Research on Polar and Decoding for 5G System

Liu Han

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Master of Engineering

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

21/09/2023

Date

Liu Han

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accordance with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

21/09/2023

Date

Erry Gunawan

Assoc Prof. Erry Gunawan

Authorship Attribution Statement

This thesis contains material from one paper published in the following peer-reviewed journal in which I am listed as an author.

Chapter 4 is published as Han Liu, Erry Gunawan, Hu Yaoyue, and Yong Liang Guan, "BP-Based Sparse Graph List Decoding of Polar Codes," *IEEE Communications Letters*, vol. 27, no. 5, pp. 1257-1261, May 2023.

The contributions of the co-authors are as follows:

- A/Prof Erry Gunawan provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Prof Guan Yong Liang and Dr. Hu Yaoyue.
- I co-designed the study with Dr Hu Yaoyue and performed all the programming work at the School of Electrical & Electronic Engineering.
- All performance simulation, including algorithm implementation, was conducted by me in the facility for analysis, characterization, and testing.

21/09/2023 ..... Liu Han

Date Liu Han

Acknowledgements

To begin with, I wish to express my greatest thanks and respect to my supervisor Assoc Prof. Erry Gunawan from School of EEE, Nanyang Technological University (NTU) for his patient instruction and precious suggestions on my research work. He taught me how to start research, how to collect valuable information from a large number of documents, and how to write papers.

Also, I would like to show my sincere appreciation to Prof. Guan Yong Liang at School of EEE, Nanyang Technological University (NTU), and Dr. Hu Yaoyue at National Mobile Communications Research Laboratory, Southeast University (SEU), for giving valuable suggestions on the direction and instructions of my research as well as helping me solve problems during my research.

Finally, I would like to express my gratitude to my parents as well as all my friends. Thanks for their companion and help without any complaints. Their precious love and support are always the motivation of my life.

"One, remember to look up at the stars and not down at your feet. Two, never give up work. Work gives you meaning and purpose and life is empty without it. Three, if you are lucky enough to find love, remember it is there and don't throw it away."

—Stephen Hawking

To my dear family

Contents

List of Figures

List of Tables

Symbols and Acronyms

Symbols

$\mathbb{R}^{T\times T}$	the $T \times T$ Euclidean space
$\oplus$	the addition in Galois field
$\otimes$	the Kronecker product
$\odot$	the convolution product of VNs
$\star$	the convolution product of CNs
$\langle \cdot, \cdot \rangle$	the inner product of two vectors
$\| \cdot \|$	the 2-norm of a vector or matrix in Euclidean space
$\tanh(\cdot)$	the hyperbolic tangent function
$\int_a^b$	the integral function from a to b
$\sum$	the continuous addition
$\prod$	the continuous multiplication
u_i^j	the bit sequence from the i-th bit to the j-th bit
u_i	the i-th component of a bit sequence
$\hat{u}$	the expected bit sequence
$\bar{x}$	the vector with the average of all components of x as each element
$\mathbf{A}^{\otimes n}$	the n-th Kronecker product of a matrix A
$W : X \to Y$	the channel model
W^N	a channel W after N times of use
W_N	a channel combined by N independent channels

$W_N^{(i)}$	the i-th subchannel of a combined channel W_N
$W(\cdot \mid \cdot)$	the channel transition probability of channel W
σ^2	the channel noise variance

Acronyms

5G	5th Generation
eMBB	Enhanced Mobile Broadband
mMTC	massive Machine Type Communication
URLLC	Ultra Reliable Low Latency communication
LTE	Long-Term Evolution
3GPP	3rd Generation Partnership Project
B-DMC	Binary-input Discrete Memoryless Channel
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
LLR	Log-Likelihood Ratio
PDF	Probability Density Function
DE	Density Evolution
GA	Gaussian Approximation
BEC	Binary Erasure Channel
LDPC	Low-Density Parity-Check
BP	Belief Propagation
BPL	BP List
SC	Successive Cancellation
SCL	SC list
CRC	cyclic Redundancy Check
CA-SCL	CRC Aided SCL
BP-SGL	BP-based Sparse Graph List
SNR	Signal Noise Ratio
d-SNR	design SNR

BER	Bit Error Rate
BLER	Block error rate
PE	Processing Element
VN	Variable Node
HVN	Hidden VN
CVN	Channel VN
CN	Check Node
R-message	Left-to-right message
L-message	Right-to-left message
MED	Minimum Euclidean Distance
PM	Path Metric
KPCA	Kernel Principal Component Analysis
NCR	Non-Coverage Rate
SOTA	State-Of-The-Art
FLOP	Floating Point Operation
CC	Clock Cycle

Abstract

As an important technology of the new-generation mobile communication system, polar codes have been selected as the control channel coding standard in the 5th generation (5G) enhanced Mobile Broadband (eMBB) scenario. In the case of finite code length, the channel polarization is incomplete, which causes a negative impact on decoding efficiency. Thus, how to construct an effective polar decoding scheme has attracted researchers in the field of communication.

In this thesis, we first introduce the background of our research, including 5G channel coding schemes, the basic concept of polar codes, and the current polar encoding and decoding algorithms. We then propose an improved belief propagation (BP) based polar decoding algorithm by taking advantage of the low-density parity-check (LDPC) like BP decoding algorithm, graph similarity analysis, and the list decoding scheme. In a more detailed explanation, our approach involves the utilization of both cosine similarity analysis and kernel principal component analysis (K-PCA) methods. These techniques are instrumental in effectively clustering the sparse BP decoding graphs based on their structural similarities. It is important to note that the structural similarity among these graphs plays a pivotal role in determining the overall decoding performance, as it greatly influences how well each decoding graph performs its decoding tasks. A sparse graph list generation algorithm is also presented for the first time. In comparison to some conventional list selection methods used previously, our proposed scheme not only approaches global optimality but also does so with higher computational efficiency. This innovation addresses the challenge of optimizing graph selection while keeping computational requirements reasonable, making it a significant contribution to our decoding framework. Simulation results show that the proposed decoding scheme

can achieve error-rate performance improvement while having low complexity and latency.

Keywords:5G communication system, Channel coding, Polar codes, Belief propagation

Chapter 1

Introduction

In this chapter, we provide the background and motivation of our research by introducing the 5th generation (5G) communication system and 5G polar coding schemes. Then we discuss the objectives and contributions of our research. Finally, the organization of the thesis is presented.

1.1 Background and Motivation

The wireless communication system is an indispensable and important means of information transmission in today's society. The first-generation mobile cellular system provides voice call service through analog communication. The system can communicate flexibly without being limited by a fixed location. The second-generation digital communication system has been significantly upgraded. Mobile phones have introduced call and text encryption, picture messaging, short message service, etc., which changed the traditional communication mode. The Internet access and data transmission rates of the third-generation communication system have greatly improved. The emergence of smartphones has transformed users from being media consumers to becoming content providers, and have promoted the goal of higher throughput of fourth-generation mobile communications.

With the continuous improvement of the communication requirements, 5G mobile Communication officially became commercial in 2020. Based on the higher peak communication rate, 5G puts more emphasis on "Internet of Everything", and builds an intelligent ecosystem where all people and all things can communicate with each other. To this end, 5G considers three flexible application scenarios: enhanced mobile broadband (eMBB), massive machine type communication

(mMTC), and ultra reliable low latency communication (URLLC) [1]. Obviously, these application scenarios put forward higher requirements for various indicators of the communication system. To ensure spectrum efficiency, energy efficiency, and cost-effectiveness, it is necessary to reach a peak rate of 100 Gbps, mobility of 1000 km/h, a delay of 1ms, a user experience rate of 0.1 1 Gbps, a connection density of $10^6/km^2$, and a traffic density of 100 Tbps/km^2 [2]. In addition, the increasing demand for capacity and the shortage of spectrum resources pose great challenges to the key technologies of communication systems.

5G adopts different channel coding methods from Long-Term evolution (LTE) and other communication systems. The actual communication environment is a non-ideal channel, which is mixed with noise and other interference factors. In order to counter noise interference, channel coding technology adds redundancy to the source information to improve transmission efficiency. In general, the more redundant information is added, the stronger the anti-interference ability of codeword transmission, but the lower the channel transmission rate. With the development of the communication system, some outstanding channel coding schemes have been applied to practical systems, such as convolution code, interlaced code, turbo code, etc. The channel coding scheme used in LTE is turbo code, while its throughput is not satisfactory at the long code length [3]. In addition, turbo code adopts iterative decoding. Even when the code length is very short, this method will lead to increased complexity and latency, which is difficult to meet the low latency requirements of 5G. Compared with turbo codes, low-density parity-check (LDPC) codes have better performance in the case of long code lengths, thus it is applied to data channel transmission in 5G eMBB scenario [4].

In 2008, E. Arıkan first proposed the concept of polar code in [5]. It is strictly proved from the mathematical formula that polar code can reach the Shannon limit in a specific channel. In 2016, under the unified comparison standard of multiple operators and terminal manufacturers around the world, polar code was recognized by the 3rd Generation Partnership Project (3GPP) as the channel coding method of the control channel in the 5G eMBB scenario [6]. Although polar code has excellent coding performance, compared with turbo code and LDPC code, the application of polar code in the practical environment is not mature, and there are still many problems to be solved. The successive cancellation list (SCL) decoding [7] and the cyclic redundancy check (CRC) aided SCL (CA-SCL) were designed as the baseline algorithms for polar code, which can achieve satisfactory error-rate performance

for infinite length codewords. However, SCL-based decoding algorithms have high decoding latency and calculation complexity with the increase of code length due to the serial decoding structure. Arıkan later proposed a pipelined implementation of polar code and the belief propagation (BP) polar decoder [8]. Due to its parallel decoding structure, the BP decoder has fewer convergence steps over successive cancellation (SC) based decoding algorithms, but its error-rate performance is worse than SCL-based schemes. Thus, it is necessary to design a polar decoding scheme that offers superior performance while maintaining low complexity.

1.2 Objectives

The objectives of this thesis are listed as follows:

- *Designing a polar decoding scheme with a good tradeoff between performance and complexity*: Researchers have been working on decreasing the memory complexity and decoding latency of the original BP decoder and simplifying the hardware implementation for practical utility. For example, [9] discussed an express polar decoding scheme to reduce decoding complexity by reducing the number of messages exchanged between nodes. Later, a sparse BP decoding method is developed to convert a BP factor graph to the corresponding sparse graph representation (like the LDPC bipartite graph) for polar codes by flattening the factor graph and pruning or merging some edges and nodes that have no contribution to the decoding process. However, due to error accumulation during iterations caused by numerous short loops in sparse graphs, the decoding performance of the LDPC-like BP decoder is not comparable to that of the original BP decoder. It is noticed that sparse graphs converted from different permuted factor graphs may have different structures and thus have different decoding performance due to the change of short loops on graphs [10]. Considering the performance gain obtained by the BPL decoder over the original BP decoder, the list decoding scheme may also help improve the decoding performance of the LDPC-like BP decoder by combining the LDPC-like BP decoder based on different sparse graphs.

- *Proposing an improved list generation algorithm*: For a polar codeword of length $N = 2^n$, whose BP factor graphs contain n stages, there are $n!$ different permuted factor graphs with the same encoding behavior. It is proved

that the different structures of the permuted factor graphs may result in different decoding performance for the same received codeword due to the changes of short loops in these permuted factor graphs [10], [12]. Based on the above observation, authors in [11] proposed a list decoding scheme based on the BP decoding algorithm for polar codes. BP list (BPL) decoder starts with L branches that function as independent parallel BP decoders and output the final decoding result by making a comparison between each result from the corresponding decoding branch to select the best one. The original BPL decoding method generates the list based on the cyclic-shift selection method and selects the final output by the minimum Euclidean distance (MED) criterion. How to effectively select decoding graphs to construct the list is the key factor that determines the decoding performance of the BPL decoder. There are many graph selection methods proposed for BPL generation by setting the reasonable evaluation criterion for factor graphs later, such as error upper-bond based selection [13] and reinforcement learning aided selection [14]. About output selection, a deep learning based scheme has later been proposed in [15] to obtain a more accurate output by comprehensively processing a list of decoding results.

- *Scientifically evaluating the performance of decoding algorithm*: The performance of the proposed decoding algorithm needs to be comprehensively analyzed under different polar code construction parameters, in terms of both error-rate performance, complexity, and latency. By comparing with the typical BP-based polar decoding algorithms and some improved schemes, the superiority of the proposed decoding scheme should be clearly reflected.

1.3 Major Contribution

The major contribution of this thesis is the proposed polar decoding scheme, named BP-based sparse graph list (BP-SGL) decoding algorithm. To generate the proposed decoder, a sparse list generation algorithm is presented for the first time. The outstanding features of the proposed BP-SGL decoding scheme are summarized as follows:

- Compared with the traditional BP decoding schemes and some current state-of-the-art (SOTA) BP-based decoding algorithms, the proposed BP-SGL decoder has performance gains accross a wide range of signal-to-noise ratios (SNRs).

- Due to the parallel decoding framework and single-stage decoding structure, the proposed BP-SGL exhibits significantly reduced latency compared to the original n-stage BP decoding algorithms. Moreover, when compared to the original BPL decoding, which also belongs to the list structure, the proposed BP-SGL consumes much lower computing power consumption.

- The proposal BP-SGL can achieve the globally optimal property. The previous BPL construction methods can only generate a locally optimal list by ranking each permuted BP factor graph. we propose a sparse graph selection scheme to construct a sparse graph list with strong overall performance. Cosine similarity analysis and kernel principal component analysis (KPCA) method are used to cluster the decoding sparse graphs. The sparse graph list is constructed by selecting one graph from each group. Testing results confirm the efficiency of the proposed list generation scheme.

1.4 Organization of the Thesis

The organization of this thesis is as follows. Chapter 1 introduces the background of our research, including the digital communication system, channel models, and 5G communication technical requirements. We analyze the limitations of some current 5G polar coding schemes as well. In Chapter 2, the basic concept and construction principle of polar codes are discussed. Different channel polarization schemes are also analyzed and compared. In Chapter 3, Two kinds of polar decoding algorithms, namely SC-based decoding algorithms and BP-based decoding algorithms, are introduced. Some improved BP-based decoding schemes are also investigated and compared. Chapter 4 proposes the BP-SGL decoder by taking advantage of the list decoding scheme and the LDPC-like BP decoding algorithm to achieve error-rate performance improvement while having low complexity and latency. The key idea of the proposed list construction method is the similarity comparison of decoding sparse graphs. Testing results verify that selecting graphs with significant structural differences is helpful in constructing a list with good

overall performance. Simulation results show that the proposed BP-SGL decoding scheme can achieve performance gains over traditional BP decoding and some SOTA BP-based decoding algorithms, with a significant reduction in complexity and latency. Finally, Chapter 5 makes a summary of the research at this stage, analyzes some open problems to be solved, and provides the recommendations for the future work.

Chapter 2

Basic Theory of Polar Codes

In this chapter, we introduce the basic concept of polar codes. Firstly, we give the principle of the polarization phenomenon. Then, we present some typical polar code construction methods. Finally, the polar code encoding scheme is introduced.

2.1 Channel Polarization Phenomenon

By combining and splitting N ($N = 2^n$) independent channels, N new interdependent channels can be obtained. With the increase of code length, the newly split channels will develop into two extremes. Part of them will be close to the perfect channels, i.e., noiseless channels with a channel capacity close to "1". The others will be close to the completely noisy channels with a channel capacity close to "0". This process of polarization of channel capacity is called the channel polarization phenomenon. As can be seen from the above description, channel polarization consists of two steps, that is channel combining and channel splitting.

The details of channel combining and channel splitting will be introduced in the following parts. In this section, a channel is denoted as $W : X \to Y$, where X and Y represent channel input and channel output, respectively. $W(y|x)$ is used to express the channel transition probability, where $x \in X$ ($x \in \{0, 1\}$), $y \in Y$. A channel W after N times of use is expressed as $W^N : X^N \to Y^N$, and the channel transition probability of W^N is $W^N(y_1^N|x_1^N)=\prod_{i=1}^{N} W(y|x)$. $W_N : X^N \to Y^N$ denotes that N original independent channels are combined into one channel.

2.1.1 Channel Combining

Channel combining is denoted as $W_N : X^N \to Y^N$, which can be achieved iteratively. When there is only one channel, $W_1 \triangleq W, W_1 : X \to Y$. If there are two independent channels, the channel combined with them is $W_2 : X^2 \to Y^2$. The channel transition probability of W_2 is

$$W_2(y_1^2|u_1^2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2). \tag{2.1}$$

Further, combining two W_2 channels can obtain $W_4 : X^4 \to Y^4$. The channel transition probability of W_4 can be calculated as

$$W_4(y_1^4|u_1^4) = W_2(y_1^2|u_1 \oplus u_2, u_3 \oplus u_4)W_2(y_3^4|u_3, u_4). \tag{2.2}$$

Figure 2.1 and Figure 2.2 show the above two processes of channel combining.

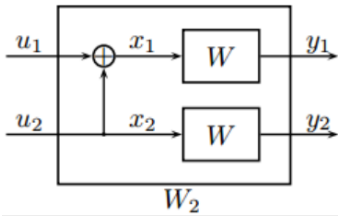

FIGURE 2.1: Channel combining for W_2.

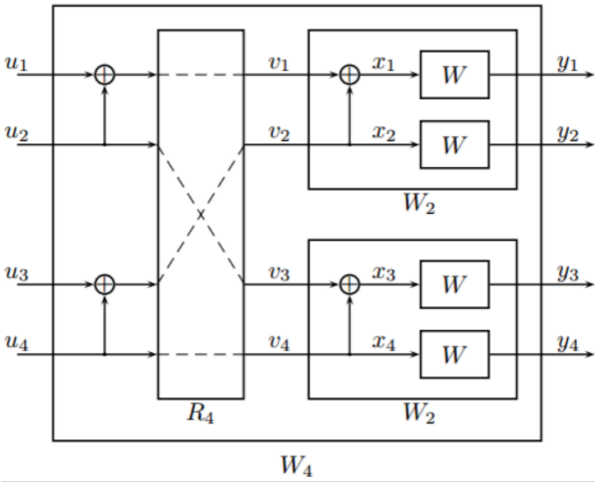

FIGURE 2.2: Channel combining for W_4.

When the number of channels is N, the general expression of channel combining can be presented, as shown in Figure 2.3. The inputting bit sequence u_1^N first transformed into s_1^N ($s_{2i-1} = u_{2i-1} \oplus u_{2i}$, ($s_{2i} = u_{2i}$, $1 \le i \le N/2$). Then, s_1^N is

mapped into x_1^N by bit order processing R. Finally, x_1^N will be inputted into two $W_{N/2}$ channels. The transformation from u_1^N to x_1^N describes the mapping from the combining channel W_N to N original channels W, which will be further discussed in Section 2.3).

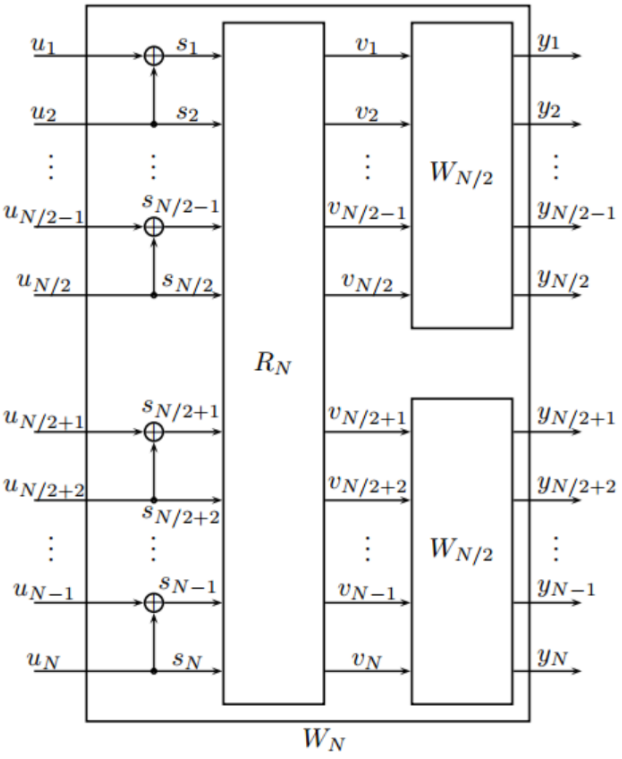

FIGURE 2.3: Channel combining for W_N.

2.1.2 Channel Splitting

Channel splitting refers to the process of converting the combining channel W_N into N independent sub-channels $W_N^{(i)} : X \to Y^N \times X^{i-1}, 1 \leq i \leq N$. The channel transition probability of $W_N^{(i)}$ can be obtained by

$$W_N^{(i)}(y_1^N, u_1^{i-1}|u_i) \triangleq \sum_{u_{i+1}^N \in X^{N-i}} \frac{W_N(y_1^N|u_1^N)}{2^{N-1}}, \tag{2.3}$$

where, u_i and (y_1^N, u_1^{i-1}) denote the input and output of channel $W_N^{(i)}$, respectively. The equation expresses that the value of u_i can be estimated, on the condition that the input bit u_i and the channel transition probability of channel $W_N^{(i)}$ from u_1^{i-1} to y_1^N are known.

To sum up, channel polarization can realize the transformation of channel capacity [16]. Assuming that the binary input symmetric capacity of the original channel W is denoted as $I(W)$, when the code length N approaches infinity, the proportion of split channels with channel capacity approaching "1" is approximately $N \cdot I(W)$, and the proportion of channel capacity approaching "0" is approximately $N \cdot (1 - I(W))$. For a reliable channel with a channel capacity of "1", the information bits can be placed directly without any encoding, that is, the code rate is "1". For unreliable channels with a channel capacity of "0", the frozen bits known to both the sender and the receiver can be placed, that is, the code rate is "0". Thus, when the code length approaches infinity, the achievable code rate of polar codes is $N \cdot I(W)/N = I(W)$, which indicates that, in theory, polar codes can be proved to reach channel capacity.

2.2 Polar Code Construction Methods

Based on the principle of channel polarization, N independent channels are converted into N interdependent channels and ranked according to channel capacity. The K most noiseless bit channels will be used for information bits transmission and the other $N-K$ bits will be set as the known frozen value to construct a source bit sequence u_1^N.

Polar codes are constructed based on the concept of channel polarization. As previously introduced, channel polarization includes channel combining and channel splitting. Through these two steps, the channel capacity of each subchannel will present a trend of two-stage differentiation. With the increase of code length (i.e. the number of the original independent channels), the capacity of part of the subchannels will tend to "1", while the capacity of the remaining subchannels will tend to "0". Polar code uses this phenomenon of channel polarization to transmit information bits on K subchannels with capacity tending to "1" (i.e. the index set of these K subchannels is represented by $\mathcal{A}$), and transmit frozen bits on the other $(N - K)$ subchannels (that is, fixed bits known to both transmitter and receiver, usually set to all zero). The codeword thus formed is the source bit sequence, and the code rate can be denoted as $R = K/N$. In this section, several methods used to calculate the subchannel capacity and thus determine $\mathcal{A}$ will be described.

2.2.1 Bhattacharyya Parameter

E. Arıkan proposed the Bhattacharyya parameter method in [5], which can evaluate the reliability of each subchannel for binary erasure channel (BEC) conditions. BEC is a subset of the binary-input discrete memoryless channel (B-DMC). For a B-DMC W, there are two important channel parameters, which are symmetric capacity

$$I(W) \triangleq \sum_{y\in Y}\sum_{x\in X}\frac{1}{2}W(y|x)\log\frac{W(y|x)}{\frac{1}{2}W(y|0)+\frac{1}{2}W(y|1)}, \tag{2.4}$$

and Bhattacharyya parameter

$$Z(W) \triangleq \sum_{y\in Y}\sqrt{W(y|0)W(y|1)}. \tag{2.5}$$

The Bhattacharyya parameter is equal to the erasure probability of BECs [17].

$I(W)$ denotes the measure of channel rate, which is the maximum rate of reliable transmission of channel W under equal probability input. The higher the value of $I(W)$, the better the transmission capacity of the channel. $Z(W)$ represents the channel reliability, which is the upper limit of the maximum likelihood decision error probability when channel W only transmits "0" or "1". Thus, contrary to $I(W)$, the greater $Z(W)$ is, the less reliable the channel is. The value range of $I(W)$ and $Z(W)$ is [0,1]. The mathematical relationship between them is $I(W_N^{(i)}) = 1 - Z(W_N^{(i)})$. It indicates that If and only if $Z(W_N^{(i)}) = 0$, $I(W_N^{(i)}) = 1$; If and only if $Z(W_N^{(i)}) = 1$, $I(W_N^{(i)}) = 0$. $Z(W)$ of each subchannel can be recursively obtained by

$$\begin{cases} \mathbf{Z}(W_N^{(2i-1)}) = 2\mathbf{Z}(W_{N/2}^{(i)}) - (\mathbf{Z}(W_{N/2}^{(i)}))^2, \\ \mathbf{Z}(W_N^{(2i)}) \leq (\mathbf{Z}(W_{N/2}^{(i)}))^2. \end{cases} \tag{2.6}$$

After obtaining $Z(W)$ of all sub-channels, the information subchannel index set $\mathcal{A}$ can be formed by sorting subchannels and selecting K subchannels with the lowest $Z(W)$. The channel transmission rate can be improved by transmitting information bits on these subchannels with smaller $Z(W)$.

2.2.2 Monte Carlo Simulation

Arıkan further proposed a polar code construction method based on Monte-Carlo simulation, to achieve the sub-channel reliability estimation for non-B-DMC channels.

Monte Carlo simulation is essentially a numerical statistical method based on the theory of probability and statistics. When applied to the construction of polar codes, the specific steps are as follows. Under a specific SNR, assume that the transmitter sends a sequence of zero bits with a length of N, and the sequence is known by the receiver. When decoding the bit from a certain subchannel and the decoding of the previous bits are all correct, if the decoding decision of the receiver is "1", it indicates that there is an error in this channel decoding, and that will be counted into the number of errors. After all testings are completed, calculate the ratio of the error number of each subchannel to the total simulation number. This ratio can be used to roughly estimate the error probability of each subchannel. If the error rate of one subchannel is small, it means that this subchannel is reliable.

The accuracy of the Monte Carlo simulation method for polar code construction is strongly affected by SNR and the number of simulations. Especially in the case of high SNR, if the number of simulations is not enough, the performance of this construction method will be degraded. The more the number of simulations, the higher the accuracy of the Monte Carlo simulation construction method, but it also in the meanwhile means high time/calculation complexity.

2.2.3 Density Evolution

Density evolution (DE) is a method to analyze the progressive performance of the modern efficient error correction and coding methods. Polar code construction by DE method was proposed in [18] and can be generally applied to non-B-DMC channels. DE is calculated based on the Tanner graph, as shown in Figure 2.4. Tanner graph consists of variable nodes (VNs) and check nodes (CNs). The message of VN is represented by the form of the log-likelihood ratio (LLR) as

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) = \log \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)}. \tag{2.7}$$

In the Tanner graph, the LLR value corresponding to each VN can be calculated by the following recursive formula

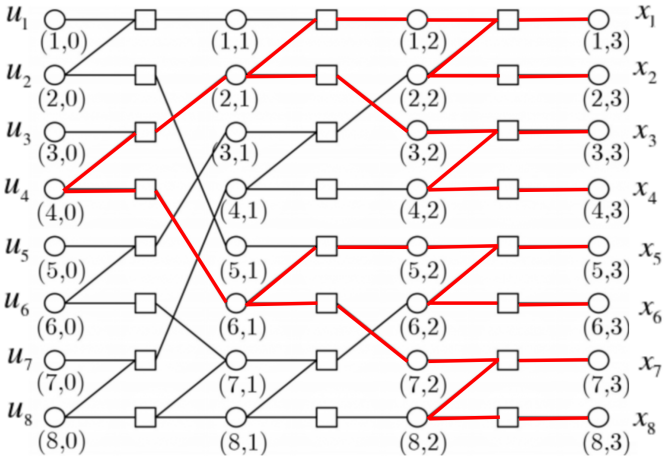

FIGURE 2.4: DE iteration based ob Tanner graph of $\mathcal{P}(8,4)$.

$$\begin{cases} L_N^{(2i-1)} = 2\tanh^{-1}\left(\tanh\left(\frac{L_{N/2}^{(i)}(y_1^{N/2},\hat{u}_{i,o}^{2,2i-2}\oplus\hat{u}_{i,e}^{1,2i-2})}{2}\right)\right)\cdot\tanh\left(\frac{L_{N/2}^{(i)}(y_{N/2+1}^{N},\hat{u}_{i,e}^{2i-2})}{2}\right), \\ L_N^{(2i)} = L_{N/2}^{(i)}(y_{N/2+1}^N,\hat{u}_{i,e}^{2i-2}) + (-1)^{\hat{u}_{2i-1}}\cdot L_{N/2}^{(i)}(y_1^{N/2},\hat{u}_{1,0}^{2i-2}\oplus\hat{u}_{1,e}^{2i-2}). \end{cases} \tag{2.8}$$

The decision formula of the expected bit is

$$\begin{cases} \hat{u}_i = 0, & L_N^{(i)}(y_1^N,\hat{u}_1^{i-1}) \geq 0, \\ \hat{u}_i = 1, & L_N^{(i)}(y_1^N,\hat{u}_1^{i-1}) < 0. \end{cases} \tag{2.9}$$

If LLR is regarded as a random variable, its probability density function (PDF) can be represented by $\rho(z)$. As mentioned above, if the transmitter sends all "0" bits and the receiver determines one bit as "1", it means that the decision is wrong. Thus, The error decision probability of a VN is denoted as

$$P_e = \int_{-\infty}^{0} \rho(z)\mathrm{d}z. \tag{2.10}$$

According to Figure 2.4, when estimating the i-th bit channel, take u_i as the root node and extend to the right to form a decoding tree. Let ρ_x denote the PDF of LLR of the rightmost bits, and $\rho_N^{(i)}$ denote the PDF corresponding to $L_N^{(i)}(y_1^N,\hat{u}_1^{i-1})$. On the premise that u_1^{i-1} is known, LLR of the i-th polarization subchannel $L_N^{(i)}$ can be obtained by

$$\begin{cases} \rho_N^{(2i-1)} = \rho_{N/2}^{(i)} \odot \rho_N^{(i)}, \\ \rho_N^{(2i)} = \rho_{N/2}^{(i)} \star \rho_N^{(i)}, \\ \rho_1^{(1)} = \rho_x \end{cases} \tag{2.11}$$

where, $\odot$ and $\star$ denote the convolution operation of VNs and CNs, respectively.

Based on DE method, the threshold value of the codeword transmission in the corresponding polarization subchannel can be calculated. The higher the threshold value, the higher the reliability of this codeword transmission, that is, the stronger its ability to withstand noise. By taking advantage of this, the reliability of each subchannel can be calculated and the polar code can then be constructed.

2.2.4 Gaussian Approximation

In the practical application, DE has high computational complexity and storage requirements. In order to ensure the coding performance under the Gaussian white noise channel with lower complexity, the Gaussian approximation (GA) method for polar code construction was proposed in [19]. GA is an approximate method of DE. Its basic idea is to simplify the infinite-dimensional message density in DE to Gaussian distribution or Gaussian mixture distribution for analysis. In linear design, this can not only simplify the problem but also greatly improve the accuracy of approximation [20].

Assuming that the transmitted signal passes through an additive white Gaussian noise (AWGN) channel, its noise variance is σ^2, and binary phase shift keying (BPSK) modulation is used. The received bit sequence y can be represented as

$$y = (1 - 2x) + z, \tag{2.12}$$

where, x denotes transmitted bit sequence, and z denotes channel noise which obeys Gaussian distribution (i.e. $z \sim N(0, \sigma^2)$). When transmitted bits are all zero, which means that $y = 1 + z$, the LLR of y is

$$\mathrm{LLR}(y) = \log \frac{P(y|x=0)}{P(y|x=1)} = \frac{2}{\sigma^2} y, \tag{2.13}$$

where, $y \sim N(1, \sigma^2)$. It can be found that LLR$\sim N(2/\sigma^2, 4/\sigma^2)$. In the decoding stage, the LLR of the expected bit $\hat{u}_i$ can be calculated as

$$\mathrm{LLR}_N^{(i)} = \log \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|u_i = 1)}. \tag{2.14}$$

From this, $\mathrm{LLR}_N^{(i)}$ also obeys Gaussian distribution, whose variance is twice the average, that is $D(\mathrm{LLR}_N^{(i)}) = 2E(\mathrm{LLR}_N^{(i)})$. According to the GA theorem, it can be calculated that

$$E(\mathrm{LLR}_N^{(1)}) = \phi^{-1}\{1 - [1 - \phi(\frac{2}{\sigma^2})]^N\}, \tag{2.15}$$

where, $\phi(x)$ is shown in the following formula

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{-\infty}^{+\infty} \tanh\frac{u}{2} e^{-\frac{(4-x)^2}{4x}} \mathrm{d}u, & x > 0 \\ 1, & x = 0 \end{cases} \tag{2.16}$$

and can be simplified as

$$\phi(x) = \begin{cases} e^{-0.45627x^{0.86}+0.0218}, & 0 < x < 10 \\ \frac{\pi}{x}(1 - \frac{10}{7x})e^{-\frac{x}{4}}, & x \geq 10 \end{cases} \tag{2.17}$$

By iteration, $E(\mathrm{LLR}_N^{(i)})$ can be obtained according to

$$\begin{cases} E(\mathrm{LLR}_N^{(2i-1)}) = \phi^{-1}\left(1 - \left(1 - \phi\left(E(\mathrm{LLR}_{N/2}^{(i)})\right)\right)^2\right), \\ E(\mathrm{LLR}_N^{(2i)}) = 2E(\mathrm{LLR}_{N/2}^{(i)}). \end{cases} \tag{2.18}$$

The initial condition is $\mathrm{LLR}_1^{(1)} = \frac{2}{\sigma^2}$.

The bit error rate (BER) performance formula under BPSK modulation is

$$P(\gamma_i) = \frac{1}{2} erfc(\sqrt{\mathrm{SNR}}), \tag{2.19}$$

where, $erfc(x)$ denotes the complementary error function and it is known that SNR $N(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$. Thus, the above formula can be transformed as

$$P(\gamma_i) = \frac{1}{2} erfc(\frac{\sqrt{E}}{2}). \tag{2.20}$$

Define $P(\varepsilon)$ as an event that there are bit errors occurring in the codeword, It is easy to obtain that

$$P(\varepsilon) = 1 - \prod(1 - P(\gamma_i)). \tag{2.21}$$

It means that the smaller $P(\gamma_i)$, the smaller $P(\varepsilon)$. Considering that $erfc(x)$ is a monotone decreasing function, the larger E, the smaller the error probability $P(\gamma_i)$. Therefore, by calculating E(LLR) of subchannels according to Eq.(2.18), the K most reliable subchannels can be picked out.

2.2.5 Design SNR

The design SNR (d-SNR) can help improve the construction efficiency of polar codes. No matter what construction method is adopted, the selection of information bits is always closely related to the value of the current SNR. When simulating the performance of polar code, SNR is often used as the abscissa of the simulation diagram, and BER or block error rate (BLER) is usually used as the ordinate. In theory, each time a new simulated SNR is brought in, a new polarization channel index set $\mathcal{A}$ must be reconstructed. This traditional polar construction method is called point-by-point construction. However, this repetitive work significantly increases the time complexity and computing power consumption of polar code construction.

It is found [21] that there is always a perfect SNR for a specific code rate to construct $\mathcal{A}$, which is defined as d-SNR. The d-SNR is not calculated, but an empirical value based on values of a large number of simulations. After determining d-SNR, it only needs to substitute this SNR to construct the polarization channel index set $\mathcal{A}$ once, and then store it. In the future, at this bit rate, no matter which SNR value is used for simulation, the polar code can be constructed by directly placing the current message into the stored information bit index. Since the information bit index set is constructed only once, the time/calculation complexity can be substantially reduced.

2.3 Polar Code Encoding Principle

Polar code with code length N and K information bits, denoted as $\mathcal{P}(N,K)$, is generally constructed by a generator matrix $\mathbf{G}_N$, and can be expressed [5] as

$$x_1^N = u_1^N \cdot \mathbf{G}_N. \tag{2.22}$$

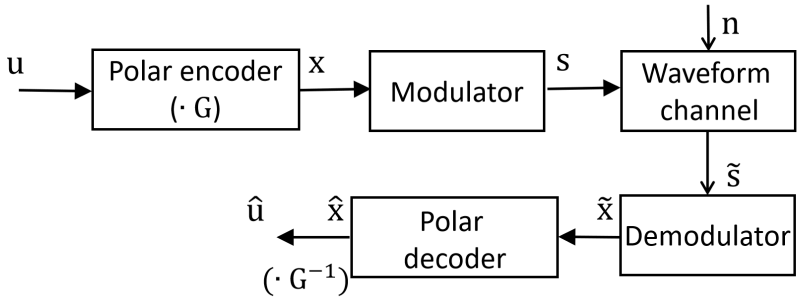

FIGURE 2.5: System diagram considered in this thesis. We employ binary discrete sequences as the signal model, with **u** representing the information bit sequence and **x** representing the coded bit sequence. BPSK modulation and AWGN channels are mainly considered in this work.

The generator matrix $\mathbf{G}_N$ can be further presented by

$$\mathbf{G}_N = \mathbf{B}_N \cdot \mathbf{F}^{\otimes n}, \tag{2.23}$$

where, $\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ is the Arıkan's kernel, which can be obtained from recursive formula $\mathbf{F}^{\otimes n} = \mathbf{F} \otimes \mathbf{F}^{\otimes n-1}$. $\mathbf{B}_N$ is the sorting matrix to complete the reverse bit reordering operation. Reverse bit reordering needs the following steps. First, the decimal sequence number $i \in \{1, 2, ..., N\}$ of each original sequence is expressed as $(i-1) \rightarrow \{b_n, b_{n-1}, ..., b_1\}$ in binary, where b_n is the most significant bit. Then reverse the above binary sequence to get $\{b_1, b_2, ..., b_n\}$. Finally, take b_1 as the most significant bit and re-present the sequence number in decimal by $\{b_1, b_2, ..., b_n\} \rightarrow (j-1)$, to make the value of the i-th element of the current output sequence the j-th element of the original sequence. The recursion of $\mathbf{B}_N$ is defined as

$$\mathbf{B}_N = \mathbf{R}_N(\mathbf{I}_2 \otimes \mathbf{B}_{N/2}), \tag{2.24}$$

where $\mathbf{I}_2$ is a two-dimensional unit matrix, and $\mathbf{B}_2 = \mathbf{I}_2$. The matrix $\mathbf{R}_N$ is a permutation matrix, which separates the odd-order elements and even-order elements from the input sequence, that is, the odd-order elements are sorted first, and then even-order elements, which can be described as $\{b_1, b_2, b_3, b_4, ..., b_N\} \times \mathbf{R}_N = \{b_1, b_3, b_5, ...b_{N-1}, b_2, b_4, b_6..., b_N\}$. In the subsequent studies, $\mathbf{B}_N$ is usually ignored when constructing generator matrix $\mathbf{G}_N$, and thus $\mathbf{G}_N = \mathbf{F}^{\otimes n}$. We follow this polar encoding scheme in our research.

Chapter 3

Polar Code Decoding Schemes

3.1 SC-based Decoding Algorithms

In this chapter, we provide an overview of prevalent polar code decoding approaches, including SC-based decoding and BP-based decoding. The strengths and weaknesses of each algorithm will also be elaborated.

3.1.1 SC Decoding Algorithm

The SC decoding algorithm was proposed in [5] by Arıkan as the baseline decoding scheme for polar codes. SC decoding uses the recursive calculation to decode all bits in turn. A specific bit is decoded based on the assumption that the previous bit is decoded correctly. In the SC decoding graph, LLR information and estimated values of previous bits are transmitted from the parent node to the child node. The bit decision rule is as follows

$$\hat{u}_i = \begin{cases} h(L_N^{(i)}(y_1^N), \hat{u}_1^{i-1}), & i \in \mathcal{A} \\ 0, & i \notin \mathcal{A} \end{cases} \tag{3.1}$$

where, $h(x)$ denotes the bit decision function of information bits. The LLR value of information bit is defined as the ratio of the probability that "0" is transmitted to the probability that "1" is transmitted under the condition that $(y_1^N, \hat{u}_1^{i-1})$ is known. The value of the LLR can be calculated as

$$\text{LLR} = \log\left(\frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1})|0}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1})|1}\right) \tag{3.2}$$

From this, it can be seen that if LLR ≥ 0, the probability that "0" is transmitted is greater than the probability that "1" is transmitted. Thus, the decoding algorithm will determine that the bit received is "0". The bit decision function $h(x)$ can be expressed as

$$\hat{u}_i = \begin{cases} 1, & L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geq 0 \\ 0, & L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) < 0 \end{cases} \tag{3.3}$$

In specific calculation, the value of $L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})$ can be obtained by $f(x)$ and $g(x)$, which are represented as follows

$$f(x, y) = \log \frac{1 + e^{x+y}}{e^x + e^y}, \tag{3.4}$$

$$g(x, y, u_k) = (-1)^{u_k} x + y. \tag{3.5}$$

Thus, the recursive formula of LLR can be calculated as

$$\begin{cases} L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) = f(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}), L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})), \\ L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = g(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}), L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}), \hat{u}_{2i-1}). \end{cases} \tag{3.6}$$

The initial item is that $L_1^{(1)}(y_i) = \log \frac{W(y|0)}{W(y|1)}$. According to Gaussian approximation theory, the LLR of y can be calculated by

$$L(y) = \log \frac{p(y|0)}{p(y|1)} = \frac{2y}{\sigma^2}, \tag{3.7}$$

where, $p(y|x)$ and σ^2 denote the Gaussian posterior distribution probability and the channel noise variance, respectively.

The whole process of the SC decoding algorithm can be presented through Figure 3.1. The rightmost column is the LLR of the received bit sequence, i.e. $\frac{2y_1^N}{\sigma^2}$. The leftmost column represents the source bit sequence $\hat{u}_1^N$ to be decoded. In the decoding stage, the LLR values at each node are calculated in turn, and the LLR updates at each node are carried out in the order from right to left. After all nodes are traversed, $L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})$ can be obtained, based on which $\hat{u}_i$ can be estimated. If the LLR ≥ 0, $\hat{u}_i = 0$, otherwise $\hat{u}_i = 1$. Since SC decoding features the serial decoding structure which performs decoding in the order of accepting bits,

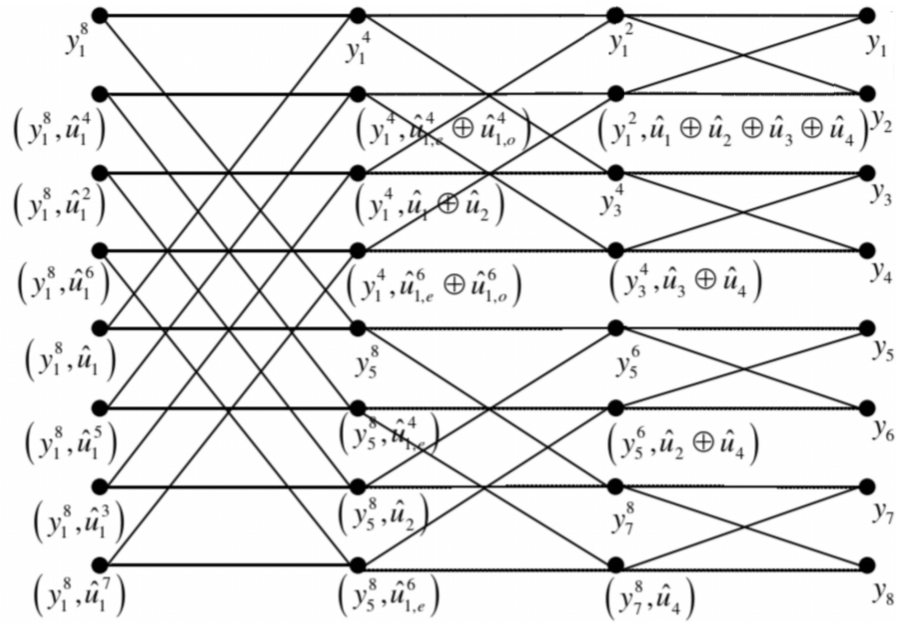

FIGURE 3.1: SC decoding diagram for $\mathcal{P}(8, 4)$.

the longer the code length, the longer the time-spending will be, which will have a great negative impact in practical applications with high real-time requirements.

In addition, when SC decoding algorithm makes decoding decisions for each bit, it needs to assume that the decoding results of the previous bits are correct. However, in the case of finite code length, due to incomplete channel polarization, there will still be some information bits that cannot be decoded correctly. When an error occurs in the decoding of the previous information bits, it will lead to a more serious error delivery since the SC decoder needs to use the estimated values of the previous information bits when decoding the next information bit. Therefore, for polar codes with finite code length, the SC decoder may not achieve ideal decoding performance.

3.1.2 SCL Decoding Algorithm

To solve the problem that SC is prone to error propagation when channel polarization is insufficient, [22] proposed the SCL decoding scheme.

According to the decoding principle, the SC decoding process can also be represented by a binary tree $\mathcal{T}(E, V)$, where E and V denote the edge set and node set in the tree, as shown in Figure 3.2. The depth of a node, denoted as d is defined as the shortest path length from the node to the root node. For polar

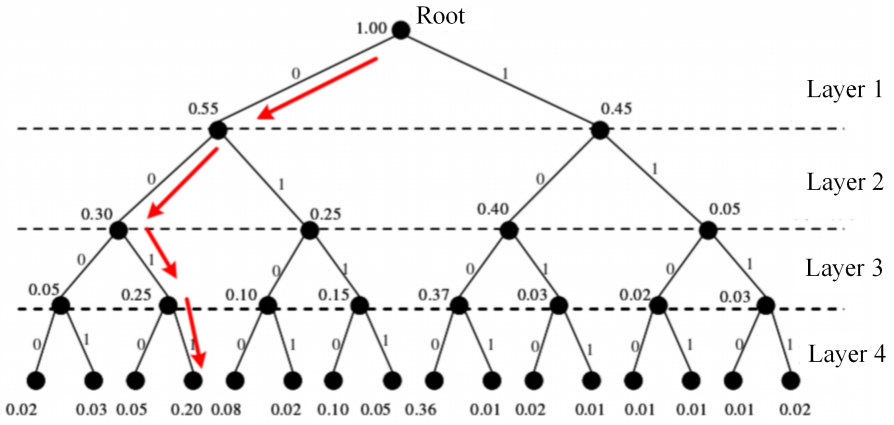

FIGURE 3.2: Binary tree decoding form for SC decoding.

codes with code length N, the node set V of the binary tree can be divided into $N+1$ subsets according to the depth d, which is recorded as V_d (d = 0,1,...,N). V_0 includes root node only (i.e. $|V_0| = 1$). In addition to the leaf node that is the node whose depth is N, each node $v \in V_d$ in the binary tree is connected to two subsequent nodes belonging to V_{d+1} through two edges marked with "0" and "1" respectively. The value of the bit sequence u_1^N corresponding to a node v is defined as the marking sequence of all edges from the root node to this node v. For example, if a node v represents the bit sequence u, its left and right successor nodes represent paths $(u_1^i, u_1^{i+1}) = 0$ and $(u_1^i, u_1^{i+1}0) = 1$, respectively. Thus, each path from root node to node $v \in V_d$ with depth d represents one possible value of u_1^d. Define the set of edges connecting nodes with depth $i-1$ and i as the i-th edge, denoted as E_i. It is easy to find that $E_i = 2^i$ for any $i \in 1, 2, ..., N$. Each path from the root node to any node corresponds to a metric. Therefore, The decoding process is actually the process of finding the appropriate path on the binary tree. At each node, the connected subsequent edge with the highest transfer probability will be selected as part of the path. For example, in Figure 3.2, the expected bit sequence is $\hat{u}_1^4 = [0, 0, 1, 1]$.

SC decoding algorithm only retains the path with the maximum transfer probability when decoding each bit, and judges this bit directly. To improve the decoding performance, SCL increases the number of candidate paths allowed to be retained after each layer of path search, from only allowing the best path for next expansion to the best L path for the next expansion, and no hard decision is directly made on the current decoding bit. Similar to SC decoding algorithm, SCL decoding algorithm still starts from the root node of the binary tree and searches the path layer by layer to the leaf node layer. The difference is that after each layer

is expanded, the subsequent paths are reserved as much as possible (the number of paths reserved for each layer should not be greater than L).

When $L = 1$, which means that only one path is reserved for each layer of the decoding tree, SCL algorithm degenerates into SC algorithm. When $L \geq 2^N$, all paths of the decoding tree are reserved, that is, maximum likelihood decoding. It can be observed that SCL decoding actually includes the above two decoding methods. When $2 \leq L < 2^N$, how to select the best L paths to be reserved from all $2L$ paths is the most important issue. Obviously, it is not feasible to only rely on the LLR of each bit, thus it is necessary to select more appropriate indicators to evaluate the cost of each path. In SCL decoding algorithm, this indicator is expressed as path metric (PM).

For any path $l \in \{1, 2, ..., L\}$ and any received bit $i \in \{1, 2, ..., N\}$ during SCL decoding, PM can be calculated by

$$\mathrm{PM}_l^{(i)} = \sum_{j-1}^{i} \ln(1 + e^{-(1-2\hat{u}_j[l]) \cdot L_N^{(j)}[l]}), \tag{3.8}$$

where, $L_N^{(j)}[l] = \log\left(\frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}[l]|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}[l]|1)}\right)$. Assuming the probability of "0" and "1" bits sent by the transmitter is the same, for any two different $l_1, l_2 \in \{1, 2, ..., L\}$, if and only if $\mathrm{PM}_{l_1}^{(i)} > \mathrm{PM}_{l_2}^{(i)}$, the following formula is established

$$W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}[l_1]|\hat{u}_1^i[l_1]) < W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}[l_2]|\hat{u}_1^i[l_2]). \tag{3.9}$$

From Eq. (3.9), the PM is inversely proportional to the path transition probability. For any decoding path, the greater the transfer probability, the higher the probability that the path is the correct path, and the smaller the PM value. Based on this, the best L path can be selected from $2L$ paths for subsequent decoding. Eq. (3.8) can be further simplified as

$$\mathrm{PM}_l^{(i)} \approx \begin{cases} \mathrm{PM}_l^{(i-1)}, & \hat{u}_i[l] = \delta(L_N^{(i)}[l]) \\ \mathrm{PM}_l^{(i-1)} + L_N^{(i)}[l], & \hat{u}_i[l] \neq \delta(L_N^{(i)}[l]) \end{cases} \tag{3.10}$$

where, $\delta(x) = \frac{1}{2}(1 - \mathrm{sign}(x))$, $\mathrm{PM}_l^{(0)} = 0$. If frozen bits are taken into account during decoding, the above formula can be written as

$$\mathrm{PM}_l^{(i)} \approx \begin{cases} \mathrm{PM}_l^{(i-1)}, & \hat{u}_i[l] = \delta(L_N^{(i)}[l]) \\ \mathrm{PM}_l^{(i-1)} + L_N^{(i)}[l], & \hat{u}_i[l] \neq \delta(L_N^{(i)}[l]) \\ +\infty, & i \notin \mathcal{A} \text{ and error in decoding of } u_i \end{cases} \tag{3.11}$$

The decision formula of the information bit is as follows

$$\hat{u}_i = \delta(L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})). \tag{3.12}$$

In sum, SCL decoding algorithm is a breadth-first search algorithm. When the algorithm is implemented, it first expands, then prunes, and finally reaches the leaf nodes. In the decoding process, the decoding performance is improved by reserving multiple paths, but the complexity of the algorithm is also greatly increased due to the increase in the number of paths.

CA-SCL is an optimization algorithm of SCL. CRC is a typical channel error detection method, which is highly operable and effective and thus widely used in the practical communication system. In CA-SCL decoding, a CRC bit sequence is added at the end of the sender's message through a specific calculation method, and the receiver can check whether the recovered data is wrong based on the CRC bit sequence.

CA-SCL first performs SCL decoding on the received codeword. Among the reserved multiple paths, the first path that passes CRC verification is selected as the output decoding result [23], not the path with the smallest PM. If the length of the original information bit sequence is k and the length of the added CRC bit sequence is m, the whole K polar code information bits can be obtained, where $K = k + m$. After polar encoding, the complete polar codeword with length N can be obtained.

Denote the set of paths reserved in the i-th layer of the polar code tree of as $L^{(i)}$, and the specific steps of CA-SCL decoding scheme with L paths are as follows:

(1) Initialize: The initial path set of the root node, that is, the path set of layer 0, is empty, and the PM value is 0.

(2) Expand: For each sequence in the list, two sequences of length i are generated which are corresponding to the decoding bit is bit "0" and "1" respectively (i.e. $L^{(i)} = \{(d_1^{i-1}, d_i) | d_1^{i-1} \in L^{(i-1)}, d^i \in \{0, 1\}\}$). For each $d^i \in L^{(i)}$, update PM.

(3) Compete: If the number of paths generated in the previous step is greater

than L, those paths with the larger PM values will be discarded, and only L paths with the smaller PM values will be retained. If the number of paths is no more than L, proceed to the next step.

(4) CRC aided decoding: Repeat step (2) and step (3) until SCL decoding is completed. Sort paths according to their PM values from small to large, and then extract the CRC bit sequence added at the end of each output information bit sequence for CRC verification. If the information bit sequence generated by one path passes the CRC verification, the decoding will be terminated immediately, and this result is selected as the decoding output. If the last path does not pass CRC verification, the result generated by the first path will be selected as the decoding output.

CA-SCL decoding algorithm provides a simple prior information for the final decoding output by adding CRC bit sequence after the original information bits, and has better decoding performance than the original SCL algorithm. However, compared with the original SCL decoding scheme, due to the addition of CRC bits and the calculation for CRC verification, the decoding throughput of CA-SCL decreases and the decoding complexity of CA-SCL increases.

3.1.3 SC-Flip Decoding Algorithm

When the code length of a polar codeword is limited, the subchannels cannot be fully polarized and some information bits cannot be correctly decoded. When an error occurs in decoding one information bit, considering that the SC decoder needs to use the estimated value of the previous information bit when decoding the subsequent information bits, it will lead to more serious error transmission. If the bit decision value of the first error can be corrected before decoding the subsequent bits, the error rate can be reduced. Based on the above idea, SC-Flip algorithm was proposed in [24]. When the CRC verification fails, SC-Flip scheme will try to correct the first wrong bit. Under the condition of high SNR, the calculation complexity of SC-Flip algorithm is close to the original SC decoding, but it can obtain more ideal decoding performance than the original SC algorithm.

The first wrong bit usually appears in the key set [25]. Figure 3.3 shows a full binary tree with code length N=16, whose 16 leaf nodes represent the input of the polar encoding module successively. The black nodes and white nodes denote information bits and frozen bits, respectively. If all child nodes of a parent node

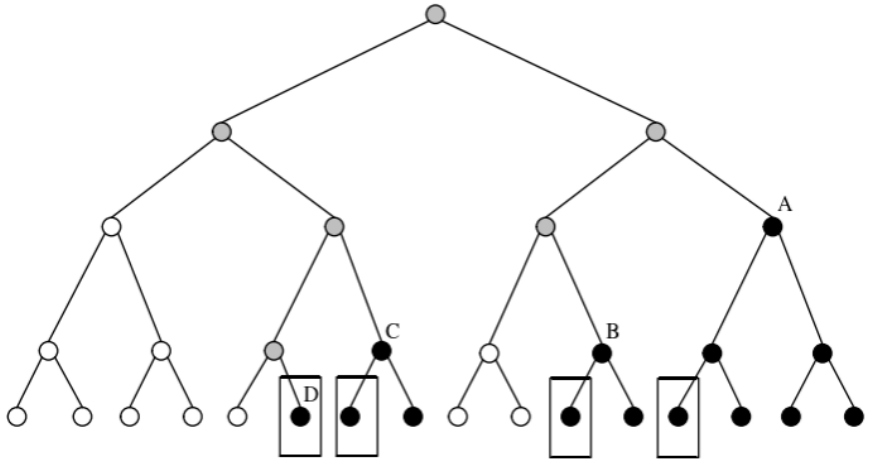

FIGURE 3.3: The key sets used in SC-Flip.

are black/white, the parent node should also be marked as black/white. If there are both black and white child nodes of a parent node, it is marked with gray. If the parent node of a black node is not black, the one with the lowest sequence number of its corresponding information bits belongs to the key set, as the nodes marked in Figure 3.3. It can be observed that whether a node belongs to the key set only depends on its position, and the first failed decoded bit in SC decoding is almost in this set. Therefore, the SC-Flip algorithm can be simplified by only flipping the specific bits.

3.2 BP-based Decoding Algorithms

3.2.1 BP Decoding Algorithm

According to the principle of polar encoding, the construction of polar codes is a problem of selecting polarization channels, and the selection of polarization channels is actually based on the optimal SC decoding performance. According to the polarization channels transition probability function, each polarization channel is not independent of each other but has a definite dependency: the polarization channel with a large serial number depends on all polarization channels with a smaller serial number. Based on this dependence between polarization channels, when SC decoding algorithm makes decoding decisions for each bit, it needs to assume that the decoding results of the previous bits are correct. And it is in SC decoding algorithm that the polar codes are proved to reach the channel capacity.

Therefore, for polar codes, the most suitable decoding algorithm should theoretically be based on SC decoding. However, in the case of finite code length, due to incomplete channel polarization, there will still be some information bits that cannot be decoded correctly. When an error occurs in the decoding of the previous information bits, it will lead to a more serious error delivery since the SC decoder needs to use the estimated values of the previous bits when decoding the next bit, thus for polar codes with finite code length, SC-based decoders may not achieve ideal performance. In addition, due to the serial decoding structure of SC-based decoding methods, the longer the code length, the longer the latency will be, which makes it inefficient for some real-time systems. To obtain low decoding latency, Arıkan arose a BP decoding algorithm for polar codes in [8].

BP decoding algorithm which features parallel decoding structure is more feasible in terms of application for long codes decoding. In BP decoding, each bit can be decoded at the same time due to the parallel architecture of BP decoder. The iterative BP decoding of polar codes is based on the factor graph. If the impact of some short loops on the factor graphs is negligible, the decoding of each bit can be considered independent without mutual influence. However, in SC-based decoding algorithms, whether the present bit can be decoded correctly is totally affected by the previous bits, which means once one bit is decoded incorrectly, all following bits will be interfered. Therefore, compared with SC-based decoding algorithms, BP decoder process each bit in parallel and 'independently', which greatly reduces the decoding time and calculation complexity.

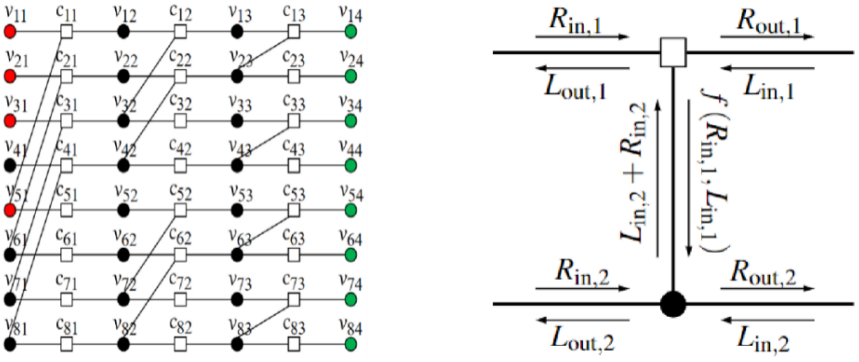

FIGURE 3.4: BP factor graph (left) and processing element (right).

Figure 3.4 shows the factor graph of $\mathcal{P}(8,4)$. The nodes at the leftmost column represent the expected bit sequence $\hat{u}_1^N$, among which red ones denote frozen bits. Nodes at the rightmost column denote the expected construction codeword $\hat{x}_1^N$. All

other black nodes in the factor graph are hidden nodes which are used to complete message processing and transmission. Two kinds of messages are propagated over the whole factor graph, that are left-to-right messages (R-messages) and right-to-left messages (L-messages). All R-messages of frozen nodes are initialized as $+\infty$, and the L-messages of $\hat{x}_1^N$ can be initialized with the LLR of the received codeword y_1^N. Any other node messages without prior knowledge will be initialized as 0. During each iteration, the L-messages and R-messages are propagated from stage to stage throughout the whole factor graph, and updated in each processing element (PE) as follows

$$\begin{cases} R_{\text{out},1} = f(R_{\text{in},1}, L_{\text{in},2} + R_{\text{in},2}) \\ R_{\text{out},2} = f(R_{\text{in},1}, L_{\text{in},1}) + R_{\text{in},2} \\ L_{\text{out},1} = f(L_{\text{in},1}, L_{\text{in},2} + R_{\text{in},2}) \\ L_{\text{out},2} = f(R_{\text{in},1}, L_{\text{in},1}) + L_{\text{in},2} \end{cases} \tag{3.13}$$

where, $f(x, y) = \ln \frac{1+e^{x+y}}{e^x+e^y} \approx \text{sign}(x)\text{sign}(y)\min(|x|, |y|)$. The expected bits can be acquired by

$$L(\hat{u}_i) = L_{i,1} + R_{i,1}, \tag{3.14}$$

$$L(\hat{x}_i) = L_{i,n+1} + R_{i,n+1}. \tag{3.15}$$

The decoding result is decided as

$$\hat{u}_i = \begin{cases} 0, & i \notin \mathcal{A} \\ 0, & i \in \mathcal{A} \text{ and } L(\hat{u}_i) \geq 0 \\ 1, & \text{otherwise} \end{cases} \tag{3.16}$$

$$\hat{x}_i = \begin{cases} 0, & L(\hat{x})i) \geq 0 \\ 1, & \text{otherwise} \end{cases} \tag{3.17}$$

The iterative decoding will stop when $\hat{x} = \hat{u} \cdot \mathbf{G}$, or the predetermined number of iterations, denoted as $N_{\text{it,max}}$, has been reached.

Compared with SC decoding, BP decoding achieves much higher throughput and lower latency due to its parallelism. By transferring and updating the soft messages between different layers of the factor graph in parallel, the confidence level can be improved and the goal of correct decoding can be achieved. However, BP decoding introduces a higher computational complexity and degraded error-rate

performance compared with SC decoding. Many efforts have been made to help performance improvement of polar BP decoding [11], [13], [26]-[29]. Among them, BPL decoding [11], [26], [28], [29] has been considered as an effective scheme to achieve impressive error-rate performance. To solve the problem of high complexity and latency, an LDPC-like polar BP decoder was proposed in [27]. In Section 3.2.2 and Section 3.2.3, the above two improved BP-based decoding schemes and their superiority and inferiority will be introduced.

3.2.2 BPL Decoding Algorithm

It is shown in [16], [30] that for a factor graph of polar code with code length $N = 2^n$, there exists n stages and thus $n!$ different representations corresponding to different permutations of the original factor graph. A permuted factor graph can be represented by a vector $\mathbf{\Pi} = \text{permute}(n)$, in which the order of elements describes the order of stages after permutation (i.e., for the original factor graph, $\mathbf{\Pi} = [0, 1, ..., n-1]$. In this manner, the original factor graph of $\mathcal{P}(8,4)$ and one stage permuted representation of that are demonstrated in Figure 3.5.

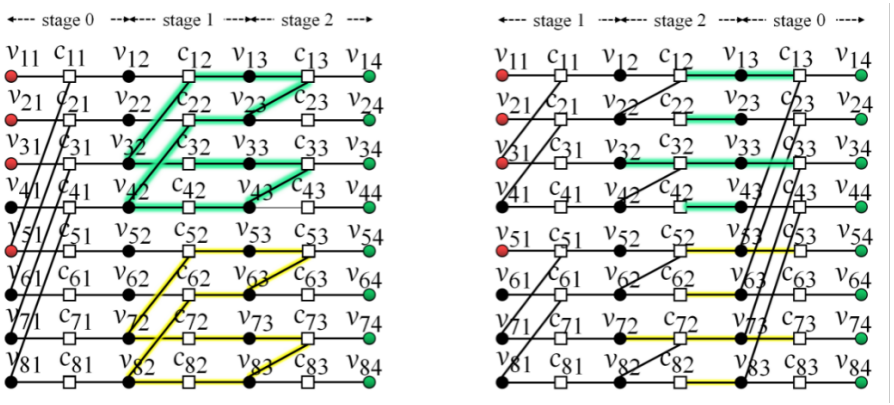

FIGURE 3.5: The original BP factor graph and its one corresponding permuted factor graph for $\mathcal{P}(8, 4)$.

It is proved that permuting stages within a polar factor graph does not change the encoded codeword when information bits and frozen bit places stay the same [31], while having a huge influence on the decoding performance. Stage permutation of factor graph can help solve error floor problem [32] and obtain performance improvement, due to changing the properties of short loops in the decoding graph. In the initial BPL decoding scheme, if the decoding based on one factor graph fails, another different permuted factor graph will be employed for the decoding operation, where the latter permuted factor graph is randomly selected [26]. To

optimize the above process, a permuted factor graph selection method was proposed in [12] to choose the better graph selectively in case of decoding failure. It is noted that in the above studies, these permuted factor graphs are utilized in a sequential manner until a correct codeword is obtained, which is similar to the decoding structure of SC/SCL. As a result, the throughput of this multiple permutations decoding scheme much decreased than the original BP decoder which is based on only one factor graph. The more the number of permuted factor graphs used in BPL, the lower the throughput of the decoding. In order to solve the above issue, a BPL decoder featuring the parallel decoding structure was proposed in [11].

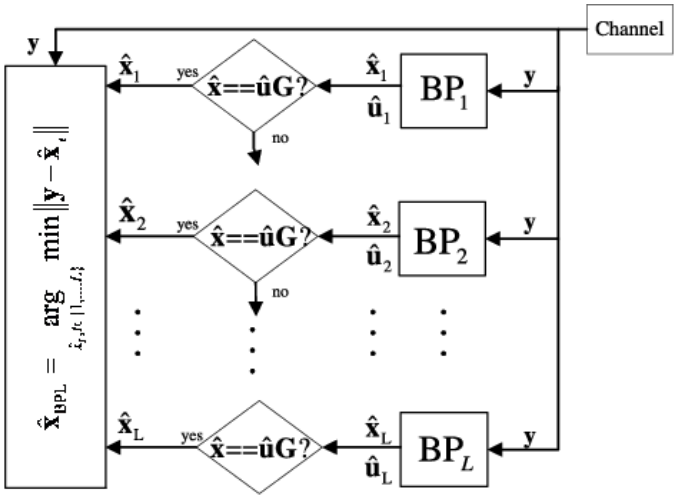

FIGURE 3.6: The diagram of BPL decoding scheme for polar codes.

Figure 3.6 shows the diagram of BPL decoding. L parallel independent BP decoders based on different permuted factor graphs are combined to operate decoding at the same time by using the same received symbol. These permuted factor graphs are generated by l $(1 \leq l \leq L)$ cyclic shifts of the original factor graph. **G**-matrix is used as early stopping criterion. Those decoded results which can pass the **G**-matrix verification will be then compared to the received symbol in terms of Euclidean distance, and the result with MED to the received codeword will be selected as the final output of BPL decoder.

Compared with the original BP decoding, BPL decoding achieve improvement in error-rate performance and latency (since the average iterations required of BPL are usually fewer than that of the original BP). Some follow-up studies focus on finding better list construction methods, as methods discussed in [33], [34], [35] which lead to faster convergence, and algorithms proposed in [13], [14], [28], [29] which help achieve further performance gains. However, in terms of calculation

complexity, BPL decoding has a multiple growth compared with the original BP decoding, due to the utilization of multiple BP decoders. This matter makes it difficult to apply BPL decoding in the actual systems, especially those with limited computing power.

3.2.3 LDPC-like BP Decoding Algorithm

LDPC code is a communication coding method proposed by Gallager in 1962 [4]. Today, with the rapid growth of information transmission rate, LDPC codes have broad application prospects. At present, in the field of 5G communication, LDPC codes and polar codes are equally popular and are the current research hot spot [36], [37], [38], [39] which have the possibility to replace turbo codes in the near future. LDPC codes have significant advantages in decoding algorithm. The construction and decoding of LDPC codes are both based on a sparse parity-check matrix, denoted as **H**-matrix, in which most elements are "0" and only a few elements are "1". This feature can help to reduce a lot of computation and improve decoding efficiency in the decoding process. At the same time, the sparse matrix makes the influence of continuous burst errors on decoding small. On FPGA, the algorithm has good parallelism and low hardware complexity, which enables LDPC codes to become the most popular codes applied in industry and arises continuous research enthusiasm in the academic community. At present, the main decoding algorithms of LDPC include BP/log-BP algorithm and min-sum algorithm. In order to combine the advantages of LDPC code and polar code, many studies have focused on the joint use of these two coding schemes. In [40], the concatenation of the inner LDPC codes with the outer polar codes was investigated, where the usage of polar codes were to remedy the error floor of LDPC codes. Concatenations of polar codes with outer LDPC codes were studied in [41] to improve the performance of polar codes with finite length. At the same time, it is found that BP algorithm has great mobility when decoding these two codes, which becomes the important foundation of the LDPC-Polar joint system. In [42], based on BP decoding for polar codes, the concatenation framework with the outer LDPC codes was discussed, where a set of encoded bits of a polar codeword is chosen to be protected by an outer LDPC codeword. The research in [43] focused on the dependence among the information bit errors and arose a method to break the dependence through the outer LDPC codes. Some research mainly dedicated to analysis about BP factor graph shows that the existing factor graph has redundancy. Authors in [44]

propose a low-complexity calculation method, in which the update of messages can be calculated by the local structure composed of some nodes as a whole but not node by node.

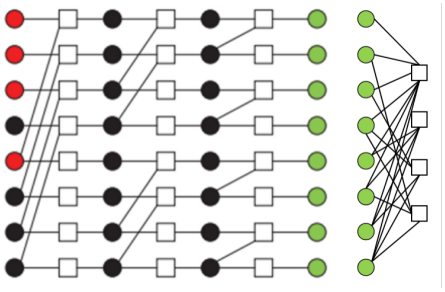

FIGURE 3.7: The original BP factor graph and the corresponding dense graph for $\mathcal{P}(8,4)$.

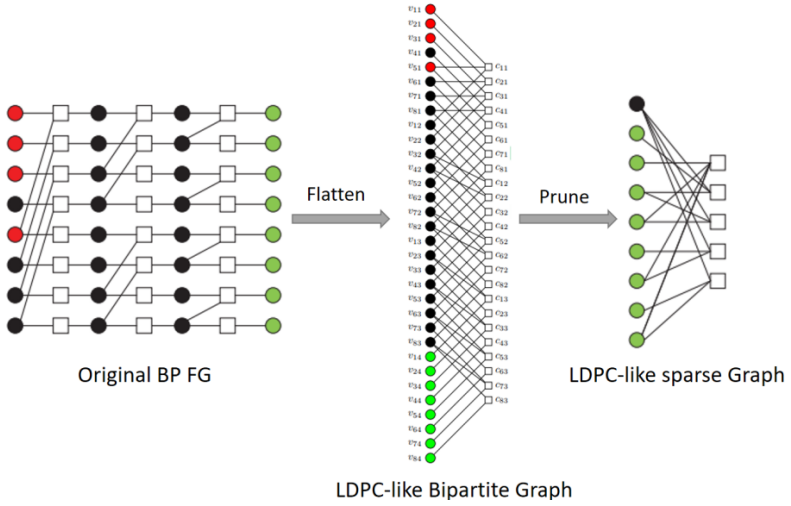

FIGURE 3.8: Converting the original BP factor graph to LDPC-like sparse graph for $\mathcal{P}(8,4)$.

The traditional n-stage polar BP factor graph can be converted to the LDPC-like structure as in Figure 3.7. However, its dense nature causes poor performance. In [27], an LDPC-like BP decoder was proposed to present a sparse representation of the original BP factor graph. A BP factor graph can be transformed to an LDPC-like sparse graph by expanding all stages, as shown in Figure 3.8, where the black VNs are hidden VNs (HVNs) and the green VNs are channel VNs (CVNs). It can be found that there are $N \cdot (n+1)$ VNs and $N \cdot n$ CNs for code length N polar code with n stages. LDPC decoding is also based on message iterative updates. The CN update can be calculated as

$$y_{c\to v} = \prod_{v'\in V_c/v} \operatorname{sign}(x_{v'\to c}) \cdot \phi\left(\sum_{v'\in V_c/v} \phi(|x_{v'\to c}|)\right), \tag{3.18}$$

where $\phi(x) = \phi^{-1}(x) = -\log(\tanh(\frac{x}{2}))$. The above equation can be simplified into

$$y_{c\to v} = \left(\prod_{v'\in V_c/v} \operatorname{sign}(x_{v'\to c})\right)\left(\min_{v'\in V_c/v} |x_{v'\to c}|\right). \tag{3.19}$$

The VN update can be computed as

$$x_{v\to c} = L_v^{\text{init}} + \sum_{c'\in C_v/c} y_{c'\to v}, \tag{3.20}$$

where L_v^{init} denotes the prior knowledge of VN v. The specific value can be calculated also by LLR of the received symbols as $L_{\text{ch},i} = \log \frac{P(x_i=0|y_i)}{P(x_i=1|y_i)}$.

Owning to the sparse feature of LDPC decoding graph, LDPC decoding itself has faster message update speed and simpler hardware implementation than the original polar BP decoding, which makes the LDPC-like BP decoding method meaningful and valuable for practical applications. However, in the simulations of LDPC-like BP decoding algorithm [27], its decoding performance is worse than that of the original BP decoder. The first reason is that although the approximation calculation can be the same as original message calculation theoretically, it will still have a small gap in practical simulation. Especially with long code length and big iteration number, the accumulation of gap will be more obvious, which leads to the larger gap with the performance of original BP decoding. The second reason is that the transformed sparse graph is just LDPC-like but not the same as the original LDPC decoding graph based on **H**-matrix. There are usually more short loops in the LDPC-like sparse graph than the conventional LDPC Tanner graph, which causes negative impact on decoding performance. At present, there is none research on optimizing the decoding structure of LDPC-like BP decoder.

3.3 Summary

In this chapter, we provide an extensive exploration of polar code decoding schemes, aiming to offer a comprehensive understanding of some common polar decoding methods within the context of modern communication systems. The

chapter is organized into two major sections. Chapter 3.1 introduces the fundamental SC-based decoding scheme and its variants, such as SCL and SC-Flip, which enhance error correction capabilities under specific conditions. We further present some vulnerabilities of the adoption of SC-based algorithms in the practical communications, such as high complexity, long decoding latency and error propagation. Then, in Chapter 3.2, we discuss the principles of BP decoding and its variants, including BPL and LDPC-like BP schemes. We show that BP decoding can be faster and more flexible in its application. Nevertheless, when compared with SC decoding, BP decoding exhibits limitations in error correction, especially for short codes. These observed challenges serve as the motivation behind our quest to explore optimized polar decoding solutions.

Chapter 4

The Proposed BP-SGL Decoding of Polar Codes

In this chapter, we will present a novel polar code decoding scheme. To begin, we will provide an overview of the scheme's background knowledge and its fundamental framework. Subsequently, we will elaborate the specific implementation methods. Lastly, we will showcase the benefits of the proposed scheme through simulation results.

4.1 Preliminaries

We hope to improve the decoding performance of the LDPC-like BP decoding while retaining its advantages of low complexity and low implementation difficulty. In order to explore the characteristics of LDPC-like sparse graphs, we first take $\mathcal{P}(8,4)$ as an example for analysis. There are total $(\log_2 8)!$ permuted factor graphs and thus six LDPC-like sparse graphs converted from the corresponding permuted factor graphs. By comparing their structures, it is found that there are three pairs of identical sparse graphs, and these different three graphs can be essentially the same by changing the position of nodes.

Two among these six sparse graphs are shown in Figure 4.1 and Figure 4.2, respectively. The sparse graph in Figure 4.1 is transformed from the original BP factor graph with $\mathbf{\Pi}$=[0,1,2] and we mark each variable node with a serial number from 1 to 8. The sparse graph in Figure 4.2 is transformed from one permuted BP factor graph with $\mathbf{\Pi}$=[1,2,0]. Observing the edges between VNs and CNs, we find that it a one-to-one correspondence can be established between the VNs in

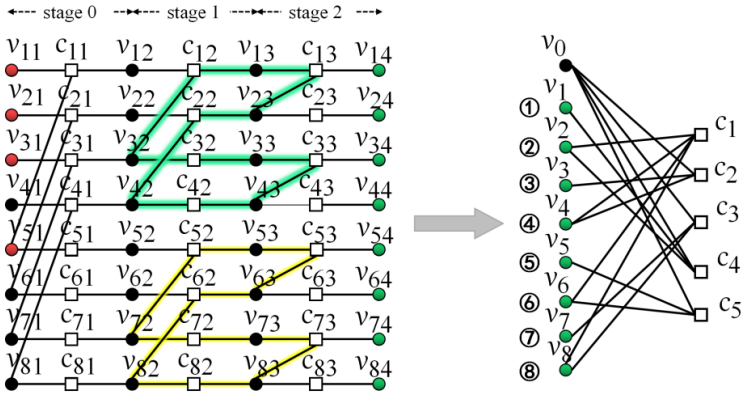

FIGURE 4.1: Sparse graph converted from factor graph with $\mathbf{\Pi}$=[0,1,2] for $\mathcal{P}$(128,64).

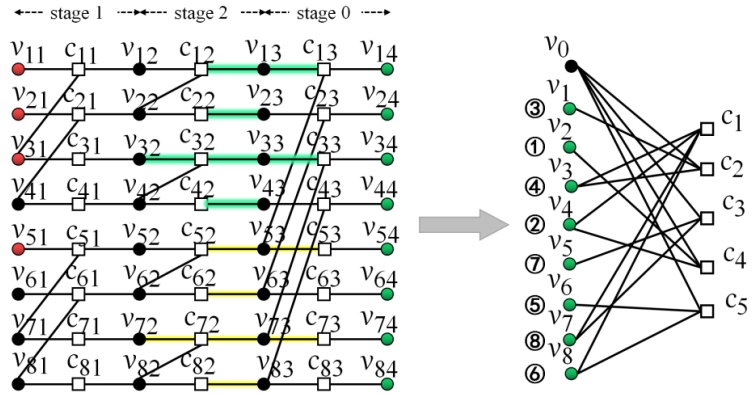

FIGURE 4.2: Sparse graph converted from factor graph with $\mathbf{\Pi}$=[1,2,0] for $\mathcal{P}$(128,64).

these two sparse graphs, as marked above. A loop passing through the VNs with the same numbers can be considered logically identical. Obviously, the received bits involved in the logically identical loops in these two sparse graphs are changed due to the permutation of VNs. Therefore, the decoding result of LDPC-like BP decoder based on one sparse graph may be better than that based on another sparse graph for the same received codeword. Considering the performance improvement of BPL decoding over the original BP decoding, list decoding scheme may also be beneficial to improve the decoding performance of LDPC-like BP decoder. We thus propose a BP-SGL decoding scheme for polar codes.

How to select effective sparse graphs to construct the sparse graph list is the most important factor that decides the performance of the BP-SGL decoder. As mentioned before, many BPL generation methods have been proposed. However,

these existing BPL generation methods are only applicable to evaluate BP factor graphs, not LDPC-like sparse graphs, and cannot guarantee a globally optimal list. By "a globally optimal list", we mean the following scenario: There are totally k lists, and we can select the best list based on the evaluation of each list. For example, [13] selects the best L permuted factor graphs by ranking the upper error bounds of each graph, where the evaluation object is each factor graph, not a list as a whole. Similarly, the list selection scheme in [28] by ranking the quality of legitimate "decoding lists" is also based on the evaluation of each factor graph, not a list as a whole. [29] constructs a list by "finding optimal permutations", not the optimal list. Even though [14] selects a list based on the evaluation of each list, it cannot guarantee a globally optimal list unless it can traverse all randomly generated lists, which is impossible in actual calculations (e.g. $\binom{127}{7} \approx 9 \times 10^{10}$ randomly generated lists for code length 128). Therefore, it is critical to explore an effective algorithm for sparse graph list generation.

In the initial experiment, it is observed that many LDPC-like BP decoders based on different sparse graphs tend to generate the same decoding result for the same received codeword by a high probability. Take $\mathcal{P}(128,64)$ as an example. There are $7! = 5040$ sparse graphs and after removing the duplicated ones, there are 2384 graphs remaining. We first select the first sparse graph, converted from the original factor graph, to build an LDPC-like BP decoder, and input a certain number of known samples, which can be considered as received codewords, to it, then collecting all unsuccessfully decoded samples as a dataset. It is obvious that this first sparse graph is not applicable for decoding those codewords in the dataset. To find valid sparse graphs for successfully decoding samples in the dataset, the necessary condition is that there are sparse graphs whose decoding results of samples in the dataset differ from the original incorrect results. We then use all the other 2383 sparse graphs to decode these codewords in the dataset. Surprisingly, it is found that there are a large number of repetitions within the decoding results, and the number of different decoding results is limited. The grouping results of these 2384 sparse graphs according to their decoding results are shown in Figure 4.3. Considering that the decoding results generated based on sparse graphs within the same group are likely to be the same, the sparse graph list for $\mathcal{P}(128,64)$ can be simply constructed by picking out one sparse graph from each group.

The above analysis shows that there may be some correlation between these sparse graphs. Thus, it will become a reasonable mechanism to efficiently select

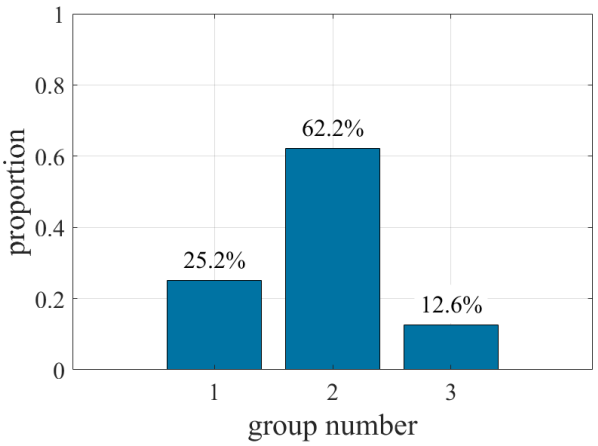

FIGURE 4.3: Sparse graphs grouping for $\mathcal{P}(128,64)$.

sparse graphs based on their correlation.

4.2 Details of the Proposed Decoding Scheme

4.2.1 List Generation Algorithm

Theoretically, the upper performance limit of BP-SGL decoding is to add all sparse graphs to the list. However, considering the implementation complexity and processing capacity of the actual system, the size of the list is not allowed to be too large. In order to approach the optimal decoding effect under the restriction of limited list size, the sparse graphs with low structural similarity should be selected as more as possible to form the list. From the perspective of a received codeword, if it can not be decoded correctly based on one sparse graph, those sparse graphs with high similarity to this one are very likely to produce the same wrong decoding result, which not only makes no contribution to decoding performance improvement but also causes waste of computing resource due to duplicate contribution. Thus, the lower the structural similarity between sparse graphs in the list, the better the overall performance of the list.

A graph $G(V,E)$, where V and E are respectively the vertex set and edge set, can be represented by an adjacency matrix $\mathbf{J}_{\text{adj}}$. Under the condition that the sparse parity-check matrix $\mathbf{H}_{\text{spa}}$ of each sparse graph is known, which is given after LDPC-like BP pruning, $\mathbf{J}_{\text{adj}}$ of a sparse graph can be expressed as $\mathbf{J}_{\text{adj}} = \begin{pmatrix} \mathbf{0} & \mathbf{H}_{\text{spa}} \\ \mathbf{H}_{\text{spa}}^{\text{T}} & \mathbf{0} \end{pmatrix}$. Graph kernel [45] is a typical graph similarity evaluation method,

where a kernel matrix is constructed based on $\mathbf{J}_{\text{adj}}$ to describe the similarity relationship between graphs. Referring to this idea, we use a low-dimensional dense similarity description vector $\mathbf{s}$ to label each sparse graph.

Assuming that there are T sparse graphs to be compared, each sparse graph can be labeled by a vector $\mathbf{s} \in \mathbb{R}^{1\times T}$. A similarity description matrix $\mathbf{S} \in \mathbb{R}^{T\times T}$, each row of which is a similarity description vector corresponding to a sparse graph, can fully describe the similarity between any two sparse graphs. In specific, $\mathbf{S}_{ij}$, the element at i-th row and j-th column in $\mathbf{S}$, describes the similarity between the i-th sparse graph and the j-th sparse graph. The label vector $\mathbf{s}$ can be obtained by cosine similarity calculation, which is a technique for finding the similarity between two vectors [46]. Cosine similarity is derived by using the Euclidean dot product formula

$$\mathbf{a}\cdot\mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\|\cos(\theta). \tag{4.1}$$

The cosine similarity s of two vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$s\left\langle \mathbf{a},\mathbf{b}\right\rangle = \cos(\theta) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\,\|\mathbf{b}\|}. \tag{4.2}$$

Considering the difference in matrix dimensions, centered cosine similarity calculation is applied in the actual simulation. Centered cosine similarity calculation is an improvement of the original cosine similarity calculation in the condition of dimension missing, and can be calculated by

$$s\left\langle \mathbf{a},\mathbf{b}\right\rangle = \frac{\sum_{i=1}^{n}(\mathbf{a}_i-\bar{\mathbf{a}})(\mathbf{b}_i-\bar{\mathbf{b}})}{\sqrt{\sum_{i=1}^{n}(\mathbf{a}_i-\bar{\mathbf{a}})^2\sum_{i=1}^{n}(\mathbf{b}_i-\bar{\mathbf{b}})^2}}, \tag{4.3}$$

where $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ represent the average of vector $\mathbf{a}$ and $\mathbf{b}$, respectively. Converting each $\mathbf{J}_{\text{adj}}$ into a one-dimensional vector, $\mathbf{S}$ can be obtained by using Eq. (4.3).

KPCA [47] method is performed to generate a visual low-dimensional matrix $\mathbf{V}$ by extracting the most important information contained in $\mathbf{S}$. Figure 4.4 shows the visualization result of $\mathbf{V}$ for $\mathcal{P}(128,64)$. Two principal components PC1 and PC2 of each $\mathbf{s}$ are extracted based on the sigmoid kernel function and the Laplacian kernel function, respectively. In a physical sense, PC1 and PC2 denote two directions in which the projection variances of all label vectors are maximum. In a mathematical sense, PC1 and PC2 represent the two largest eigenvectors of the covariance matrix of $\mathbf{S}$. According to Figure 4.4, sparse graphs of $\mathcal{P}(128,64)$ can be clustered into 4

Algorithm 1: BP-SGL Generation Algorithm

Input :

N, n, K, $\mathcal{A}$; // Parameters of the polar code.
L; // Number of sparse graphs (SGs) in the list.
$\mathbf{\Pi}^i, 1 \leq i \leq n!$; // Stage permutations.
numComponent, kernelFunction; // Parameters of KPCA.

Output : $\mathcal{L}_{|\mathrm{BP-SGL}|}$; //BP-based sparse graph list.

// Initialization.
$\mathbb{H}_{\mathrm{spa}} \leftarrow \emptyset$; // A set of sparse parity check matrix $\mathbf{H}_{\mathrm{spa}}$.
$\mathbb{J}_{\mathrm{adj}} \leftarrow \emptyset$; // A set of adjacency matrix $\mathbf{J}_{\mathrm{adj}}$.
$\mathcal{L}_{|\mathrm{BP-SGL}|} \leftarrow \emptyset$;
// Generate $\mathbb{H}_{\mathrm{spa}}$.
for $i = 1, 2, ..., n!$ **do**
  $\mathbf{H}^i_{\mathrm{spa}} = \mathrm{GenerateSG}(N, K, \mathcal{A}, \mathbf{\Pi}^i)$;
  $\mathbf{H}^i_{\mathrm{spa}} = \mathrm{ZeroPadding}(\mathbf{H}^i_{\mathrm{spa}})$; // Make all $\mathbf{H}^i_{\mathrm{spa}}$ the same size.
  $\mathbf{H}^i_{\mathrm{spa}} = \mathrm{SortRows}(\mathbf{H}^i_{\mathrm{spa}})$; // Permute rows in $\mathbf{H}^i_{\mathrm{spa}}$.
  if $\mathrm{IsNotEqual}(\mathbf{H}^{i[\mathrm{HVNs}]}_{\mathrm{spa}}, \forall\ \mathbf{H}_{\mathrm{spa}} \in \mathbb{H}^{[\mathrm{HVNs}]}_{\mathrm{spa}})$ **then**
    // $\mathbf{H}^{[\mathrm{HVNs}]}$ means the part in $\mathbf{H}$ corresponding to HVNs.
    $\mathbb{H}_{\mathrm{spa}} \leftarrow \mathbf{H}^i_{\mathrm{spa}}$; // Delete duplicate $\mathbf{H}_{\mathrm{spa}}$.
  end if
end for
// Generate $\mathbb{J}_{\mathrm{adj}}$.
$T = |\mathbb{H}_{\mathrm{spa}}|$; // Number of $\mathbf{H}_{\mathrm{spa}}$ in $\mathbb{H}_{\mathrm{spa}}$.
for $i = 1, 2, ..., T$ **do**
  $\mathbf{J}^i_{\mathrm{adj}} = \mathrm{ConvertToAdj}(\mathbf{H}^i_{\mathrm{spa}})$;
  $\mathbb{J}_{\mathrm{adj}} \leftarrow \mathbf{J}^i_{\mathrm{adj}}$;
end for
// Generate similarity description matrix $\mathbf{S}$.
for $i = 1, 2, ..., T$ **do**
  $\mathbf{J}^i_{\mathrm{adj}} = \mathrm{Flatten}(\mathbf{J}^i_{\mathrm{adj}})$; // Flatten into one dimension.
  for $j = 1, 2, ..., T$ **do**
    $\mathbf{J}^j_{\mathrm{adj}} = \mathrm{Flatten}(\mathbf{J}^j_{\mathrm{adj}})$;
    $\mathbf{S}_{ij} = s(\mathbf{J}^i_{\mathrm{adj}}, \mathbf{J}^j_{\mathrm{adj}})$ by Eq. (5);
  end for
end for
// Group SGs by KPCA.
KPCA = BuildModel(numComponent,kernelFunction);
$\mathbf{V}$ = Kernal PCA.train($\mathbf{S}$); // Extract information from $\mathbf{S}$.
$\mathcal{G}^{[k]}_{|\mathrm{SG}|} = \mathrm{Group}(\mathbf{V})$; // k groups generated.
// Select SGs to construct the list.
for $i = 1, 2, ...,$ minimum(L,k) **do**
  $\mathcal{L}_{|\mathrm{BP-SGL}|} \leftarrow \mathrm{SG} \in \mathcal{G}^i_{|\mathrm{SG}|}$;
end for

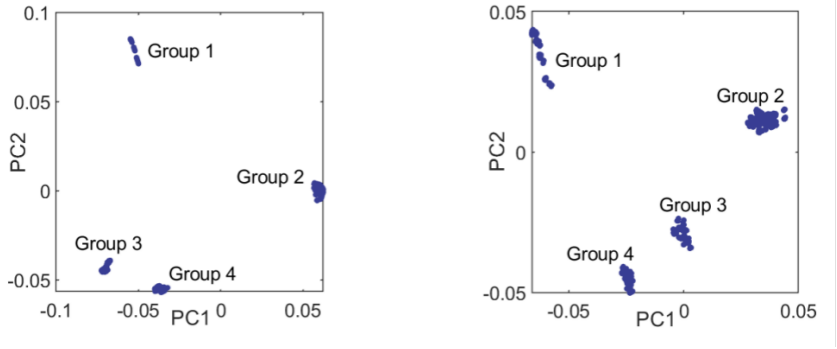

FIGURE 4.4: Sparse graph grouping results based on the sigmoid kernel function (left) and the Laplacian kernel function (right) for $\mathcal{P}(128,64)$.

groups. Pick out one sparse graph from each group and finally a list containing L (i.e. L=4 in the case of Figure 4.4) sparse graphs can be constructed. The detailed BP-SGL generation algorithm is presented in Algorithm 1.

4.2.2 Validity Test of the Generated List

The clustering results of sparse graphs based on two different kernel functions are completely the same, as shown in Figure 4.5. It indicates that the constructed similarity description matrix $\mathbf{S}$ truly contains effective structural similarity information of the sparse graphs, and the main features extracted by KPCA can help distinguish sparse graphs with different structural features. According to the above clustering results, sparse graphs of $\mathcal{P}(128, 64)$ can be clustered into four groups. The list can be constructed by picking out one sparse graph from each group. In our simulation, the sequence of the final selected sparse graph is #1, #2, #10, #403. It is found that #1, #2 and #10 belongs to three groups in Figure 4.3 respectively. It indicates that selecting sparse graphs according to the clustering results can indeed cover the graphs with different structural features.

More validation tests are performed, as shown in Figure 4.6. Let $c = \frac{|\mathcal{D}_T \cap \mathcal{D}_L|}{|\mathcal{D}_T|}$ describe the coverage of the list for all possible decoding results of a codeword, where set $\mathcal{D}_T$ contains decoding results of a codeword by all T sparse graphs and set $\mathcal{D}_L$ contains decoding results of the same codeword by the list with L sparse graphs. Testing results show that the average of c, denoted as $\bar{c}$, is at least 98.4% at different testing SNRs.

From another perspective, the non-coverage rate (NCR) of the generated list, which can be considered as the lower-bound on BLER of BP-SGL and can be

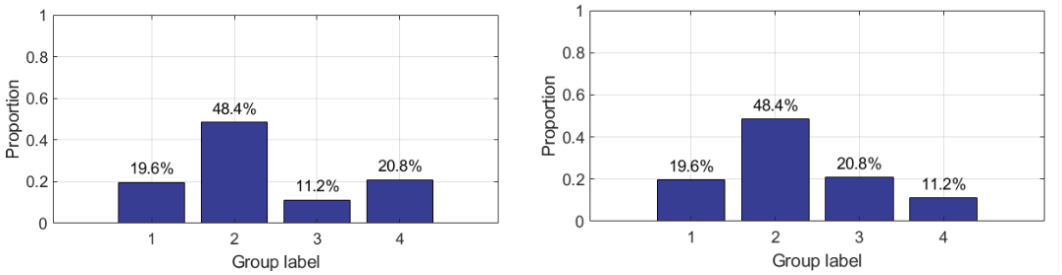

FIGURE 4.5: Statistics of sparse graph grouping results (Figure 4.4) based on the sigmoid kernel function (upper) and the Laplacian kernel function (under) for $\mathcal{P}(128,64)$.

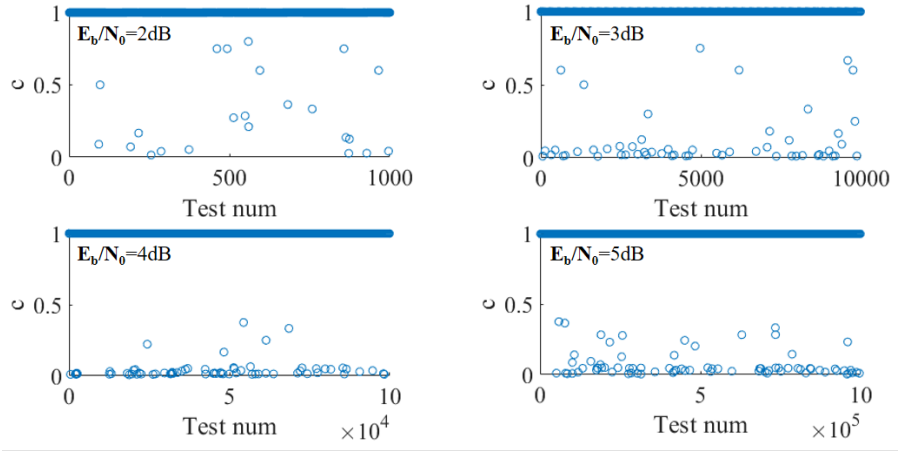

FIGURE 4.6: List grouping validity testing for $\mathcal{P}(128,64)$ at different SNRs; Test num denotes the number of codewords tested.

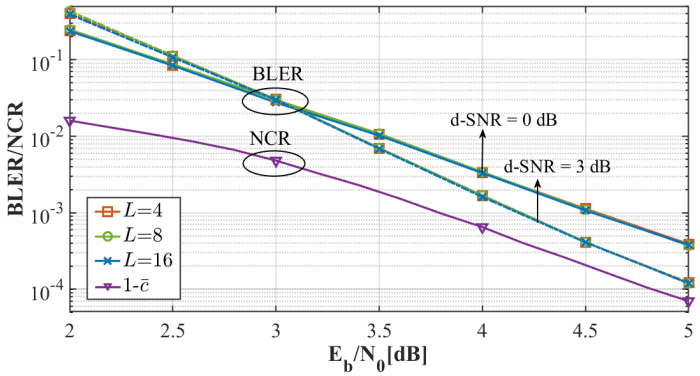

FIGURE 4.7: Non-coverage rate (NCR) of the generated sparse graph lists.

calculated by 1-$\bar{c}$ as shown in Figure 4.7, is extremely low. It means that the decoding contribution of the list containing L ($L \ll T$) SGs is almost equivalent to that of all T SGs. Adding any other sparse graph to the list hardly contributes to decoding performance improvement, since its decoding result will almost certainly be the same as at least one of the results produced by the list. As Figure 4.7 shows, the decoding performance of BP-SGL with list size 4, 8, and 16 for $\mathcal{P}$(128,64) is almost the same. Using different polar code design parameters, such as changing the d-SNR, the above conclusion is still established. It indicates that the proposed BP-SGL generation algorithm can not only select the effective sparse graphs but also avoid calculation redundancy.

4.3 Performance Evaluation

In this section, we compare the error-rate performance and complexity and latency between some existing BP-based polar decoding algorithms and the proposed BP-SGL decoding. We carry out the system simulation and experiment on MATLAB. The system architecture of the simulation is presented in Figure 4.8. About the polar encoder, we generate the polarization channels based on the Bhattacharyya parameter with a d-SNR of 0 dB, which enables the structure of joint channels to be the fixed and known information. The maximum iterations is fixed as $N_{\text{it,max}} = 200$. $\mathbf{u}$ denotes the information bit sequence with the fixed frozen positions. $\mathbf{x}$ is the encoded polar codeword. BPSK modulation is applied as the modulator. The simulation is executed on AWGN channels, thus the channel noise distribution conforms to Gaussian distribution, and noise added on each bit is the independent noise point which is irrelevant to each other. In the receiving end, $\widetilde{\mathbf{s}}$ denotes the received signal mixed with channel noise. A list of decoding results can be generated by the BP-SGL decoder, and the final output will be obtained after the post verification.

4.3.1 Error-rate Performance Comparison

The BLER performance comparisons between the proposed BP-SGL and some typical BP-based decoding schemes are presented. Figure 4.9 and Figure 4.10 show the BLER comparison of Arıkan's original BP, LDPC-like BP, the original BPL, and our proposed BP-SGL for $\mathcal{P}$(128,64) and $\mathcal{P}$(256,128), respectively.

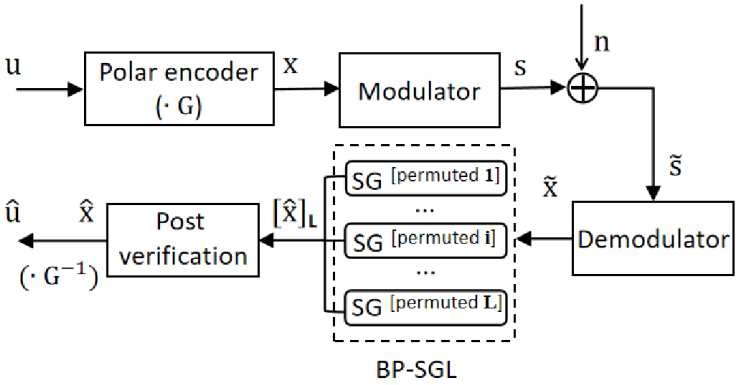

FIGURE 4.8: System framework of simulation.

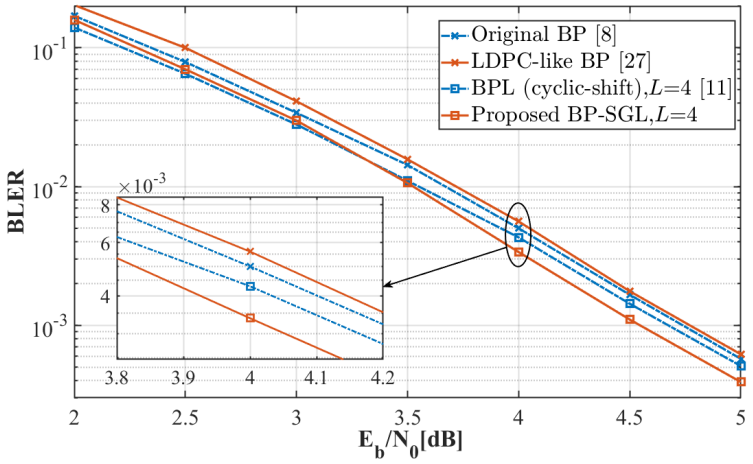

FIGURE 4.9: BLER comparison between Arıkan's original BP [8], LDPC-like BP [27], BPL (cyclic-shift) [11], and the proposed BP-SGL for $\mathcal{P}(128,64)$; $N_{\text{it,max}} = 200$.

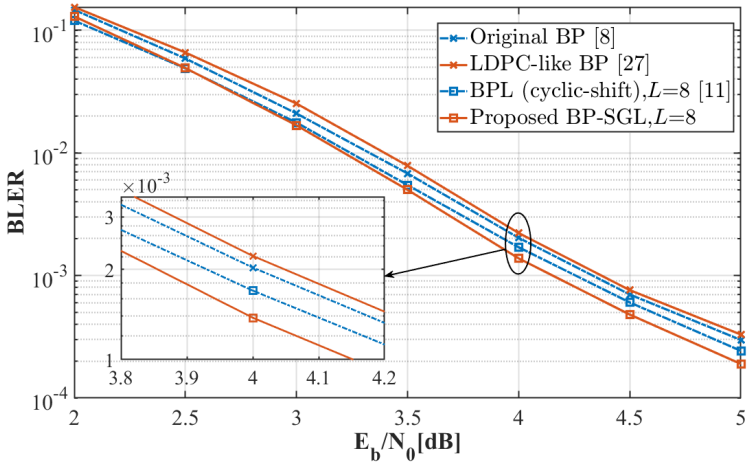

FIGURE 4.10: BLER comparison between Arıkan's original BP [8], LDPC-like BP [27], BPL (cyclic-shift) [11], and the proposed BP-SGL for $\mathcal{P}(256,128)$; $N_{\text{it,max}} = 200$.

Results show that the proposed BP-SGL decoder has 0.2 dB and 0.17 dB performance gains over the LDPC-like BP decoder and the original BP decoder, respectively. When compared with the original BPL (cyclic-shift) decoding, it can be observed that the proposed BP-SGL achieves 0.1 dB performance gain over BPL (cyclic-shift) in the high SNR region. LDPC-like BP decoder has poor performance due to error accumulation caused by some short loops on one sparse graph. Such error may be effectively reduced by decoding on another sparse graph with a more appropriate structure. Therefore, the proposed BP-SGL decoder has a great performance improvement over the LDPC-like BP decoder and can even outperform the BPL (cyclic-shift) decoder.

Further comparison between SOTA BP-based decoding algorithms [28], [48] and the proposed decoding scheme is presented in Figure 4.11. It can be observed that the proposed BP-SGL can outperform BPL (EXIT-Chart) [28] when $E_b/N_0 >$ 3.6 dB, and can achieve near 0.1 dB performance gain at around BLER $= 10^{-4}$. To visually show the performance gain brought by the list selection scheme, we compare the EXIT-Chart aided selection method and our proposed list selection method, for SGL with list size $L = 8$. As shown in Figure 4.11, under the same decoder architecture, our generated BP-SGL can achieve 0.14 dB performance gain over EXIT-Chart based SGL.

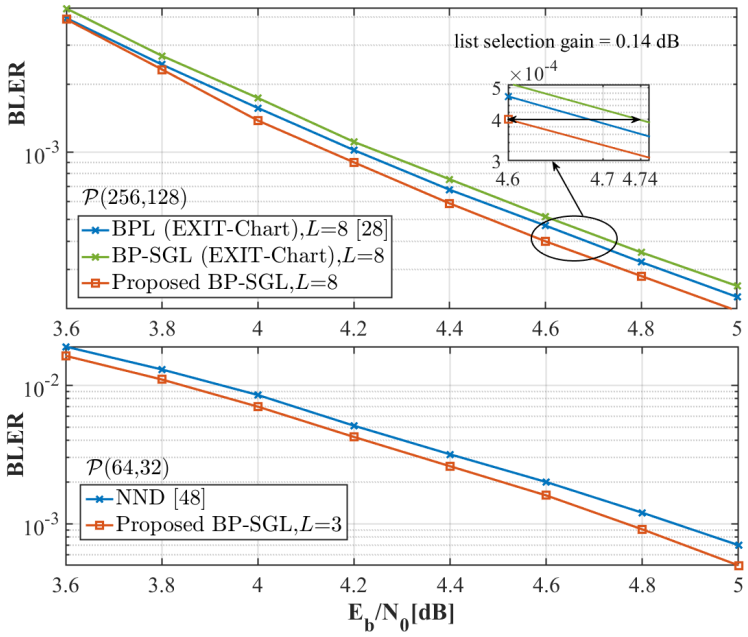

FIGURE 4.11: BLER comparison between SOTA BP-based decoding algorithms [28], [48] and the proposed BP-SGL.

Deep learning methods have shown great value in many aspects such as signal and image processing, object detection and tracking, speech processing, and behavior recognition. It has also become a popular trend to improve traditional communication technology by using deep learning methodologies. Many deep learning aided BP-based decoding schemes have been explored in recent years [48], [49], [50]. As shown in Figure 4.11, the proposed BP-SGL can obtain 0.1 dB performance gain over NND proposed in [48] for $\mathcal{P}$(64,32). It further presents the superiority of the proposed decoding method.

4.3.2 Complexity and Latency Analysis

In order to quantitatively compare the complexity, the number of additions, multiplications and comparisons in one iteration of decoding are counted. For BPL decoding, there are $L \cdot 2N \cdot \log_2 N$ additions, multiplications and comparisons in one iteration, where L is list size and N is code length. For the proposed BP-SGL, let D_{v_i} and D_{c_j} denote the degree of VN v_i and CN c_j in the sparse graph, respectively. It can be calculated that in one iteration of BP-SGL decoding, there are $L \cdot \sum_{i=1}^{\text{num(VN)}} 2D_{v_i}$ additions and $L \cdot \sum_{i=1}^{\text{num(CN)}} D_{c_j}(D_{c_j} - 2)$ comparisons. To facilitate the overall comparison of complexity, we use the number of floating point operations (FLOPs) required to quantify the calculation complexity [51]. In specific, each addition/comparison requires two FLOPs and each multiplication requires six FLOPs.

TABLE 4.1: Complexity and latency comparison between BPL (cyclic-shift) [11] and the proposed BP-SGL at BLER = 10^{-3}.

Polar codes	Decoder	Iterations		Complexity ($\times 10^5$ FLOPs)		Latency (CCs)	
		I_{avg}	$I_{\max}$	C_{avg}	$C_{\max}$	t_{avg}	$t_{\max}$
$\mathcal{P}$(128,64)	Proposed BP-SGL	6.2	9.0	1.4	2.1	12.4	18.0
	BPL (cyclic-shift) [11]	2.6	4.0	1.9	2.9	36.4	56.0
$\mathcal{P}$(256,128)	Proposed BP-SGL	8.4	12.0	7.0	10.0	16.8	24.0
	BPL (cyclic-shift) [11]	3.2	5.0	10.5	16.4	51.2	80.0

Table 4.1 compares the calculation complexity and decoding latency of the original BPL (cyclic-shift) and the proposed BP-SGL at BLER=10^{-3}. Both the

average (i.e. I_{avg}, C_{avg}, and t_{avg}) and the worst case (i.e. $I_{\max}$, $C_{\max}$, and $t_{\max}$) are considered. Even though BP-SGL uses more iterations than BPL (cyclic-shift), it has 26.3%∼33.3% lower C_{avg} and 27.6%∼39.0% lower $C_{\max}$ compared with BPL (cyclic-shift), due to less processing elements. The latency is quantified as the number of clock cycles (CCs) during the decoding process and can be calculated by $2 \cdot I \cdot \log_2 N$ and $2 \cdot I$ for polar-like BP decoding and LDPC-like BP decoding, respectively, where I is the number of iterations. As Table 4.1 shows, the proposed BP-SGL has 65.9%∼67.2% t_{avg} reduction and 67.9%∼70.0% $t_{\max}$ reduction over BPL (cyclic-shift), due to its single stage structure, which gives BP-SGL advantages in real-time systems.

TABLE 4.2: Complexity and latency comparison between SOTA BP-based algorithms [28], [48] and the proposed BP-SGL at BLER = 10^{-3}.

Polar codes	Decoder	Iterations		Complexity ($\times 10^5$ FLOPs)		Latency (CCs)	
		I_{avg}	$I_{\max}$	C_{avg}	$C_{\max}$	t_{avg}	$t_{\max}$
$\mathcal{P}$(256,128)	Proposed BP-SGL	8.4	12.0	7.0	10.0	16.8	24.0
	BPL (EXIT-Chart) [28]	3.1	5.0	10.2	16.4	49.6	80.0
$\mathcal{P}$(64,32)	Proposed BP-SGL	4.6	6.0	0.3	0.4	9.2	12.0
	NND [48]	5.0	5.0	0.4	0.4	60.0	60.0

Moreover, Table 4.2 shows that the proposed BP-SGL has 31.4% lower C_{avg} and 66.1% lower t_{avg} than BPL (EXIT-Chart). The reduction of $C_{\max}$ and $t_{\max}$ of BP-SGL over BPL (EXIT-Chart) can achieve 39.0% and 70.0%, respectively. When compared with NND [48], the proposed BP-SGL can obtain 25.0% C_{avg} reduction and 84.7% t_{avg} reduction. Even considering the worst case, the proposed BP-SGL has 80.0% lower $t_{\max}$ than NND and no increase in $C_{\max}$. In conclusion, the proposed BP-SGL has much superior comprehensive performance, especially in latency-sensitive and low computational budget application scenarios.

4.4 Discussion

In this section, we make a further discussion on some issues that may attract people's concern.

- *The flexibility about list generation*: How easy is it to adjust the list generation may be one concern. In the simulation, the time spent on list construction is related to the specific code design parameters, mainly code length. For code length 128 (with 7!=5040 sparse graphs) and 256 (with 8!=40320 sparse graphs), it takes about 1.2 and 10.5 hours to construct the list. What we need to emphasize is that the above values are the time consuming for constructing the optimal list. If only a sub-optimal list is required, which does not search the entire space of $(\log_2 N)!$, as the existing list construction methods do, the time required will be greatly reduced. For example, if we fix the number of sparse graphs to be classified as 3000, it only takes about 0.6 and 0.8 hour to construct the list for $\mathcal{P}(128,64)$ and $\mathcal{P}(256,128)$, respectively. The shorter the code length, the less computing time is required. Considering that polar codes are used for control channel, where code length between 20 to 200 is mainly considered, the time spent on BP-SGL construction for polar codes is acceptable. In addition, the list construction is offline, which means that for fixed parameters, the list only needs to be built once and does not need to be searched again.

- *The influence of* $N_{\text{it,max}}$: In our simulation, we fix the maximum iteration as $N_{\text{it,max}} = 200$. We show the influence of different $N_{\text{it,max}}$ at $\text{E}_\text{b}/\text{N}_0 = 4$ dB in Figure 4.12 and Figure 4.13. It can be observed that with the increase of $N_{\text{it,max}}$, the decoding performance gets better, while when $N_{\text{it,max}}$ exceeds a certain value (i.e. $N_{\text{it,max}} = 80$ for BPL (cyclic-shift) and $N_{\text{it,max}} = 100$ for the proposed BP-SGL), the performance will not continue to improve. In the meanwhile, BP/BPL decoding converges faster than LDPC-like BP/BP-SGL.

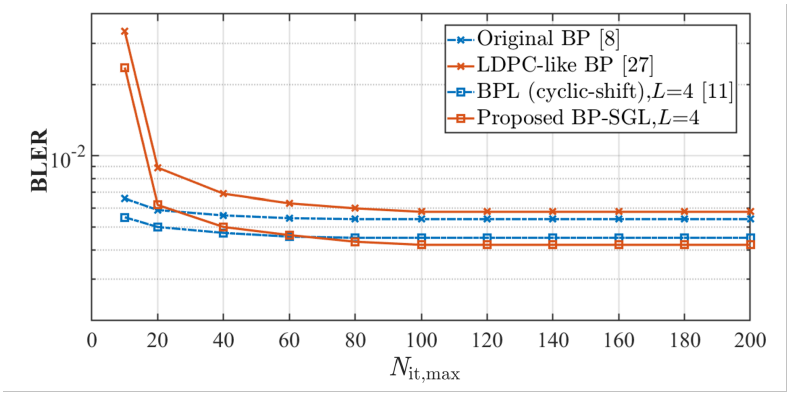

FIGURE 4.12: BLER comparison between Arıkan's original BP [8], LDPC-like BP [27], BPL (cyclic-shift) [11], and the proposed BP-SGL for $\mathcal{P}(128,64)$.

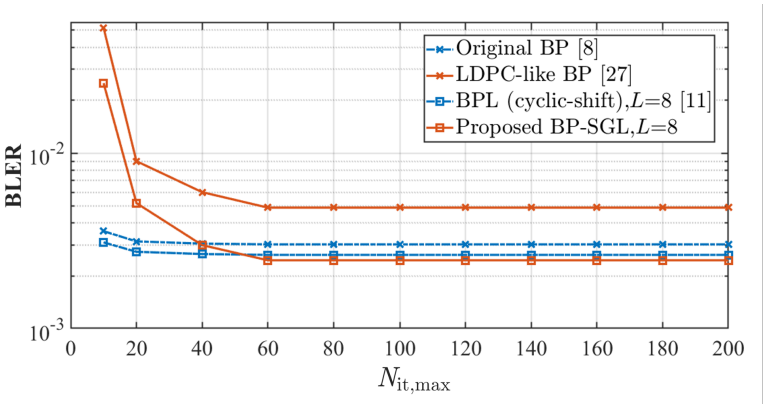

FIGURE 4.13: BLER comparison between Arıkan's original BP [8], LDPC-like BP [27], BPL (cyclic-shift) [11], and the proposed BP-SGL for $\mathcal{P}(256,128)$.

We further present the decoding performance comparisons of BP-SGL with different $N_{\text{it,max}}$ in Figure 4.14. It can be found that at low SNRs, the decoding performance at larger $N_{\text{it,max}}$ is better than that at smaller $N_{\text{it,max}}$. With the increase of simulation SNR, the performances at different maximum iterations tend to converge. It indicates that in bad channel conditions (i.e. low SNR region), The comprehensive benefit of using large $N_{\text{it,max}}$ is higher. While in good channel conditions (i.e. high SNR region), the cost performance will be better by using a smaller $N_{\text{it,max}}$.

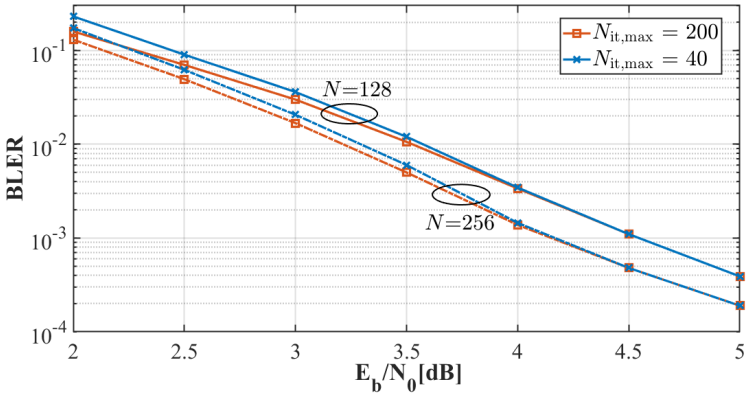

FIGURE 4.14: BLER performance comparison of the proposed BP-SGL under different maximum iterations.

- *Error-rate performance under different code rates*: Whether it can perform well under different code rates R is also a consideration dimension to measure whether a decoding scheme is excellent. For this, we present the BLER comparisons of the proposed BP-SGL and BPL (cyclic-shift), with $R = 1/3$ and $R = 2/3$ for $\mathcal{P}(128,64)$, in Figure 4.15. It can be found that with different

code rates, the proposed BP-SGL always shows better BLER performance than BPL (cyclic-shift) in the high SNR region.

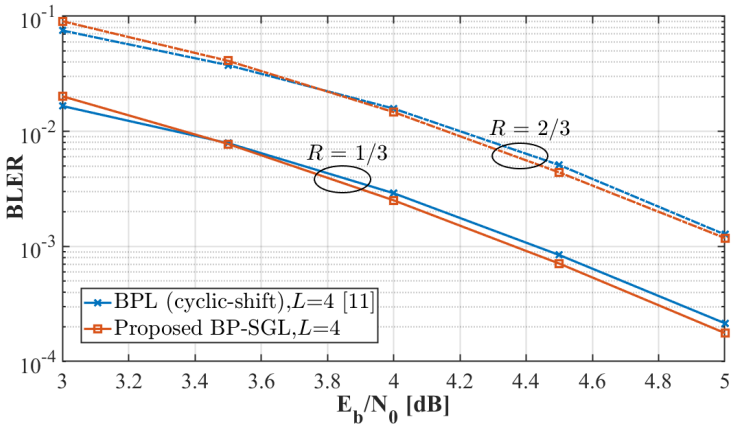

FIGURE 4.15: BLER performance comparison between the proposed BP-SGL with different code rates R for $\mathcal{P}$(128,64).

- *The influence of d-SNR*: In the simulations, we set the d-SNR as 0 dB for polar code construction, which is a commonly selected d-SNR under Bhattacharya parameter method.

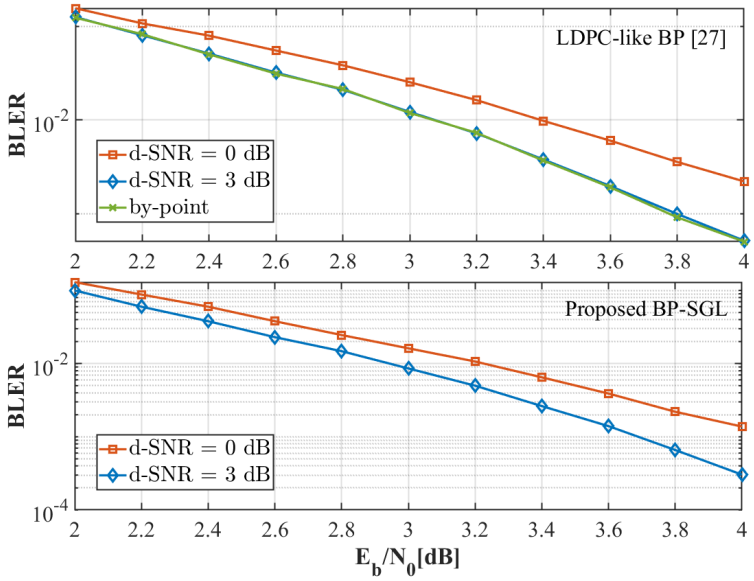

FIGURE 4.16: BLER performance comparison of LDPC-like BP decoding [27] and the proposed BP-SGL based on different construction parameters for $\mathcal{P}$(256,128).

However, based on our extensive testing, we find that the best d-SNR for LDPC-like BP decoding under Bhattacharya parameter is 3 dB, as shown in Figure 4.16. It can be observed that the BLER performance of LDPC-like BP

decoding based on d-SNR=3 dB is the same as that based on by-point SNR. We thus further investigate the BLER performance of the proposed BP-SGL under d-SNR = 3 dB. As shown in Figure 4.16, 0.4 dB performance gain can be obtained by changing d-SNR from 0 dB to 3 dB. While in the same time, because the structure of the sparse graph is changed due to different d-SNRs, the calculation complexity per iteration of BP-SGL with d-SNR = 3 dB is increased by 27.2% than that of BP-SGL with d-SNR = 0 dB.

- *Adding CRC to the proposed decoding process*: One big problem in SOTA polar decoding is to find decoders that can benefit from CRC. In our simulation, we try to add an outer CRC-6 code from 5G standard $g(x) = x^6 + x^5 + 1$ to polar codes to help error detection and early termination. It is observed that the proposed BP-SGL hardly achieves performance gain from the outer CRC code, as shown in Figure 4.17.

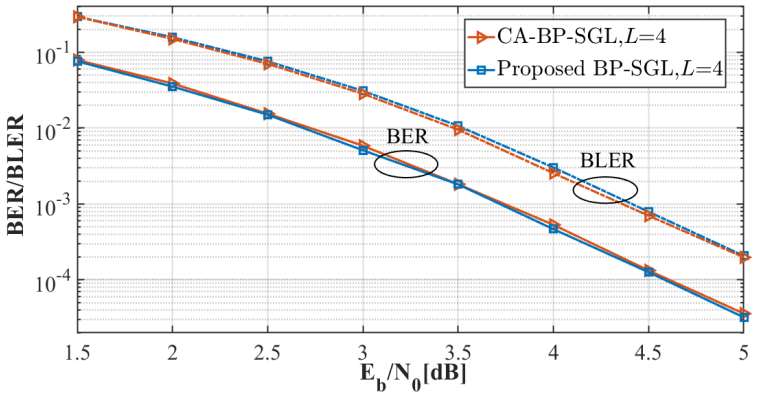

FIGURE 4.17: Performance comparison between CRC-aided BP-SGL and BP-SGL for $\mathcal{P}(128, 64)$.

One difficulty to apply the CRC-aided scheme for LDPC-like BP decoding is that assuming the default code rate remains constant, the outer CRC code needs to be placed into the original frozen position, which will change the feature of some nodes in the original LDPC-like dense bipartite graph. Thus, the pruning and merging standard needs to be adjusted accordingly, and it is likely to disrupt the similarity grouping conclusion of sparse graphs. How to well design CRC bits for BP-SGL decoding is an open problem worthy of attention.

- *Performance evaluation on fading channel scenario*: Since fading channels are quite common in real-world scenarios, we also provide here a brief performance comparison analysis under Rayleigh fading channel.

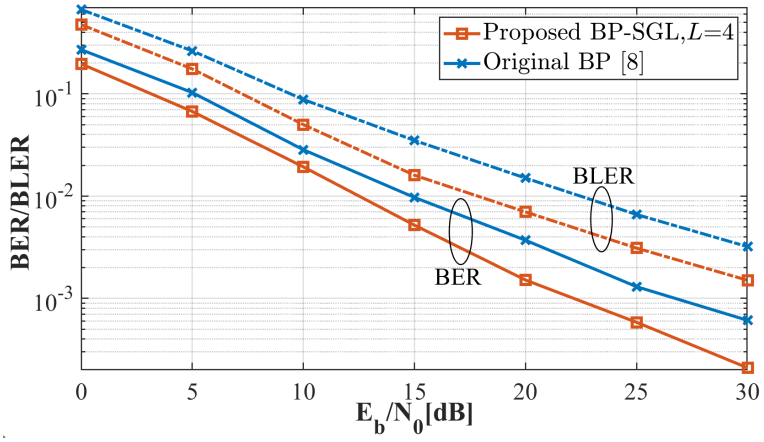

FIGURE 4.18: Performance comparison between the original BP [8] and the proposed BP-SGL under Rayleigh fading channel (only amplitude considered) for $\mathcal{P}(128, 64)$.

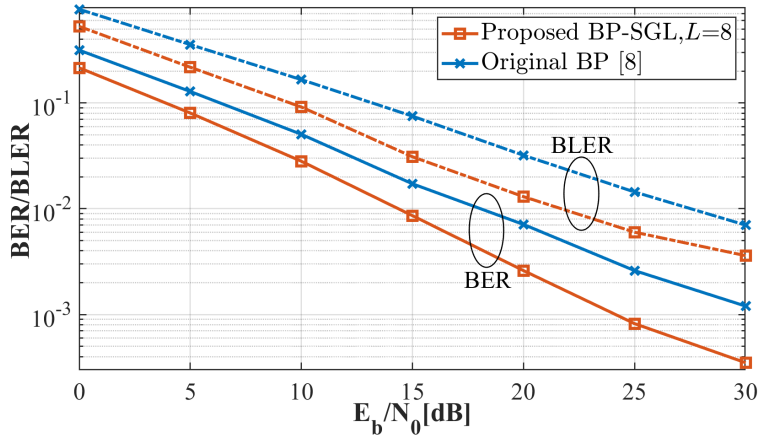

FIGURE 4.19: Performance comparison between the original BP [8] and the proposed BP-SGL under Rayleigh fading channel (only amplitude considered) for $\mathcal{P}(256, 128)$.

As shown in Figure 4.18 and Figure 4.19,we compare the BER and BLER performance between our proposed BP-SGL decoding scheme and the original BP decoding for $\mathcal{P}(128, 64)$ and $\mathcal{P}(256, 128)$, respectively, where we only consider the signal amplitude attenuation caused by channel environment (i.e. flat fading). It can be observed that the proposed decoding scheme still exhibits a significant performance advantage when considering the influence

of channel fading. This advantage is most prominent when observing BLER, especially at very high SNRs.

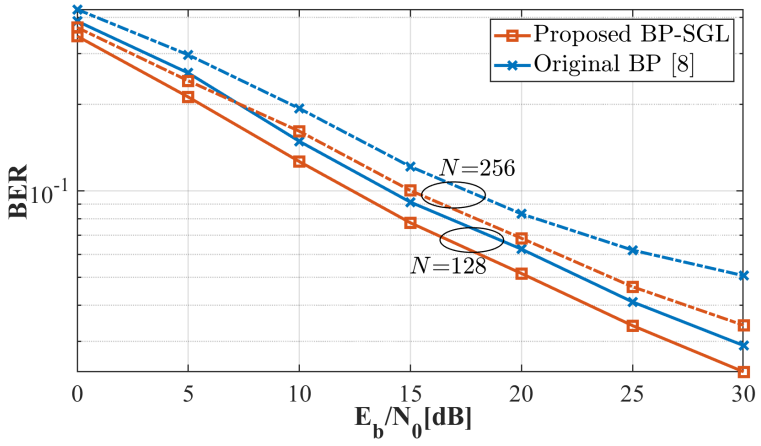

FIGURE 4.20: Performance comparison between the original BP [8] and the proposed BP-SGL under Rayleigh fading channel (only amplitude considered) for $\mathcal{P}(256, 128)$.

In Figure 4.20, we further consider the normal fading scenario where amplitude and phase both fluctuate. In such scenario, the performance further deteriorates, while the fact is still held that the proposed BP-SGL outperforms the original BP. One thing to note is that different from the performances on AWGN channel, in Rayleigh fading channel, the longer code length has worse performance. This happens because we do not use multipath, so the increase in code length also affects the number of bits that are interfered.

4.5 Summary

In this chapter, we first clarify the presence of redundancy, or repetition, within the sparse graphs corresponding to the BP decoding graphs of a fixed code length. Based on that, we develop a sparse list generation algorithm in Chapter 4.2 to help produce a list decoder with the globally optimal decoding performance. We provide detailed algorithmic steps and explanations and demonstrate the effectiveness of the proposed algorithm. In Chapter 4.3, we compare our proposed decoding scheme with existing decoding schemes, also with some SOTA solutions, in terms of both error rate and computing complexity. Comprehensive simulation results indicate that our proposed scheme achieves an improvement in error rate while simultaneously reducing complexity and latency. To further enhance our work, in

Chapter 4.4, we conducted additional discussions regarding the impact of other design parameters and different channel models during the simulation process and the fading channel scenario. These discussions and analyses, while highlighting our contributions, also pointing out areas where further consideration and improvement can be made.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we first introduced the background of the research in Chapter 1. The basic concept and construction principle of the 5G polar codes were presented in Chapter 2. Also, the polar encoding method was shown in Chapter 2. In Chapter 3, several typical polar decoding algorithms and their specific implementation processes were introduced. Two SOTA polar decoding schemes (i.e. BPL decoding, and LDPC-like BP decoding) were also discussed in Chapter 3. The BPL decoder has performance improvement over the traditional BP decoder but much higher complexity, and the LDPC-like BP decoder contributes to the complexity and latency reduction but it has performance degradation compared to the original BP decoder. In Chapter 4, with the purpose of combining the strong points of both the list decoding method and the LDPC-like BP decoding algorithm, we propose the BP-SGL decoding method for the first time. A sparse graph list generation algorithm based on graph similarity comparison was presented for the first time. Considering that the graph selection criteria used by the existing BPL construction methods are only applicable to evaluating BP factor graphs, not LDPC-like sparse graphs, and can not guarantee a globally optimal list, we propose a sparse graph selection scheme to generate a sparse graph list with good overall performance. By constructing a similarity description matrix and extracting the main features by KPCA method, all sparse graphs can be clustered into different groups. A sparse graph list can be constructed by picking out one sparse graph from each group. Assuming that the difference in decoding performance based on sparse graphs within the same group is negligible, testing results verify that the

decoding results generated by the sparse graph list can cover that by all sparse graphs to at least 98.40%. It means that a small-size list containing only a very limited number of sparse graphs can be constructed, and its decoding performance is almost equivalent to a super-large list containing all sparse graphs. The MED criterion is applied as the output selection method. Simulation results show that the proposed BP-SGL decoding has 0.2 dB performance gain over LDPC-like BP decoding, and 0.1 dB performance gain over BPL decoding in the high SNR region with up to 33.3% average complexity reduction and 67.2% average latency reduction. Compared with the deep learning aided BP decoding, BP-SGL can achieve 0.1 dB performance gain with 25.0% average complexity reduction and 84.7% average latency reduction. In conclusion, the proposed BP-SGL has excellent comprehensive performance in terms of both performance and complexity and latency, especially in application scenarios with high real-time requirements and limited computing power, such as control channels in 5G eMBB scenario. At the end of Chapter 4, some issues which may incur people's concerns were discussed. Finally, in Chapter 5, the conclusion of the thesis and the recommendations for future works involving 5G polar coding have been presented. The details of the possible future research are listed in Section 5.2.

5.2 Recommendations for Future Work

This thesis mainly focuses on BP-based decoding schemes for 5G polar codes. We have presented a better decoding algorithm in terms of both decoding performance and efficiency. In the future, more efforts should be made in the following aspects:

- *Research on the proposed decoding algorithm in different fading channels*: The BP-SGL decoding algorithm proposed in this thesis was designed and simulated only in AWGN channels and the basic Rayleigh fading channels. In other more complicated fading channels, the decoding scheme has not been analyzed and tested. Considering that the decoding process of polar codes will change in different fading channels, it is necessary to further study the universality of the proposed decoding algorithm in other types of channels.

- *Research on graph similarity evaluation methods*: Cosine similarity calculation is applied as the graph similarity evaluation criterion in this thesis. Exploring more effective graph similarity evaluation methods may help polish the proposed list generation algorithm. Some typical graph kernels can directly be used to evaluate the similarity between labeled graphs, while sparse graphs have no labels. It may help improve list generation by effectively labeling the nodes in the sparse graph and then using the graph kernel method to evaluate the graph similarity.

- *Research on polar code construction methods*: In this thesis, only the polar decoding scheme is optimized based on LDPC-like BP decoding algorithm. [27] mentioned that the proposed BP factor graph pruning method may also be helpful for further research on polar encoding. For example, based on the sparse graph, an improved selection scheme of frozen bits may be possible. In addition, if an outer CRC code can be effectively designed, it may also help achieve performance gain.

List of Author's Awards, Patents, and Publications

Journal Articles

- **Han Liu**, Erry Gunawan, Hu Yaoyue, and Yong Liang Guan, "BP-Based Sparse Graph List Decoding of Polar Codes," *IEEE Communications Letters*, vol. 27, no. 5, pp. 1257-1261, May 2023.

Bibliography

[1] G. Kakkavas, M. Diamanti, A. Stamou, V. Karyotis, S. Papavassiliou, F. Bouali, and K. Moessner, "5G network requirement analysis and slice dimensioning for sustainable vehicular services," in *Proc. IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, Jul. 2021, pp. 495–502. 2

[2] "IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond", *International Telecommunications Union (ITU) Std. Recommendation ITU-R M.2083-0*, Sep. 2015. 2

[3] E. Ankan, N. U. Hassan, M. Lentmaier, G. Montorsi, and J. Sayir, "Challenges and some new directions in channel coding," *J. Commun. Netw.*, vol. 17, no. 4, pp. 328–338, Aug. 2015. 2

[4] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962. 2, 30

[5] E. Arıkan, "Channel polarization: A method for constructing capacity achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009. 2, 11, 16, 18

[6] "Evaluation of TBCC and polar codes for small block lengths," Huawei, Shenzhen, China, Tech. Rep. 3GPP TSG RAN WG1 N.85, May 2016. 2

[7] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015. 2

[8] E. Arıkan, "Polar codes: A pipelined implementation," in *Proc. 4th Int. Symp. Broadcast. Commun. (ISBC)*, Jul. 2010, pp. 11–14. ix, x, 3, 26, 43, 47, 48, 51, 52

[9] J. Xu, T. Che, and G. Choi, "XJ-BP: Express Journey Belief Propagation Decoding for Polar Codes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1-6. 3

[10] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2009, pp. 1488–1492. 3, 4

[11] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1536–1539, Aug. 2018. ix, x, xi, 4, 28, 29, 43, 45, 47, 48

[12] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "On the decoding of polar codes on permuted factor graphs," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6. 4, 29

[13] B. Li, B. Bai, M. Zhu, and S. Zhou, "Improved belief propagation list decoding for polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 1-6. 4, 28, 29, 36

[14] N. Doan, S. A. Hashemi, and W. J. Gross, "Decoding polar codes with reinforcement learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1-6. 4, 29, 36

[15] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE J. Sel. Top. Signal Process.*, vol. 12, no. 1, pp. 144-159, Feb. 2018. 4

[16] E. Sasoglu, "Polar Codes: A New Paradigm for Coding Theory," *Foundations and Trends in Communications and Information Theory*, vol. 7, no. 3-4, pp. 155-313, 2011. 10, 28

[17] F. Abbasi, H. Mahdavifar, and E. Viterbo, "Polar Coded Repetition," *IEEE Transactions on Communications*, vol. 70, no. 10, pp. 6399-6409, Oct. 2022. 11

[18] Mori R and Tanaka T, "Performance of polar codes with the construction using density evolution," *IEEE Communications Letters*, vol. 13, no. 7, pp. 519-521, Jul. 2009. 12

[19] Wu D, Li Y, and Sun Y, "Construction and block error rate analysis of polar codes over AWGN channel based on Gaussian approximation," *IEEE communication letters*, vol. 18, no. 7, pp. 1099-1102, Jul. 2014. 14

[20] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001. 14

[21] Vangala H, Viterbo E, and Hong Y, "A comparative study of polar code constructions for the AWGN channel," arXiv preprint arXiv:1501.02473, 2015. 16

[22] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015. 20

[23] K. Niu and Chen K. Chen, "CRC-aided decoding of polar codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, Sep. 2012. 23

[24] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," *in Proc. IEEE Asilomar Conference on Signals, Systems and Computers*, Nov. 2014, pp. 2116-2120. 24

[25] Z. Zhang, K. Qin, L. Zhang, et al. "Progressive bitflipping decoding of polar codes over layered critical sets," *in Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2017, pp. 1-6. 24

[26] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation decoding of polar codes on permuted factor graphs," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6. 28

[27] S. Cammerer, M. Ebada, A. Elkelesh, and S. ten Brink, "Sparse graphs for belief propagation decoding of polar codes," in *Proc. IEEE Inter. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1465–1469. ix, x, 28, 31, 32, 43, 47, 48, 49, 56

[28] Z. Q. Chen, L. Li, Z. Ma, and P. Z. Fan, "List selection and decision fusion scheme for belief propagation list decoding of polar codes," in *Proc. IEEE Inter. Symp. on Personal, Indoor and Mobile Radio Commun. (PIMRC)*, Sep. 2019, pp. 1-7. x, xi, 28, 29, 36, 44, 46

[29] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "CRC-aided belief propagation list decoding of polar codes," *in Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 395–400. 28, 29, 36

[30] E. Arikan, "A performance comparison of polar codes and reed-muller codes," *IEEE Communications Letters*, vol. 12, no. 6, pp. 447–449, Jun. 2008. 28

[31] A. G. Peker, "Belief propagation decoding of polar codes under factor graph permutations," Master's thesis, Middle East Technical University, 2018. 28

[32] A. Elkelesh, S. Cammerer, et al., "Mitigating clipping effects on error floors under belief propagation decoding of polar codes," in *Proc. IEEE International Symposium on Wireless Communication Systems (ISWCS)*, Aug. 2017, pp. 384–389. 28

[33] Y. Ren, Y. Shen, Z. Zhang, X. You, and C. Zhang, "Efficient belief propagation polar decoder with loop simplification based factor graphs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5657–5660, Mar. 2020. 29

[34] V. Ranasinghe, N. Rajatheva, and M. Latva-aho, "Partially permuted multi-trellis belief propagation for polar codes," in *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2020, pp. 1-6 29

[35] L. Li and L. Liu, "Belief propagation with permutated graphs of polar codes," *IEEE Access*, vol. 8, pp. 17632–17641, Jan. 2020. 29

[36] 3GPP, "Multiplexing and channel coding (release 10) 3gpp ts 21.101 v10.4.0," 2018. [Online]. Available: http://www.3gpp.org/ftp/Specs/2018-09/Rel-10/21-series/21101-a40.zip 30

[37] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5g new radio," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 28–34, Mar. 2018. 30

[38] L. Xiang, Z. B. K. Egilmez, R.G. Maunder, and L. Hanzo, "CRC-aided logarithmic stack decoding of polar codes for ultra reliable low latency communication in 3gpp new radio," *IEEE Access*, vol. 7, pp. 28559–28573, Feb. 2019. 30

[39] A. Sharma and M. Salim, "Polar code: The channel code contender for 5g scenarios," in *Proc. IEEE International conference on computer, communications and electronics (Comptelix).* , Jul. 2017, pp. 676–682. 30

[40] A. Eslami and H. Pishro-Nik, "On Finite-Length Performance of Polar Codes: Stopping Sets, Error Floor, and Concatenated Design," *IEEE Trans. Commun. (T-COM)*, vol. 61, no. 3, pp. 919-929, Mar. 2013. 30

[41] X. Li, Q. Yu, Z. Shi, Y. Li, and Q. Yan, "Concatenations of polar codes with outer nonbinary LDPC codes," in *Proc. IEEE International Conference on Communication Technology (ICCT)*, Nov. 2017, pp. 117-121. 30

[42] S. M. Abbas, Y. Fan, J. Chen, and C. -Y. Tsui, "Concatenated LDPC-polar codes decoding through belief propagation," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1-4. 30

[43] Y. Meng, L. Li, and Y. Hu, "A Novel Interleaving Scheme for Polar Codes," in *Proc. IEEE Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1-5. 30

[44] J. Xu, T. Che, and G. Choi, "XJ-BP: Express Journey Belief Propagation Decoding for Polar Codes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1-6. 30

[45] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler–Lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, no. 9, pp. 2539–2561, Sep. 2011. 37

[46] S. Dewangan and R. S. Rao, "Design pattern detection by using cosine similarity technique," in *Proc. IEEE Int. Conf. Comput. Commun. Appl. (ICCCA)*, Dec. 2021, pp. 171-175. 38

[47] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299-1319, Jul. 1998. 38

[48] W. Xu, Z. Wu, Y.-L. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2017, pp. 1-6. x, xi, 44, 45, 46

[49] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Jun. 2017, pp. 1361–1365. 45

[50] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling deep learning based decoding of polar codes via partitioning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1-6. 45

[51] P. R. Balogun, I. D. Marsland, R. H. Gohary, and H. Yanikomeroglu, "Polar code design for irregular multidimensional constellations," *IEEE Access*, vol. 5, pp. 21941-21953, 2017. 45