

Shift Pruning-Based SCL Decoding for Polar Codes

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

¹ Faculty of Electronic Information Engineering, Huaiyin Institute of Technology, Huaian 223003, China; 12024030@hyit.edu.cn

² School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101408, China

³ School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China; xuanyang@cumt.edu.cn (X.Y.); jiaqiyan@cumt.edu.cn (J.Y.); ghua@cumt.edu.cn (G.H.)

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

Abstract: In the context of the high-speed development of 5G communications, high-performance decoding schemes for polar codes are a hot spot in channel coding research. Shift pruning successive cancellation list (SP-SCL) decoding aims to recover the correct path by shift pruning in the extra SCL decoding. However, the current SP-SCL decoding is inflexible in determining the shift positions. In this paper, a flexible shift pruning SCL (FSP-SCL) decoding is proposed. Firstly, the reasons for movement and the eliminated states of the correct path are analyzed in detail using the path metric range (PMR), and on this basis, the validity of the method adopted in this paper for determining the shift priority of the information bits is verified. Secondly, the FSP-SCL decoding proposes two methods for determining the shift positions. One is the log-likelihood ratio (LLR) threshold method, which compares the LLR values of the eliminated paths on the shift bit with the corresponding LLR threshold to determine the shift positions. The other is the path distance method. It combines the minimum distance between the eliminated paths and the received vector with the path metrics to determine the shift positions. Both methods are more flexible and practical, as they can calculate the corresponding shift positions online based on a specific shift bit, avoiding the high complexity caused by the simulation method. Finally, this paper designs various experimental schemes to verify the decoding performance of the FSP-SCL. The experimental results show that in terms of error-correction performance, the LLR threshold-based FSP-SCL (FSPL (LLR threshold)) decoding, the path distance-based FSP-SCL (FSPL (path distance)) decoding and the existing SP-SCL decoding are roughly equal overall. In terms of decoding complexity, FSPL (LLR threshold) decoding is slightly better than FSPL (path distance) decoding, and the decoding complexity of both is lower than that of SP-SCL decoding, with the difference being more pronounced in the medium to high SNR regions.

Keywords: polar codes; successive cancellation list decoding; shift pruning algorithm; shift-position determination

MSC: 94A24



Citation: Wang, D.; Yin, J.; Xu, Y.; Yang, X.; Yan, J.; Hua, G. Shift Pruning-Based SCL Decoding for Polar Codes. *Mathematics* **2024**, *12*, 2937. <https://doi.org/10.3390/math12182937>

Academic Editor: Ioannis Tsoulos

Received: 19 May 2024

Revised: 19 June 2024

Accepted: 28 June 2024

Published: 21 September 2024



Copyright: © 2024 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 were first proposed by Arikan in 2009 and have been proved to achieve the channel capacity of binary-input discrete memoryless channels under successive cancellation (SC) decoding, as the code length tends to infinity [1]. Polar codes have been selected as the channel coding scheme for control information in the 5G enhanced mobile broadband scenario due to their explicit construction method and low-complexity coding process. Therefore, after more than a decade of development, polar codes have achieved many research results in the fields of mobile communications, underwater communications, optical communications, etc. [2–6].

However, at more practical lengths, SC decoding suffers from a significant degradation in error-correction performance. To overcome this problem, successive cancellation list

(SCL) decoding has been proposed [7]. The SCL decoder can retain multiple paths simultaneously, reducing the likelihood of the correct path being eliminated. The reference [8] proposed a cyclic redundancy check (CRC)-aided SCL (CA-SCL) decoder, which helps the SCL decoder to make the final selection from the candidate paths by concatenating a CRC code at the end of the codeword. Obviously, the error-correction performance of SCL decoding can be directly improved by increasing the list size (L), but this will also significantly increase the decoding complexity. In fact, the decoding complexity of the SCL decoder is L times the decoding complexity of the SC decoder. In particular, if L is too large, the decoding complexity will continue to increase, but the improvement in error-correction performance will be extremely slow. Therefore, various schemes have been proposed in the references [9–14] to reduce the complexity of SCL decoding.

Inspired by the idea of bit-flipping [15–21], SCL bit-flipping (SCLF) decoding algorithms have been successively proposed [22–26]. If the initial decoding fails the CRC check, these decoding algorithms will flip the erroneous decoded bit in the extra decoding attempts, which can improve the error-correction performance to varying degrees. References [22–24] have developed various metrics to evaluate erroneous decoded bits caused by channel noise, with the aim of accurately locating these bits and correcting them with minimum complexity. The references [25,26] used the idea of segmented checking, where the CRC code is divided into several sub-segments and placed in the middle of the codeword. Segmented checks are used to verify the decoding results so that incorrect decoding can be corrected in time to reduce complexity. The above SCLF decoding algorithms improve the error-correction performance of SCL decoding, and some of them limit the increase in decoding complexity very well. However, they rely heavily on the precise location of the erroneous decoded bits, and when selecting the reserved paths for the flip bits in the extra decoding, they always make the opposite decision to the initial decoding and cannot select the reserved paths according to the actual situation.

The shift pruning SCL (SP-SCL) decoder [27,28], on the other hand, proposes a new idea of flexible selecting the shift positions (i.e., the reserved paths in the SCLF decoder) for the shift bit (i.e., the erroneous decoded bit in the SCLF decoder). Reference [27] first proposed to select the paths $\{k+1, k+2, \dots, k+L\}$, $k \in \{1, 2, \dots, L\}$ as the shift positions, instead of the paths $\{L+1, L+2, \dots, 2L\}$ in the SCLF decoders among the $2L$ sorted paths at the shift bit. Then, it proved the effectiveness of the algorithm through experiments, but it did not explicitly propose a method to determine the value of the k . Reference [28] proposed an improved SP-SCL decoder based on [27]. The decoder proposes a new metric for determining the shift priority of information bits and uses simulation to count the distribution of the correct path after it is eliminated to determine the shift positions. Compared with SCLF decoding, references [27,28] provided more flexibility in selecting the reserved paths at the erroneous decoded bit, and the proposed decoding method reduces the dependence on the accuracy of the locating of the erroneous decoded bit, which is a novel idea. However, the shift-position determination method in [28] only applicable to certain decoding conditions, which is inflexible and less applicable.

Based on the above analysis, this paper proposes a flexible shift pruning SCL (FSP-SCL) decoder. The FSP-SCL decoder first adopts the metric proposed in reference [28] to determine the shift priority of the information bits, and further verifies the validity of the metric based on a detailed analysis of the movement and the eliminated state of the correct path. Then, the FSP-SCL decoding designs two methods to determine the shift positions, i.e., the LLR threshold method and the path distance method, both of which can dynamically select L continuous or non-continuous sorted paths $\{k_{t_1}, k_{t_2}, \dots, k_{t_j}, \dots, k_{t_L}\}$, $k_{t_j} \in \{1, 2, \dots, 2L\}$, as the shift positions. These flexible shift-position determination methods allow FSP-SCL decoding to better balance error-correction performance and decoding complexity.

The remainder of this paper is organized as follows. Section 2 introduces the polar codes, CA-SCL decoder, and SP-SCL decoder. Section 3 details the movement and elimination state of the correct path. Section 4 provides a detailed description of the pro-

posed FSP-SCL decoding algorithm. Simulation results are provided in Section 5, and the conclusions are drawn in Section 6.

2. Preliminaries

2.1. Polar Codes

$N = 2^n$ independent and identically distributed binary-input discrete memoryless symmetric channels can be formed into N symmetric sub-channels with different channel capacities after channel polarization. The reliability of information transmission can be effectively improved by selecting the K sub-channels with the largest channel capacity to transmit information bits, while the remaining sub-channels transmit frozen bits (which are usually set to 0 and known to both the receiver and the transmitter). In this paper, we denote the information bit index set and the frozen bit index set by $\mathcal{A}$ and $\mathcal{A}^c$, respectively. For a polar code $PC(N, K, \mathcal{A})$, it can be encoded by:

$$x_1^N = u_1^N \cdot \mathbf{G}_N \quad (1)$$

where $x_1^N = (x_1, x_2, \dots, x_N)$ denotes the encoded vector, $u_1^N = (u_1, u_2, \dots, u_N)$ denotes the data vector and contains K information bits and $N - K$ frozen bits. $\mathbf{G}_N$ denotes the generator matrix and $\mathbf{G}_N = B_N F^{\otimes n}$. $B_N = R_N(I_2 \otimes B_{N/2})$ presents a bit-reversal permutation matrix, where R_N is an odd-even separation matrix. $F_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $(\cdot)^{\otimes n}$ denotes the n th Kronecker power.

For any information bit u_i , its hard decision estimate with SC decoding can be obtained by:

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

where L_i is the LLR corresponding to u_i and $\text{sign}(0) = \pm 1$ with equal probability. L_i is defined as:

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

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

In this paper, we select a CRC code as the outer code and concatenate it with an inner polar code. Therefore, the information bits of a polar code can be expressed as $K = k + r$, where k and r are the number of information bits and check bits of the outer CRC code, respectively. For the sake of simplicity, in the remainder of this paper, we will denote the polar code by $PC(N, k + r)$.

2.2. CA-SCL Decoder

The decoding process of CA-SCL decoder is shown in Figure 1. Unlike SC decoding, the CA-SCL decoder maintains two possible outcomes for each information bit by path splitting, thus increasing the number of candidate paths to increase the probability of obtaining the correct path among the candidates. On the other hand, for frozen bits, it directly uses the predefined value, and the number of candidate paths remains unchanged. If the number of candidate paths reaches $2L$, the decoder sorts the candidate paths in ascending order according to their path metric (PM) values and reserves the paths $\{1, 2, \dots, L\}$ through path competition. At the end of decoding, the CA-SCL decoder selects the path with the lowest PM value from the paths that pass the CRC check as the final decoding output. Note that the SCL decoder directly selects the first path as the decoding output without CRC checking.

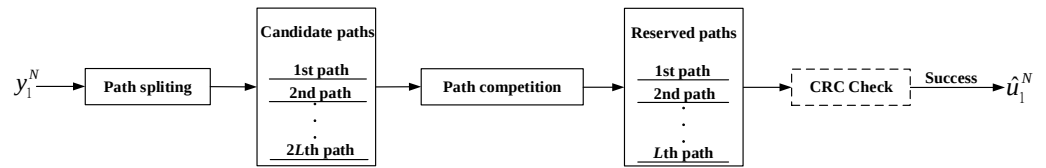


Figure 1. The decoding process of the CA-SCL decoder.

For the CA-SCL decoder, we define $\hat{u}_{1,l}^i = (\hat{u}_{1,l}, \hat{u}_{2,l}, \dots, \hat{u}_{i,l})$ as the hard decision estimate of $u_{1,l}^i$ for the l th path, and the path metric of $\hat{u}_{i,l}$ can be computed as:

$$PM_l^i = \sum_{j=1}^i \log \left(1 + \exp \left(-(1 - 2\hat{u}_{j,l}) L_{j,l} \right) \right) \quad (4)$$

where the initial value of PM_l^i is 0, $L_{j,l} = \log \left(\frac{\Pr(u_{i,l}=0|y_1^N, \hat{u}_{1,l}^{i-1})}{\Pr(u_{i,l}=1|y_1^N, \hat{u}_{1,l}^{i-1})} \right)$ denotes the LLR for $\hat{u}_{i,l}$. After simplification, Equation (4) can be rewritten as:

$$PM_l^i = \begin{cases} PM_l^{i-1} & \text{if } \hat{u}_{i,l} = h(L_{i,l}) \\ PM_l^{i-1} + |L_{i,l}| & \text{if } \hat{u}_{i,l} \neq h(L_{i,l}) \end{cases} \quad (5)$$

2.3. SP-SCL Decoder

The decoding process of the SP-SCL decoder [28] is shown in Figure 2 and the main steps are as follows. First, the SP-SCL decoder performs standard SCL decoding with CRC check. If the decoding is successful, it directly outputs the decoding result; otherwise, the shift bit calculator determines the candidate shift bit list according to the decoding result and starts the extra decoding attempts. The decoding process of the extra decoding is basically the same as that of the SCL decoding, and the only difference is that when the extra decoding decodes at the shift bit, it performs the shift pruning on the candidate paths and reserved the paths $\{k+1, k+2, \dots, k+L\}$ (i.e., the shift positions), $k \in \{1, 2, \dots, L\}$. 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. In this paper, we define $\mathcal{A} \setminus \mathcal{A}_0$ the set consisting of all information indices $i \notin \mathcal{A}_0$, where $\mathcal{A}_0$ is the index set of the first $\log_2 L$ information bits. In particular, when the SP-SCL decoder is executed to shift bit u_j , $j \in \mathcal{A} \setminus \mathcal{A}_0$, assuming that $2L$ candidate paths satisfy $PM_1^j < PM_2^j < \dots < PM_{2L}^j$, the paths with indexes $\{k+1, k+2, \dots, k+L\}$ are reserved in the SP-SCL decoding. Note that the SP-SCL decoder only corrects one candidate shift bit in each extra decoding. Meanwhile, the reserved paths are summarized by Monte Carlo simulation and kept unchanged in SP-SCL decoding, which lacks flexibility, as the reserved paths cannot be adjusted according to the specific shifted bits.

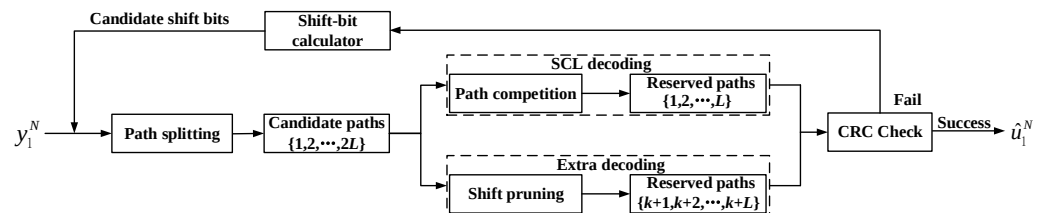


Figure 2. The decoding process of the SP-SCL decoder.

3. The Movement and Elimination States of the Correct Path

In SCL decoding, there may be a situation where a certain information bit u_i of the correct path does not match its hard estimate decision $\hat{u}_i$, i.e., $u_i = 1 - \hat{u}_i$, which we call a path penalty. Obviously, the correct path will be shifted in the $l = L$ direction due to path

penalties, etc. In this section, we first analyze in detail the reasons for the movement of the correct path and its states after elimination. Then, the effectiveness of the metric adopted in this paper to determine the shift priority of the information bits is further verified.

From the analyses in [27,28], the list size is an important factor in the ability of the correct path to withstand path penalties. Figure 3 shows the relative frequency of the number of penalties that lead to the elimination of the correct path for different list sizes for $PC(512, 256 + 16)$ when $SNR = 1.5$ dB. Figure 3 shows that as the list size increases, a greater number of penalties are required to eliminate the correct path. This suggests that as the list size increases, SCL decoding becomes more tolerant of path penalties and has greater gains in error-correction performance. When the list size is small, a single path penalty has a significant impact on the performance of the SCL decoder, e.g., when $L = 2$, a single path penalty caused more than 40% of the frame errors.

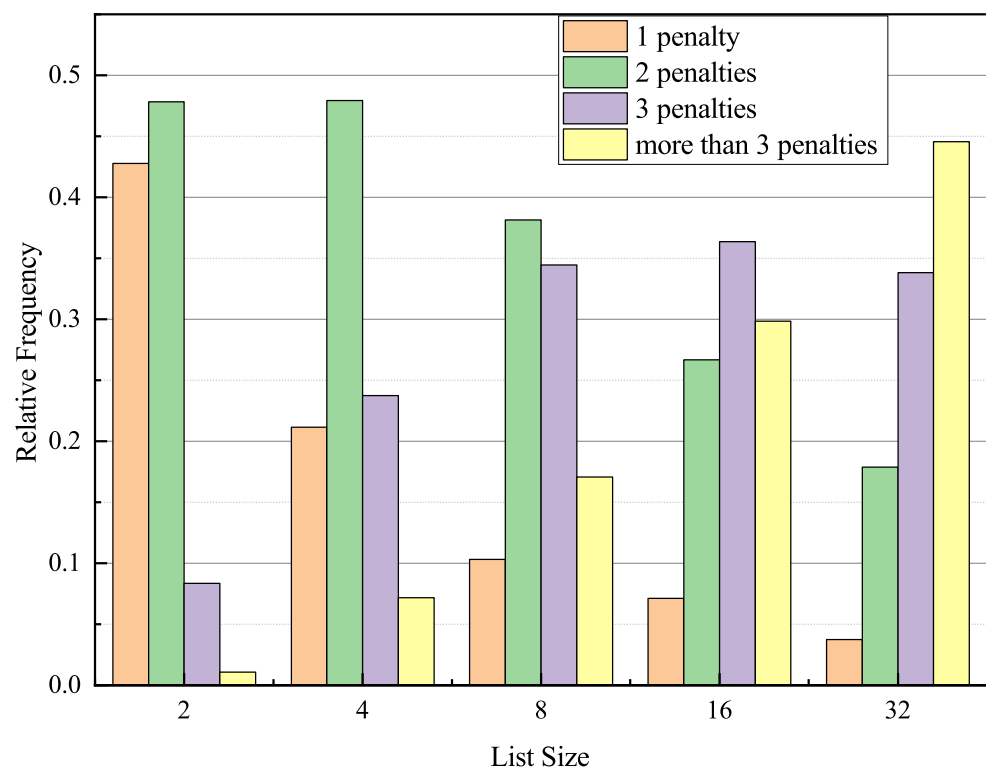


Figure 3. The relative frequency of number of penalties leading to elimination of correct path at different list sizes for $PC(512, 256 + 16)$ when $SNR = 1.5$ dB.

It is important to explore the movement characteristics of the correct path during SCL decoding, which provides new ideas and solutions for recovering the correct path. For example, the shift pruning operation can be used to compensate for the movement before it is eliminated, thus avoiding the occurrence of erroneous decoding in advance and reducing the dependence on the accurate identification of the first erroneous decoded bit. However, it is very difficult to obtain the law of movement of the correct path from theory; only some indicators can be proposed indirectly to indicate the movement of the correct path or to assist in judging whether elimination of the correct path occurs. Reference [27] uses the path metric range (PMR), i.e., $PMR_i = PM_L^i - PM_1^i$, to elucidate the movement characteristics of the correct path in the SCL decoding. Figure 4 shows the average PMR of each bit in failure decoding and the average $|LLR|$ for each bit for $PC(512, 256 + 16)$ at $SNR = 1.5$ dB and $L = 8$.

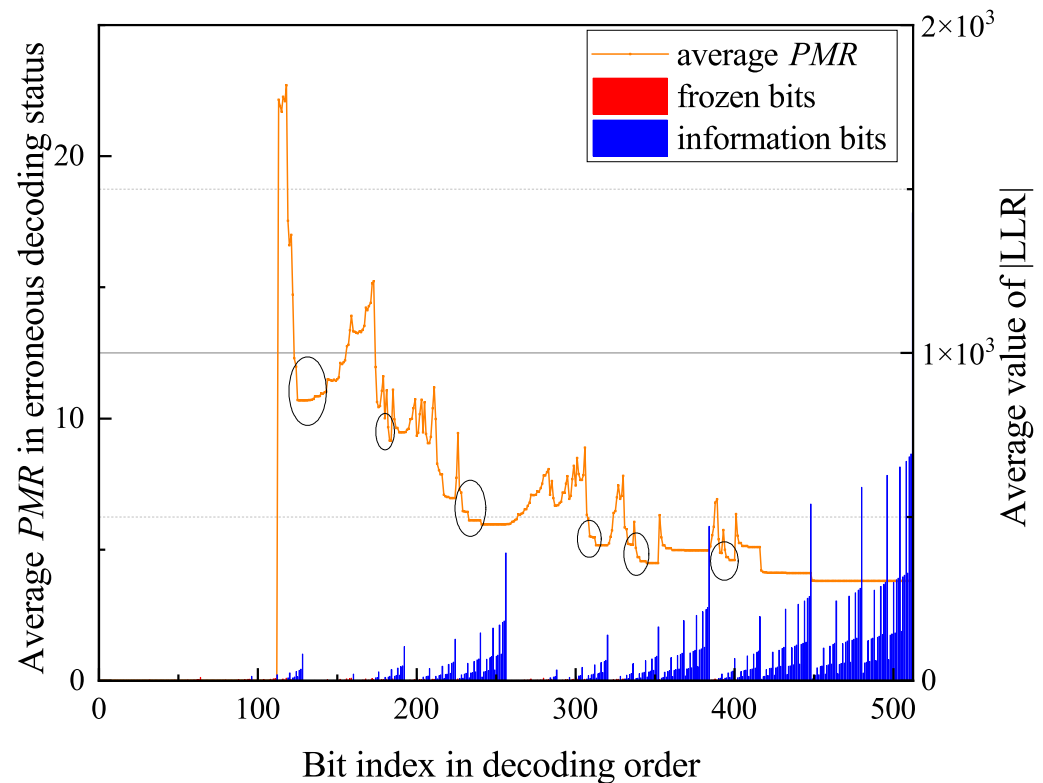


Figure 4. The average PMR of each bit in erroneous decoding and the average $|LLR|$ for each bit for $PC(512, 256 + 16)$ at $SNR = 1.5$ dB and $L = 8$. The points marked with black circles are the local minima of PMR .

It can be seen in Figure 4 that the decreasing part of the PMR curve tends to occur at low-reliability information bits due to the fact that the low-reliability information bits usually correspond to smaller $|LLR|$ values, and there is a high probability that both sub-paths of the low-reliability information bit are reserved in the path competition. It is worth noting that after the i th decoding step, if there exists a path l and a correct path l_c , $l < l_c \leq L$, and their path metrics are PM_l^i and $PM_{l_c}^i$, respectively. From Lemma 1 in [28], if the $L_{i+1,l}$ of path l satisfies $PM_l^i + |L_{i+1,l}| < PM_{l_c}^i$ after the $i + 1$ th decoding step, then the correct path must move in the direction of L , regardless of whether it is penalized or not. Therefore, based on the above analysis, we can conclude that when decoding over a series of low-reliability information bits, the correct path moves in the direction of $l = L$ not only due to the path penalty, but also due to the fact that the metrics of the two sub-paths of the path l ($l < l_c$) are smaller than the metric of the correct path. In this paper, we define the former as active movement and the latter as passive movement. If the correct path is eliminated at the i th decoding step in CA-SCL decoding, then $\hat{u}_{1,l_c}^{i-1} \in \mathcal{L}^{i-1}$ and $\hat{u}_{1,l_c}^i \in \tilde{\mathcal{L}}^i$, where $\mathcal{L}$ and $\tilde{\mathcal{L}}$ represent the list of reserved and eliminated paths, respectively. Assuming that $\hat{u}_{1,\bar{l}_c}^i$ is the other sub-path of $\hat{u}_{1,l_c}^{i-1}$, according to the position of $\hat{u}_{1,\bar{l}_c}^i$ in the sorted paths, this paper divides the states of the eliminated correct path into two types: The first is the eliminated state. In this state, the path $\bar{l}_c$ is also eliminated after the path competition, i.e., $\bar{l}_c > L$. If u_i satisfies the hard decision estimate, i.e., $L < l_c < \bar{l}_c$, then the correct path must have undergone a passive movement, and conversely, for $L < \bar{l}_c < l_c$, the correct path must have undergone an active movement. The second is the preserved state. In this state, the path $\bar{l}_c$ is preserved in the path competition, i.e., $\bar{l}_c < L < l_c$, and it is clear that u_i does not satisfy the hard decision estimate, and the correct path must have undergone an active movement.

In addition, the local minima on the PMR curve (marked with black circles) in Figure 4 appear after a series of low-reliability bits, where most errors are observed. Therefore,

reference [26] combines *PMR* with the critical set to prioritize the shift bits. In this paper, to track the movement of the correct path in the decoding process, we use the metric proposed in [28], which is calculated as

$$\sigma_i = \sum_{l=1}^{2L} PM_l^i \quad (6)$$

Figure 5 shows the relative frequency at which the first erroneous decision occurs in each bit and the average value of σ_i for each bit in the erroneous state for $PC(512, 256 + 16)$ when $SNR = 1.5$ dB and $L = 8$. It is obvious that the overall trend of σ_i is similar to that of the average value of $|LLR|$ in Figure 4. The values of σ_i show an overall upward trend as the bit index increases, and its value also increases as the bit index increases in each segment. In addition, the information bits with the smallest values of σ_i overlap well with the information bits that are most prone to erroneous decoding. This means that if the information bits are sorted in descending order of σ_i , the first erroneous decision caused by channel noise is ranked high and can be quickly corrected in extra decoding. Using σ_i as the metric to determine the shift priority has better applicability, as it neither requires the help of the critical set to narrow down the candidate bits, nor avoids the complexity of computing the critical set.

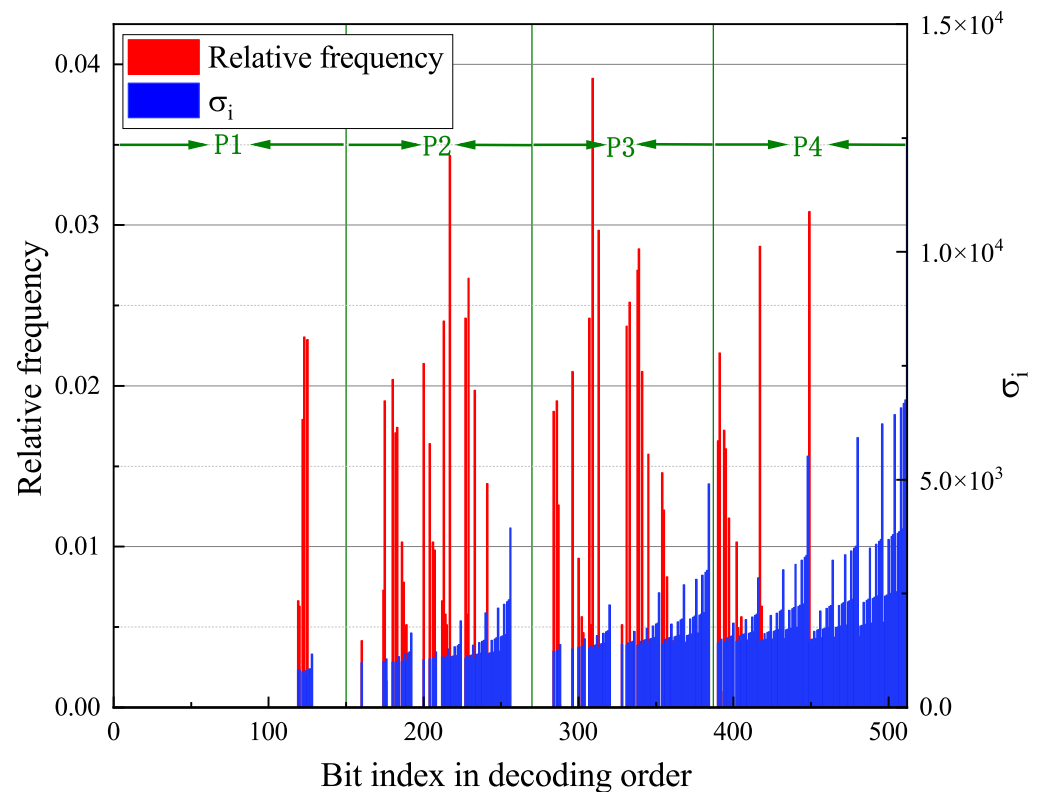


Figure 5. The relative frequency at which the first erroneous decision occurs in each bit and the average value of σ_i for each bit in erroneous state for $PC(512, 256 + 16)$ when $SNR = 1.5$ dB and $L = 8$.

4. The Flexible Shift-Pruning List Decoder

Figure 6 shows the process of the FSP-SCL decoding, which is roughly the same as the process of the SP-SCL decoding but differs in that it proposes two methods for determining the shift positions. The most important feature of these two methods is that they are able to calculate the corresponding shift positions based on the specific shift bit online during the extra decoding, thus providing greater flexibility and avoiding the increase in complexity caused by the simulation method of determining the shift position offline.

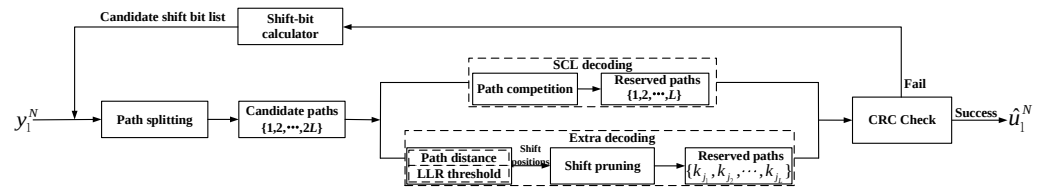


Figure 6. The decoding process of the FSP-SCL decoder.

4.1. Flexible Shift-Position Selection

The correct path is shifted to the eliminated paths list in the path competition due to path penalties and so on. How to narrow down the range of correct path among these eliminated paths is important for shift-pruning decoding. Therefore, we propose two flexible methods to determine the shift positions. The first is the LLR threshold method, which uses the LLR threshold to determine the more reliable paths in the eliminated path list to determine the shift positions. The second is the path distance method, which determines the shift positions by taking the path with the smallest distance from the received vector among the eliminated paths as the largest path. Both methods can adapt to different decoding conditions by flexibly selecting the shift positions during the entire decoding process.

LLR Threshold Method

In SC decoding, the probability of the first erroneous decision occurring at the i th ($i \in \mathcal{A}$) step is defined as:

$$P_e(u_i) = \Pr(\hat{u}_i \neq u_i, \hat{u}_1^{i-1} = u_1^{i-1}, y_1^N) \quad (7)$$

By a simple derivation, Equation (7) can be rewritten as:

$$P_e(u_i) = \Pr(\hat{u}_i \neq u_i | \hat{u}_1^{i-1} = u_1^{i-1}, y_1^N) \cdot \Pr(\hat{u}_1^{i-1} = u_1^{i-1} | y_1^N) \quad (8)$$

Supposing that the u_1^{i-1} is correctly decoded, the $P_e(u_i)$ depends on $\Pr(\hat{u}_i \neq u_i | \hat{u}_1^{i-1} = u_1^{i-1}, y_1^N)$, and it is an arduous task to compute $\Pr(\hat{u}_i \neq u_i | \hat{u}_1^{i-1} = u_1^{i-1}, y_1^N)$. An approximation is proposed in the reference [19]:

$$\Pr(\hat{u}_i \neq u_i | \hat{u}_1^{i-1} = u_1^{i-1}, y_1^N) \approx \frac{1}{1 + \exp(\alpha |L_i|)} \quad (9)$$

where α is the perturbation parameter, and $\alpha > 0$. It can be seen from Equations (3) and (9) that the decoding result of u_i in SC decoding is closely related to L_i , where the sign of L_i determines the value of $\hat{u}_i$, and the value of L_i reflects the reliability of the hard decision estimate. A larger value of L_i indicates a higher probability of $\hat{u}_i = u_i$, i.e., a smaller value of $P_e(u_i)$. Therefore, finding an LLR threshold (L_{th}) of the information bit is crucial to determine the reliability of the sub-paths after path splitting. Once the LLR threshold (L_{th}^i) of u_i is determined, the reliability of $\hat{u}_i$ can be evaluated with L_{th}^i at the i th decoding step. If $|L_i| > L_{th}^i$, the hard decision estimate of u_i can be judged to be correct; otherwise, it is difficult to make a conclusion about whether the hard decision estimate of u_i is correct, i.e., it is probably correct.

Similar to SC decoding, when CA-SCL decodes over a series of low-reliability information bits, it is assumed that the correct path is eliminated at the i th step, then $\hat{u}_{1,l_c}^{i-1} \in \mathcal{L}^{i-1}$, $\hat{u}_{1,l_c}^i \in \tilde{\mathcal{L}}^i$. Since each path can be considered SC decoding, we can use the LLR threshold to narrow down the candidates for the correct path among the eliminated paths. For $\forall l \in \{L+1, L+2, \dots, 2L\}$, we will treat it as a candidate for the correct path if it satisfies one of the following two conditions. In the first condition, when $|L_{i,l}| < L_{th}^i$, path l is listed as a candidate path. This is because the first erroneous decoded bit usually occurs in the sub-channels with low reliability, and the $|LLR|$ is usually smaller. In the second

condition, when $|L_{i,l}| \geq L_{th}^i$, the path with $\hat{u}_i$ that matches the hard decision estimate can be considered a candidate correct path. This is because if $\hat{u}_{i,l}$ satisfies the hard estimate decision, the higher the value of $L_{i,l}$, the higher the probability of correct decoding.

Calculating L_{th}^i is the key to narrowing down the range of the correct path, based on the path splitting inequality in [14]:

$$\hat{u}_i = \begin{cases} 0, & \text{if } L_{i,l} > \log \frac{1 - p_e(u_i)}{p_e(u_i)}, \\ 1, & \text{if } L_{i,l} < -\log \frac{1 - p_e(u_i)}{p_e(u_i)}, \\ \text{split,} & \text{otherwise.} \end{cases} \quad (10)$$

for the additive white Gaussian noise channels, we conclude that the L_{th}^i can be computed as:

$$L_{th}^i = \left| \log \frac{1 - p_e(u_i)}{p_e(u_i)} \right|, \quad i \in \mathcal{A} \quad (11)$$

For any path l in CA-SCL decoding, $p_e(u_i)$ can be calculated as:

$$p_e(u_i) = Q(\sqrt{\mathbf{E}[L_i]}/2) \quad (12)$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} e^{-\frac{t^2}{2}} dt$ and $\mathbf{E}[L_i]$ is the mathematical expectation of L_i (the subscript denoting the path index are omitted for ease of presentation). See reference [29] for the calculation method of $\mathbf{E}[L_i]$.

Figure 7 illustrates the process of the LLR threshold method, and the function SSP1(.) refers to the LLR threshold method, which is detailed in Algorithm 1. The parameters $\{L_{i,L+1}, L_{i,L+2}, \dots, L_{i,2L}\}$ in the SSP1(.) function are arranged in ascending order. Firstly, the LLR threshold of the shift bit is calculate by Equation (11) (line 2), and secondly, the LLR values of these paths $\{L+1, L+2, \dots, 2L\}$ at the shift bit u_i are compared sequentially with L_{th}^i . Then, the indexes of the paths that satisfy the first condition (line 5) and the second condition (line 8) are collected in the set ζ . Finally, the number of elements in ζ is counted (line 12) and used to determine the shift positions sp . The elements in sp are the path indexes to be preserved at the shift bit in the extra decoding. Note that the path indexes in the set sp may not be contiguous.

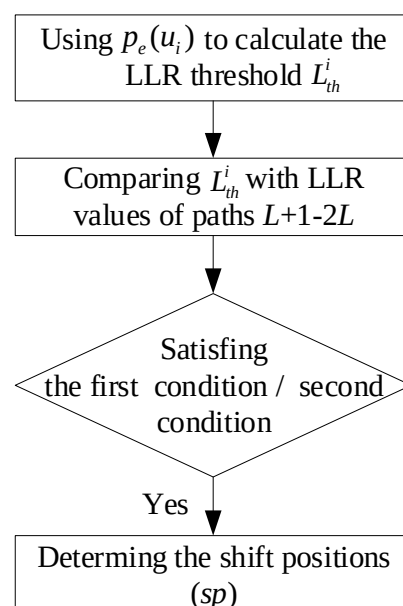


Figure 7. The process of the LLR threshold.

Algorithm 1: SSP1($\{L_{i,L+1}, L_{i,L+2}, \dots, L_{i,2L}\}, p_e(u_i)$)

```

1  $\xi = \emptyset$ 
2  $L_{th}^i = \left\lceil \log \frac{1-p_e(u_i)}{p_e(u_i)} \right\rceil$ 
3 for  $l = L + 1$  to  $2L$  do
4   if  $|L_{i,l}| < L_{th}^i$  then
5      $\xi = \xi \cup \{l\}$ 
6   else
7     if  $(1 - 2\hat{u}_{i,l})L_{i,l} > 0$  then
8        $\xi = \xi \cup \{l\}$ 
9     end
10  end
11 end
12  $t = \text{length}(\xi)$ 
13  $sp = \{t + 1, t + 2, \dots, L\} \cup \xi$  // determine the shift positions
14 return  $sp$ 

```

4.2. Path Distance Method

The LLR threshold method determines the candidate range of the correct path by comparing the LLR value of the shift bit with the threshold, which does not take into account the effect of the path metric. Generally speaking, the smaller the distance between the decoding path and the receiving vector y_1^N , the higher the probability that the path is the correct decoding path. Therefore, FSP-SCL decoding combines the path distance and path metric to propose the path distance method. It first calculates the path with the minimum distance from the received vector among the eliminated paths. Second, using the metric of the path with the minimum distance as an upper bound, the consecutive L paths whose metrics are not more than the upper bound are selected as the candidate set of the correct path.

Figure 8 illustrates the process of the path distance method, and the function SSP2(.) refers to the path distance method, which is detailed in Algorithm 2. The parameters of $\{\hat{u}_{1,L+1}^i, \hat{u}_{1,L+2}^i, \dots, \hat{u}_{1,2L}^i\}$ in the SSP2(.) function are arranged in ascending order according to their PM values. Firstly, the Euclidean distance between the path in the list $\{\hat{u}_{1,L+1}^i, \hat{u}_{1,L+2}^i, \dots, \hat{u}_{1,2L}^i\}$ and y_1^i is calculated in turn (lines 2–5). Then, the index of the path corresponding to the minimum distance among the $2L$ paths is calculated and denoted as st (line 6). Finally, the consecutive L path with indexes less than st is used to determine the shift positions (line 7).

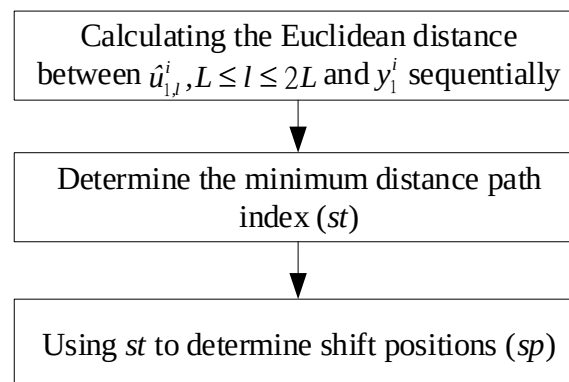


Figure 8. The process of the path distance.

Algorithm 2: SSP2($\{\hat{u}_{1,L+1}^i, \hat{u}_{1,L+2}^i, \dots, \hat{u}_{1,2L}^i\}, y_1^N$)

```

1  $d = \text{zeros}(1, L)$  // initialize the set  $d$  with  $L$  zeros
2 for  $l = L + 1$  to  $2L$  do
3    $d(l - L) = \sqrt{\sum_{k=1}^i (\hat{u}_{k,l} - y_k)^2}$ 
4   // calculate the Euclidean distance between  $\hat{u}_{1,l}^i$  and  $y_1^i$ 
5 end
6  $st = \arg \min_{1 \leq l \leq L} d + L$ 
7  $sp = \{st - L + 1, st - L + 2, \dots, st\}$  // determine the shift positions
8 return  $sp$ 

```

4.3. The Detailed Description of the FSP-SCL Decoding Algorithm

Based on the above analysis, the detailed description of the FSP-SCL decoding is shown in Algorithm 3. First, the CA-SCL(.) decoding is performed, where if there is a path that passes the CRC check, the path is returned directly and the decoding is terminated. Otherwise, the σ is constructed (line 11) and sorted in ascending order according to its elements to obtain the candidate shift bit list $indexSet$ (line 12). Then, the extra decoding is performed, and in each extra decoding, the function FSP-SCL(.) (line 14) performs a shift pruning operation on one element of $indexSet$ until the decoding is successful or the maximum number of extra decoding attempts T is reached (lines 14–22).

Algorithm 3: The FSP-SCL decoding.

Input: the received vector y_1^N , information bit index set $\mathcal{A}$, list size L , maximum extra decoding attempts T

Output: the decoding output $\hat{u}_1^N$

```

1  $crcPass \leftarrow false$ 
2  $[\hat{u}_{1,1}^N, \hat{u}_{1,2}^N, \dots, \hat{u}_{1,L}^N] = \text{CA-SCL}(y_1^N, \mathcal{A})$ 
3 for  $l = 1$  to  $L$  do
4   if  $\text{CRC check}(\hat{u}_{1,l}^N)$  success then
5      $crcPass \leftarrow true$ 
6      $\hat{u}_1^N = \hat{u}_{1,l}^N$ 
7     return  $\hat{u}_1^N$ 
8   end
9 end
10 if  $crcPass == false$  then
11    $\sigma = \{\sigma_i | \sigma_i = \text{sum}(PM_1^i : PM_{2L}^i), i \in \mathcal{A} \setminus \mathcal{A}_0\}$ 
12    $indexSet = \text{sort}(\sigma, 'ascend')$ 
13   // determine the candidate shift bit list
14   for  $t = 1$  to  $T$  do
15      $[\tilde{u}_{1,1}^N, \tilde{u}_{1,2}^N, \dots, \tilde{u}_{1,L}^N] = \text{FSP-SCL}(y_1^N, \mathcal{A}, indexSet(t))$  // Algorithm 4
16     for  $l = 1$  to  $L$  do
17       if  $\text{CRC check}(\tilde{u}_{1,l}^N)$  success then
18          $\hat{u}_1^N = \tilde{u}_{1,l}^N$ 
19         return  $\hat{u}_1^N$ 
20       end
21     end
22   end
23 end

```

Algorithm 4: FSP-SCL($y_1^N, \mathcal{A}, \text{indexSet}(t)$)

```

1  $\mathcal{L} \leftarrow \{1\}$  //initialize the path list
2 for  $i = 1$  to  $N$  do
3   perform standard SCL decoding
4   path pruning when  $i \in \mathcal{A} \setminus \mathcal{A}_0$ 
5   if  $i == \text{indexSet}(t)$  then
6      $sp = \text{SSP1}(\{L_{i,L+1}, L_{i,L+2}, \dots, L_{i,2L}\}, p_e(u_i))$  // Algorithm 1 or
7      $sp = \text{SSP2}(\{\tilde{u}_{1,L+1}^i, \tilde{u}_{1,L+2}^i, \dots, \tilde{u}_{1,2L}^i\}, y_1^i)$  // Algorithm 2
8      $\mathcal{L} \leftarrow sp$ 
9   else
10     $\mathcal{L} \leftarrow \{1, 2, \dots, L\}$ 
11  end
12 end
13 return  $[\tilde{u}_{1,1}^N, \tilde{u}_{1,2}^N, \dots, \tilde{u}_{1,L}^N]$ 

```

4.4. Theoretical Analysis of Complexity

The decoding complexity of SP-SCL and FSP-SCL decoding consists of three main components: average extra decoding complexity, shift bit determination complexity and shift-position determination complexity. The average extra decoding complexity can be calculated as $O(T_{avg}LN \log N)$, where T_{avg} is the average number of extra decoding attempts, and it is clear that the average extra decoding complexity depends on T_{avg} for a given decoding condition. Shift bit determination mainly involves the calculation and sorting of σ . The method of shift bit determination is the same for SP-SCL and FSP-SCL decoding, so the complexity is also the same and will not be analyzed in detail in this paper.

To compute the shift positions, SP-SCL decoding determines the shift positions by counting the distribution of correct paths using simulations. Assuming that the number of simulations is M , the computational complexity of the shift positions in SP-SCL decoding is $O(MLN \log N)$. In FSPL (LLR threshold) decoding, the shift positions are determined using the SSP1(.) function, which first calculates the LLR threshold of the corresponding shift bit using Equation (9). The $p_e(u_i)$ in Equation (9) is the intermediate result of the polar codes construction, and its computational complexity depends on the computational complexity of $E[L_i]$ and the $Q(\cdot)$ function. According to the reference [29], the computational complexity of $E[L_i]$ is $O(N)$. The computational complexity of the $Q(\cdot)$ function depends on the number of shift bits τ , which is $O(\tau)$. Therefore, the computational complexity of $p_e(u_i)$ is $O(N)$. Additionally, in Equation (10), the direct computational complexity of the LLR threshold is $O(\tau)$. Since the computational complexity of the remaining steps in Algorithm 1 is much lower than that of the LLR threshold, the complexity of determining the shifted position in ASPL decoding based on threshold comparison is $O(N)$. The FSPL (path distance) decoding uses the SSP2(.) function to determine the shift positions, which first computes the distance between the L paths and the received vector, assuming that the shift bit is indexed by i , then the complexity of calculating the path distance is $O(\tau i L)$, and the sorting complexity of L paths is $O(\tau L^2)$. In general, $L < N$, and the maximum complexity of determining the shift positions in FSPL (path distance) decoding is $O(\tau NL)$. Therefore, the complexity of computing shift positions for FSPL (path distance) decoding and FSPL (LLR threshold) decoding is lower than that of SP-SCL decoding.

5. Simulation Results

Error-correction performance and decoding complexity are two important indicators for evaluating the decoding algorithms. However, error-correction performance and complexity are usually contradictory, i.e., improvement in error-correction performance is often at the expense of decoding complexity. In order to comprehensively evaluate the decoding algorithm proposed in this paper, we design several experiments to verify the

error-correction performance and the decoding complexity with different extra decoding attempts T , list sizes L , code lengths N , and code rates R , respectively. Furthermore, all the experiments in this paper are performed under the binary-input additive white noise Gaussian channel, and the information bits are selected at a certain SNR using the Gaussian approximation algorithm.

5.1. Error-Correction Performance

To demonstrate the error-correction performance of the FSP-SCL decoding, we compare the FSP-SCL decoding with SP-SCL [28] decoding and CA-SCL decoding, where the method of determining the shift positions based on the LLR threshold and the path distance in the FSP-SCL decoding is notated as FSPL (LLR threshold) and FSPL (path distance). Figure 9 shows the FER curves for different decoders under different conditions.

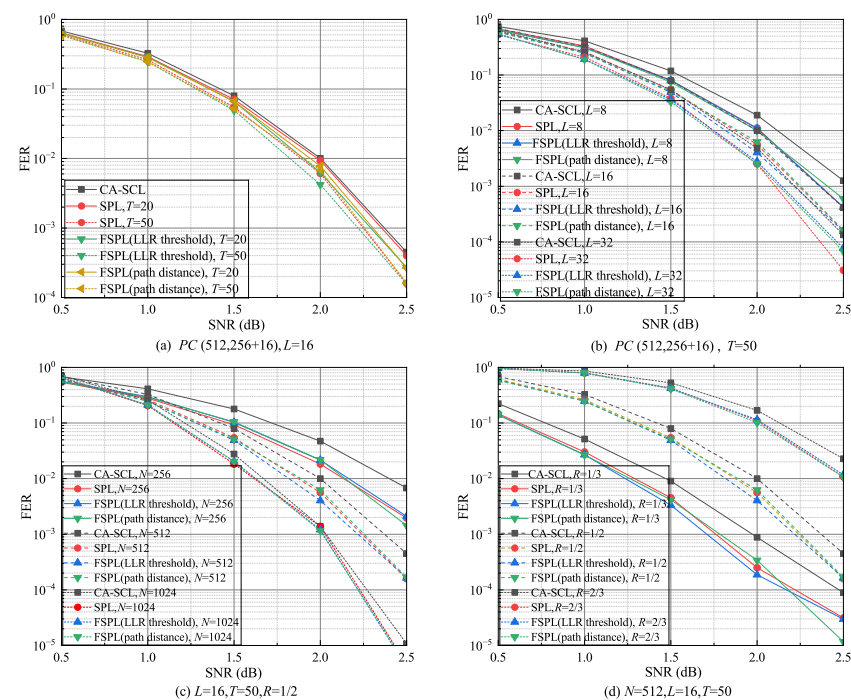


Figure 9. Error-correction performance comparison for different decoders under different conditions.

Figure 9a shows the error-correction performance of $PC(512, 256 + 16)$ with different decoders for $L = 16$, $T = \{20, 50\}$. At $T = 20$, the FER values of FSP-SCL decoding are smaller than those of SP-SCL decoding, especially in the medium to high SNR regions, i.e., $SNR > 1.5$ dB. At $T = 50$, the FER values of SP-SCL and FSPL (path distance) decoding are approximately the same, while the FSPL (LLR threshold) decoding have the smallest FER values among the three decoders, especially at $SNR > 1.5$ dB. Figure 9b shows a comparison of the error-correction performance of different decoders for $PC(512, 256 + 16)$ with $T = 50$, $L = \{8, 16, 32\}$. It is clear that the FER values of FSPL (LLR threshold) decoding and FSPL (path distance) decoding are basically the same and slightly lower than those of SP-SCL decoding for different L conditions at $0.5 \text{ dB} < SNR < 1.5 \text{ dB}$. At $SNR > 1.5 \text{ dB}$, the FER values of the three decoders are basically the same. Figure 9c shows the FER curves of $PC(256, 128 + 16)$, $PC(512, 256 + 16)$ and $PC(1024, 512 + 16)$ with different decoding algorithms at $T = 50$ and $L = 16$. The three decoders as a whole have similar FER values at different values of N , while the SP-SCL and FSPL (LLR threshold) decoders have the smallest FER values in the medium to high SNR regions at $N = 256$ and $N = 512$ respectively. Figure 9d shows the FER curves of $PC(512, 171 + 12)$, $PC(512, 256 + 16)$ and $PC(512, 342 + 16)$ with three decoding algorithms at $T = 50$ and $L = 16$. Obviously, the FER curves of the FSPL (path distance) and SP-SCL decoders basically overlap, while the FSPL (LLR threshold) decoder has the smallest FER values, especially when $R < 2/3$.

Based on the above analysis, the three decoders have similar error-correction performance, with FSPL (LLR threshold) slightly outperforming the other two decoders.

5.2. Decoding Complexity

To compare the average extra decoding complexity of SP-SCL and FSP-SCL decoding in more detail, Figure 10 shows the average number of extra decoding attempts for the two decoding algorithms, and (a), (b), (c), and (d) in Figure 10 correspond to (a), (b), (c), and (d) in Figure 9, respectively. Figure 10a shows that when $T = 20$, the average number of extra decoding attempts of the three decoders are basically the same. When $T = 50$, the curves of the FSPL (LLR threshold) decoding and the FSPL (path distance) decoding are lower than that of the SP-SCL decoding. In addition, the FSPL (LLR threshold) decoding has a lower average number of extra decoding attempts than the FSPL (path distance) decoding, especially in the $SNR > 1.0$ dB regions. For example, the FSPL (LLR threshold) decoding has only 81% and 89% of the average number of extra decoding attempts of the SP-SCL decoding and FSPL (path distance) decoding at $SNR = 2.0$ dB. Figure 10b shows that for the same L values, the average number of extra decoding attempts for SP-SCL decoding is greater than that for FSPL (LLR threshold) and FSPL (path distance) decoding, and the difference is more pronounced at high SNR values. The FSPL (LLR threshold) and FSPL (path distance) decoding curves are intertwined and the overall difference is small. As can be seen in Figure 10c, overall, with the same code lengths, the curves of the FSPL (LLR threshold) decoding are slightly lower than the curves of the FSPL (path distance) decoding, and they are both lower than the curves of the SP-SCL decoding with the same code lengths. In particular, when $N > 256$ and $SNR = 2.0$ dB, the average number of extra decoding attempts for the FSPL (LLR threshold) decoding and FSPL (path distance) decoding is approximately 90% of that for the SP-SCL decoding. Figure 10d shows that the average number of extra decoding attempts of FSPL (LLR threshold) and FSPL (path distance) decoding is lower than that of SP-SCL decoding overall for different R values, and the difference increases with decreasing R values. For $R < 2/3$, the average number of extra decoding attempts of FSPL (LLR threshold) decoding is lower than that of FSPL (path distance) decoding, and the difference is greater in the medium to high SNR regions.

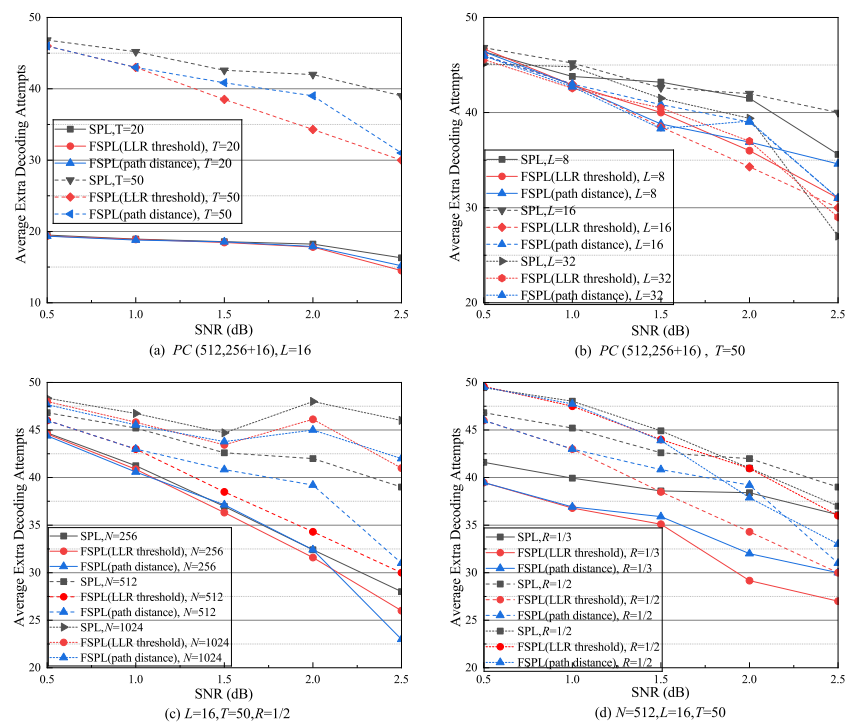


Figure 10. Average extra decoding attempts for different decoders with various conditions.

Based on the above analyses, under the condition that FSP-SCL decoding and SP-SCL decoding have the same shift bit determination complexity, FSP-SCL decoding has low complexity in average extra decoding and shift-positions determination, so the FSP-SCL decoding proposed in this paper has the advantage of decoding complexity compared to SP-SCL decoding.

6. Conclusions

In this paper, we study the reasons for the shift in the correct path during SCL decoding, and analyze the state of the correct path after it is eliminated. Based on this, the FSP-SCL decoding algorithm is proposed, which has two methods to determine the shift positions, namely, the LLR threshold method and the path distance method. In practical applications, the offline simulation experiment method used for SPL decoding needs to calculate the shift positions in advance according to the decoding conditions and set the shift positions in the system in advance, whereas the algorithms proposed in this paper can flexibly calculate the corresponding shift positions online according to the specific shift bits, which is characterized by high flexibility and low complexity. The experimental results show that the FSP-SCL decoding proposed in this paper has some improvement in decoding complexity compared with the SP-SCL algorithm, and the error-correction performance is basically the same.

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

Funding: This research was funded by the National Natural Science Foundation of China under Grant 62271466 and 51574232.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

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

References

1. Erdal, A. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Trans. Inf. Theory* **2009**, *55*, 3051–3073.
2. Lidström, V. Evaluation of Polar-Coded Noncoherent Acoustic Underwater Communication. *IEEE J. Oceanic Eng.* **2023**, 1–14. [\[CrossRef\]](#)
3. Zhai, Y.; Li, J.; Feng, H.; Hong, F. Application research of polar coded OFDM underwater acoustic communications. *J. Wirel. Commun. Netw.* **2023**, *2023*, 26. [\[CrossRef\]](#)
4. Liu, F.; Wu, Q.; Li, C.; Xu, Y. Polar Coding Aided by Adaptive Channel Equalization for Underwater Acoustic Communication. *Trans. Inst. Electron. Inf. Commun. Eng. Sect. E* **2023**, *E106-A*, 83–87. [\[CrossRef\]](#)
5. Yang, F.; Fang, J.; Xiao, S.; Chen, Z.; Hu, W. Shaped polar codes for dimmable visible light communication. *Opt. Commun.* **2021**, *496*, 127126. [\[CrossRef\]](#)
6. Wang, H.; Kim, S. Adaptive Puncturing Method for Dimming in Visible Light Communication With Polar Codes. *IEEE Photonics J.* **2022**, *30*, 1780–1783. [\[CrossRef\]](#)
7. Tal, I.; Vardy, A. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory* **2015**, *61*, 2213–2226. [\[CrossRef\]](#)
8. Niu, K.; Chen, K. CRC-Aided Decoding of Polar Codes. *IEEE Commun. Lett.* **2023**, *16*, 1668–1671. [\[CrossRef\]](#)
9. Xu, Q.; Pan, Z.; Liu, N.; You, X. A complexity-reduced fast successive cancellation list decoder for polar codes. *Sci. China Inf. Sci.* **2017**, *61*, 022309. [\[CrossRef\]](#)
10. Wang, C.H.; Pan, Y.; Lin, Y.; Ueng, Y. Post-Processing for CRC-Aided Successive Cancellation List Decoding of Polar Codes. *IEEE Commun. Lett.* **2020**, *24*, 1395–1399. [\[CrossRef\]](#)
11. 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. [\[CrossRef\]](#)
12. Ardakani, M.H.; Hanif, M.; Ardakani, M.; Tellambura, C. Fast Successive-Cancellation-Based Decoders of Polar Codes. *IEEE Trans. Commun.* **2019**, *67*, 4562–4574. [\[CrossRef\]](#)
13. 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. [\[CrossRef\]](#)

14. 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. [\[CrossRef\]](#)
15. 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. [\[CrossRef\]](#)
16. Ercan, F.; Condo, C.; Hashemi, S.A.; Gross, W.J. Partitioned Successive-Cancellation Flip Decoding of Polar Codes. In Proceedings of the 2018 IEEE International Conference on Communications, Kansas City, MO, USA, 20–24 May 2018.
17. 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 (Wcnw), Barcelona, Spain, 15–18 April 2018; pp. 19–24.
18. 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. [\[CrossRef\]](#)
19. Chandesaris, L.; Savin, V.; Declercq, D. Dynamic-SCFlip Decoding of Polar Codes. *IEEE Trans. Commun.* **2018**, *66*, 2333–2345. [\[CrossRef\]](#)
20. Zhang, X.; Liu, Y.; Chen, S. BER Evaluation Based SCFlip Algorithm for Polar Codes Decoding. *IEEE Access.* **2020**, *8*, 3042–3054. [\[CrossRef\]](#)
21. Wang, D.S.; Yin, J.H.; Xu, Y.G.; Yang, X.; Hua, G. Threshold Based D-SCFlip Decoding of Polar Codes. *IEICE Trans. Commun.* **2023**, *E106.B*, 635–644. [\[CrossRef\]](#)
22. Yu, Y.Y.; Pan, Z.W.; Liu, N.; You, X.H. Successive Cancellation List Bit-flip Decoder for Polar Codes. In Proceedings of the 10th International Conference on Wireless Communications and Signal Processing (WCSP), Hangzhou, China, 18–20 October 2018; pp. 1–6.
23. Cheng, F.; Liu, A.; Zhang, Y.; Ren, J. Bit-Flip Algorithm for Successive Cancellation List Decoder of Polar Codes. *IEEE Access.* **2019**, *7*, 58346–58352. [\[CrossRef\]](#)
24. 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.
25. Cao, A.; Zhang, L.; Qiao, J. An LLR-Based Segmented Flipped SCL Decoding Algorithm for Polar Codes. In Proceedings of the 2019 IEEE/CIC International Conference on Communications in China (ICCC), Changchun, China, 11–13 August 2019; pp. 724–729.
26. Peng, Y.; Liu, X.; Bao, J. An Improved Segmented Flipped Successive Cancellation List Decoder for Polar Codes. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 94–99.
27. Rowshan, M.; Viterbo, E. Improved List Decoding of Polar Codes by Shifted-pruning. In Proceedings of the 2019 IEEE Information Theory Workshop (ITW), Gotland, Sweden, 25–28 August 2019; pp. 1–5.
28. Rowshan, M.; Viterbo, E. Shifted Pruning for Path Recovery in List Decoding of Polar Codes. In Proceedings of the IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 27–30 January 2021; pp. 1179–1184.
29. Wu, D.; Li, Y.; Sun, Y. Construction and Block Error Rate Analysis of Polar Codes Over AWGN Channel Based on Gaussian Approximation. *IEEE Commun. Lett.* **2014**, *18*, 1099–1102. [\[CrossRef\]](#)

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.