



Article

An Improved Bit-Flipping Algorithm of Successive Cancellation List Decoding for Polar Codes

Desheng Wang ¹, Jihang Yin ^{2,*}, Yonggang Xu ^{1,*}, Xuan Yang ¹, Qiuwei Xu ³ and Gang Hua ¹

¹ School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China; tb18060007b0@cumt.edu.cn (D.W.); xuanyang@cumt.edu.cn (X.Y.); ghua@cumt.edu.cn (G.H.)

² School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

³ College of Information Engineering, Yangzhou University, Yangzhou 225127, China; mz220220345@stu.yzu.edu.cn

* Correspondence: yinjihang20@mails.ucas.ac.cn (J.Y.); xygang@cumt.edu.cn (Y.X.)

Abstract: Polar codes, as the coding scheme for the control channel in fifth-generation mobile communication technology (5G), have attracted widespread attention since their proposal. As a mainstream decoding algorithm for polar codes, the successive cancellation list (SCL) decoder usually improves the error correction performance by increasing the list size, but this method suffers from the problems of high decoding complexity. To address this problem, this paper proposes a layered-search bit-flipping (LS-SCLF) decoding algorithm based on SCL decoding. Firstly, a new flip-bit metric is proposed, which derives a formula to approximate the probability of an error occurring in an information bit. This formula introduces a perturbation parameter to improve the calculation accuracy. Secondly, a compromise scheme for determining the perturbation parameter is proposed. The scheme uses Monte Carlo simulation to determine an optimized parameter for the precise positioning of the first erroneous decoded bit under different decoding conditions. Finally, a layered search strategy is adopted to sequentially search the erroneous decoded bits from the low order to high order, which can correct up to multiple bits at the same time. Simulation results show that the proposed algorithm achieves improved error correction performance with a slight increase in decoding complexity compared to the generalized SCL-Flip (GSCLF) decoding algorithm. This algorithm also achieves a good balance between the error correction performance and decoding complexity.

Keywords: 5G; polar codes; successive cancellation list decoder; bit-flipping algorithm

MSC: 94B35



Citation: Wang, D.; Yin, J.; Xu, Y.; Yang, X.; Xu, Q.; Hua, G. An Improved Bit-Flipping Algorithm of Successive Cancellation List Decoding for Polar Codes.

Mathematics **2023**, *11*, 4462. <https://doi.org/10.3390/math11214462>

Academic Editor: Janez Žerovnik

Received: 6 October 2023

Revised: 24 October 2023

Accepted: 25 October 2023

Published: 27 October 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Polar codes are the first channel coding scheme that can be rigorously proven to achieve channel capacity [1], with clear construction methods and low-complexity encoding and decoding algorithms compared to LDPC codes [2] and turbo codes [3]. Therefore, polar codes have been regarded as an important innovation in channel coding in recent years and have great theoretical value and application potential in future wireless communications, thus attracting much attention from researchers. In 2016, polar codes have been identified as the control channel coding scheme for the fifth-generation enhanced mobile broadband (eMBB) scenario, marking the leap from theoretical research to practical application. Meanwhile, after more than a decade of development, polar codes have achieved many research achievements in deep space communications [4,5], underwater acoustic communications [6–8], optical communications [9–12], and so on.

Although polar codes have many advantages, they still face challenges in practice. In particular, the error correction performance of polar codes for short and moderate lengths under successive cancellation (SC) decoding is unsatisfactory due to insufficient channel polarization. To break the bottleneck of SC decoding performance, successive cancellation list (SCL) decoding [13] was proposed, which greatly improved the error correction performance at finite lengths by retaining L (L is denoted as list size) decoding paths at each information bit. Subsequently, cyclic redundancy check (CRC)-aided SCL (CA-SCL) decoding [14] further improved the error correction performance by concatenating a CRC code at the end of the polar code to assist in making the final output among the L candidate decoding paths. In early SCL decoding, increasing L was the most direct way to improve the error correction performance. However, as L increases, this method faces two problems: (1) If L is too large, the error correction performance gain becomes very small and diminishes as L continues to increase. (2) It requires more hardware resources and more decoding overhead. For this reason, many decoding algorithms have been studied in terms of adaptive lists [15], path-splitting strategies [16–18], and fast decoding [19–21] to compensate for the increase in decoding complexity. However, these decoders do not address the problem of improving the error correction performance of CA-SCL decoding with a limited list size.

Inspired by the bit-flipping algorithms [22–27] based on SC decoding, the idea of bit-flipping was introduced in SCL decoding. The reference [28] first proposed a successive cancellation list bit-flipping (SCLF) decoding algorithm. This decoding algorithm improves the critical set in [25] to identify the candidate flip-bit and flips one bit in each extra decoding. Experimental results show that the SCLF decoding has better error correction performance than SCL decoding in the medium to high SNR regions. However, the bit-flipping strategy used in this algorithm may lead to the appearance of two identical candidate paths, interfering with the final decoding choice. Reference [29] proposed a shift-pruning decoding algorithm, the essence of which is also to correct the erroneous decoded bits by extra decoding. This algorithm can better balance the error correction performance and decoding complexity, but it lacks flexibility in determining the shift positions of the flip-bit, which leads to its poor applicability. Reference [30] also proposed a bit-flipping-based SCL (BF-SCL) decoder, which developed a new criterion for determining the flipping priority of information bits and a specific bit-flipping strategy. However, the proposed bit-flipping metric only considers the effect of information bits on erroneous decoding, which limits the improvement in error correction performance and also introduces a perturbation parameter that varies with the decoding conditions, causing complex complexity. Reference [31] proposes the generalized SCL-Flip (GSCLF) decoding algorithm, which uses the path metrics of the reserved and eliminated paths to devise a new method for determining the flipping priority of information bits that can quickly locate the erroneous decoded bits. Compared to the decoding algorithms proposed in [28–30], the GSCLF decoding is able to correct up to ω erroneous decoded bits caused by channel noise at the same time, thus providing better error correction performance while maintaining a low decoding complexity. These decoding algorithms improve the error correction performance with a limited list size by correcting the erroneous decoded bits in extra decoding attempts. They can achieve the same or better error correction performance as SCL decoders with larger list sizes.

The flip-bit metric is essential for determining the flipping priority of the information bits, which directly affects the error correction performance and the decoding complexity. In this paper, we draw on the metric proposed in D-SCFlip decoding [27] to design a new flip-bit metric to approximate the probability of the erroneous decision occurrence on an information bit. The metric introduces a perturbation parameter that is optimized by simulations to improve the calculation accuracy. Unlike other methods, this metric reflects the sequential nature of SCL decoding, and the metrics between the forward and backward information bits are closely correlated, allowing good localization of erroneous decoded bits. Based on this metric, we propose the layered-search bit-flipping (LS-SCLF)

decoder, which searches for erroneous decoded bits sequentially from low order to high order and can correct up to ω bits in one extra decoding attempt. Simulation results show that the proposed decoding algorithm can achieve a good balance between error correction performance and decoding complexity.

The remainder of the paper is organized as follows. Section 2 introduces the polar codes, SCL decoder, CA-SCL decoder, and SCL-Flip decoder. Section 3 details the proposed flip-bit metric and the the LS-SCLF decoding algorithm. Simulation results are provided in Section 4, and the conclusions are drawn in Section 5.

2. Preliminaries

2.1. Polar Codes

In this paper, u_i^j ($i < j$) denotes the vector $\{u_i, u_{i+1}, \dots, u_j\}$. We denote by $PC(N, K)$ the polar code, where N is the code length and K is the length of information bits. It should be noted that if a CRC code is concatenated at the end of a polar code to improve the error detection capability of the polar code, then $K = k + r$, where k is the number of information bits of the CRC code and r is the number of check bits of the CRC code. The rate is denoted by R and is calculated as $R = k/N$.

Since N independent and identically distributed binary symmetric channels can form N sub-channels with different channel capacities after polarization, we select the first K sub-channels with the largest channel capacity to transmit information bits and denote the index set of these reliable sub-channels by \mathcal{A} . The remaining subchannels are used to transmit frozen bits, which are usually set to 0 and known by both the transmitter and receiver. The index set of frozen bits is denoted as \mathcal{A}^c .

The encoding of polar codes is expressed as:

$$x_1^N = u_1^N \cdot \mathbf{G}_N \tag{1}$$

where u_1^N denotes the data vector, and it consists of K information bits and $N - K$ frozen bits. x_1^N denotes the encoded vector and \mathbf{G}_N denotes the generator matrix. For any information bit $u_i, i \in \mathcal{A}$, when successive cancellation decoding is used, its hard decision estimation can be expressed as:

$$\hat{u}_i = h(L_i) = \begin{cases} u_i & \text{if } i \in \mathcal{A}^c \\ \frac{1 - \text{sign}(L_i)}{2} & \text{if } i \in \mathcal{A} \end{cases} \tag{2}$$

where L_i is the LLR of u_i , and it can be computed as:

$$L_i = \log \left(\frac{\Pr(u_i = 0 | y_1^N, \hat{u}_1^{i-1})}{\Pr(u_i = 1 | y_1^N, \hat{u}_1^{i-1})} \right) \tag{3}$$

where $\Pr(u_i = 0 | y_1^N, \hat{u}_1^{i-1})$ and $\Pr(u_i = 1 | y_1^N, \hat{u}_1^{i-1})$ denote the probabilities of $u_i = 0$ and $u_i = 1$ under the condition (y_1^N, \hat{u}_1^{i-1}) , respectively. y_1^N denotes the received vector of the receiver.

2.2. SCL Decoder and CA-SCL Decoder

The framework of the SCL (CA-SCL) decoder is shown in Figure 1. Firstly, to improve the error correction performance of SC decoding, the SCL decoder retains both sub-paths after the path splitting at any information bit to increase the probability that the correct path is obtained in the candidate paths. Secondly, if the number of candidate paths reaches $2L$, they are ranked in ascending order by the path metric, and the L paths with the smallest metric are reserved through path competition. Finally, the SCL decoder selects the path with the smallest metric as the decoding result \hat{u}_1^N at the end of decoding. CA-SCL decoding is

almost the same as SCL decoding, and the only difference is that CA-SCL decoding selects the path that passes the CRC check as the decoding output.

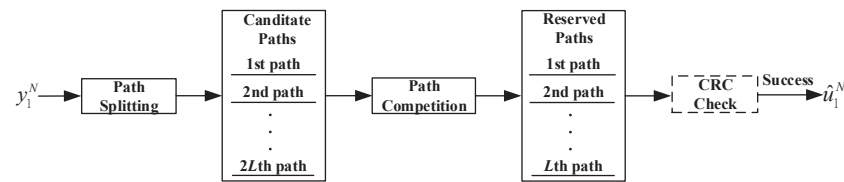


Figure 1. Framework of the SCL (CA-SCL) decoder.

2.3. SCL-Flip Decoder

The SCL-Flip decoder originates from the CA-SCL decoder, with the difference that when the CA-SCL decoder fails the CRC check, the SCL-Flip decoder performs extra decoding attempts to correct the suspiciously erroneous decoded bits, and its framework is shown in Figure 2. The extra decoding procedures for the SCL-Flip decoder are as follows. Firstly, if the CA-SCL decoding (initial SCL decoding) fails the CRC check, the metric calculator calculates the flip-bit metric, and the flip-bit sorter sorts and selects the candidate flip-bit based on their metrics. Secondly, the SCL-Flip decoding performs extra decoding and adds bit-flipping to the path competition. In other words, it makes the opposite decision to CA-SCL decoding for the element in the candidate flip-bit; i.e., it keeps the L decoded paths with the largest path metric (the $L + 1$ th to $2L$ th paths of the candidate paths) while decoding the other bits as CA-SCL decoding. If the extra decoding fails the CRC check, the above operation continues until the decoding is successful or the maximum number of extra decoding attempts is reached.

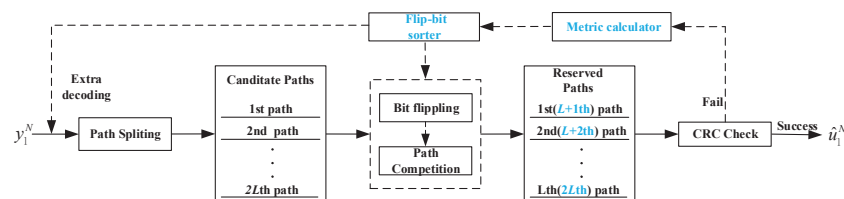


Figure 2. Framework of the SCL-Flip decoder.

In particular, for a better interpretation of the SCL-Flip decoding, we denote $\varepsilon_\omega = \{i_1, i_2, \dots, i_\omega\}$ as a flip-bit set of any order ω , where $\omega \geq 1, i_1 < i_2 < \dots < i_\omega, \varepsilon_\omega \subset \mathcal{I}, \mathcal{I} = \mathcal{A} / \mathcal{A}_0$, and \mathcal{A}_0 is the index set of the first $\log_2 L$ information bits. Without loss of generality, $\text{SCLF}(\varepsilon_\omega)$ is denoted as the decoding of ε_ω by SCL-Flip decoder, and $\text{SCLF}(\varepsilon_\omega^{(t)})$ is denoted as that $\text{SCLF}(\varepsilon_\omega)$ has finished decoding the first $t - 1$ elements and to decode the t th element, obviously, $\varepsilon_\omega^{(t)} = \{i_1, i_2, \dots, i_t\} \subset \varepsilon_\omega$. In the remainder of this paper, $\text{SCLF}(0)$ is denoted as the initial SCL decoding. To clearly indicate the decoding step and distinguish each decoding path, for $\forall j \in \mathcal{A}, \mathcal{L}^j[\varepsilon_\omega^{(t)}] = \{\hat{u}_{1,l}^j[\varepsilon_\omega^{(t)}] | 1 \leq l \leq L\}$ and $\tilde{\mathcal{L}}^j[\varepsilon_\omega^{(t)}] = \{\hat{u}_{1,l+L}^j[\varepsilon_\omega^{(t)}] | 1 \leq l \leq L\}$ respectively denote the reserved paths list and eliminated paths list of $\text{SCLF}(\varepsilon_\omega^{(t)})$ at the step that decoding u_j , where $\hat{u}_{1,l}^j[\varepsilon_\omega^{(t)}] = \{\hat{u}_{1,l}^j[\varepsilon_\omega^{(t)}]_1, \dots, \hat{u}_{1,l}^j[\varepsilon_\omega^{(t)}]_j\}$ stands for the l th decoding path and $\hat{u}_{1,l+L}^j[\varepsilon_\omega^{(t)}] = \{\hat{u}_{1,l+L}^j[\varepsilon_\omega^{(t)}]_1, \dots, \hat{u}_{1,l+L}^j[\varepsilon_\omega^{(t)}]_k\}$ stands for the $l + L$ th decoding path with $1 \leq l \leq L$. For ease of presentation, let $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]$ be denoted as both paths simultaneously. For $j > i_t, \text{SCLF}(\varepsilon_\omega^{(t)})$ decodes u_j the same way as $\text{SCLF}(0)$. Similarly, $\mathcal{L}^j[0] = \{\hat{u}_{1,l}^j[0] | 1 \leq l \leq L\}$ and $\tilde{\mathcal{L}}^j[0] = \{\hat{u}_{1,l+L}^j[0] | 1 \leq l \leq L\}$ correspond to the reserved paths list and eliminated paths list of $\text{SCLF}(0)$ when decoding u_j , respectively. Specifically, for $\forall j \in \mathcal{A}_0, \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(t)}] = \emptyset$ and $\tilde{\mathcal{L}}^j[0] = \emptyset$.

In addition, we denote $PM_{l(+L)}^j[\varepsilon_\omega^{(t)}]$ as the PM of $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]$. For $k < j$, let $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_1^k = \{\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_1, \dots, \hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k\}$ and the corresponding PM is denoted as $PM_{l(+L)}^j[\varepsilon_\omega^{(t)}]_k$. Obviously, $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_1^k$ is the sub-vector of $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]$ and it consists of the first k elements in $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]$. For $k = j$, $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_1^k = \hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]$, $PM_{l(+L)}^j[\varepsilon_\omega^{(t)}]_k = PM_{l(+L)}^j[\varepsilon_\omega^{(t)}]$. For $k > j$, both $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_1^k$ and $PM_{l(+L)}^j[\varepsilon_\omega^{(t)}]_k$ are null vectors. $PM_{l(+L)}^j[\varepsilon_\omega^{(t)}]$ can be computed as:

$$PM_{l(+L)}^j[\varepsilon_\omega^{(t)}] = \sum_{k=1}^j \log\left(1 + \exp\left(-\left(1 - 2\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k\right) \cdot L_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k\right)\right) \tag{4}$$

where $L_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k$ is the LLR of $\hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k$ and can be obtained by:

$$L_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k = \log\left(\frac{\Pr\left(u_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k = 0 | y_1^N, \hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_1^{k-1}\right)}{\Pr\left(u_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_k = 1 | y_1^N, \hat{u}_{1,l(+L)}^j[\varepsilon_\omega^{(t)}]_1^{k-1}\right)}\right) \tag{5}$$

Assume that $\varepsilon_\omega = \{i_1, i_2, \dots, i_\omega\}$ is the trajectory that all its elements are indexes of the erroneous decoded bits caused by channel noise. When the SCLF($\varepsilon_\omega^{(t)}$) decoding is executed to $u_{i_{t+1}}$ ($i_t < i_{t+1} \leq i_\omega$), $2L$ candidate paths are obtained through path splitting, and these candidate paths are sorted by their metrics to satisfy $PM_{1,1}^{i_{t+1}}[\varepsilon_\omega^{(t)}] < PM_{1,2}^{i_{t+1}}[\varepsilon_\omega^{(t)}] < \dots < PM_{1,2L}^{i_{t+1}}[\varepsilon_\omega^{(t)}]$. After path competition, we can obtain the reserved paths list; i.e., $\mathcal{L}^{i_{t+1}} = \{\hat{u}_{1,L+1}^{i_{t+1}}[\varepsilon_\omega^{(t)}], \hat{u}_{1,L+2}^{i_{t+1}}[\varepsilon_\omega^{(t)}], \dots, \hat{u}_{1,2L}^{i_{t+1}}[\varepsilon_\omega^{(t)}]\}$. Based on the definition of SCL-Flip decoding, SCLF($\varepsilon_\omega^{(t)}$) and SCLF($\varepsilon_\omega^{(t+1)}$) satisfy the following relationship:

$$\begin{cases} u_1^{i_{t+1}} \in \tilde{\mathcal{L}}^{i_{t+1}}[\varepsilon_\omega^{(t)}] \\ u_1^{i_{t+1}} \in \mathcal{L}^{i_{t+1}}[\varepsilon_\omega^{(t+1)}] \end{cases}, \begin{cases} u_1^{i_t} \in \mathcal{L}^{i_t}[\varepsilon_\omega^{(t)}] \\ u_1^{i_t} \in \mathcal{L}^{i_t}[\varepsilon_\omega^{(t+1)}], u_1^{i_t} \in \tilde{\mathcal{L}}^{i_t}[\varepsilon_\omega^{(t+1)}] \end{cases} \tag{6}$$

$$\begin{cases} \mathcal{L}^j[\varepsilon_\omega^{(t)}] = \mathcal{L}^j[\varepsilon_\omega^{(t+1)}], j < i_{t+1} \\ \mathcal{L}^j[\varepsilon_\omega^{(t)}] = \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(t+1)}], j = i_{t+1} \end{cases} \tag{7}$$

$$\begin{cases} PM_l^j[\varepsilon_\omega^{(t)}] = PM_l^j[\varepsilon_\omega^{(t+1)}], j < i_{t+1} \\ PM_l^j[\varepsilon_\omega^{(t)}] = PM_{l+L}^j[\varepsilon_\omega^{(t+1)}], j = i_{t+1} \end{cases} \tag{8}$$

$$\begin{cases} L_{1,l}^j[\varepsilon_\omega^{(t)}] = L_{1,l}^j[\varepsilon_\omega^{(t+1)}], j < i_{t+1} \\ L_{1,l}^j[\varepsilon_\omega^{(t)}] = L_{1,l+L}^j[\varepsilon_\omega^{(t+1)}], j = i_{t+1} \end{cases} \tag{9}$$

In particular, for a trajectory $\varepsilon_1 = \{i_1\}$ of order 1, the above relationship can be written as:

$$\begin{cases} \mathcal{L}^j[\varepsilon_1] = \mathcal{L}^j[0], j \leq i_1 - 1 \\ \mathcal{L}^j[\varepsilon_1] = \tilde{\mathcal{L}}^j[0], j = i_1 \end{cases} \tag{10}$$

$$\begin{cases} PM_l^j[\varepsilon_1] = PM_l^j[0], j \leq i_1 - 1 \\ PM_l^j[\varepsilon_1] = PM_{l+L}^j[0], j = i_1 \end{cases} \tag{11}$$

$$\begin{cases} L_{1,l}^j[\varepsilon_1] = L_{1,l}^j[0], j \leq i_1 - 1 \\ L_{1,l}^j[\varepsilon_1]_j = L_{1,l+l}^j[0]_j, j = i_1 \end{cases} \tag{12}$$

3. The LS-SCLF Decoder

3.1. Framework of the LS-SCLF- ω Decoder

Channel noise and error propagation are the two causes of erroneous decoding decisions in SCL decoding. In particular, the first erroneous decoded bit is entirely caused by the channel noise so that the error propagation does not affect the word-error rate (WER) performance. In practice, especially under low signal-to-noise ratio (SNR) conditions, there are often several erroneous decoding decisions that are caused by channel noise. Therefore, the bit-flipping algorithm for the correction of the first erroneous decision alone does not solve the case of multiple erroneous decoded bits caused by channel noise.

Based on the above analysis, this paper proposes the layered-search SCL-Flip decoder that can correct up to ω erroneous decoded bits simultaneously, i.e., the LS-SCLF- ω decoder. The framework of LS-SCLF- ω is shown in Figure 3, which is almost the same as the SCL-Flip decoder, it but has features in terms of the flip-bit metric and the bit-flipping strategy.

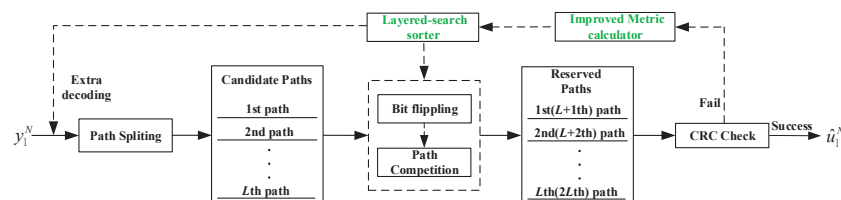


Figure 3. Framework of the LS-SCLF- ω decoder.

3.2. The Proposed Flip-Bit Metric

For SCLF(0), we define the first erroneous decision occurring at information bit $u_i(\forall i \in \mathcal{I})$ as event E_i , i.e., $\hat{u}_1^{i-1} = \mathcal{L}^{i-1}[0]$, $\hat{u}_1^i = \tilde{\mathcal{L}}^i[0]$. Let $\Pr(E_i)$ denote the probability of E_i , then the mathematical definition of $\Pr(E_i)$ can be expressed as:

$$\Pr(E_i) \stackrel{def}{=} \Pr\left(u_1^{i-1} \in \mathcal{L}^{i-1}[0], u_1^i \notin \mathcal{L}^i[0] | y_1^N\right) \tag{13}$$

For $\forall i \in \mathcal{A}_0$, since there is no path competition, $\Pr(E_i) = 0$. The calculation of $\Pr(E_i)$ is an arduous task, due to the fact that all the previous bits must be correctly decoded. To simplify the calculation, we approximate it with the following equation:

$$\Pr(E_i) \approx \Pr\left(u_1^i \in \tilde{\mathcal{L}}^i[0] | y_1^N\right) = \sum_{l=1}^L \Pr\left(\hat{u}_{1,l+l}^i[0] = u_1^i | y_1^N\right) \tag{14}$$

One can calculate $\Pr(E_i)$ by taking the definition of $PM_l^i[0]$ [32]:

$$PM_l^i[0] \stackrel{def}{=} -\log\left(\Pr\left(\hat{u}_{1,l}^i[0] = u_1^i | y_1^N\right)\right) \tag{15}$$

Based on (15), [30] derives the following equation:

$$\begin{aligned} & \Pr\left(\hat{u}_{1,l+l}^i[0] = u_1^i | y_1^N\right) \\ & \approx \exp\left(-\left(PM_{1,l+l}^i[0] - \sum_{j \notin \mathcal{A}} \left(PM_{1,l+l}^i[0]_j - PM_{1,l+l}^i[0]_{j-1}\right)\right)\right) \\ & = \prod_{j=1}^i \left(1 + \exp\left(-\left(1 - 2u_{1,l+l}^i[0]_j\right)L_{1,l+l}^i[0]_j\right)\right)^{-1} \end{aligned} \tag{16}$$

In [30], (16) is carried directly into (14), and a perturbation parameter is introduced to calculate the flip-bit metric to determine the priority of the flip-bit. However, the method only considers the effect of information bits and ignores the role of frozen bits for the erroneous decoded bits. To solve this problem, this paper proposes a new metric. Let Λ_i ($i \in \mathcal{I}$) be denoted as the ratio that is the sum of the probability of the reserved paths to that of the eliminated ones for u_i and referred to as Probability Ratio (PB), i.e.,

$$\Lambda_i = \frac{\sum_{l=1}^L \Pr(\hat{u}_{1,l}^i[0] = u_1^i | y_1^N)}{\sum_{l=1}^L \Pr(\hat{u}_{1,l+L}^i[0] = u_1^i | y_1^N)} \tag{17}$$

$\Lambda_i > 0$ and a higher value of Λ_i indicates a higher probability of $u_1^i \in \mathcal{L}^i[0]$, while a smaller value of indicates a higher probability of $u_1^i \in \tilde{\mathcal{L}}^i[0]$. Since the decoding paths in $\mathcal{L}^i[0]$ can be treated as the input to the next bit in the SCLF(0) decoding, the PB of the previously decoded bits affects the result of the following bits if the L decoding paths are considered as a whole. Therefore, PB is to SCLF(0) decoding what LLR is to SC decoding.

For a flip-bit $\varepsilon_\omega = \{i_1, i_2, \dots, i_\omega\} \subset \mathcal{I}$, the probability that ε_ω is the erroneous decoded path is denoted as $\Pr(\varepsilon_\omega)$. Referring to (12)–(13) in [27], $\Pr(\varepsilon_\omega)$ can be computed as:

$$\begin{aligned} \Pr(\varepsilon_\omega) &= \Pr(u_1^{i_\omega} \in \mathcal{L}^{i_\omega}[\varepsilon_\omega] | y_1^N) \\ &= \Pr(u_1^{i_\omega-1} \in \mathcal{L}^{i_\omega-1}[\varepsilon_\omega^{(\omega-1)}], u_1^{i_\omega} \in \tilde{\mathcal{L}}^{i_\omega}[\varepsilon_\omega^{(\omega-1)}] | y_1^N) \\ &= p_e(u_1^{i_\omega} \in \tilde{\mathcal{L}}^{i_\omega}[\varepsilon_\omega^{(\omega-1)}]) \cdot \prod_{\substack{j=i_\omega-1+1 \\ j \in \mathcal{I}}}^{i_\omega-1} (1 - p_e(u_1^j \in \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(\omega-1)}])) \cdot \Pr(\varepsilon_\omega^{(\omega-1)}) \end{aligned} \tag{18}$$

where

$$p_e(u_1^j \in \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(\omega-1)}]) \stackrel{def}{=} \Pr(u_1^j \in \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(\omega-1)}] | u_1^{j-1} \in \mathcal{L}^{j-1}[\varepsilon_\omega^{(\omega-1)}], y_1^N) \tag{19}$$

By simple calculation, the above equation can be unfolded to the following expression:

$$\Pr(\varepsilon_\omega) = \prod_{j=1}^\omega p_e(u_1^{i_j} \in \tilde{\mathcal{L}}^{i_j}[\varepsilon_\omega^{(j-1)}]) \cdot \prod_{j < i_\omega, j \in \mathcal{I}} (1 - p_e(u_1^j \in \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(\omega-1)}])) \tag{20}$$

Note that the second product of the right-hand side term of (20) is taken only over indexes $j \in \mathcal{I}$, since $p_e(u_1^j \in \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(\omega-1)}]) = 0$ for $j \notin \mathcal{I}$. In addition, the calculation of $p_e(u_1^j \in \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(\omega-1)}])$ is also a complex task. This paper draws on (14) in [27] and proposes a formula to approximate it:

$$p_e(u_1^j \in \tilde{\mathcal{L}}^j[\varepsilon_\omega^{(\omega-1)}]) \approx \frac{1}{1 + \exp(\alpha \Lambda'_j)}, i \in \mathcal{I} \tag{21}$$

where α is the perturbation parameter and $\alpha > 0$, Λ'_j denotes the value of Λ_j in the logarithmic domain; i.e., $\Lambda'_j = \log \Lambda_j$. Such a perturbation directly affects the results of (21). In practice, α can be optimized by Monte Carlo simulation. On the basis of (20) and (21), the proposed metric associated with flip-bit $\varepsilon_\omega = \{i_1, i_2, \dots, i_\omega\}$ is defined as:

$$M_\alpha(\varepsilon_\omega) = \prod_{j \in \varepsilon_\omega} \frac{1}{1 + \exp(\alpha \Lambda'_j)} \cdot \prod_{\substack{j < i_\omega \\ j \in \mathcal{I}}} \frac{1}{1 + \exp(-\alpha \Lambda'_j)} \tag{22}$$

It is clear that for set $\varepsilon_{\omega-1} = \{i_1, i_2, \dots, i_{\omega-1}\}$, consisting of the first $\omega - 1$ elements of ε_ω , the following relationship is satisfied between its metric $M_\alpha(\varepsilon_{\omega-1})$ and $M_\alpha(\varepsilon_\omega)$:

$$M_\alpha(\varepsilon_\omega) = M_\alpha(\varepsilon_{\omega-1}) \cdot \frac{1}{1 + \exp(\alpha\Lambda'_{i_\omega})} \cdot \prod_{\substack{j=i_{\omega-1}+1, \\ j \in \mathcal{I}}}^{j=i_\omega-1} \frac{1}{1 + \exp(-\alpha\Lambda'_j)} \tag{23}$$

Equation (23) shows that $M_\alpha(\varepsilon_{\omega-1})$ can be regarded as part of $M_\alpha(\varepsilon_\omega)$, which fully reflects the sequential nature of SCL decoding. In particular, for a one-order flip-bit $\varepsilon_1 = \{i_1\}$, the above metric can be rewritten as:

$$M_\alpha(\varepsilon_1) = \frac{1}{1 + \exp(\alpha\Lambda'_{i_1})} \cdot \prod_{\substack{j < i_1, \\ j \in \mathcal{I}}} \frac{1}{1 + \exp(-\alpha\Lambda'_j)} \tag{24}$$

Based on the fact that $\frac{1}{1+\exp(x)} = \frac{\exp(-x)}{1+\exp(-x)}$, (22) can be further rewritten as:

$$M_\alpha(\varepsilon_\omega) = \prod_{j \in \varepsilon_\omega} \exp(-\alpha\Lambda'_j) \cdot \prod_{\substack{j \leq i_\omega, \\ j \in \mathcal{I}}} \frac{1}{1 + \exp(-\alpha\Lambda'_j)} \tag{25}$$

To improve the numerical stability, by taking the logarithm of (22), i.e., $M'_\alpha(\varepsilon_\omega) = -\frac{1}{\alpha} \log(M_\alpha(\varepsilon_\omega))$, we obtain the equivalent logarithmic domain metric:

$$M'_\alpha(\varepsilon_\omega) = \sum_{j \in \varepsilon_\omega} \Lambda'_j + \frac{1}{\alpha} \sum_{\substack{j \leq i_\omega, \\ j \in \mathcal{I}}} \log(1 + \exp(-\alpha\Lambda'_j)) \tag{26}$$

Based on the above considerations, the priority of a flip-bit should be in descending order of M_α or ascending order of M'_α . In other words, the higher the value of M_α (or the smaller the value of M'_α), the higher the flipping priority of the corresponding flip-bit. For convenience, the remainder of this paper will use M_α as the path metric.

3.3. Optimization of the Perturbation Parameter

The parameter α is a key factor for calculating the flip-bit metric. This subsection investigates the optimization of α to give higher flipping priority to the channel-induced erroneous decoded bits, thus achieving the goal of correcting the erroneous decoding with a minimum number of additional decoding attempts. In contrast to the D-SCFlip decoding [27], the factors that affect the α in the proposed decoding are not only N , R , and SNR but also L . Therefore, the complexity of optimizing the perturbation parameter should be very high, and it cannot even find a global optimal parameter. To improve the practicality, this paper proposes a compromise method of optimizing the perturbation parameter for the first erroneous decoded bit only.

We denote by $i_{E_1} (i_{E_1} \in \mathcal{I})$ the index where the first erroneous decision occurs in the initial SCL decoding, and the corresponding information bit is denoted as $u_{i_{E_1}}$. Let $\mathcal{L}_{\alpha, E_1}$ denote the list of all the one-order flip-bit, ordered according to decreasing values of their metrics. We denote by $rk_\alpha(i_{E_1})$ the rank of $u_{i_{E_1}}$ within $\mathcal{L}_{\alpha, E_1}$. The smaller the value of $rk_\alpha(i_{E_1})$, the higher the flipping priority of the information bit $u_{i_{E_1}}$, and thus we denote by α_{opt} the optimized perturbation parameter and define it as:

$$\alpha_{opt} = \arg \min_{\alpha} \mathbf{E}(rk_\alpha(i_{E_1})) \tag{27}$$

where $\mathbf{E}(rk_\alpha(i_{E_1}))$ denotes the expected value of random variable $rk_\alpha(i_{E_1})$. Since it is very difficult to solve $\mathbf{E}(rk_\alpha(i_{E_1}))$ directly, we will investigate the law between the expected value of $rk_\alpha(i_{E_1})$ and α for different decoding conditions to obtain indirectly. The main ideas are as follows. Firstly, let $E(rk_\alpha(i_{E_1}))$ denote the expectation value of $rk_\alpha(i_{E_1})$ for a given decoding condition and investigate the relationship between $E(rk_\alpha(i_{E_1}))$ and α under

different conditions of N, R, SNR , and L separately. Secondly, we analyze and summarize the laws reflected in the above relationships. Finally, the data fitting and other methods are used to derive the α_{opt} .

To investigate the relationship between α and $E(rk_\alpha(i_{E_1}))$ under different conditions, we take the relationship between α and $E(rk_\alpha(i_{E_1}))$ under various SNR values as an example, and the process is as follows:

1. Fix parameters N, R , and L , and set the value of SNR .
2. Set the range and step size of α , and the number of Monte Carlo simulations.
3. Perform the Monte Carlo simulation for the current α and SNR values. If the initial SCL decoding does not pass the CRC check, first construct the list $\mathcal{L}_{\alpha, E_1}$, then find the rank of i_{E_1} and note it as $rk_\alpha(i_{E_1})$. Finally, calculate the average value of $rk_\alpha(i_{E_1})$ after all experiments are completed and denoted as $E(rk_\alpha(i_{E_1}))$.
4. Update the α value and return to step (3). Continue to calculate $E(rk_\alpha(i_{E_1}))$ for the new α value until the traversal of α is complete. Plot the relationship between α and $E(rk_\alpha(i_{E_1}))$, then derive the relationship between the two at that SNR value.
5. Update the SNR values and complete steps (1)–(4) again to obtain the relationship between the α and $E(rk_\alpha(i_{E_1}))$ for different SNR values.

Specifically, in this paper, to obtain the relationship between α and $E(rk_\alpha(i_{E_1}))$ where SNR is the variable, we first set the $SNR = \{0.5, 1.0, 1.5, 2.0, 2.5\}$ dB and the fixed parameters $N = 512, R = 1/2, L = 8, \alpha \in [0.1, 10]$ (theoretically $\alpha \in [0.1, \infty)$, but in our experiments we set the range according to the specific situation as $\alpha \in [0.1, 10]$); the step size is 0.2, and the number of Monte Carlo simulations is 10^5 . Second, 10^5 independent CA-SCL decoding experiments are performed for $PC(512, 256 + 16)$ at $\alpha = 0.1$ and $SNR = 0.5$ dB according to step (3), and the corresponding $E(rk_\alpha(i_{E_1}))$ value is obtained. Then, according to step (4), the value of α is updated according to the step size and returned to step (3) to obtain a new $E(rk_\alpha(i_{E_1}))$ value, and so on; when the traversal of α is completed, the relationship curve between α and $E(rk_\alpha(i_{E_1}))$ at $SNR = 0.5$ dB can be obtained. Finally, according to step (5), the SNR value is sequentially updated and the above experiment is repeated to obtain the relationship between α and $E(rk_\alpha(i_{E_1}))$ under various SNR values.

Following the above steps, we can also obtain the relationship between α and $E(rk_\alpha(i_{E_1}))$ when L, N , and R are the variables, respectively. For L as the variable, we set $L = \{2, 8, 16, 32\}$; the fixed parameters $N = 512, R = 1/2, SNR = 1.0$ dB; and the $\alpha \in [0.1, 10]$. For N as the variable, we set $N = \{256, 512, 1024\}$; the fixed parameters $R = 1/2, L = 8, SNR = 1.0$ dB; and the $\alpha \in [0.1, 10]$. Similarly, for R as the variable, we set $R = \{1/3, 1/2, 2/3\}$; the fixed parameters $N = 512, L = 8, SNR = 1.0$ dB; and $\alpha \in [0.1, 10]$.

3.4. The Detailed Description of the Proposed Decoding Algorithm

For ease of presentation, we denote $\mathcal{L}_f^l = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_{T_l}\}$ as the flip-bit list, in which every element is of order $l (l \geq 1)$, and T_l is the number of elements in \mathcal{L}_f^l . For $\forall \varepsilon_t \in \mathcal{L}_f^l$, it contains l elements and ordered in ascending order. $\mathcal{M}_f^l = \{M_\alpha(\varepsilon_1), M_\alpha(\varepsilon_2), \dots, M_\alpha(\varepsilon_{T_l})\}$ is denoted as the metric list corresponding to \mathcal{L}_f^l . For $\forall M_\alpha(\varepsilon_t) \in \mathcal{M}_f^l, M_\alpha(\varepsilon_t)$ is the metric of ε_t where $\varepsilon_t \in \mathcal{L}_f^l$.

The detailed algorithm of LS-SCLF- ω decoding is described in Algorithm 1, and the main decoding procedure is as follows: if the initial SCL decoding fails, the \mathcal{M}_f^l and \mathcal{L}_f^l are initialized by the function $\text{Init}(\cdot)$, and the bit-flipping is performed by $\text{SCLF}(\cdot)$ for the elements in \mathcal{L}_f^l one by one. Whenever incorrect decoding occurs, the \mathcal{L}_f^2 and \mathcal{M}_f^2 are updated by the $\text{Update}(\cdot)$ function based on the current flip-bit. Once a successful decoding occurs, the decoding ends, and the final path is output. If and only if all the extra decoding attempts for the one-order flip-bit fail the CRC check, the decoder proceeds to the next round of additional decoding and follows the above process until a successful decoding occurs or the maximum number of extra decoding is reached. The following is the specific description about the key functions in LS-SCLF- ω decoding:

Algorithm 1: LS-SCLF- ω Decoder.

Input: $y_1^N, \omega, L, \mathcal{A}, \mathcal{I}, \{T_1, T_2, \dots, T_\omega\}, \mathcal{L}_f^1 = \mathcal{L}_f^2 = \dots = \mathcal{L}_f^\omega = \emptyset, \mathcal{M}_f^1 = \mathcal{M}_f^2 = \dots = \mathcal{M}_f^\omega = \emptyset$
Output: \hat{u}_1^N

```

1  $[\hat{u}_1^N, \Lambda'_{\mathcal{I}}] = \text{CA-SCLF}(y_1^N, \mathcal{A})$ 
2 if CRC check( $\hat{u}_1^N$ ) fails then
3   | Init( $\mathcal{L}_f^1, \mathcal{M}_f^1, \Lambda'_{\mathcal{I}}, T_1, \mathcal{I}$ )
4   | flag=0
5   | for  $l = 1$  to  $\omega$  do
6     | for  $j = 1$  to  $T_l$  do
7       |  $[\tilde{u}_1^N, \tilde{\Lambda}'_{\mathcal{I}}] = \text{SCLF}(y_1^N, \mathcal{A}, \varepsilon_j)$ 
8       | if CRC check( $\tilde{u}_1^N$ ) fails and  $l \leq \sqrt{T_{l+1}}$  then
9         | | Update( $\mathcal{L}_f^{l+1}, \mathcal{M}_f^{l+1}, \tilde{\Lambda}'_{\mathcal{I}}, T_{l+1}, \mathcal{A}, \varepsilon_j$ )
10        | | else
11          | |  $\hat{u}_1^N = \tilde{u}_1^N$ 
12          | | flag=1
13          | | Break
14        | | end
15        | | end
16        | | if flag == 1 then
17          | | | Break
18        | | end
19        | | end
20      | end
21      | if flag == 1 then
22        | | Break
23      | end
24    | end
25  | end
26  | return  $\hat{u}_1^N$ 

```

- **Init**($\mathcal{L}_f^1, \mathcal{M}_f^1, \Lambda'_{\mathcal{I}}, T_1, \mathcal{I}$) ($\Lambda'_{\mathcal{I}} = \{\Lambda'_i | i \in \mathcal{I}\}$): The initialization strategy is detailed in Algorithm 2. The preliminary flip-bit list ξ is constructed as $\xi = \{M_\alpha(\varepsilon) | \varepsilon = \{i\}, i \in \mathcal{I}\}$, and (24) gives the calculation of $M_\alpha(\varepsilon)$. Then, the elements in ξ are sorted in descending order by the sort(.) function, with the largest T_1 elements constructing \mathcal{M}_f^1 and the corresponding indexes constructing \mathcal{L}_f^1 .
- **SCLF**($y_1^N, \mathcal{A}, \varepsilon_j$): ε_j is the j th element in \mathcal{L}_f^l , i.e., $\varepsilon_j = \mathcal{L}_f^l(j)$. The **SCLF**(.) function flips all the information bits corresponding to the ε_j in turn in an extra decoding attempt.
- **Update**($\mathcal{L}_f^{l+1}, \mathcal{M}_f^{l+1}, \tilde{\Lambda}'_{\mathcal{I}}, T_{l+1}, \mathcal{A}, \varepsilon_j$): The update algorithm is detailed in Algorithm 3. If **SCLF**($y_1^N, \mathcal{A}, \varepsilon_j$) fails to pass the CRC check and $l \leq \sqrt{T_{l+1}}$; it first calculates the $M_\alpha(\varepsilon')$ of ε' by (25), where $\varepsilon' = \varepsilon_j \cup k, k = \varepsilon_j(\text{end}) + 1, \dots, N, k \in \mathcal{A}$ and $\varepsilon_j(\text{end})$ represents the last element of ε_j . Then, the **Insert**(.) function inserts ε' and $M_\alpha(\varepsilon')$ into the appropriate positions in \mathcal{L}_f^{l+1} and \mathcal{M}_f^{l+1} , respectively, according to the value of $M_\alpha(\varepsilon')$. If the number of elements in \mathcal{L}_f^{l+1} exceeds T_{l+1} , then only the first T_{l+1} elements of \mathcal{L}_f^{l+1} and \mathcal{M}_f^{l+1} are retained. Since $M_\alpha(\varepsilon_j) > M_\alpha(\varepsilon')$, ε' will be inserted into a position behind ε_j .

Algorithm 2: **Init**($\mathcal{L}_f^1, \mathcal{M}_f^1, \Lambda'_{\mathcal{I}}, T_1, \mathcal{I}$).

```

1  $\xi = \{M_\alpha(\varepsilon) | \varepsilon = \{i\}, i \in \mathcal{I}\}$ 
2  $[\mathcal{L}_f^1, \mathcal{M}_f^1] = \text{sort}(\xi, T_1)$ 
3 return  $[\mathcal{L}_f^1, \mathcal{M}_f^1]$ 

```

Algorithm 3: Update $(\mathcal{L}_f^{l+1}, \mathcal{M}_f^{l+1}, \tilde{N}'_{\mathcal{I}}, T_{l+1}, \mathcal{A}, \varepsilon_j)$.

```

1 for  $k = \varepsilon_j(end) + 1, \dots, N$  and  $k \in \mathcal{I}$  do
2    $\varepsilon' = \varepsilon_j \cup k, m = M_\alpha(\varepsilon')$ 
3   Insert  $(\mathcal{L}_f^{l+1}, \mathcal{M}_f^{l+1}, m, \varepsilon', T_{l+1})$ 
4 end
5 return  $[\mathcal{L}_f^{l+1}, \mathcal{M}_f^{l+1}]$ 

```

4. Experiment Design and Simulation Analysis

In this section, two experiments are conducted to verify the performance of the LS-SCLF- ω decoder. The first experiment is to optimize the perturbation parameter. We use Monte Carlo simulation to derive the relationships between α and N, R, L , and SNR and then summarize the applicable range or a certain value under the decoding conditions in this paper. Based on the conclusion of the first experiment, the second experiment simulates the LS-SCLF- ω decoding and compares it with related decoding methods in terms of error correction performance and complexity to verify the advantages and disadvantages of the proposed algorithm. It should be noted that all experiments in this paper are carried out under the binary-input additive white Gaussian noise channel, and the information bits are selected using the Gaussian approximation algorithm for specific SNR values.

4.1. Perturbation Parameter Optimization

In this subsection, the experiments on the relationship between α and $E(rk_\alpha(i_{E_1}))$ are carried out under different SNR, L, N , and R conditions according to the strategy given in Section 3.3 and the relationship curves are shown in Figure 4a–d, respectively. In Figure 4a, all of the curves gradually decrease as α increases, and they reach their minimum value at around $\alpha = 0.9$, after which the curves stabilize. This suggests that the minimum value of $E(rk_\alpha(i_{E_1}))$ is not impacted by the values of SNR , and that the first erroneous decoded bit has the highest flipping priority at $\alpha = 0.9$ under any SNR condition. In Figure 4b, all the curves decrease with increasing α and reach a minimum value around $\alpha = 0.9$, and they gradually converge as α continues to increase. This indicates that the minimum value of $E(rk_\alpha(i_{E_1}))$ is also not affected by the value of L , and the first erroneous decoded bit obtains the highest flipping priority around $\alpha = 0.9$ with any L . Moreover, the curves in Figure 4c,d have approximately the same trend as those in Figure 4a,b, and with the exception of a few curves, they all reach their minimum value at $\alpha = 0.9$ with different N and R values. Therefore, we conclude that the minimum value of $E(rk_\alpha(i_{E_1}))$ is independent of N, R, L , and SNR ; i.e., the minimum value of $E(rk_\alpha(i_{E_1}))$ does not change as the decoding conditions change. Based on the above analysis, we can conclude that the $E(rk_\alpha(i_{E_1}))$ achieves its minimum value at $\alpha = 0.9$, so $\alpha_{opt} = 0.9$.

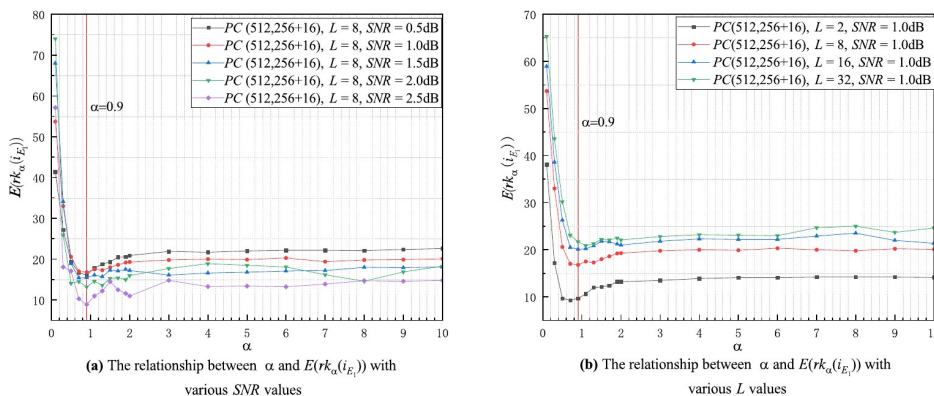


Figure 4. Cont.

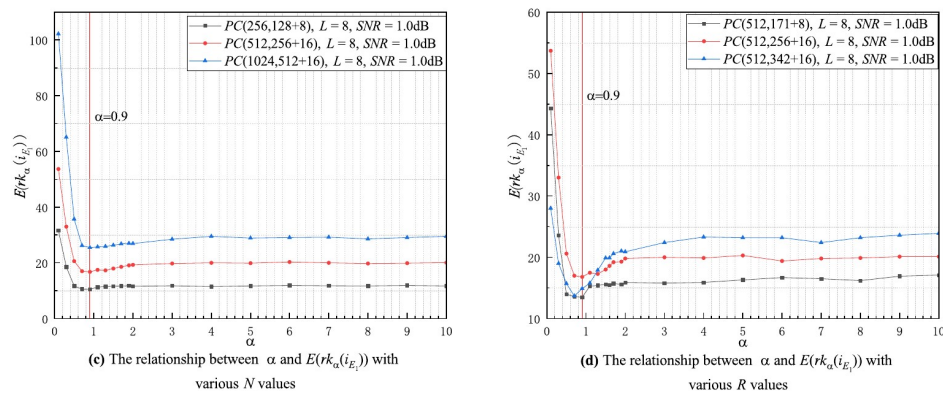


Figure 4. The relationship between α and $E(rk_\alpha(i_{E_1}))$ with various decoding conditions.

4.2. Simulation Results and Performance Analysis

4.2.1. Error Correction Performance Analysis

To verify the error correction performance of LS-SCLF- ω decoding, we simulate it for different conditions and compare it with BF-SCL decoding [30], GSCLF- ω decoding [31], and oracle-assisted SCL- ω (OA-SCL- ω) decoding, where the OA-SCL- ω decoding corresponds to the genie-aided SCL decoding in [31]. In the paper, we choose $\omega = \{1, 2, 3\}$ to perform experiments on each of the above decoders. The WER curves for the LS-SCLF-1 decoding, GSCLF-1 decoding, and BF-SCL decoding under different $T, L, N,$ and R conditions are shown in (a), (b), (c), and (d), respectively, in Figure 5. From Figure 5a, all the WER curves of the above decoders decrease as T increases, and the WER performance of the LS-SCLF-1 and GSCLF-1 decoders is better than that of the BF-SCL decoder. At $T = 10$, the WER values of the LS-SCLF-1 decoder are slightly lower than those of the GSCLF-1 decoder. At $T = 50$, the WER values of the both decoders are basically the same and very close to the WER values of the BF-SCL decoder. From Figure 5b, the WER curves of all the decoders decrease as the L increases, and the WER performance of the LS-SCLF-1 and GSCLF-1 decoders is better than that of the BF-SCL decoder. For $L = 8$, The WER curves of the LS-SCLF-1 and GSCLF-1 decoders are basically the same and very close to the WER curve of the OA-SCL-1 decoder. For $L = 16$ and $L = 32$, the WER values of the LS-SCLF-1 decoder are slightly lower than those of the GSCLF-1 decoder, especially in the regions of $1.0 \text{ dB} \leq \text{SNR} \leq 2.0 \text{ dB}$, where the difference between the two is more obvious. As can be seen in Figure 5c, almost all the WER curves for these decoders roughly intersect at $\text{SNR} = 1.0 \text{ dB}$. In the regions of $\text{SNR} \leq 1.0 \text{ dB}$, the WER values increase as the N increases and vice versa in the $\text{SNR} > 1.0 \text{ dB}$ regions. In addition, at $N = 256$ and $N = 512$, the WER curves of the LS-SCLF-1 and GSCLF-1 decoders are basically the same and slightly higher than that of the OA-SCL-1 decoder, while at $N = 1024$, the WER curve of the LS-SCLF-1 decoder is slightly lower than that of the GSCLF-1 decoder in the regions of $1.0 \text{ dB} \leq \text{SNR} \leq 2.0 \text{ dB}$. As illustrated in Figure 5d, when $R = 2/3$, the WER values of the LS-SCLF-1 decoder are slightly lower than those of the GSCLF-1 decoder, especially in the regions of $1.0 \text{ dB} \leq \text{SNR} \leq 2.0 \text{ dB}$, and the difference between the two is more obvious. When $R = 1/2$ and $R = 2/3$, the WER values of the LS-SCLF-1 decoder are approximately the same as those of the GSCLF-1 decoder, but in the regions around $\text{SNR} = 1.0 \text{ dB}$, the WER values of the LS-SCLF-1 decoder are lower than those of the GSCLF-1 decoder. In summary, Figure 5 shows that the LS-SCLF-1 decoder can correct more decoding caused by a single erroneous bit with the same number of extra decoding than the GSCLF-1 decoder, further proving the effectiveness of the bit-flipping metric and perturbation parameter optimization proposed in this paper. This also implies that the LS-SCLF decoder will also perform better for correcting higher-order erroneous bits.

Figure 6a–c show the WER performance of LS-SCLF-2 decoding and GSCLF-2 decoding for various $L, N,$ and R values. From Figure 6a, it can be seen that the WER values of the LS-SCLF-2 decoder are lower than those of the GSCLF-2 decoder in the

1.0 dB \leq SNR \leq 2.0 dB regions for both $L = 4$ and $L = 8$. From Figure 6b, it can be seen that the WER curves of the LS-SCLF-2 decoder and the GSCLF-2 decoder are basically the same at $N = 1024$ and $N = 256$, while the WER values of the LS-SCLF-2 decoder are slightly lower than those of the GSCLF-2 decoder at $N = 512$. As can be seen in Figure 6c, the WER curves of the LS-SCLF-2 decoder are lower than those of the GSCLF-2 decoder under different R values, and the difference between them gradually increases with increasing SNR. In summary, the overall error correction performance of the LS-SCLF-2 decoder is higher than that of the GSCLF-2 decoder, and the advantage of the LS-SCLF-2 decoder is more obvious in the region of 1.0 dB \leq SNR \leq 2.0 dB. Figure 7 shows the comparison of the error correction performance of different decoders at $\omega = 3$, where $T_1 = 50$, $T_2 = 49$, $T_3 = 100$. Analyzing Figure 7, we can come to a similar conclusion as in Figures 5 and 6; i.e., the error correction performance of the LS-SCLF-3 decoder is slightly better than that of the GSCLF-3 decoder under different decoding conditions.

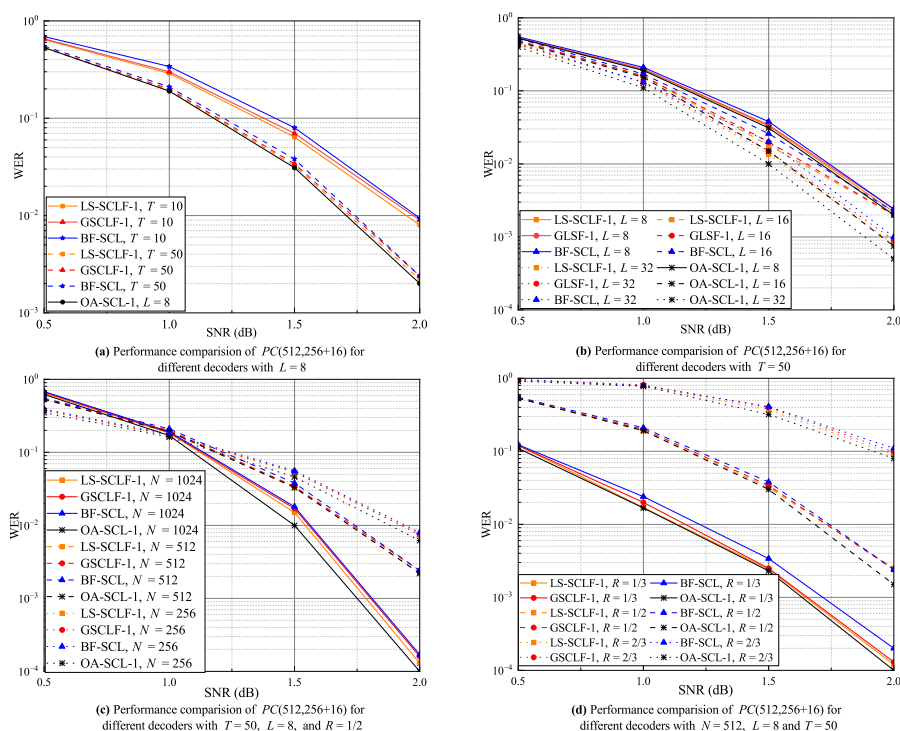


Figure 5. WER curves for different decoders under various decoding conditions with $\omega = 1$.

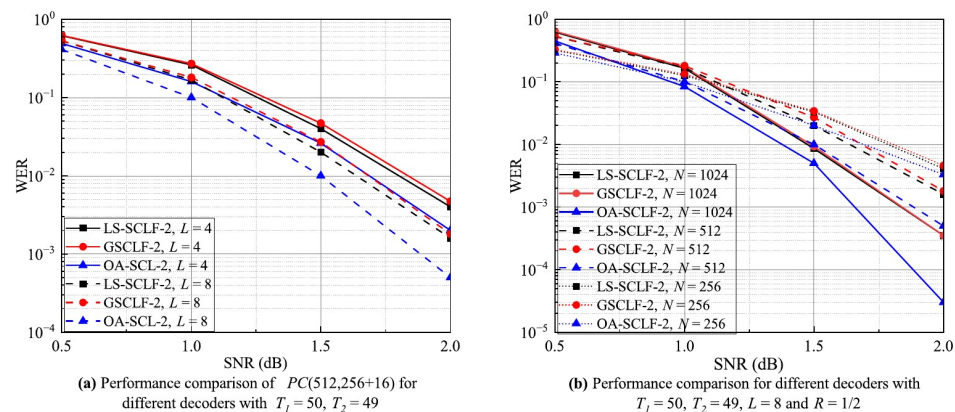
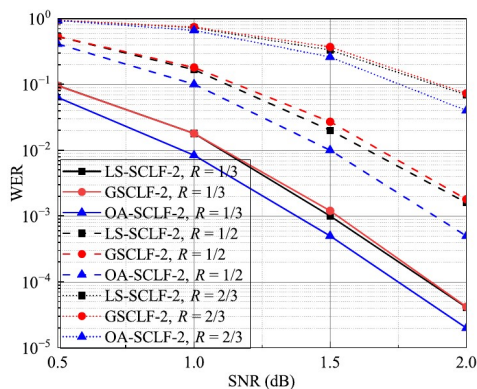


Figure 6. Cont.



(c) Performance comparison for different decoders with $T_1 = 50, T_2 = 49, L = 8,$ and $N = 512$

Figure 6. WER curves for different decoders under various decoding conditions with $\omega = 2$.

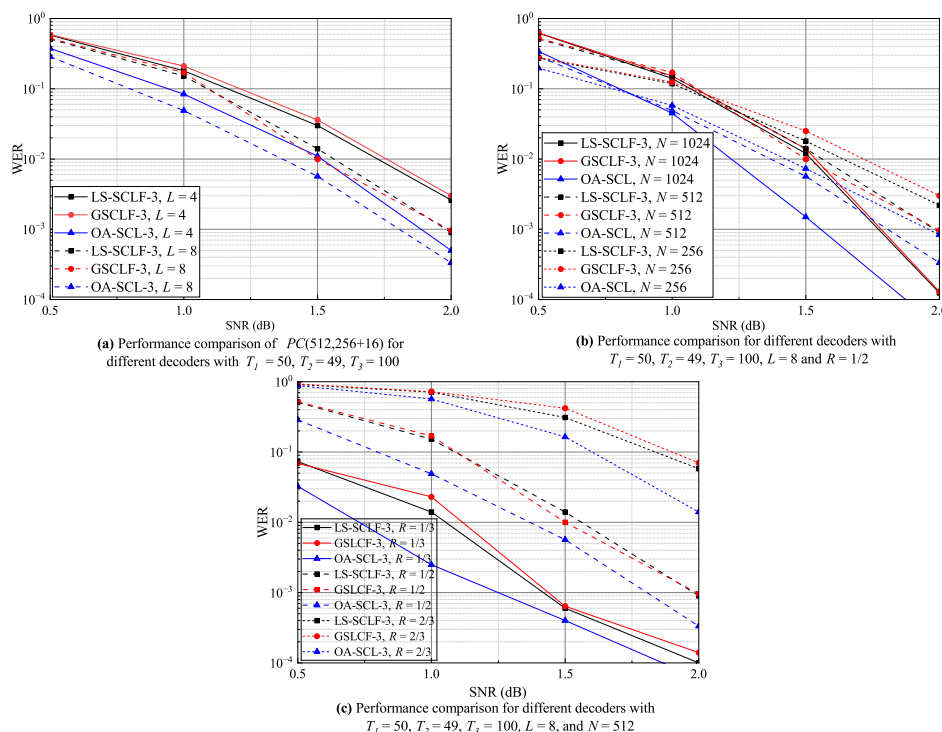


Figure 7. WER curves for different decoders under various decoding conditions with $\omega = 3$.

4.2.2. Decoding Complexity Analysis

The complexity of bit-flipping-based SCL decoding is mainly composed of two parts: the flipping priority determination complexity and the extra decoding complexity, where the former is much smaller than the latter, especially with a large list size. Therefore, this paper mainly considers the extra decoding complexity, which can be expressed as $T_{avg} \cdot (LN \log N)$, where T_{avg} is denoted as the average number of extra decoding attempts. It is clear that the average number of extra decoding attempts determines the complexity of the decoding, and this subsection focuses on the average number of extra decoding attempts. Figures 8–10 show the corresponding average extra decoding attempts for Figures 5, 6, and 7, respectively.

From Figure 8a, at $T = 10$, the decoders BF-SCL, LS-SCLF-1, and GSCLF-1 exhibit an equivalent average number of extra decoding attempts. At $T = 50$, the BF-SCL decoder shows the largest number of extra decoding attempts, whereas the LS-SCLF-1 and GSCLF-1 decoders have similar average extra decoding attempts. As can be seen from Figure 8b, under different L conditions, the average extra decoding attempt curves of the LS-SCLF-1

decoder are all between those of the BF-SCL and GSCLF-1 decoders, indicating that the extra decoding complexity of LS-SCLF-1 decoder is smaller than that of the GSCLF-1 decoder and larger than that of the BF-SCL decoder. In the regions of $1.5 \text{ dB} \leq \text{SNR} \leq 2.0 \text{ dB}$, the difference between the LS-SCLF-1 and GSCLF-1 decoders gradually increases, but with $L = 32$, for example, the difference between them is no more than 3. From Figure 8c, when $N = 1024$ and $N = 512$, the average extra decoding attempts of the LS-SCLF-1 and GSCLF-1 decoders are approximately the same, and both of them are smaller than those of the BF-SCL decoder. When $N = 256$, the average extra decoding attempts of LS-SCLF-1 decoder are slightly larger than those of the GSCLF-1 decoder, but both of them are higher than the average extra decoding attempts of the BF-SCL decoder. As can be seen from Figure 8d, the average extra decoding attempts of the BF-SCL decoder are higher than those of the other two decoders under different R conditions. The LS-SCLF-1 decoder also needs more extra decoding attempts than the GSCLF-1 decoder, and in the regions of $1.5 \text{ dB} \leq \text{SNR} \leq 2.0 \text{ dB}$, the difference between them slightly increases, but it does not exceed 4. Based on the analysis of Figure 8, it can be concluded that overall, the BF-SCL decoder has the highest extra decoding complexity, while the LS-SCLF-1 decoder has slightly higher extra decoding complexity than the GSCLF-1 decoder.

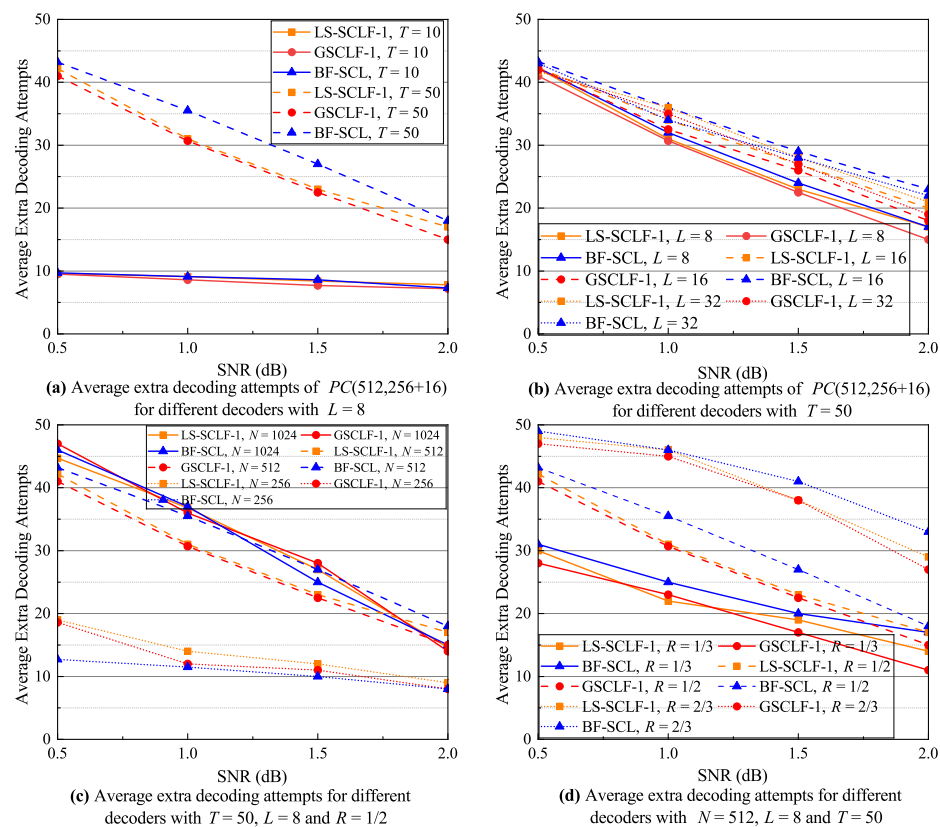


Figure 8. Average extra decoding attempts of decoders under various decoding conditions with $\omega = 1$.

As shown in Figure 9a, the average extra decoding attempts for the LS-SCLF-2 decoder are marginally greater than for the GSCLF-2 decoder at varying L conditions. The discrepancy between the two is more noticeable at $L = 8$, but it does not exceed 2 at the most. As shown in Figure 9b, the average extra decoding attempts of the LS-SCLF-2 decoder and the GSCLF-2 decoder are nearly equal under varying N conditions, and the six curves converge around $\text{SNR} = 1.5 \text{ dB}$. However, the difference between the average extra decoding attempts of the two decoder marginally rises in the $1.5 \text{ dB} \leq \text{SNR} \leq 2.0 \text{ dB}$ regions in comparison with other regions. From Figure 9c, we can see that the extra decoding attempts of both decoders rises considerably as R increases. For R values of one-third, the

average extra decoding attempts for the LS-SCLF-2 decoder and the GSCLF-2 decoder are almost identical, but for values greater than one-half, the average extra decoding attempts of the LS-SCLF-2 decoder are marginally larger than those of the GSCLF-2 decoder. The analysis in Figure 9 demonstrates that the decoding complexity of the LS-SCLF-2 decoder is slightly greater than that of the GSCLF-2 decoder. In Figure 10, we can also observe a similar phenomenon as in Figure 9; i.e., when $\omega = 3$, the average number of extra decoding attempts of the LS-SCLF decoder is higher than those of the GSCLF decoding under different conditions, indicating that the LS-SCLF decoder has a higher average extra decoding complexity compared to the GSCLF decoder. However, the difference is that the gap between the average extra decoding attempts of the two decoders is much larger with $\omega = 3$; e.g., in Figure 10b, when $N = 1024$, the gap between them can reach more than 10 in the regions of $1.0 \text{ dB} \leq \text{SNR} \leq 2.0 \text{ dB}$.

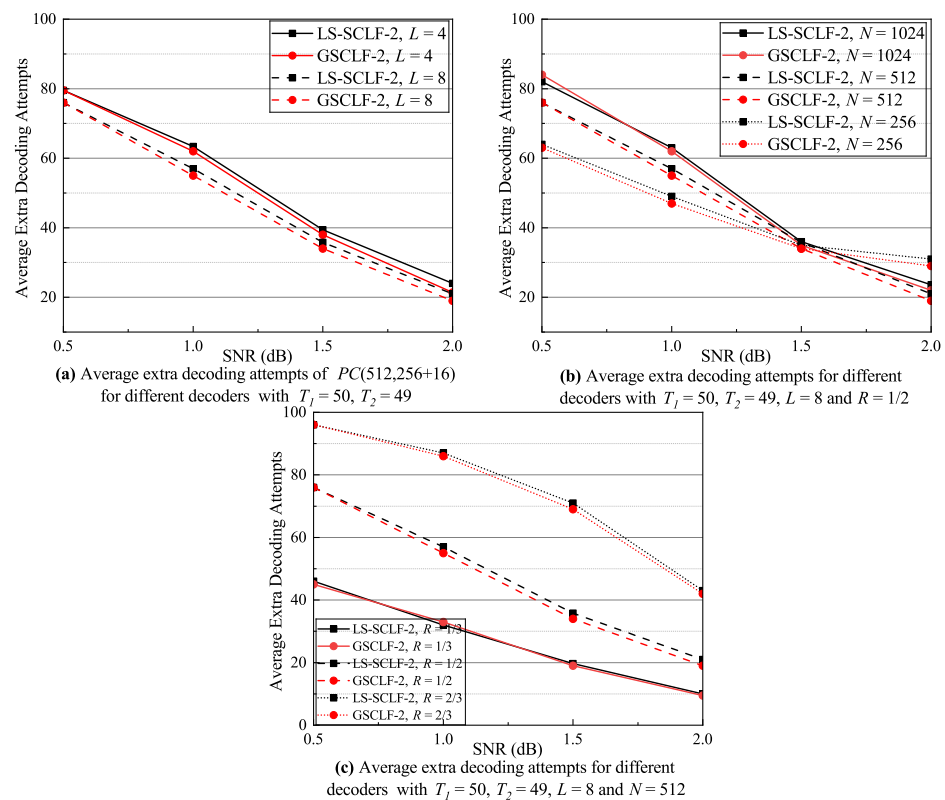


Figure 9. Average extra decoding attempts of decoders under various decoding conditions with $\omega = 2$.

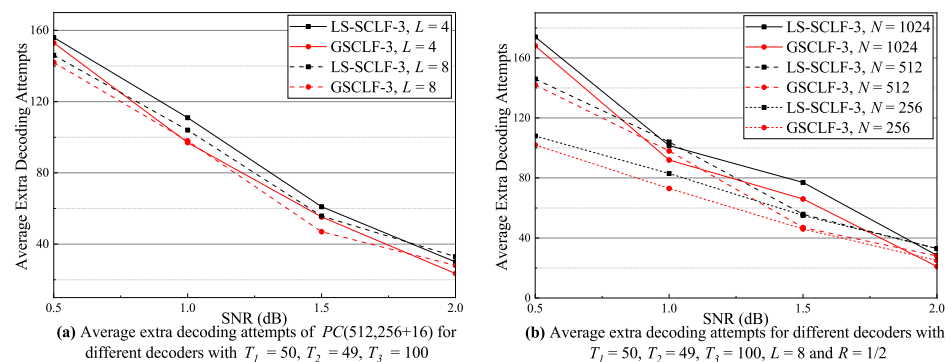


Figure 10. Cont.

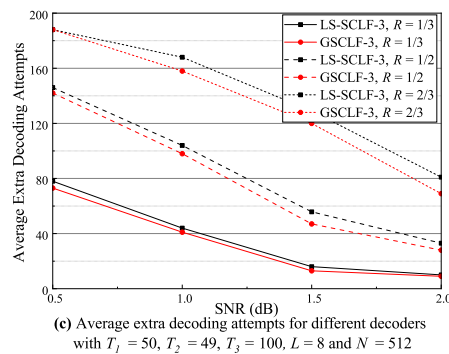


Figure 10. Average extra decoding attempts of decoders under various decoding conditions with $\omega = 3$.

5. Conclusions

In this paper, we propose LS-SCLF- ω decoding, which can simultaneously correct ω erroneous decoded bits. Firstly, a new flip-bit metric is proposed. The metric takes into account the sequential nature of SCL decoding, i.e., the effect of the previous decoded bits on the bits to be decoded is introduced into the calculation of the metric. Then, a perturbation parameter is also introduced to make the metric as close as possible to the error probability of the flip-bit. Finally, a bit-flipping strategy is proposed, which searches and locates the erroneous decoded bits sequentially from lower to higher orders. Simulation results show that the LS-SCLF-1 decoder has better performance than the BF-SCL decoder in both error correction and decoding complexity. The LS-SCLF- ω decoder has better error correction performance than the GSCLF- ω decoder, especially at the medium to high SNR regions, but its decoding complexity is slightly higher. In a word, the proposed decoder can achieve a good balance between the error correction performance and decoding complexity.

However, the perturbation parameter α involved in this paper needs to be optimized through a large number of simulations. Meanwhile, the optimized parameter $\alpha_{opt} = 0.9$ is not the optimal value under all conditions; for example, in Figure 4d, for $PC(512, 342 + 16)$, $\alpha = 0.7$ is its optimal parameter value, while $\alpha = 0.9$ is only a sub-optimal value under all conditions considered in a comprehensive way, which is the reason why the complexity of the proposed decoding algorithm is higher than that of the GSCLF decoder. To solve this problem, we hope to improve the decoding algorithm in the following ways in future work. (1) To reduce the excessive occupation of computing resources, the optimized parameter can be stored in the system in advance. (2) The parameter optimization problem can be solved from a theoretical point of view. The functional relationship between different decoding conditions and the corresponding optimal parameter values is summarized so as to avoid the instability caused by experimental statistics.

Author Contributions: Conceptualization, D.W., J.Y., Y.X. and G.H.; methodology, D.W., J.Y. and Y.X.; software, D.W., Q.X. and X.Y.; validation, Q.X. and X.Y.; supervision, J.Y., Y.X. and G.H.; funding acquisition, J.Y., Y.X. and G.H. All authors have read and agreed to the published version of the manuscript.

Funding: National Natural Science Foundation of China: 62271446, 51574232; Xinjiang Key Research and Development Special Task: 2022B03003-3.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Erdal, A. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input. *IEEE Trans. Inf. Theory* **2009**, *55*, 3051–3073.
2. Lidström, V. Low-density parity-check codes. *IEEE Trans. Inf. Theory* **1962**, *12*, 21–28.
3. Berrou, C.; Glavieux, A. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In Proceedings of the ICC 93-IEEE International Conference on Communications, Geneva, Switzerland, 23–26 May 1993.

4. Liang, H.; Liu, A.; Tong, X.; Gong, C. Raptor-like Rateless Spinal Codes using Outer Systematic Polar Codes for Reliable Deep Space Communications. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020; pp. 1045–1050.
5. Feng, B.; Jiao, J.; Wu, S.; Zhang, Q. How to apply polar codes in high throughput space communications. *Sci. China Tech. Sci.* **2020**, *63*, 1371–1382. [[CrossRef](#)]
6. Zhai, Y.; Li, J.; Feng, H.; Hong, F. Application research of polar coded OFDM underwater acoustic communications. *J. Wireless. Com. Netw.* **2020**, *63*, 1371–1382. [[CrossRef](#)]
7. Liu, F.; Wu, Q.; Li, C.; Chen, F.; Xu, Y. Polar Coding Aided by Adaptive Channel Equalization for Underwater Acoustic Communication. *IEICE Tech. Fund. Electr.* **2023**, *E106.A*, 83–87. [[CrossRef](#)]
8. Lidstrom, V. Evaluation of Polar-Coded Noncoherent Acoustic Underwater Communication. *IEEE J. Ocean. Eng.* **2023**, *2023*, 1–14. [[CrossRef](#)]
9. Wang, H.; Kim, S. Dimming Control Systems with Polar Codes in Visible Light Communication. *IEEE Photonics Technol. Lett.* **2017**, *29*, 1651–1654. [[CrossRef](#)]
10. Wang, H.; Kim, S. Design of Polar Codes for Run-Length Limited Codes in Visible Light Communications. *IEEE Photonics Technol. Lett.* **2019**, *31*, 27–30. [[CrossRef](#)]
11. Wang, H.; Kim, S. Adaptive Puncturing Method for Dimming in Visible Light Communication with Polar Codes. *IEEE Photonics Technol. Lett.* **2018**, *30*, 1780–1783. [[CrossRef](#)]
12. Zheng, H.; Wu, K.; Chen, B. Experimental Demonstration of 9.6 Gbit/s Polar Coded Infrared Light Communication System. *IEEE Photonics Technol. Lett.* **2020**, *32*, 1539–1542. [[CrossRef](#)]
13. Tal, I.; Vardy, A. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory* **2015**, *61*, 2213–2226. [[CrossRef](#)]
14. Niu, K.; Chen, K. CRC-Aided Decoding of Polar Codes. *IEEE Commun. Lett.* **2012**, *16*, 1668–1671. [[CrossRef](#)]
15. Li, B.; Shen, H.; Tse, D. An Adaptive Successive Cancellation List Decoder for Polar Codes with Cyclic Redundancy Check. *IEEE Commun. Lett.* **2012**, *16*, 2044–2047. [[CrossRef](#)]
16. Gao, C.; Liu, R.; Dai, B.; Han, X. Path Splitting Selecting Strategy-Aided Successive Cancellation List Algorithm for Polar Codes. *IEEE Commun. Lett.* **2019**, *23*, 422–425. [[CrossRef](#)]
17. Wang, X.; Zhang, H.; Li, J.; Bao, X.; Xie, K. An Improved Path Splitting Decision-Aided SCL Decoding Algorithm for Polar Codes. *IEEE Commun. Lett.* **2021**, *25*, 3463–3467. [[CrossRef](#)]
18. Zhang, Z.; Zhang, L.; Wang, X.; Zhong, C.; Poor, H.V. A Split-Reduced Successive Cancellation List Decoder for Polar Codes. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 292–302. [[CrossRef](#)]
19. Doan, N.; Hashemi, S.A.; Gross, W.J. Fast Successive-Cancellation List Flip Decoding of Polar Codes. *IEEE Access* **2022**, *10*, 5568–5584. [[CrossRef](#)]
20. Ardakani, M.H.; Hanif, M.; Ardakani, M.; Tellambura, C. Fast Successive-Cancellation-Based Decoders of Polar Codes. *IEEE Trans. Commun.* **2019**, *67*, 4562–4574. [[CrossRef](#)]
21. Sarkis, G.; Giard, P.; Vardy, A.; Thibeault, C.; Gross, W.J. Fast List Decoders for Polar Codes. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 318–328. [[CrossRef](#)]
22. Afisiadis, O. A low-complexity improved successive cancellation decoder for polar codes. In Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 2–5 November 2014; pp. 2116–2120.
23. Ercan, F.; Condo, C.; Gross, W.J. Improved Bit-Flipping Algorithm for Successive Cancellation Decoding of Polar Codes. *IEEE Trans. Commun.* **2019**, *67*, 61–72. [[CrossRef](#)]
24. Condo, C.; Ercan, F.; Gross, W.J. Improved Successive Cancellation Flip Decoding of Polar Codes Based on Error Distribution. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Barcelona, Spain, 15–18 April 2018; pp. 19–24.
25. Zhang, Z.; Qin, K.; Zhang, L.; Chen, G.T. Progressive Bit-Flipping Decoding of Polar Codes: A Critical-Set Based Tree Search Approach. *IEEE Access* **2018**, *6*, 57738–57750. [[CrossRef](#)]
26. Wang, D.; Yin, J.; Xu, Y.; Yang, X.; Hua, G. Threshold Based D-SCFlip Decoding of Polar Codes. *IEICE Trans. Commun.* **2023**, *E106.B*, 635–644. [[CrossRef](#)]
27. Chandesris, L.; Savin, V.; Declercq, D. Dynamic-SCFlip Decoding of Polar Codes. *IEEE Trans. Commun.* **2018**, *66*, 2333–2345. [[CrossRef](#)]
28. Yu, Y.Y.; Pan, Z.W.; Liu, N.; You, X.H. Successive Cancellation List Bit-flip Decoder for Polar Codes. In Proceedings of the 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP), Hangzhou, China, 18–20 October 2018; pp. 1–6.
29. Rowshan, M.; Viterbo, E. Shifted Pruning for Path Recovery in List Decoding of Polar Codes. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 27–30 January 2021; pp. 1179–1184.
30. Cheng, F.; Liu, A.; Ren, J.D. Bit-Flip Algorithm for Successive Cancellation List Decoder of Polar Codes. *IEEE Access* **2019**, *7*, 58346–58352. [[CrossRef](#)]
31. Pan, Y.H.; Wang, C.H.; Ueng, Y.L. Generalized SCL-Flip Decoding of Polar Codes. In Proceedings of the GLOBECOM 2020—2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6.

32. Balatsoukas-Stimming, A.; Parizi, M.B.; Burg, A. LLR-Based Successive Cancellation List Decoding of Polar Codes. *IEEE Trans. Signal. Process.* **2015**, *63*, 5165–5179. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.