The degree distribution polynomials in Equation (2.1) and Equation (2.2) are defined from an *edge-perspective*. The degree distributions can also be given from a *node-perspective*, denoted by $\Lambda(X)$ and $\mathrm{P}(X)$, shown in Equation (2.3) and Equation (2.4), respectively. Different from λ_d and ρ_d, Λ_d and P_d denote the fraction of nodes of degree d,

$$\Lambda\left(x\right) = \sum_{d=1}^{d_v} \Lambda_d x^d, \tag{2.3}$$

$$\mathrm{P}\left(x\right) = \sum_{d=1}^{d_c} \mathrm{P}_d x^d. \tag{2.4}$$

The degree distribution from an edge-perspective and a node-perspective can be easily transformed to each other by [11],

$$\Lambda_d = \frac{\lambda_d/d}{\int_0^1 \lambda(X)dX}, \tag{2.5}$$

$$\mathrm{P}_d = \frac{\rho_d/d}{\int_0^1 \rho(X)dX}. \tag{2.6}$$

Consider a $(7,4)$ Hamming code as an example. The parity check matrix **H** is

$$\mathbf{H} = \begin{bmatrix} 1&1&1&0&1&0&0\\ 0&1&1&1&0&1&0\\ 1&1&0&1&0&0&1 \end{bmatrix}. \tag{2.7}$$

The degree distributions from an edge-perspective are:

$$\lambda(X) = \frac{1}{4} + \frac{1}{2}X + \frac{1}{4}X^2,$$
$$\rho(X) = X^3,$$

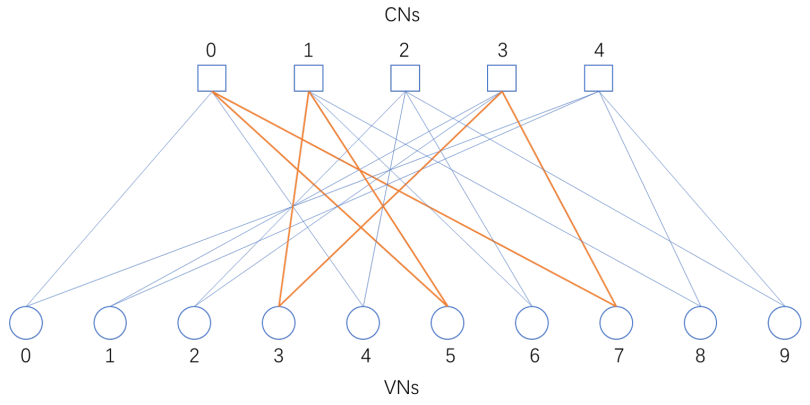


Figure 2.2: An example of a cycle in a Tanner graph.

while from a node-perspective they are:

$$\Lambda(x) = \frac{3}{7}x + \frac{3}{7}x^2 + \frac{1}{7}x^3,$$
$$\mathrm{P}(x) = x^4.$$

The code rate R_L of an LDPC code can be computed based on the degree distributions as

$$R_L = 1 - \frac{\int_0^1 \rho(X)dX}{\int_0^1 \lambda(X)\, dX}. \tag{2.8}$$

Properties in the Tanner graph of an LDPC code will decide the performance of the iterative decoder for the LDPC code, and one of the most significant properties is the presence of a cycle. A cycle in the Tanner graph is a closed path formed by a sequence of edges. The number of edges a cycle contains is the length of the cycle. The minimum cycle length in the Tanner graph is called girth. Taking the Tanner graph in Figure 2.2 as an example, the orange edges form a cycle of length six. The girth of this Tanner graph is six as well. During the construction of an LDPC code, short cycles should be avoided since they degrade the performance of the iterative decoder [11].

2.2 Decoding Algorithms

In 1960, in addition to introducing LDPC codes, Gallager provided a near-optimal decoding algorithm as well, the sum-product algorithm (SPA). This algorithm is also often referred to as belief propagation (BP). This algorithm initializes with message values coming from the channel, which are denoted by $\boldsymbol{m}_{\mathrm{ch}}$. Take the binary additive white Gaussian noise channel (BI-AWGNC) as an example. The channel message for the j-th received value is $m_{\mathrm{ch},j} = 2y_j/\sigma^2$ [11], where y_j is the j-th received value and σ is the standard deviation for the BI-AWGNC.

After initialization, the message values will be passed between CNs and VNs, iteratively, until a stopping criteria is satisfied. We denote by c_i and v_j CN i and

Algorithm 1: SPA for Decoding

```
1. Initialization: Initialize $m_{\mathrm{ch},j}$ according to the channel model. Then for every
   VN set $m_{v_j \to c_i} = m_{\mathrm{ch},j}$.

2. CN Update: With the extrinsic message value $m_{v_j \to c_i}$, compute the outgoing
   message value $m_{c_i \to v_j}$ as follows,

   $$m_{c_i \to v_j} = 2\tanh^{-1}\left(\prod_{v_{j'} \in N(c_i)\backslash\{v_j\}} \tanh\left(\frac{1}{2} m_{v_{j'} \to c_i}\right)\right). \quad (2.9)$$

3. VN Update: Compute outgoing message value $m_{v_j \to c_i}$ based on $m_{c_i \to v_j}$ as
   follows,

   $$m_{v_j \to c_i} = m_{\mathrm{ch},j} + \sum_{c_{i'} \in N(v_j)\backslash\{c_i\}} m_{c_{i'} \to v_j}. \quad (2.10)$$

4. Bit Decision: Calculate bit decision $\mathbf{v}$ based on

   $$\mathbf{v} = \mathrm{sign}\left(m_{\mathrm{ch}} + \sum_{c_i \in N(v_j)} m_{c_i \to v_j}\right). \quad (2.11)$$

5. Stopping Criteria: If the following statement is satisfied, or the number of
   iterations exceeds the maximum limit, stop; otherwise, go to Step 2:

   $$\mathbf{v}\mathbf{H}^T = \mathbf{0}. \quad (2.12)$$
```

VN j, respectively. The message from CN i to VN j is denoted by $m_{c_i \to v_j}$, and the message from VN j to CN i is denoted by $m_{v_j \to c_i}$. $N(c_i)$ denotes the set of VNs that are connected to CN i, and $N(v_j)$ denotes the set of CNs that are connected to VN j. The details of the SPA for decoding are shown in Algorithm 1 [11].

2.3 Construction of LDPC Code Based on PEG Algorithm

As mentioned previously, short cycles in the Tanner graph will cause problems for decoders of LDPC codes. As an algorithm widely utilized for computer-based code design, the progressive-edge-growth (PEG) algorithm is effective to construct a Tanner graph with large girth [12]. The algorithm starts with a Tanner graph with a set of VNs and a set of CNs without edges connecting nodes in one set to nodes in the other. In order to maximize the girth of the Tanner graph, only one edge is built in the graph at a time based on specific rules in order to make sure each edge added in the graph maximizes the local girth.

There are some concepts that need to be presented before the details of the PEG algorithm. The PEG algorithm is initialized with the number of VNs and CNs, n

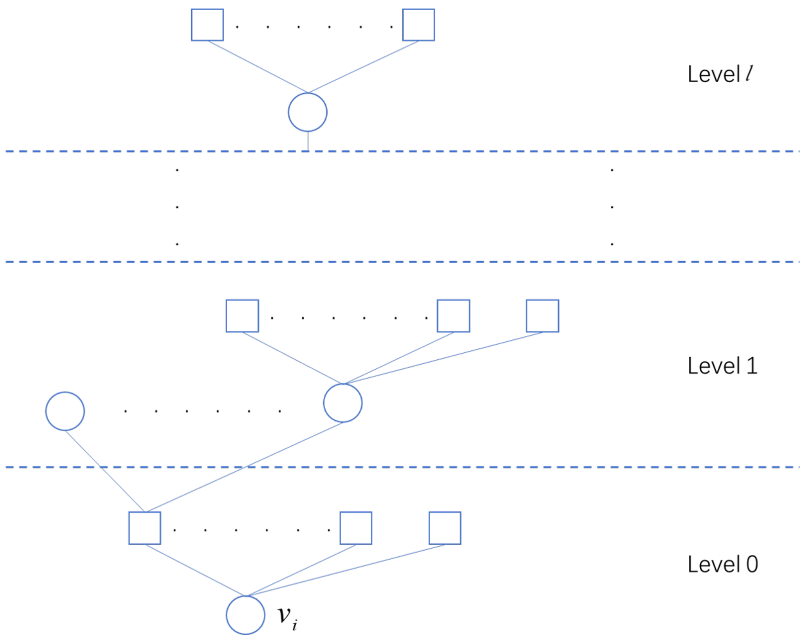


Figure 2.3: Tree graph $\mathcal{R}_i$ expanded from Tanner graph.

and m, respectively, and the VN degrees sequence D_v, which contains the degrees for the VNs. Denote a Tanner graph as $\mathcal{G} = (\mathcal{V}, \mathcal{C})$ with VN set $\mathcal{V} = \{v_0, v_1, ..., v_{n-1}\}$ and CN set $\mathcal{C} = \{c_0, c_1, ..., c_{n-1}\}$. Considering variable node v_i, let g_i be the local girth of v_i, which is the shortest cycle passing through v_i. The edge-growth rules are to maximize the girth of the current subgraph. Let $N_i^{(l)}$ denote the set of CNs in $\mathcal{G}$ that are connecting to VN v_i within a distance $2l+1$, and $\bar{N}_i^{(l)}$ is the complementary set of $N_i^{(l)}$. As shown in Figure 2.3, with a VN v_i as the root, the Tanner graph can be expanded as a tree graph $\mathcal{R}_i$. Each node will only appear once. There are two levels of nodes in each level of the tree, which includes one VN level and one CN level. The details of the PEG algorithm are shown in Algorithm 2 [11, 13].

Algorithm 2: PEG Algorithm

```
for $i = 0$ to $n-1$ do
    for $k = 0$ to $d_{v_i} - 1$ do
        if $k = 0$ then
            Add an edge $\mathcal{E}_i^{(0)}$ from $v_i$ to $c_j$. The edge should be the first
            edge connecting variable node $v_i$. The check node $c_j$ should have
            the lowest degree for current Tanner graph.
        else
            Based on current Tanner graph, grow a path tree $\mathcal{R}_i$ with $v_i$ as
            the root to a level such that $\bar{N}_i^{(l)} \neq \varnothing$ but $\bar{N}_i^{(l+1)} = \varnothing$.
            Choose a check node $c_j$ which has the lowest degree from $\bar{N}_i^{(l)}$
            and add the $k$-th edge $\mathcal{E}_i^{(k)}$ from $v_i$ to $c_j$.
        end
    end
end
```

2.4 Density Evolution for LDPC Codes

DE tracks the evolution of the probability density functions (pdfs) of the messages being passed around in the iterative decoder across iterations [11]. It is one powerful tool for predicting the performance and error correction capability (decoding threshold) for an LDPC code. The decoding threshold is the channel parameter value that effects in which communication is reliable beyond it and unreliable below it. DE determines the decoding threshold given a degree distribution pair, $\lambda(X)$ and $\rho(X)$, as given in Equation (2.1) and Equation (2.2). Let $\boldsymbol{p}^{(l)}$ denote the pdfs of messages being passed from VNs to CNs at the l-th iteration, and $\boldsymbol{q}^{(l)}$ denote the pdfs of messages being passed from CNs to VNs at the l-th iteration. Let $\boldsymbol{m}$ denote the pdfs of log-likelihood ratios (LLRs) coming from the channel, and p_e is a predetermined minimum error probability. DE can be implemented as shown in Algorithm 3 [11], where Γ and Γ^{-1} are two functions proposed in [14], $\boldsymbol{p} * \boldsymbol{q}$ denotes convolution between $\boldsymbol{p}$ and $\boldsymbol{q}$, and $\boldsymbol{p}^{*(d)}$ denotes d-fold convolution of $\boldsymbol{p}$ with itself.

Algorithm 3: DE Algorithm

```
1. Choose a value for $\alpha$, which is expected to be less than the decoding threshold
   $\alpha^\star$. Set iteration $l = 0$. Set initial pdfs of CNs $\boldsymbol{q}^{(l)} = \boldsymbol{m}$.

2. Given $\boldsymbol{q}^{(l)}$, compute $\boldsymbol{p}^{(l)}$ based on
   $\boldsymbol{p}^{(l)} = \boldsymbol{q}^{(0)} * \sum_{d=1}^{d_v} \lambda_d \cdot \left(\boldsymbol{q}^{(l-1)}\right)^{*(d-1)}.$   (2.13)

3. Increment $l$ by 1. Given $\boldsymbol{p}^{(l)}$, compute $\boldsymbol{q}^{(l)}$ based on
   $\boldsymbol{q}^{(l)} = \Gamma^{-1}\left[\sum_{d=1}^{d_c} \rho_d \cdot \left(\Gamma\left[\boldsymbol{p}^{(l)}\right]\right)^{*(d-1)}\right].$   (2.14)

4. Given $\boldsymbol{q}^{(l)}$, compute $\boldsymbol{p}^{(l)}$ based on
   $\boldsymbol{p}^{(l)} = \boldsymbol{q}^{(0)} * \sum_{d=1}^{d_v} \lambda_d \cdot \Gamma^{-1}\left[\sum_{d=1}^{d_c} \rho_d \cdot \left(\Gamma\left[\boldsymbol{p}^{(l-1)}\right]\right)^{*d-1}\right]^{*(d-1)}.$   (2.15)

5. Check conditions:
    a. $l < l_{\max}$ and $\int_{-\infty}^{0} \boldsymbol{p}^{(l)}(\tau)d\tau \leq p_e$: Increment $\alpha$ by a small amount and go to
       Step 2.
    b. $l < l_{\max}$ and $\int_{-\infty}^{0} \boldsymbol{p}^{(l)}(\tau)d\tau > p_e$: Increment $\alpha$ by a small amount and go to
       Step 3.
    c. $l = l_{\max}$ and $\int_{-\infty}^{0} \boldsymbol{p}^{(l)}(\tau)d\tau > p_e$: The current $\alpha$ is the decoding threshold
       $\alpha^\star$
```

2.5 EXIT Charts for LDPC Codes

The extrinsic-information-transfer (EXIT) chart technique is an alternative tool for estimating the decoding threshold for LDPC codes and turbo codes [11]. The idea behind EXIT chart is to predict the convergence behaviour by examining the evolution of input/output mutual information (MI) exchange between two constituent decoders in consecutive iterations. In the LDPC decoder, the two consitituent decoders are the VN processor (VNP) and the CN processor (CNP), as shown in Figure 2.4. In the LDPC decoder, these processors work cooperatively and iteratively. The transfer curves plotting the input MI versus the output MI can be obtained for both processors. Among them, the transfer curve for VNP depends on channel parameters. Two transfer curves can be plotted on the same axes and aid to predict the decoding threshold.

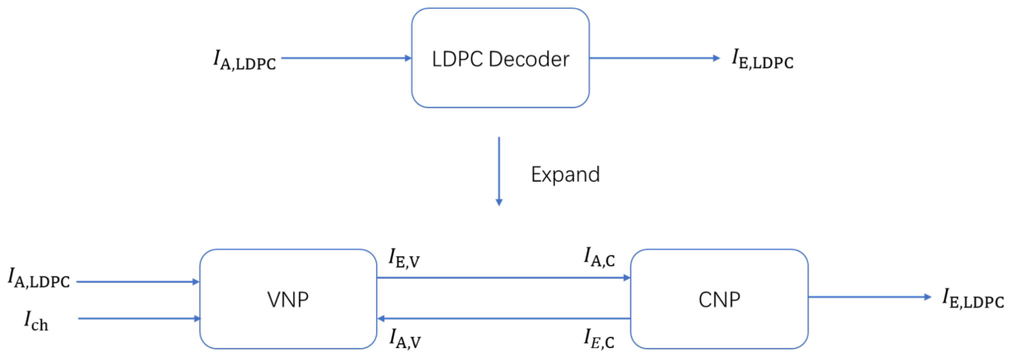


Figure 2.4: Iterative decoder for LDPC codes.

An example of an EXIT chart for a regular (3,6) LDPC code over the BI-AWGNC with signal-to-noise ratio (SNR) $\sigma_{\text{ch}} = 2.3$ is shown in Figure 2.5. The solid curve is the EXIT curve for the VNP. It plots the MI for the extrinsic information coming out of the VNP (denoted by $I_{\text{E,V}}$) against the MI for the extrinsic information entering into the VNP (denoted by $I_{\text{A,V}}$). The dashed curve is the EXIT curve for the CNP. Unlike the curve of VNP, the curve of CNP plots the MI for the extrinsic information entering into the CNP (denoted by $I_{\text{A,C}}$) against the MI for the extrinsic information coming out of the CNP (denoted by $I_{\text{E,C}}$). The EXIT curves for VNP and CNP can be computed by [11]

$$I_{\text{E,V}} = \sum_{d=1}^{d_v} \lambda_d J\left(\sqrt{(d-1)\left[J^{-1}\left(I_{\text{A,V}}\right)\right]^2 + \sigma_{\text{ch}}^2}\right) \tag{2.16}$$

and

$$I_{\text{E,C}} = \sum_{d=1}^{d_c} \rho_d \left(1 - J\left(\sqrt{(d-1)\left[J^{-1}\left(1 - I_{\text{A,C}}\right)\right]^2}\right)\right), \tag{2.17}$$

where $J()$ and $J^{-1}()$ are functions proposed in [15]. The space between two transfer curves is the so-called tunnel. As mentioned before, the transfer curve of VNP depends on channel parameters. For example, for the BI-AWGNC, the curve for VNP shifts upward and the tunnel widens when the channel SNR increases, and the

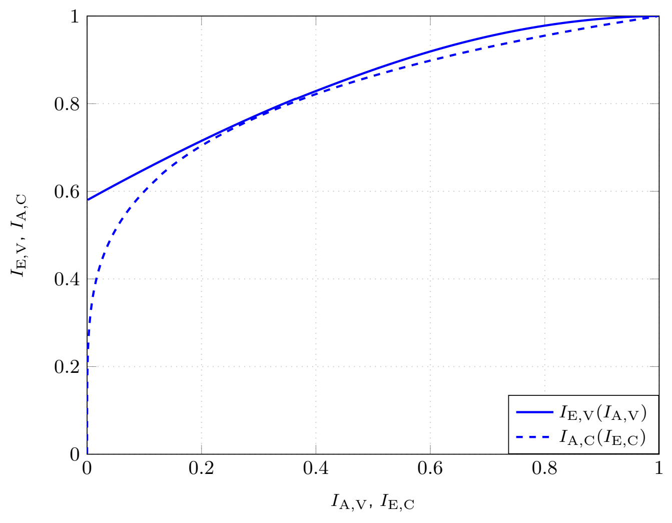
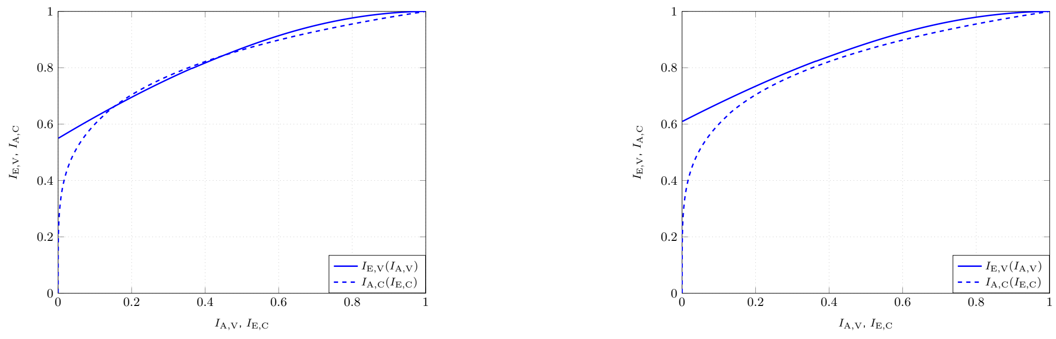


Figure 2.5: An EXIT chart example for a regular (3,6) LDPC code over the BI-AWGNC when $\sigma_{\mathrm{ch}} = 2.3$.

curve shifts downward and the tunnel narrows when the channel SNR decreases. We can implement curve-fitting based on this property to predict the decoder threshold for an LDPC code by adjusting the value of the channel parameter. The decoding threshold is the channel parameter when two curves are as close as possible, but do not cross.

Two curves for the same LDPC code over the BI-AWGNC with different values of SNR are shown in Figure 2.6. Comparing them to the curves in Figure 2.5, it is obvious that when $\sigma_{\mathrm{ch}} = 2.2$, the two curves cross, while when $\sigma_{\mathrm{ch}} = 2.4$ the two curves are not close enough. Therefore, the decoding threshold for this LDPC code over the BI-AWGNC is $\sigma_{\mathrm{ch}} = 2.3$ (when precision is 0.1).



(a) $\sigma_{\mathrm{ch}} = 2.2$ **(b)** $\sigma_{\mathrm{ch}} = 2.4$

Figure 2.6: EXIT charts for a regular (3,6) LDPC code over the BI-AWGNC.

Similarly, EXIT charts can also be implemented for turbo codes. There is one

Algorithm 4: EXIT Charts for LDPC Codes

```
1. Set $\sigma_{\text{ch}} = J^{-1}(I_{\text{A,LDPC}})$, $I_{\text{A,V}} = 0$, $l = 0$. Set a large enough number $N$ denoting
   the maximum number of iterations of exchaning MI between VNP and CNP.
2. Calculate $I_{\text{E,V}}$ based on Equation (2.16). Set $I_{\text{A,C}} = I_{\text{E,V}}$.
3. Calculate $I_{\text{E,C}}$ based on Equation (2.17). Set $I_{\text{A,V}} = I_{\text{E,C}}$.
4. Check Conditions:
    a. $l < N$: Increment $l$ by one and go to Step 2.
    b. $l = N$: Calculate $I_{\text{E,LDPC}}$ based on:
```

$$I_{\text{E,LDPC}} = \sum_{d=1}^{d_v} \lambda_d J\left(\sqrt{d[J^{-1}(I_{\text{A,V}})]^2}\right). \tag{2.18}$$

EXIT curve for each constituent code in a turbo code and curve-fitting can be implemented between the EXIT curves of the two constituents. Considering the situation when an LDPC code is one of constituents in a turbo code, let $I_{\text{A,LDPC}}$ denote the MI for the extrinsic information entering into the LDPC decoder, and let $I_{\text{E,LDPC}}$ denote the MI for the extrinsic information coming out of the LDPC decoder. The transfer curve for the entire LDPC decoder can be calculated by Algorithm 4 [15]. Take the same LDPC code in Figure 2.5 as an example. The EXIT curve of the LDPC code is shown in Figure 2.7.

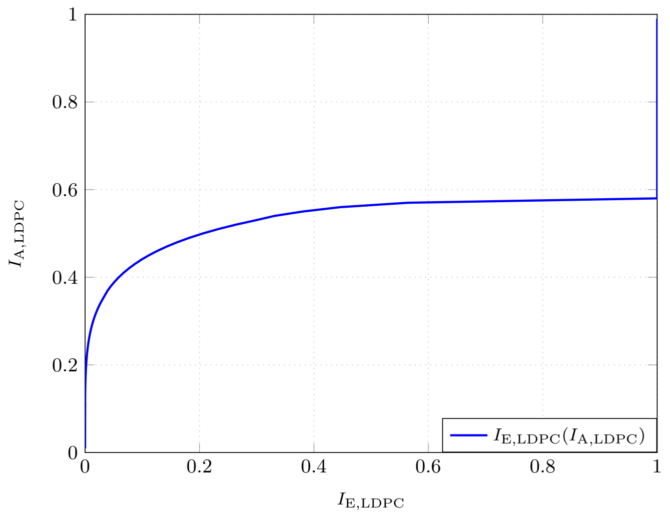


Figure 2.7: An example of an EXIT curve for a regular (3,6) LDPC code.

3

Coding for DNA-Based Storage

3.1 DNA-Based Storage Channel Model

As mentioned before, Davey and Mackay [1] proposed a method dealing with insertion, deletion, and substitution errors. The channel model proposed by Davey and Mackay is also considered in this project.

Let $\boldsymbol{x} = (x_1, ..., x_N)$ be the sequence of length N to be transmitted over the channel. The transmitted symbol x_i can be thought of as being placed in a queue. At a specific instant t_i, x_i enters the queue. As shown in Figure 3.1, there are three possible events for x_i:

- A symbol picked uniformly at random is inserted into the queue just before x_i with probability P_{I}, and the inserted symbol is transmitted over the channel.
- x_i is deleted with probability P_{D}, and nothing is transmitted over the channel.
- x_i is transmitted over the channel with probability $P_{\mathrm{T}} = 1 - P_{\mathrm{I}} - P_{\mathrm{D}}$. In this case, the transmitted symbol can be substituted by another symbol x'_i with probability P_{S}, or transmitted through the channel without errors with probability $1 - P_{\mathrm{S}}$.

The procedure will finish after the last symbol x_N leaves the queue. At the receiver side, we have the output sequence $\boldsymbol{y} = (y_1, ..., y_{N'})$. The sequence length N' for $\boldsymbol{y}$ depends on the channel procedure. The process is called insertion-deletion-substitution (IDS) process. In the IDS model, the position of x_i in the transmitted sequence can be different to the position of the corresponding received symbol $y_{i'}$ (suppose x_i is not deleted in the channel). The change of the position is referred to theas synchronization *drift*, denoted by d_i. The synchronization drift at x_i should be calculated by the numbers of insertions N_{I} and deletions N_{D} from the first transmitted symbol x_1 to the point in which x_i is transmitted, $d_i = N_{\mathrm{I}} - N_{\mathrm{D}}$. This is a Hidden Markov Model (HMM) parameterized by the probabilities P_{I}, P_{D} and P_{S}.

3.2 Coding for DNA-Based Storage Channel

3.2.1 Coding Scheme

Davey and Mackay proposed a concatenated coding scheme for correcting errors [1], as shown in Figure 1.4. As mentioned previously, the decoding algorithm in their

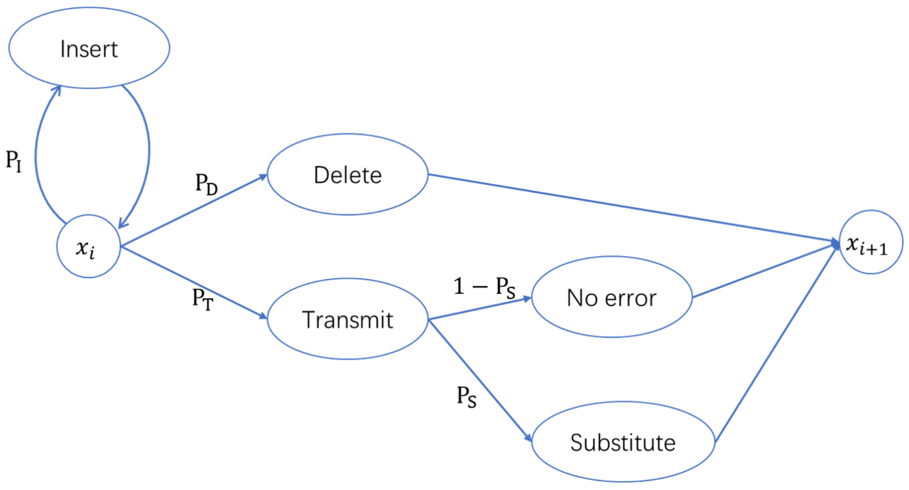


Figure 3.1: DNA-based storage channel model (IDS model).

coding scheme is referred to as bit-level synchronization, but in the DNA-based channel the data alphabet should be expanded from binary to quaternary. Hence, the coding scheme in this project is developed from the coding scheme in Figure 1.4, but has some differences.

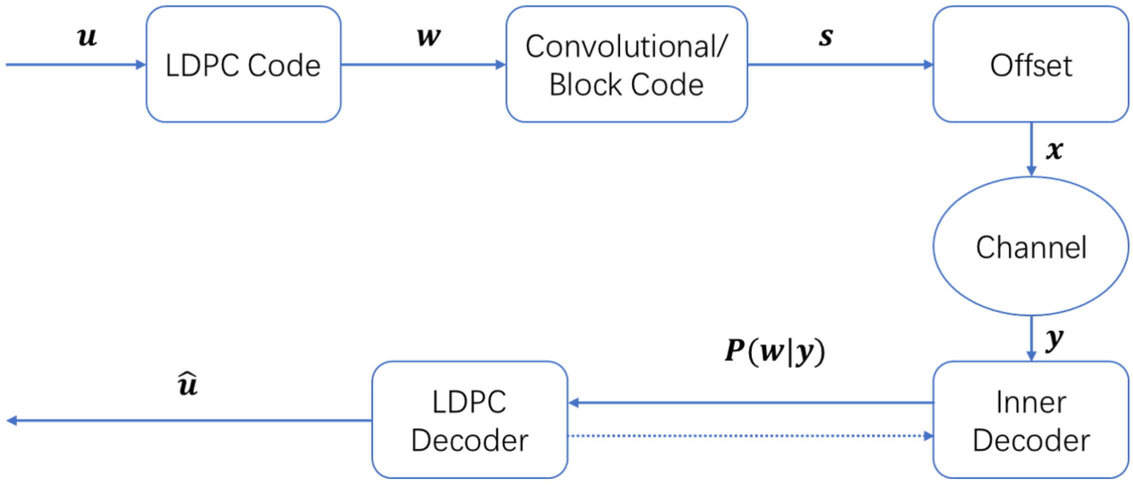


Figure 3.2: Coding scheme in this project.

The coding scheme utilized in this thesis is shown in Figure 3.2 [16]. The information sequence $\boldsymbol{u} = (u_1, ..., u_K)$, $u_i \in \mathbb{F}_{2^k}$ of length K is encoded by an LDPC code of code rate $R_{\text{out}} = \frac{K}{N_{\text{out}}}$ onto the codeword $\boldsymbol{w} = (w_1, ..., w_{N_{\text{out}}})$ of length N_{out}. Then the codeword $\boldsymbol{w}$ is encoded by a convolutional code or a block code of code rate $R_{\text{in}} = \frac{N_{\text{out}}}{N_{\text{in}}}$, resulting in the codeword $\boldsymbol{s} = (s_1, ..., s_{N_{\text{in}}})$ of length N_{in}. Then $\boldsymbol{s}$ is offset by a watermark sequence, a pseudo-random sequence which is known by both transmitter and receiver. Then codeword $\boldsymbol{x} = (x_1, ..., x_{N_{\text{in}}})$ is transmitted through the channel. The overall code rate is $R = \frac{K}{N_{\text{in}}}$.

In this thesis, the channel will be considered as a single IDS channel (Figure 3.1) at the start. Moreover, because several copies of DNA strands are accessed at the

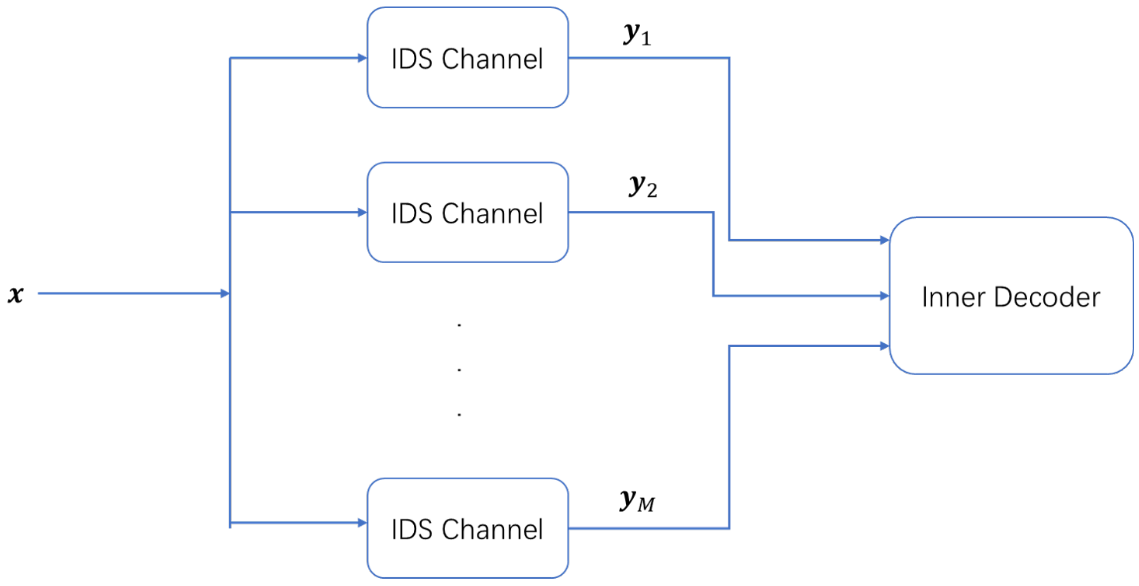


Figure 3.3: Communication over multiple IDS channel.

receiver, multiple transmissions over parallel IDS channels is considered as a channel model [16], as shown in Figure 3.3. The transmitted sequence $\boldsymbol{x}$ will be transmitted through several different IDS channels, and the inner decoder will receive several sequences $\{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_M\}$, where M is the number of received sequences. These received sequences contain the same information but may have different errors in them. The decoding process is based on these sequences together.

3.2.2 Decoding Algorithm

The decoding algorithm for the LDPC code has been introduced in Algorithm 1. Thus, in this section, we focus on the inner decoder, which is symbol-wise maximum a posteriori (MAP) decoding.

For a single IDS channel, the a posteriori likelihoods at the receiver are

$$p(w_i|\boldsymbol{y}) = \frac{p(\boldsymbol{y}, w_i)}{p(\boldsymbol{y})}. \tag{3.1}$$

Assume $\boldsymbol{w}$ is uniformly and identically distributed (u.i.d). The first step is to compute $p(\boldsymbol{y}, w_i)$. Let s_i denote the state variable of a convolutional code and d_j denote the state variable of the drift mentioned above. Denote by σ_k the joint state variable of s_i and d_j. Set $\boldsymbol{x}_a^b = (x_a, x_{a+1}, ..., x_b)$. Now we obtain:

$$p(\boldsymbol{y}, w_i) = \sum_{(\sigma,\sigma'):w_i} p\left(\boldsymbol{y}, \sigma, \sigma'\right). \tag{3.2}$$

Because of the property of Markov chains, $p(\boldsymbol{y}, \sigma, \sigma')$ can be written as:

$$p(\boldsymbol{y}, \sigma, \sigma') = p\left(\boldsymbol{y}_1^{(i-1)n+d}, \sigma\right) p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, \sigma'|\sigma\right) p\left(\boldsymbol{y}_{in+d'+1}^{N'}|\sigma'\right). \tag{3.3}$$

The terms in the equation above can be replaced by $\alpha_{i-1}(\sigma)$, $\gamma_i(\sigma, \sigma')$ and $\beta_i(\sigma')$, respectively, and computed recursively as:

$$\alpha_i(\sigma') = \sum_{\sigma} \alpha_{i-1}(\sigma)\gamma_i(\sigma, \sigma') \tag{3.4}$$

and

$$\beta_i(\sigma) = \sum_{\sigma'} \beta_{i+1}(\sigma')\gamma_{i+1}(\sigma, \sigma'). \tag{3.5}$$

The branch metric, $\gamma_{i+1}(\sigma, \sigma')$, can be decomposed into:

$$\begin{aligned}\gamma_{i+1}(\sigma, \sigma') &= p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d', s'|d, s\right) \\ &= p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right) p(s'|d, s).\end{aligned} \tag{3.6}$$

Since the convolutional memory state and the drift are independent events, and the memory state s only depends on the input w_i, we get

$$p(s'|d, s) = p(s'|s) = p(w_i). \tag{3.7}$$

We obtain:

$$\gamma_{i+1}(\sigma, \sigma') = p(w_i)p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right), \tag{3.8}$$

where $p(w_i) = \frac{1}{2^k}$ and $p\left(\boldsymbol{y}_{i-1)n+d+1}^{(in+d'}, d'|d, s, s'\right)$ can be efficiently computed using a lattice structure [17, 18] using the transmit sequence corresponding to the convolutional state transition $s \to s'$. Now the APPs at receiver can be calculated by using the BCJR algorithm [19].

If the channel is a multiple IDS channel, at the receiver part, the a posteriori likelihoods can be approximated as [16]:

$$p(w_i|\boldsymbol{y}_1, ...\boldsymbol{y}_M) \propto \frac{\prod_{j=1}^{M} p\left(w_i|\boldsymbol{y}\right)}{p(w_i)^{M-1}}. \tag{3.9}$$

In order to simplify the decoding complexity, some limitations should be set. In the IDS model, each transmitted symbol x_i will result in receiving 0 to ∞ symbols. It depends on how many symbols are inserted before x_i. In order to avoid the situation with infinity, a maximum number of insertions I is typically set during decoding. Then each transmitted symbol x_i will result in receiving 0 to $I + 1$ symbols. Define the drift state d_i of the symbol x_i as the position that symbol x_i enters the channel. Set the state of x_i as $d_i = a$ and the state of x_{i+1} as $d_{i+1} = b$. The transition probability $P_{ab} = P(x_{i+1} = b|x_i = a)$ is as follows [1]:

$$P_{ab} = \begin{cases} P_{\mathrm{D}}, & \text{if } b = a - 1, \\ \alpha_I P_{\mathrm{I}} P_{\mathrm{D}} + P_{\mathrm{T}}, & \text{if } b = a, \\ \alpha_I \left((P_{\mathrm{I}})^{b-a+1} P_{\mathrm{D}} + (P_{\mathrm{I}})^{b-a} P_{\mathrm{T}}\right), & \text{if } a < b < a + I, \\ \alpha_I (P_{\mathrm{I}})^I P_{\mathrm{T}}, & \text{if } b = a + I, \\ 0, & \text{otherwise,} \end{cases} \tag{3.10}$$

where the scaling parameter is $\alpha_I = \frac{1}{\left(1-(P_\mathrm{I})^I\right)}$. Besides, for the same reason, the values of drifts should be limited to a maximum value $|d_i| = d_{\max}$. The value of $d_{\max}$ can be chosen to be several times larger than the standard deviation of the synchronization over one block length [20].

4

Achievable Information Rates

4.1 Basic Theory

An achievable information rate (AIR) plot is a curve reflecting the data rate that can be achieved by a transceiver [21]. Assume $\boldsymbol{X} = (X_1, X_2, ..., X_n)$ and $\boldsymbol{Y} = (Y_1, Y_2, ..., Y_n)$ are the input and output process of a channel with memory. An AIR is given by [22]

$$I(\boldsymbol{X};\boldsymbol{Y}) \triangleq \lim_{n\to\infty} \frac{1}{n} \mathrm{I}(X_1, ..., X_n; Y_1, ...Y_n), \tag{4.1}$$

where $I(\cdot;\cdot)$ denotes MI. AIRs can be accurately estimated by a forward sum-product recursion based on sampling long sequences of both input, and corresponding channel output [23], as shown in Algorithm 5. The notations $\boldsymbol{x}^n \triangleq (x_1, x_2, ..., x_n)$ and $\boldsymbol{y}^n \triangleq (y_1, y_2, ..., y_n)$ denote the transmitted and the received sequences, respectively.

Algorithm 5: Algorithm for Estimating AIRs

1. **Sampling**: Sample very long sequences for both input and corresponding channel output.
2. **Computing**: Compute $\log p(\boldsymbol{x}^n)$, $\log p(\boldsymbol{y}^n)$ and $\log p(\boldsymbol{x}^n, \boldsymbol{y}^n)$.
3. **Concluding**: Conclude with the estimation:

$$\widehat{I}(\boldsymbol{X};\boldsymbol{Y}) \triangleq -\frac{1}{n}\log p(\boldsymbol{y}^n) - \frac{1}{n}\log p(\boldsymbol{x}^n) + \frac{1}{n}\log p(\boldsymbol{x}^n, \boldsymbol{y}^n) \tag{4.2}$$

In Step 2 in Algorithm 5, the computations can be carried out by the forward recursion of the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [19]. Taking $p(\boldsymbol{x}^n)$ as an example, it can be calculated as:

$$p(\boldsymbol{x}^n) = \sum_{\boldsymbol{y}^n} \sum_{\boldsymbol{s}_0^n} p(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{s}_0^n), \tag{4.3}$$

where $\boldsymbol{s}$ denotes state in trellis and $\boldsymbol{s}_0^k \triangleq (s_0, s_1, ..., s_k)$. Define the state metric as $\mu_k(s_k) \triangleq p(s_k, \boldsymbol{x}^k)$. By implementing the SPA, it can be recursively computed by

$$\mu_k(s_k) = \lambda_k \sum_{y_k} \sum_{s_{k-1}} \mu_{k-1}(s_{k-1}) p\left(x_k, y_k, s_k | s_{k-1}\right), \tag{4.4}$$

where λ_k are positive scalars which are chosen such that $\sum_{s_n}\mu_n(s_n) = 1$. Then the value of $-\frac{1}{n}\log p(\boldsymbol{x}^n)$ equals to the average of the scalars [23],

$$-\frac{1}{n}\log p(\boldsymbol{x}^n) = \frac{1}{n}\sum_{k=1}^{n}\log\lambda_k. \tag{4.5}$$

The value of $-\frac{1}{n}\log p(\boldsymbol{y}^n)$ and $-\frac{1}{n}\log p(\boldsymbol{x}^n,\boldsymbol{y}^n)$ can be calculated similarly by just replacing Equation 4.4. For $-\frac{1}{n}\log p(\boldsymbol{y}^n)$:

$$\mu_k(s_k) = \lambda_k\sum_{x_k}\sum_{s_{k-1}}\mu_{k-1}(s_{k-1})p\left(x_k, y_k, s_k|s_{k-1}\right). \tag{4.6}$$

And for $-\frac{1}{n}\log p(\boldsymbol{x}^n,\boldsymbol{y}^n)$:

$$\mu_k(s_k) = \lambda_k\sum_{s_{k-1}}\mu_{k-1}(s_{k-1})p\left(x_k, y_k, s_k|s_{k-1}\right). \tag{4.7}$$

4.2 AIR Calculation

The achievable information rate of the coding scheme in this project will be calculated based on Algorithm 5. Now starting from Equation (4.2), calculate the three parts in this equation one by one. The recursion for calculating $\log p(\boldsymbol{y}^n)$ is as follows:

$$\mu_k\left(s_k\right) = \sum_{x_k}\sum_{s_{k-1}}\mu_{k-1}\left(s_{k-1}\right)p\left(x_k, y_k, s_k|s_{k-1}\right).$$

Replace the notation in the equation and continue:

$$\begin{aligned}
\mu_k\left(\sigma'\right) &= \sum_{w_i}\sum_{\sigma}\mu_{k-1}\left(\sigma\right)p\left(w_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, \sigma'|\sigma\right)\\
&= \sum_{w_i}\sum_{\sigma}\mu_{k-1}\left(\sigma\right)p\left(w_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d', s'|d, s\right)\\
&= \sum_{w_i}\sum_{\sigma}\mu_{k-1}\left(\sigma\right)p\left(w_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right)p(s'|d, s)\\
&= \sum_{w_i}\sum_{\sigma}\mu_{k-1}\left(\sigma\right)p\left(w_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right)p(w_i)\\
&= \sum_{\sigma}\mu_{k-1}\left(\sigma\right)\sum_{w_i}p\left(w_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right)p\left(w_i\right)\\
&= \sum_{\sigma}\mu_{k-1}\left(\sigma\right)p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right).
\end{aligned} \tag{4.8}$$

Only $p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right)$ is left in these terms, and $p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d'|d, s, s'\right)$ can be calculated as mentioned above. Similarly, for $p(\boldsymbol{x}^n,\boldsymbol{y}^n)$:

$$\begin{aligned}
\mu_k\left(s_k\right) &= \sum_{s_{k-1}} \mu_{k-1}\left(s_{k-1}\right) p\left(x_k, y_k, s_k | s_{k-1}\right), \\
\mu_k\left(\sigma'\right) &= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) p\left(\omega_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, \sigma' | \sigma\right) \\
&= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) p\left(\omega_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d', s' | d, s\right) \\
&= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) p\left(\omega_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' | d, s, s'\right) p(s'|d,s) \\
&= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) p\left(\omega_i, \boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' | d, s, s'\right) p(\omega_i) \\
&= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) \frac{1}{2^k} p\left(\boldsymbol{y}_{(i-1)n+d+1}^{in+d'}, d' | d, s, s'\right).
\end{aligned} \tag{4.9}$$

For $p(\boldsymbol{x}^n)$:

$$\begin{aligned}
\mu_k\left(s_k\right) &= \sum_{s_{k-1}} \mu_{k-1}\left(s_{k-1}\right) p\left(x_k, s_k | s_{k-1}\right), \\
\mu_k\left(\sigma'\right) &= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) p\left(\omega_i, \sigma' | \sigma\right) \\
&= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) p\left(\omega_i | \sigma, \sigma'\right) p\left(\sigma' | \sigma\right) \\
&= \sum_{\sigma} \mu_{k-1}\left(\sigma\right) p\left(\omega_i\right) p\left(\sigma' | \sigma\right) \\
&= \mu_{k-1}\left(\sigma\right) p\left(\omega_i\right).
\end{aligned} \tag{4.10}$$

With the corresponding μ_k, it's easy to calculate the AIR for our coding scheme.

5
LDPC Code Optimization

In this chapter, an optimization mechanism for searching the most suitable LDPC codes for our coding scheme is proposed. The optimization process is based on the decoding threshold calculated based on two different methods presented in Chapter 2. It is defined as the channel parameter that divides the channel into two regions. The first region is characterized by reliable communication, where the probability of decoding error approaches zero, and communication in the other region is unreliable, where the probability of decoding error is bounded away from zero. An irregular LDPC code with the same degree distribution as a World Interoperability for Microwave Access (WiMax) LDPC code [24, 25] (code rate $R_{\text{WiMax}} = \frac{1}{2}$) will be tested for threshold computation. The code degree distributions are,

$$\lambda(X) = 0.2895X + 0.3158X^2 + 0.3947X^5,$$
$$\rho(X) = 0.6316X^5 + 0.3684X^6.$$

For simplicity, in the results shown in Chapters 5-6, the probabilities of insertion and deletion are assumed to be the same, denoted by p_{id}, and the substitution probability is assumed to be 0.

5.1 Threshold Computation

5.1.1 Without Inner-Outer Iterative Decoding

As mentioned previously, DE and EXIT charts are two techniques that allow to predict the decoding threshold for LDPC codes. The LLRs from the channel are necessary inputs for both of them. To simplify the problem, the case where no iterations are performed between the inner and outer decoders is considered.

5.1.1.1 Threshold Computation Based on Density Evolution

Since the goal is to optimize the outer code, the inner code and the channel can be combined and regarded as a new channel (as shown in Figure 5.1, circled by the orange dashed line). The LLRs of this channel, which has memory, can be obtained via Monte Carlo simulations by sampling an u.i.d random input sequence $\boldsymbol{w}$ and the inferred likelihoods for the symbols in $\boldsymbol{w}$. With these LLRs generated from the channel, it is easy to implement DE using Algorithm 3.

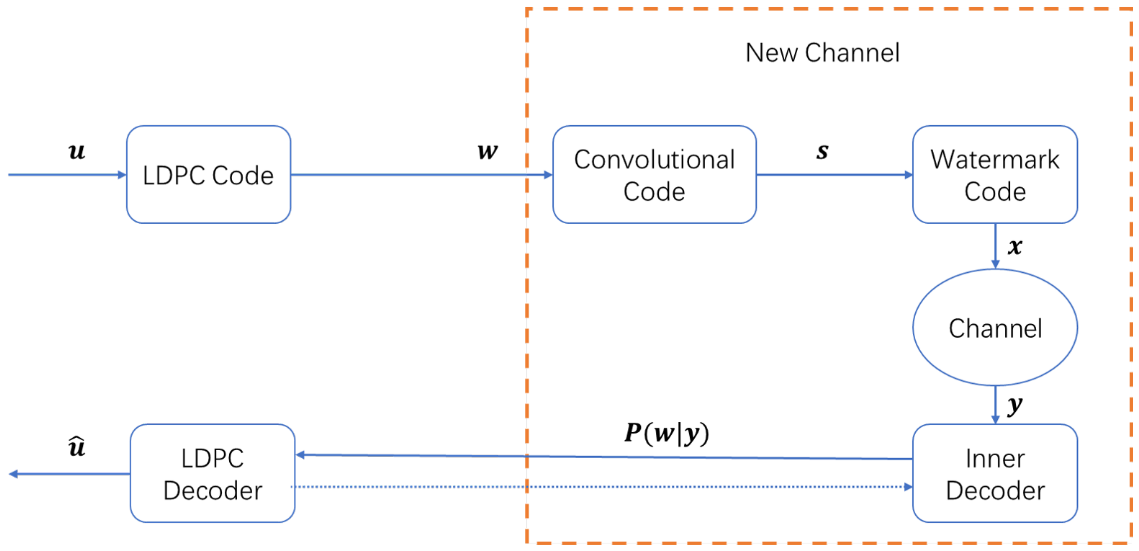


Figure 5.1: New channel for threshold computation.

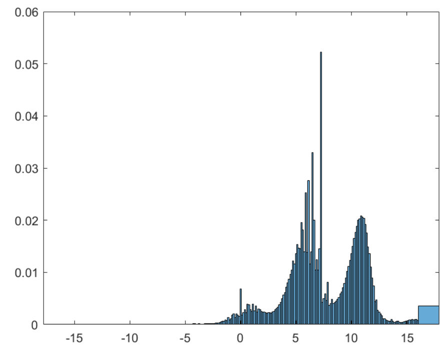
5.1.1.2 Threshold Computation Based on EXIT Charts

Different from DE, the sequence of LLRs cannot be utilized directly in the EXIT charts since it tracks the MI between the VNP and the CNP. Hence, in order to initialize the calculation of EXIT charts introduced in Equation (2.16) and Equation (2.17), the channel parameter σ_{ch} as used in Equation (2.16) should be calculated based on the sequence of LLRs. Figures (5.2a - 5.2d) show the histograms of the pdfs for different values of p_{id} (0.05, 0.10, 0.15, 0.20) with $M = 1$ (single IDS channel), and Figure 5.2e and Figure 5.2f show the histograms for $M = 2, 5$ (multiple IDS channels) when $p_{\text{id}} = 0.15$. As shown in these histograms, it is clear that when p_{id} is large enough (e.g., $p_{\text{id}} \geq 0.01$), the distribution can be approximated by a Gaussian distribution. Note that it is unnecessary to test when p_{id} is small (e.g., $p_{\text{id}} < 0.01$). Therefore, we assume that the LLRs are Gaussian distributed and we use the Gaussian approximation (GA) to obtain σ_{ch} for these channels. With the values of σ_{ch}, it is easy to implement EXIT charts based on Equation (2.16) and Equation (2.17).

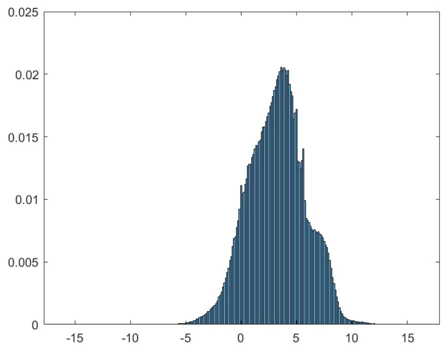
5.1.1.3 Results Verification

The decoding thresholds of the WiMax LDPC code for different numbers of received sequences M computed based on both DE and EXIT charts are presented in Table 5.1. Based on the results, we conclude that there is no significant difference between the two methods when a single IDS channel is considered ($M = 1$). However, when multiple IDS channels are considered, with increasing M, the gap between the results from DE and EXIT charts increases. For $M = 5$, the gap is as large as 0.018.

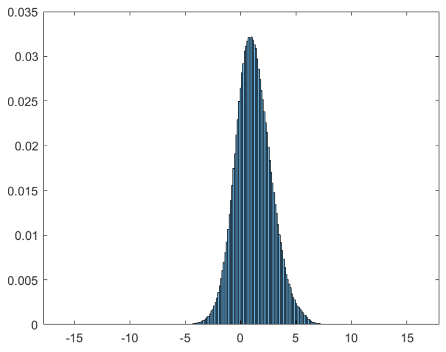
Figure 5.3 presents simulation results for the WiMax LDPC code for $M = 2$ and $M = 5$. The blue solid curve corresponds to the bit error rate (BER), while the blue dashed curve corresponds to the frame error rate (FER) of the simulation. The red dashed line is the threshold calculated based on DE, while the green dashed



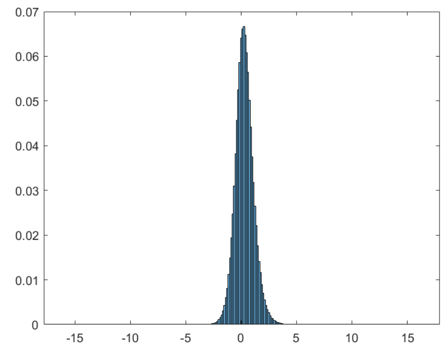
(a) $p_{\mathrm{id}} = 0.05,\ M = 1$



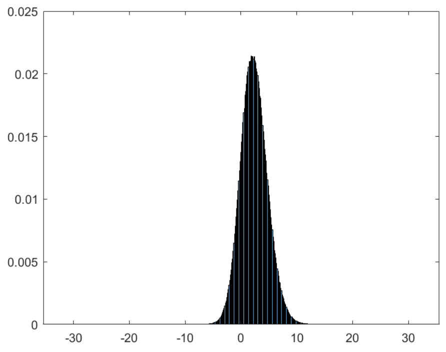
(b) $p_{\mathrm{id}} = 0.1,\ M = 1$



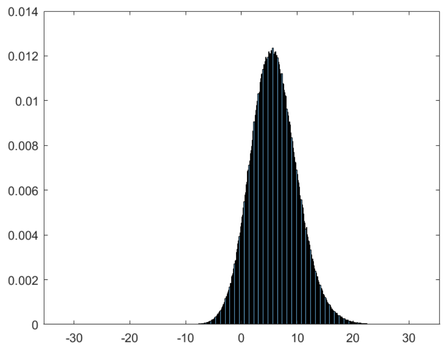
(c) $p_{\mathrm{id}} = 0.15,\ M = 1$



(d) $p_{\mathrm{id}} = 0.2,\ M = 1$



(e) $p_{\mathrm{id}} = 0.15,\ M = 2$



(f) $p_{\mathrm{id}} = 0.15,\ M = 5$

Figure 5.2: Histograms of the channel LLRs (including the inner code and the IDS channel) for different values of the insertion/deletion probability and different numbers of received sequences M.

Table 5.1: Thresholds of the WiMax code without inner-outer iterative decoding.

M	1	2	5
Threshold (DE)	0.116	0.145	0.172
Threshold (EXIT)	0.118	0.152	0.19

line is the threshold calculated based on EXIT charts. We observe that the starting point when the BER and FER start going down sharply is closer to the threshold calculated based on DE than the threshold calculated based on EXIT charts. This phenomenon is more evident when M increases. Indeed, the thresholds based on DE are more accurate. However, even though the results have less accuracy than the results based on DE, EXIT charts is still a useful method. Since σ_{ch} has already been pre-calculated based on the sequence of LLRs, it takes a shorter time to complete the computation during the optimization process. This is significant as in the optimization process (which will be introduced later in this chapter) the calculation of the decoding threshold needs to be implemented thousands of times.

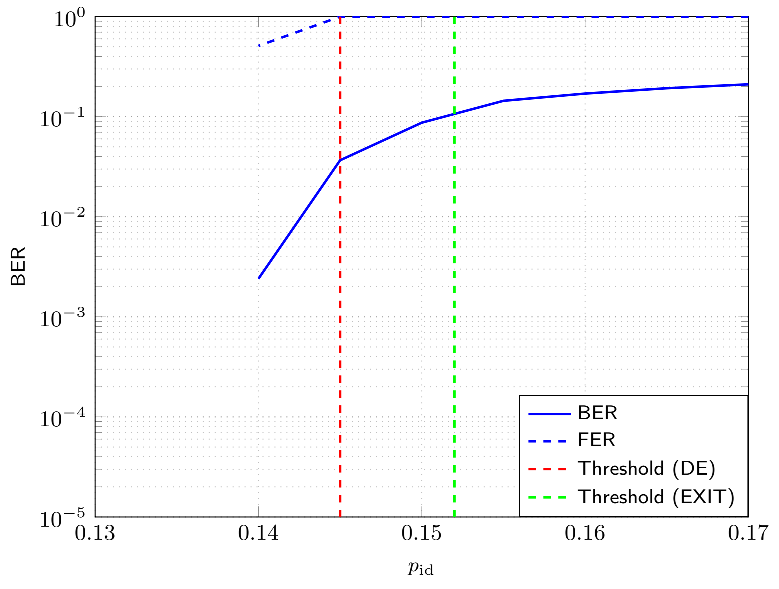
5.1.2 With Inner-Outer Iterative Decoding

5.1.2.1 Threshold Computation Based on Density Evolution

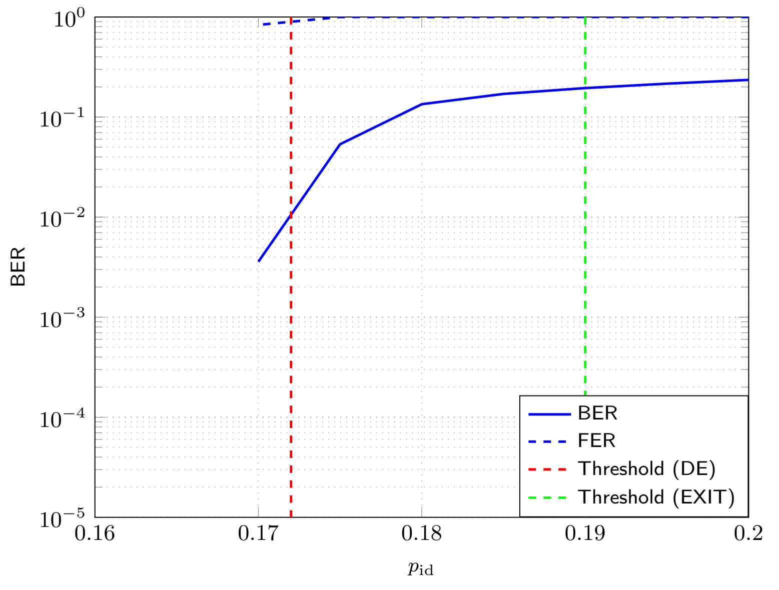
If inner-outer iterative decoding is considered, the LLRs from the channel can be obtained via Monte Carlo simulations as well. However, the computation of the threshold is different. As shown in Figure 5.4, since two decoders exchange extrinsic information iteratively, the values of inferred likelihoods from the inner decoder are different at different iterations, and both the inner and outer decoders affect these values. Thus, to ensure that the results are accurate, the inferred likelihoods should be sampled after enough iterations between the inner and outer decoders, and different sampling should be implemented for different LDPC codes being tested. These properties exacerbate the shortcoming of computation based on DE since the computational complexity is significantly higher than before. Therefore, it is no longer possible to utilize it in the optimization process. However, the results are still accurate so that this method can be used for result verification after optimization via EXIT charts.

5.1.2.2 Threshold Computation Based on EXIT Charts

The curve-fitting procedure when computing the threshold without considering inner-outer iterative decoding is implemented between the EXIT curves of VNPs and CNPs. However, as mentioned in Chapter 2, the curve-fitting can also be implemented between the EXIT curve of the inner code and the EXIT curve of the LDPC code (e.g., the curve shown in Figure 2.7). Similar to the flow chart in Figure 5.4, Figure 5.5 presents the flow chart for iterative decoding between the inner and outer decoders. The EXIT curve for the outer decoder, which is an LDPC decoder, plots the MI of the extrinsic information entering into the outer decoder (denoted



(a) $M = 2$



(b) $M = 5$

Figure 5.3: Simulation results for the WiMax code with different values of M.

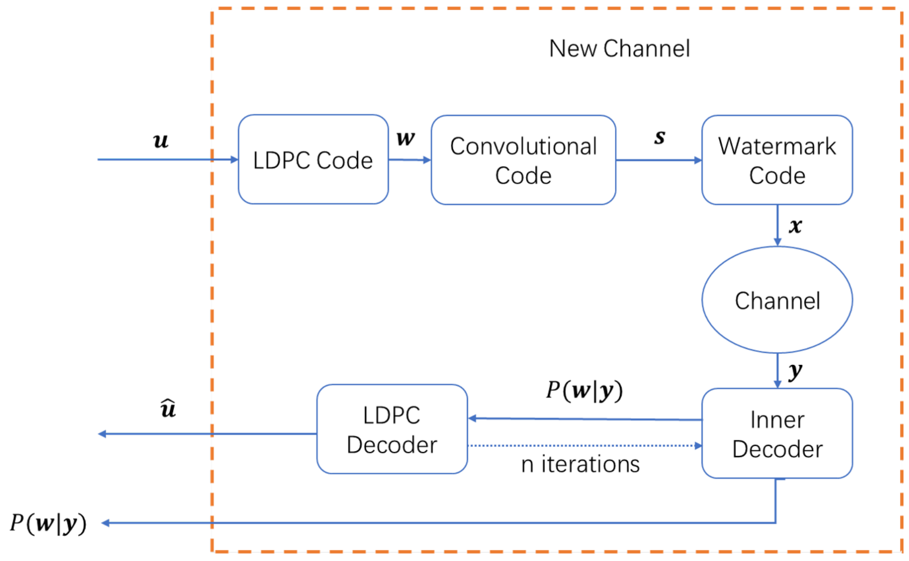


Figure 5.4: New channel for threshold computation (considering inner-outer iterative decoding).

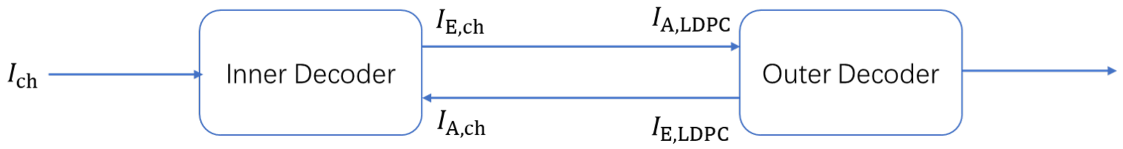


Figure 5.5: Iterative decoding between the inner and outer decoders.

by $I_{\text{A,LDPC}}$) against the MI of the extrinsic information coming out of the outer decoder (denoted by $I_{\text{E,LDPC}}$). The details of algorithm for generating EXIT curves for LDPC codes have been presented in Chapter 2. The EXIT curve for the inner decoder plots the MI of the exitrinsic information coming out of the inner decoder (denoted by $I_{\text{E,ch}}$) against the MI of the extrinsic information entering into the inner decoder (denoted by $I_{\text{A,ch}}$). The EXIT curve for the inner decoder can be obtained via Monte Carlo simulation by sampling of a sequence of a priori LLRs, denoted by L_{A}, and a sequence of its output extrinsic LLRs, denoted by L_{E}. L_{A} can be modeled as an sequence of independent zero-mean Gaussian random variables with a variance of σ_{A}^2 as [26]

$$L_{\text{A}} = \mu_{\text{A}} \cdot x + \sigma_{\text{A}}^2, \tag{5.1}$$

where $\mu_{\text{A}} = \sigma_{\text{A}}^2/2$ and x is the corresponding input bit. $I_{\text{A,ch}}$ can be computed from σ_{A} using the $J()$ function defined in Chapter 2 by

$$J(\sigma_{\text{A}}) = I_{\text{A,ch}}. \tag{5.2}$$

By sending the sequence of a priori LLRs L_{A} into the inner decoder, a sequence of output extrinsic LLRs L_{E} is obtained using the decoding algorithm introduced in

Table 5.2: Thresholds of the WiMax code with inner-outer iterative decoding.

M	1	2	5
Threshold (DE)	0.145	0.165	0.18
Threshold (EXIT)	0.13	0.155	0.17

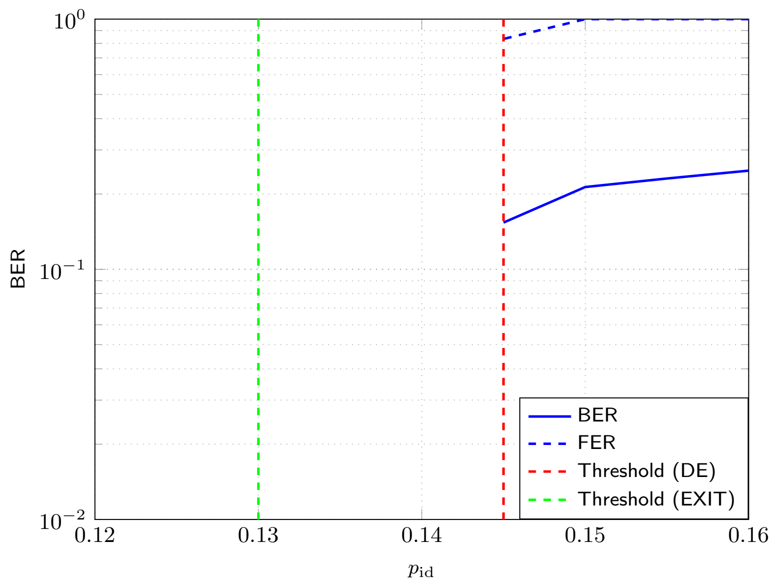
Chapter 3. The value of $I_{\mathrm{E,ch}}$ can be computed from the sequence of output extrinsic LLRs L_{E} as follows [27],

$$I_{\mathrm{E,ch}} = 1 - E\left\{\log_2\left(1 + e^{-L_{\mathrm{E}}}\right)\right\} \approx 1 - \frac{1}{N}\sum_{n=1}^{N}\log_2\left(1 + e^{-x_n \cdot L_{\mathrm{E},n}}\right), \tag{5.3}$$

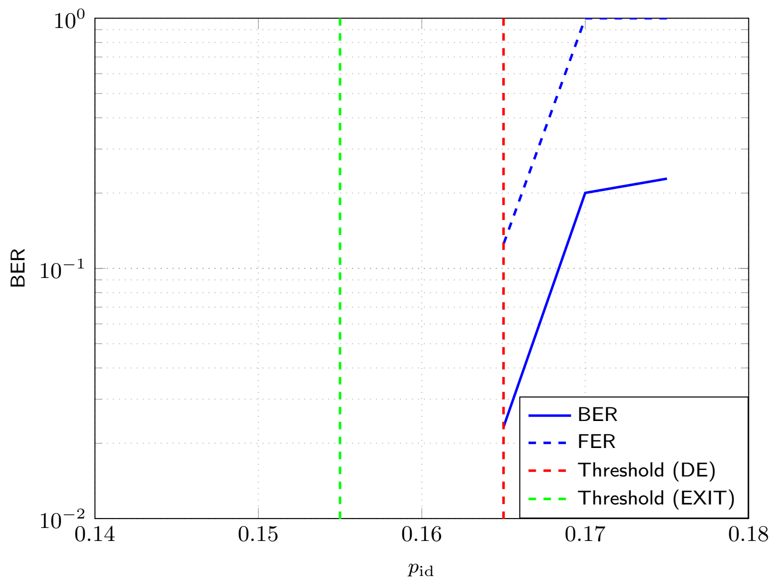
where N is the LLR sequence length, x_n the n-th transmitted bit, and $L_{\mathrm{E},n}$ is the n-th output extrinsic LLR. In order to reliably compute $I_{\mathrm{E,ch}}$, the sequence length N should be large. Having computed $I_{\mathrm{A,ch}}$ and $I_{\mathrm{E,ch}}$, the EXIT curve for the channel can be plotted and utilized for implementing curve-fitting with the EXIT curve of LDPC codes.

5.1.2.3 Results Verification

The thresholds of the WiMax code considering inner-outer iterative decoding are presented in Table 5.2. Comparing with the results in Table 5.1, the accuracy of threshold computation based on EXIT charts is lower. The accuracy will also be discussed in Chapter 6. Figure 5.6 shows the simulation for the WiMax code with inner-outer iterative decoding for $M = 1$ and $M = 2$. The maximum number of iterations between the inner decoder and the outer decoder is 50.



(a) $M = 1$



(b) $M = 2$

Figure 5.6: Simulation results for the WiMax code considering inner-outer iterative decoding.

5.2 Code Optimization

As a nonlinear cost function minimization problem [28], LDPC code optimization can be implemented by differential evolution, which is a simple and efficient approach for optimizing a problem by iteratively improving the candidate solutions based on an evolutionary process [29]. It is a universal method that has been implemented for several different channels, such as the BEC [30] and the Rayleigh fading channel [28]. The goal for this optimization is to find LDPC codes, specifically, degree distribution pairs $(\boldsymbol{\lambda}, \boldsymbol{\rho})$, which give the best performance (i.e., the best decoding threshold) for specific channels. The coding rate R_L and maximum degrees d_l and d_r are given before the optimization starts. Therefore, the optimization problem is to maximize the threshold p_{th} under the following conditions:

$$\sum_{i=2}^{d_l} \lambda_i = \sum_{j=2}^{d_r} \rho_j = 1, \tag{5.4}$$

$$\lambda_i \geq 0, \rho_j \geq 0, \forall i \in [2, d_l], \forall j \in [2, d_r], \tag{5.5}$$

$$1 - \frac{\int_0^1 \rho(X)dX}{\int_0^1 \lambda(X)dX} = R_L. \tag{5.6}$$

The details of the differential evolution algorithm are shown in Algorithm 6 [31, 32]. Let Z be the population size of the LDPC codes in the optimization, G be the maximum number of iteration, g be the current iteration, F_1 and F_2 be scaling parameters of mutation. $\lambda_{i,z}$ denotes the i-th element in the z-th degree distribution $\boldsymbol{\lambda}_z$, and similar for $\rho_{i,z}$. The threshold will be computed based on one of the methods introduced above. Besides, the authors in [14] recommended some tricks in order to reduce the search space of differential evolution:

- **Force small values to zero**: During the process of optimization, if the value of an element, $\lambda_{i,z}$ or $\rho_{j,z}$, is smaller than a specific value t, it should be set to zero.

- **Choose degree distribution in a special way**: During initialization, pick up some special elements in the degree distribution pairs and assign values, and set other elements to zeros. Specifically, for $\boldsymbol{\lambda}$, only pick up degree 2, 3, a very high degree h (e.g., $20 \leq h \leq 30$), and a degree in the middle m. On the other hand, for $\boldsymbol{\rho}$, choose two consecutive degrees, (e.g., 7 and 8, or 8 and 9). For example:

$$\lambda(X) = 0.2X + 0.15X^2 + 0.35X^{10} + 0.3X^{27},$$
$$\rho(X) = 0.64X^7 + 0.36X^8.$$

Both suggestions from [14] have been tested in the optimization based on our channel. The first suggestion is helpful to accelerate the optimization. However, the particular degree distribution introduced in the second suggestion is not suitable for our channel. Shibata *et al.* [32] also verified that the codes with this kind of degree

Algorithm 6: Differential Evolution Algorithm

1. **Initialization**: Set $g = 0$. Randomly generate Z pairs of degree distributions for the LDPC codes that satisfy the constrains in (5.4) to (5.6). Compute the decoding thresholds p_{th} for each degree distribution pair.

2. **Mutation**: Generate the mutant pair $(\tilde{\boldsymbol{\lambda}}_z, \tilde{\boldsymbol{\rho}}_z)$ for $1 \leq z \leq Z$ based on (5.7) and (5.8) as follows:

$$\tilde{\lambda}_{i,z} = \lambda_{i,z} + F_1(\lambda_{i,r_1} - \lambda_{i,z}) + F_2(\lambda_{i,r_2} - \lambda_{i,r_3}), 3 \leq i \leq d_l - 1, \tag{5.7}$$

$$\tilde{\rho}_{j,z} = \rho_{j,z} + F_1\left(\rho_{j,r_1} - \rho_{j,z}\right) + F_2\left(\rho_{j,r_2} - \rho_{j,r_3}\right), 3 \leq j \leq d_r, \tag{5.8}$$

   where r_1, r_2 and r_3 are three random distinct values chosen from $[1, Z]$. The values of the left elements in $(\tilde{\boldsymbol{\lambda}}_z, \tilde{\boldsymbol{\rho}}_z)$, including $\tilde{\lambda}_{d_l,z}$, $\tilde{\lambda}_{2,z}$, and $\tilde{\rho}_{2,z}$, are adjusted in order to make sure that the degree distribution pairs fulfill the constrains in (5.4) to (5.6).

3. **Selection**: Compute the decoding thresholds for the new degree distribution pairs. For each degree distribution pair, if the new decoding threshold $\tilde{p}_{\text{th}}$ is better than the old decoding threshold p_{th}, replace $(\boldsymbol{\lambda}_z, \boldsymbol{\rho}_z)$ by $(\tilde{\boldsymbol{\lambda}}_z, \tilde{\boldsymbol{\rho}}_z)$ in the population. Otherwise, keep $(\boldsymbol{\lambda}_z, \boldsymbol{\rho}_z)$ in the population.

4. **Termination criterion**: If $g < G$, set $g = g + 1$, and jump to Step 2. Otherwise, select the LDPC code in the population which has the best decoding threshold. The selected LDPC code is the output of the algorithm.

distributions do not have the best performance for channels with synchronization errors. Both results from Shibata's paper and from our optimization show that the existence of low degree CNs (especially degree-2 and degree-3 CNs) is significant to yield an excellent decoding threshold in channels with synchronization errors.

6 Results

In this chapter, we present numerical results for the AIRs, LDPC code optimization, and simulations for the error-rate performances will be presented successively. For the AIRs, we only present results for $M = 1$, while results for the case of $M > 1$ are presented for the LDPC code optimization and error-rate performances.

In order to compare the results conveniently, the block length in the coding scheme shown in Figure 3.2 should be set. The message sequence $\boldsymbol{u}$ is a binary sequence of length of $25,000$. It is encoded by an LDPC encoder of code rate $R_{\text{out}} = 0.5$. The output is an LDPC codeword $\boldsymbol{w}$ of length of $50,000$. A convolutional code with two memory elements and code rate $R_{\text{in}} = 0.5$ is utilized as a part of the inner code. The convolutional code encodes $\boldsymbol{w}$ into $\boldsymbol{s}$, which has length $100,000$. Then $\boldsymbol{s}$ is offset by a watermark sequence, which is a pseudo-random sequence of the same length as $\boldsymbol{s}$. Before tranmission, $\boldsymbol{s}$ is mapped into a sequence of DNA symbols $\boldsymbol{x}$ of length of $50,000$. All results presented in this chapter are under these assumptions.

6.1 Achievable Information Rates

Based on the method introduced in Chapter 3, the AIR curve for the channel when $M = 1$ is shown in Figure 6.1. Since both the outer code rate R_{out} and the inner code rate R_{in} are 0.5, the total code rate $R_L = 0.25$, shown as a green dashed line in the figure. The performance of the searched LDPC codes under the above assumptions will be located on the green line and they are always on the left side of the AIR curve. This figure will be utilized later to reflect the performance of the optimized LDPC codes based on their decoding thresholds. The closer the code performance to the AIR curve (solid blue line in Figure 6.1), the better the performance is.

6.2 Optimization Results

In this section, the optimization results under different conditions are presented. When implementing threshold computation, the precision is 0.001. The parameters used in Algorithm 6 are set as follows: $F_1 = F_2 = 0.5$, $Z = 500$ and $G = 250$. The values of the maximum degrees for VNs and CNs, d_l and d_r, are adjusted for finding the best threshold.

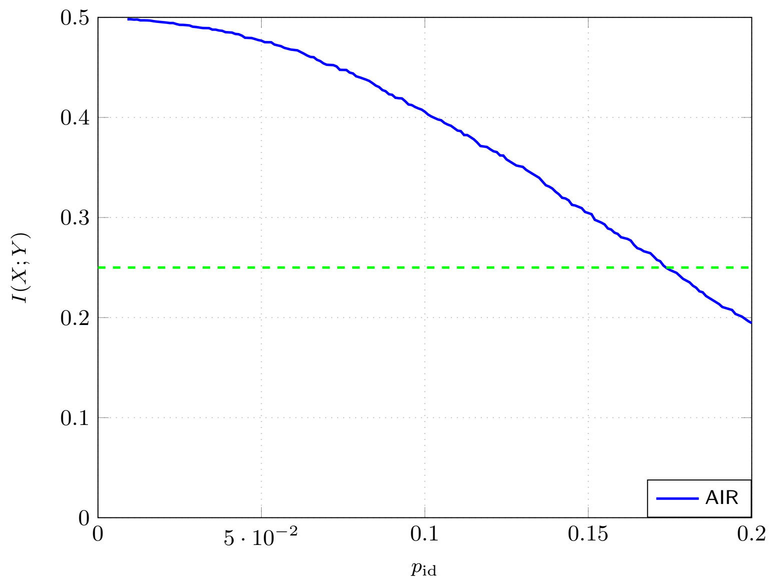


Figure 6.1: The AIR curve for $M = 1$.

Table 6.1: Best thresholds without considering inner-outer iterative decoding.

M	DE	Verified by EXIT	EXIT	Verified by DE
1	0.119	0.122	0.122	0.119
2	0.147	0.156	0.156	0.147
5	0.174	0.193	0.193	0.174

6.2.1 Without Inner-Outer Iterative Decoding

Starting from not considering inner-outer iterative decoding, the optimization based on different methods of threshold computation are implemented, as shown in Table 6.1. The results from one method can be verified using the result from the other method. In other words, the results from different methods have a fixed gap. Combining with the results for the WiMax code shown in Table 5.1, it can be concluded that the gaps are $[0.2, 0.3]$ when $M = 1$, $[0.7, 0.9]$ when $M = 2$ and $[0.17, 0.19]$ when $M = 5$. The gaps increase with the value of M since the Gaussian approximation is less accurate when M increases. Therefore, although the results are different, codes of almost the same performance can be found from code optimization based on different methods. Since the precision is not small enough, several codes are found at the end of the optimization with the same threshold. Table 6.2 presents some examples.

Table 6.2: Degree distributions searched from code optimization without considering inner-outer iterative decoding.

	Code 1	Code 2	Code 3	Code 4	Code 5	Code 6
M	1		2		5	
λ_2	0.267957	0.254924	0.221509	0.223464	0.226126	0.239622
λ_3	0.281633	0.304387	0.300010	0.299172	0.230817	0.328595
λ_4					0.154234	
λ_6	0.072002					
λ_7	0.030845					0.037356
λ_8		0.100373				
λ_9		0.050626	0.112572	0.155971		0.097966
λ_{10}	0.347563	0.28969	0.365908	0.321394	0.388822	0.296462
ρ_3						0.018024
ρ_4					0.023622	
ρ_5	0.058643				0.012706	0.018791
ρ_6	0.081797		0.226706	0.307611	0.273643	0.063051
ρ_7	0.450534	0.880597	0.416592	0.253611	0.312781	0.565518
ρ_8	0.337472	0.071173	0.050688	0.072576		0.183247
ρ_9	0.054467			0.05216	0.131189	
ρ_{10}			0.125118	0.198955	0.085489	0.054837
ρ_{11}	0.017086		0.016263			0.056243
ρ_{12}			0.014595		0.012673	0.022837
ρ_{13}			0.070114	0.017428	0.046185	
ρ_{14}		0.022427	0.079925	0.074	0.101712	
ρ_{15}		0.025803		0.023658		0.017452
$p_{\text{th,DE}}$	0.119		0.147		0.174	
$p_{\text{th,EXIT}}$	0.122		0.155		0.192	

6.2.2 With Inner-Outer Iterative Decoding

If inner-outer iterative decoding is considered at the receiver, optimization based on DE is not feasible because of the complexity and long running time. Therefore, the optimization process in this scenario is based on EXIT charts first, and then DE is utilized to verify the obtained code. The best thresholds based on the values verified by DE are shown in Table 6.3. During the optimization process, we observe that the gaps between the results from the two methods are not fixed within a small range (e.g., the gaps are $[0.02, 0.03]$ when $M = 1$ and $[0.17, 0.19]$ when $M = 5$). In this case, it is possible that the code searched by code optimization based on EXIT charts is not the best one after being verified by DE, and this deviation especially affects the cases for $M \geq 2$. Some examples of degree distributions of optimized LDPC codes are shown in Table 6.4.

Table 6.3: Best threshold searched by optimization procedure considering inner-outer iterative decoding.

M	EXIT	Verified by DE
1	0.175	0.168
2	0.2	0.2
5	0.205	0.21

Table 6.4: Degree distributions searched by code optimization considering inner-outer iterative decoding.

	Code 1	Code 2	Code 3
M	1	2	5
λ_2	0.282575	0.389230	0.788335
λ_3	0.361403	0.247290	
λ_4			0.015214
λ_5		0.069600	
λ_{18}			0.196451
λ_{20}	0.356022	0.293880	
ρ_2			0.103702
ρ_3	0.167580	0.187601	0.241582
ρ_5			0.081554
ρ_6	0.068519		
ρ_8	0.358732	0.483086	0.020628
ρ_9			0.382847
ρ_{11}		0.329313	
ρ_{13}	0.217256		
ρ_{16}	0.074224		0.169687
ρ_{18}	0.113689		
$p_{\mathrm{th,DE}}$	0.168	0.2	0.21
$p_{\mathrm{th,EXIT}}$	0.175	0.2	0.205

The optimized codes obtained for $M = 1$ can be compared with the WiMax code in the AIR curve. In Figure 6.2, the red circle on the green dashed line corresponds to the threshold of the optimized code, while the black circle is the WiMax code's threshold. The gap between the two circles is the improvement achieved by the optimization mechanism, which is 0.023 on p_{id}. Note that there is still a gap between the red circle and the AIR curve, which indicates a possible further improvement of around 0.006 on the insertion/deletion probability p_{id}. The optimization mechanism in this thesis is not guaranteed to find the theoretically best code.

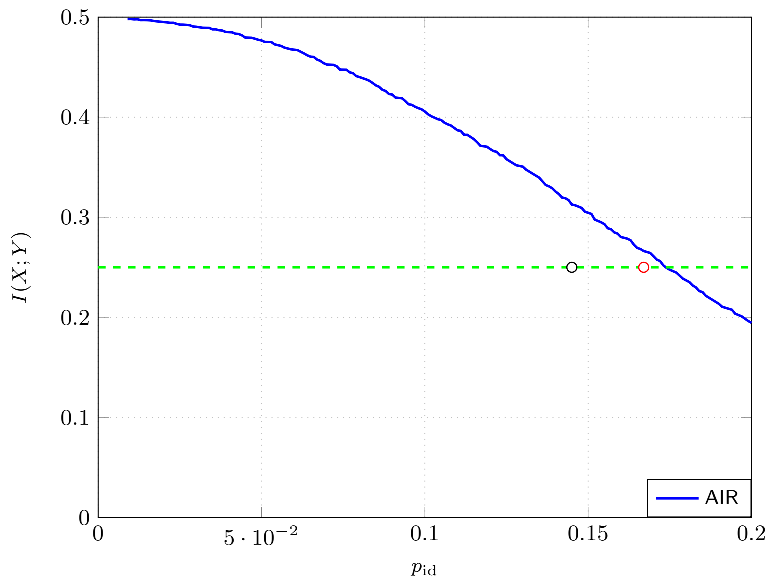
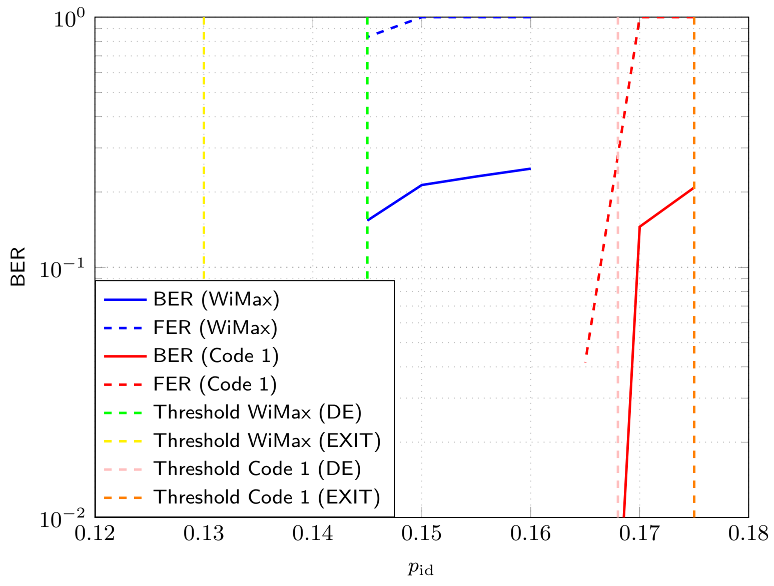


Figure 6.2: Comparing the searched code and the WiMax code to the AIR curve.

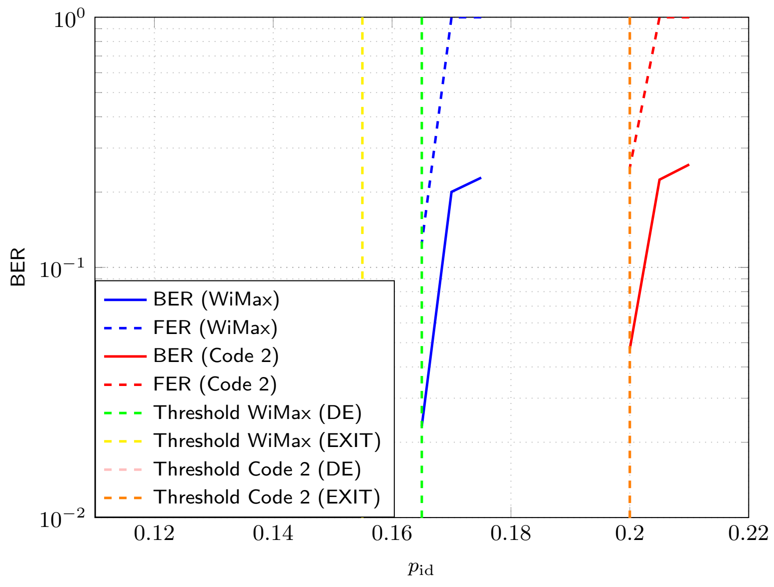
6.3 Error-Rate Results

Since there is no significant difference between the thresholds of the WiMax code and the optimized code without considering inner-outer iterative decoding (only $0.002 - 0.003$), the simulation results for them are similar to the results shown in Figure 5.3. Therefore, only simulation results under inner-outer iterative decoding will be presented in this section.

The simulation results for the optimized codes (code 1 and code 2 in Table 6.4) and the WiMax code are shown in Figure 6.3. The optimized LDPC codes show significant improvements. Besides, comparing the simulation curves with thresholds computed by DE and EXIT charts, we can observe that, as explained, the results from DE are more accurate than the results from EXIT charts.



(a) $M = 1$



(b) $M = 2$

Figure 6.3: Simulation results for the optimized LDPC codes in Table 6.4 comparing with the WiMax code.

7 Conclusion

This thesis started with the introduction of a coding scheme designed for DNA-based storage, where both synchronization and substitution errors are introduced during transmission. The coding scheme consists of two codes, an outer code, which is an LDPC code, and an inner code, which is either a convolutional code or a block code, both combined with a watermark sequence. The purpose of the inner code is to resolve synchronization errors, and the outer LDPC code is utilized for correcting substitution errors. The coding scheme is implemented, tested, and simulated under different conditions, including with or without inner-outer iterative decoding at the receiver side and with single or multiple received DNA sequences.

Moreover, a mechanism for optimizing LDPC codes for this scenario is also proposed in this thesis. The optimization is based on the decoding threshold computed by two different methods: DE and EXIT charts. Although with deviation, the results from the two methods can be compared with each other and with simulation results. The differences in performance between DE and EXIT charts are discussed in the thesis. The optimization based on DE is more accurate than the one based on EXIT charts, while the optimization based on EXIT charts is more efficient. Furthermore, in order to explore the potential of error tolerance for the channel, the method for calculating AIRs based on [23] is presented in this thesis. The method is implemented focusing on the case with inner-outer iterative decoding at the receiver and with a single received sequence. The obtained AIR curve is utilized for evaluating the performance of the optimized LDPC codes.

Specifically, the optimized LDPC codes under no iterations between the inner and outer decoders and a single or multiple received DNA sequences show an improvement of around 0.003 on the insertion/deletion probability compared with the WiMax code. When we consider inner-outer iterative decoding, the optimized LDPC codes show improvements of 0.02 on the insertion/deletion probability compared to the WiMax code. There is a gap between the best threshold and the AIR, which is around 0.007. The optimization process considering inner-outer iterative decoding and a channel with multiple received sequences is also implemented. The best threshold of the optimized LDPC code in this thesis is 0.21 considering inner-outer iterative decoding with 5 received DNA sequences.

Some factors may be taken into account for further work. First, only up to 5 received DNA sequences are simultaneously considered because of the computational complexity. This number could be increased by using devices with more computational power, such as computer clusters. Second, AIR results for multiple received DNA sequences could be calculated in the future. Third, there may be some alternative codes that can improve the performance of the coding scheme, such as spatially-coupled LDPC codes, could be implemented and tested as further work.

Bibliography

[1] M. C. Davey and D. J. C. Mackay. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 47(2):687–698, Feb 2001.

[2] Seagate. Rethink data: Put more of your business data to work – from edge to cloud, 2020. `https://www.seagate.com/gb/en/our-story/rethink-data/`. [Accessed: 2021-06-25].

[3] L. Ceze, J. Nivala, and K. Strauss. Molecular digital data storage using DNA. *Nature Reviews Genetics*, 20(8):456–466, May 2019.

[4] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic. DNA-based storage: trends and methods. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 1(3):230–248, Sep 2015.

[5] S. Taluja, J. Bhupal, and S. R. Krishnan. A survey paper on DNA-based data storage. In *Proc. International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–4, Feb 2020.

[6] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi. Coding over sets for DNA storage. *IEEE Transactions on Information Theory*, 66(4):2331–2351, Dec 2020.

[7] M. Kovačević and V. Tan. Codes in the space of multisets—coding for permutation channels with impairments. *IEEE Transactions on Information Theory*, 64(7):5156–5169, Jul 2018.

[8] M. Kovačević and D. Vukobratovic. Multiset codes for permutation channels. *ArXiv*, abs/1301.7564, 2013.

[9] J. A. Briffa, H. G. Schaathun, and S. Wesemeyer. An improved decoding algorithm for the Davey-MacKay construction. In *Proc. IEEE International Conference on Communications (ICC)*, pages 1–5, May 2010.

[10] G. Ungerboeck. Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, 28(1):55–67, Jan 1982.

[11] W. Ryan and S. Lin. *Channel codes: classical and modern.* Cambridge University Press, 2009.

[12] J. Hou, P.H. Siegel, L.B. Milstein, and D. Pfister. Multilevel coding with low-density parity-check component codes. In *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1016–1020, Nov 2001.

[13] Y. Li and W. E. Ryan. Bit-reliability mapping in LDPC-coded modulation systems. *IEEE Communications Letters*, 9(1):1–3, Jan 2005.

[14] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, Feb 2001.

[15] S. ten Brink, G. Kramer, and A. Ashikhmin. Design of low-density parity-check codes for modulation and detection. *IEEE Transactions on Communications*, 52(4):670–678, May 2004.

[16] A. Lenz, I. Maarouf, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. Graell i Amat. Concatenated codes for recovery from multiple reads of DNA sequences. In *Proc. IEEE Information Theory Workshop (ITW)*, pages 1–5, Apr 2021.

[17] V. Buttigieg and N. Farrugia. Improved bit error rate performance of convolutional codes with synchronization errors. In *Proc. IEEE International Conference on Communications (ICC)*, pages 4077–4082, Jan 2015.

[18] L. Bahl and F. Jelinek. Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *IEEE Transactions on Information Theory*, 21(4):404–411, Jul 1975.

[19] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (Corresp.). *IEEE Transactions on Information Theory*, 20(2):284–287, Mar 1974.

[20] M. C. Davey. *Error-correction using low density parity check codes*. PhD thesis, University of Cambridge, 1999.

[21] D. Semrau, T. Xu, N. Shevchenko, M. Paskov, A. Alvarado, R. Killey, and P. Bayvel. Achievable information rates estimates in optically-amplified transmission systems using nonlinearity compensation and probabilistic shaping. *Optics Letters*, 42(1):121–124, Jan 2017.

[22] J. Hu, T. M. Duman, M. F. Erden, and A. Kavcic. Achievable information rates for channels with insertions, deletions, and intersymbol interference with i.i.d. inputs. *IEEE Transactions on Communications*, 58(4):1102–1111, Apr 2010.

[23] D. M. Arnold, H. A. Loeliger, P. O. Vontobel, A. Kavcic, and W. Zeng. Simulation-based computation of information rates for channels with memory. *IEEE Transactions on Information Theory*, 52(8):3498–3508, Aug 2006.

[24] H. Ding, Q. Liu, Z. Ding, J. Xu, Z. Zhang, W. Liu, and X. Chen. Low complexity iterative receiver with lossless information transfer for non-binary LDPC coded PDMA system. *IEEE Access*, 8:150964–150973, Aug 2020.

[25] IEEE standard for local and metropolitan area networks part 16: Air interface for broadband wireless access systems. *IEEE Std 802.16-2009 (Revision of IEEE Std 802.16-2004)*, pages 1–2080, 2009.

[26] M. El-Hajjar and L. Hanzo. EXIT charts for system design and analysis. *IEEE Communications Surveys & Tutorials*, 16(1):127–153, Mar 2014.

[27] J. Hagenauer. The EXIT chart - introduction to extrinsic information transfer in iterative processing. In *Proc. 12th European Signal Processing Conference (EUSIPCO)*, pages 1541–1548, Sep 2004.

[28] J. Hou, P. H. Siegel, and L. B. Milstein. Performance analysis and code optimization of low density parity-check codes on Rayleigh fading channels. *IEEE Journal on Selected Areas in Communications*, 19(5):924–934, May 2001.

[29] M. Georgioudakis and V. Plevris. A comparative study of differential evolution variants in constrained structural optimization. *Frontiers in Built Environment*, 6:102, Jul 2020.

[30] A. Shokrollahi and R. Storn. Design of efficient erasure codes with differential evolution (ISIT). In *Proc. IEEE International Symposium on Information Theory*, pages 5–, Jun 2000.

[31] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, Jan 1997.

[32] R. Shibata, G. Hosoya, and H. Yashima. Design of irregular LDPC codes without markers for insertion/deletion channels. In *Proc. IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019.