An improved path splitting strategy on successive cancellation list decoder for polar codes

Yunlong Peng1 | Jingyun Bao | Xingcheng Liu1,2,3

1 School of Electronics and Information Technology (SEIT), Sun Yat-sen University, Guangzhou, China
2 School of Information Science, Xinhua College of Sun Yat-sen University, Guangzhou, China
3 Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai), Zhuhai, China

Abstract

The performance of the cyclic redundancy check aided successive cancellation list (CA-SCL) decoder for polar codes exceeds that of the Turbo codes and the Low Density Parity Check (LDPC) codes adopted in the World Interoperability for Microwave Access (WiMAX) proposal. However, CA-SCL decoder has a high computational complexity, resulting in a long time delay and high memory complexity. In order to alleviate this problem, an improved path splitting strategy on successive cancellation list (IPSS-SCL) decoder was proposed, which can significantly reduce the average list size. The influence of splitting on CA-SCL decoder was analysed at first and a new selective splitting criterion using the Gaussian approximation method was proposed according to the analysis. In addition, a path contraction mechanism based on the location estimation of the correct path was proposed to further reduce the average number of unnecessary candidate paths. Compared with existing path splitting strategies, simulation results show that the proposed IPSS-SCL decoding algorithm can reduce the average computational complexity significantly over the additive white Gaussian noise (AWGN) channel with almost no performance loss.

1 INTRODUCTION

Polar codes [1] is a major area of interest within the field of channel coding recently for it has been proven to be the capacity achieving code for binary-input discrete memoryless channels (BI-DMC). Successive cancellation (SC) decoding scheme [1] is proposed as the original Polar decoding scheme with low complexity and it could approach asymptotically capacity achieving over the BI-DMC when the block length goes infinity.

Unfortunately, the main challenge faced by many researchers is that the performance of SC decoder is dissatisfactory in the practical cases with finite-length blocks. To improve it, a successive cancellation list (SC-List) decoding algorithm [2] approaches the performance of maximum likelihood decoding in the high signal-to-noise ratio (SNR) regime. Furthermore, a cyclic redundancy check (CRC) aided successive cancellation list (CA-SCL) [2] decoding algorithm is comparable with the performance of current state-of-the-art LDPC codes. Considering the computational complexity of CA-SCL algorithm, $O(LN \log N)$, an adaptive CA-SCL decoding algorithm which uses adaptive list size $L$ instead of constant $L$ for decoding [3].

It can achieve the same performance but with significantly lower the number of the list in the high SNR regime. Moreover, a segmented CA-SCL polar decoding scheme [4] was proposed for a better trade-off between performance and complexity, which provides high complexity reduction in the lower SNR regime.

Besides, SC-based decoding algorithms also include depth-first SC stack decoding algorithm [5] and SC-Flip decoding algorithm [6]. SC stack algorithm has the same performance as the SC-List algorithm and can approach that of the maximum likelihood algorithm but the computational complexity of the SC stack decoder is much lower than that of the SC-List decoder in the high SNR regime. Different from SC-List algorithm and SC stack algorithm, SC-Flip algorithm [6] was proposed to solve the problem that SC algorithm has the effect of error propagation. The SC-Flip decoder preserves the low memory requirements of the basic SC decoder and improves the block error rate (BLER) performance by retrying alternative decisions.

SC-List decoder achieves superior performance compared to the SC decoder at the price of increased complexity and CA-SCL decoder outperform current state-of-the-art turbo and LDPC codes. However, the computational complexities of the...
SC-List decoder and CA-SCL decoder are relatively high. Therefore, lots of researchers have proposed improved algorithms based on SC-List decoding. For example, fast SC decoder idea [7] was applied to SC-List algorithm, and the significant reduction in the decoding latency was observed without sacrificing the bit error rate performance of the codes [8]. In addition, a split-reduced SC-List decoder [9] was proposed and it can achieve substantial complexity reduction. At the best of our knowledge, most of the work can reduce the computational complexity of SC-List algorithm with almost no performance loss, but it is not suitable for CA-SCL algorithm.

Obviously, there is no need to split all the decoding paths when performing CA-SCL algorithm. This is because the splitting is unnecessary if the reliability of deciding the unfrozen bit \( u_i \) is extremely high, especially for the bits transmitted through the sufficient polarized sub-channels. Thus, we focus on the splitting operation at unfrozen bits in order to reduce the computational complexity of CA-SCL algorithm. The main contributions can be summarized as follows:

- We analysed the effect of path splits on decoding performance, aiming to find the causes of performance degradation in the selective path splitting process on CA-SCL decoder.
- Then, we proposed an improved path splitting strategy to determine whether the path extension operation should be performed based on the Gaussian approximation (GA). In this way, the speed of the list expansion can be effectively curbed with negligible performance loss.
- A path contraction mechanism was proposed with a dynamic updated threshold. This mechanism can dynamically decide whether the path deletion operation can be performed at the related information bits after the paths competition finished. Thanks to the contractile list size, the decoding complexity is further reduced without significant performance loss.

The rest of this work is organized as follows. The preliminaries about the polar codes and the path splitting strategy on SC-List (PSS-SC) decoder [10] are introduced in Section 2. In Section 3, a new path splitting selecting decoder is proposed, including the splitting strategy and the contraction mechanism. Then, the numerical simulation results are demonstrated in Section 4. Finally, conclusions are drawn in Section 5.

2 | PRELIMINARIES

2.1 | Polar codes and SC decoding

A polar code [1] is characterized by a four-tuple \((N, K, \mathcal{A}, \mathcal{A'})\), where \( N = 2^n \) is the length of the code, \( K \) is the number of information bits, \( \mathcal{A} \subseteq \{1, \ldots, N\} \) is a set indicating the positions of the \( K \) information bits and \( \mathcal{A'} = \{1, \ldots, N\} \setminus \mathcal{A} \). After channels polarization, the indices of the \( K \) best sub-channels are taken into \( \mathcal{A} \). Bits corresponding to positions \( i \in \mathcal{A'} \) are referred to as frozen bits and are set to constant values known at both the encoder and decoder.

For any \( N = 2^n \), the encoded vector \((x_1, \ldots, x_N)\), denoted by \( x_1^N \), is obtained by the formula \( x_i^N = u_i^N F^i \), where source vector \( u_i^N \) denotes row vector \((u_1, \ldots, u_N)\), \( F^i \) denote \( i \)th power of matrix \( F \), and \( F = \left[ \begin{array}{c} 1 \quad 0 \\ 1 \quad 1 \end{array} \right] \). In this work, the encoded codeword \( x_1^N \) is sent to AWGN channel using binary phase shift keying (BPSK) modulation.

At the receiver, SC decoding algorithm [1] is taken to obtain the estimate of source vector, denoted by \( \hat{x}_1^N \). Based on the previous results \( \hat{x}_1^N \) as well as the received values \( y_1^N \), the SC decoding algorithm computes the probability of \( u_i \) denoted by \( \hat{W}_N(y_i^N, \hat{x}_1^N | u_i) \). Since \( u_i \in \mathcal{A} \) are known to the receiver, the real task of SC decoding is to estimate \( u_i \), \( i \in \mathcal{A} \). The log-likelihood ratio (LLR) for bit \( u_i \) is defined as

\[
L_i^N(y_1^N, \hat{x}_1^N) \triangleq \ln \frac{\hat{W}_N(y_i^N, \hat{x}_1^N | u_i = 0)}{\hat{W}_N(y_i^N, \hat{x}_1^N | u_i = 1)}.
\]

Hard decisions are taken according to

\[
\hat{u}_i = \begin{cases} 0, & i \in \mathcal{A}' \\ 0, & L_i^N(y_1^N, \hat{x}_1^N) \geq 0, \ i \in \mathcal{A} \\ 1, & L_i^N(y_1^N, \hat{x}_1^N) < 0, \ i \in \mathcal{A}. \end{cases}
\]

2.2 | SC-List-based decoding algorithms of polar codes

SC decoder makes hard decision in each bit position and does nothing to remedy when the decision is wrong. To overcome this problem, SC-List decoding algorithm [2] preserves both 0 and 1 probabilities at each position \( i \in \mathcal{A} \) and splits the decoding path into two parallel decoding threads, continuing in either possible direction. In this work, SC-List decoder based on probability measurement is called probability-based SC-List decoder, corresponding to LLR-based SC-List decoder [11]. Each of the threads is estimated with probability

\[
\Pr (\hat{x}_i^N | u_i^N) = \hat{W}_N(y_i^N, \hat{x}_1^N | u_i^N),
\]

where \( \Pr (\hat{x}_i^N | u_i^N) \) denote the probability of whether the \( i \)th path at \( i \)th position is the correct path, \( \hat{W}_N(y_i^N, \hat{x}_1^N | u_i) \) is computed by SC algorithm. In order to constrain the exponentially growing complexity, the authors in [2] considered a pruning procedure to discard all but the \( L \) most likely paths. At the end of the SC-List decoding process, the final output is selected with the following formula:

\[
\hat{x}_1^N = \arg \max_{\hat{x}_i^N | u_i^N} \Pr (\hat{x}_i^N | u_i^N),
\]

where \( \mathcal{L} = \{1, \ldots, L\} \) is a set indicating the alive paths. The formula (4) indicates that SC-List algorithm picks the estimated codeword \( \hat{x}_i \) with the maximum probability.
Actually, considering the fact that the path with the maximum probability is not necessarily correct, a cyclic redundancy check aided SC-List (CA-SCL) decoder [2, 12] was shown to improve the performance of SC-List decoder. In CA-SCL decoder, the path that passes the CRC check is selected as the final output. If there are more than one path pass the CRC check, then the most likely one is picked. The simulation results showed that CA-SCL algorithm can obtain 1 dB gain at $10^{-5}$ comparing to SC-List decoder for polar code ($1024, 512$) with list size $L = 16$.

2.3 An exiting path splitting selecting strategy for SC-List-based algorithms

In order to reduce the computational complexity of SC-List algorithm, a simple idea is to modify the decoding tree. Instead of splitting the paths at each sub-channel, an alternative method is to split the paths only at unreliable sub-channels.

Critical set [13] includes the erroneous bits caused by channel noise with high probability (more than 99% for SC decoding). These erroneous bits induced by channel noise are called noise erroneous bits for simplicity.

We take a polar code $\mathcal{P}(16, 8)$ for example, a full binary tree with 16 leaf nodes is constructed in Figure 1. The frozen bits and the information bits are denoted by white circles and black circles, respectively, at level 4. For each non-leaf node in the tree, if its two descendants are of the same colour, then it is marked with that colour. Otherwise, it is marked with grey one. The first leaf node of the rate-1 node is taken into the critical set $\mathcal{S}$. As shown in Figure 1, the critical set $\mathcal{S} = \{u_8, u_{10}, u_{11}, u_{13}\}$.

Critical set is first proposed to be a reference when performing SC-Flip algorithm for it contains more than 99% indices of channel noise erroneous bits. If the first SC-Flip decoding attempt fail to pass CRC check, the bits in critical set are flipped to correct the channel noise erroneous bits. Similarly, SC-List-based decoders can perform path splitting operation only at the channel noise erroneous bits, for other information bits are almost infallible according to the property of polar code. Thus, the critical set was adopted to the CA-SCL algorithm [10]. Simulation results showed that the computational complexity of CA-SCL algorithm can be reduced if paths splitting is only performed with critical set-based strategy. As shown in Figure 2, if critical set is applied to SC-List algorithm, the computational complexity of SC-List decoder is slightly reduced with no performance loss. However, simulation results in Section IV demonstrated that the method may be unsuitable for CA-SCL algorithm. It is because critical set cannot achieve the accuracy of the channel noise erroneous bits detection compared to the SC algorithm especially in a large list size, leading to the performance degradation.

3 AN IMPROVED PATH SPLITTING SELECTING STRATEGY AIDED SCL ALGORITHM

To reduce the computational complexity with negligible performance loss of CA-SCL decoder, we first study the relevant properties of CA-SCL decoding algorithm and a new path splitting selecting strategy is proposed in this section. Then, we propose a path contraction mechanism to reduce the average list size further.

3.1 Analysis of path splitting selecting strategy

The CA-SCL decoding algorithm is different from the SC-List algorithm for it picks the final path with following formula:

$$\hat{u}_1^N = \arg \max_{\hat{u}_1^N} \Pr (\hat{u}_1^N | \hat{u}_m^N) = u_1^N),$$ (5)
where \( \Pr(\hat{x}^N_1[l]|\omega) = \mu^N_1 \) denote the probability of the path that passes CRC check. Equation (5) implies that CA-SCL decoding algorithm fails for two reasons. One is the correct path does not exist in the list, the other is the wrong path passes the CRC check and the probability of it is larger than the correct one, which causes a false alarm. To further evaluate the CA-SCL algorithm and identify which situation dominates the error decoding, we study the frequency that the correct path exiting in the list and the ratio of unsuccessful decoding caused by false alarm through Monte Carlo simulations with a 16 bits CRC codes with \( g(D) = D^{16} + D^{15} + D^2 + 1 \), which is shown in Table 1.

According to above numerical evaluation, it can be observed that the performance of CA-SCL algorithm is mainly determined by the probability that the correct path kept in the list. Therefore, defines the probability that the correct path kept in the list in the whole decoding process and in a certain information bit \( u_i \) as \( \Pr(\hat{x}^N_1[\mathcal{L}] = u^N_1) \) and \( \Pr(\hat{x}_1[\mathcal{L}] = u_i) \), respectively, then we have

\[
\Pr(\hat{x}^N_1[\mathcal{L}] = u^N_1) = \prod_{i=1}^{N} \Pr(\hat{x}_1[\mathcal{L}] = u_i) = \prod_{i=1}^{N} \sum_{i'=1}^{\mathcal{L}} \Pr(\hat{x}_1[l] = u_i) \\
\geq \prod_{i=1}^{\mathcal{L}} \sum_{i'=1}^{\mathcal{L}} \Pr(\hat{x}_1[l] = u_i) \times \prod_{i=1}^{\mathcal{L}} \sum_{i'-1}^{\mathcal{L}} W_{\mathcal{L}}(\hat{x}_1^N, \hat{x}_1^{-1}[l]|u_i),
\]

where \( \Omega \) is the splitting set which contains all the bits performed splitting operation and \( \overline{\Omega} \) is the complementary set of \( \Omega, \Omega \cup \overline{\Omega} = \mathcal{A} \). According to inequality (6), the performance of CA-SCL algorithm cannot be improved because of the lack of path competition when path splitting selecting strategy is used. Instead, it must deteriorate if the set \( \Omega \) is constructed unreasonably. In the PSS-SCL decoder [10], \( \Omega \) is replaced by critical set. To further reduce the computational complexity, a threshold \( \theta \) is considered. If the maximum LLR value of \( |\mathcal{L}| \) paths less than \( \theta \), then the decoder will split the decoding paths.

However, critical set is not suitable to be a path splitting selecting strategy in CA-SCL decoding for it does not include certain unreliable information bit \( u_i \), make

\[
\sum_{l=1}^{\mathcal{L}} W_{\mathcal{L}}(\hat{x}_1^N, \hat{x}_1^{-1}[l]|u_i) < \sum_{l=1}^{\mathcal{L}} \Pr(\hat{x}_1[l] = u_i),
\]

which causes performance loss significantly.

### 3.2 An improved path splitting selecting strategy

According to the above analysis, when \( i \) th sub-channel condition is ideal enough, we have

\[
\sum_{l=1}^{\mathcal{L}} \Pr(\hat{x}_1[l] = u_i) = \sum_{l=1}^{\mathcal{L}} W_{\mathcal{L}}(\hat{x}_1^N, \hat{x}_1^{-1}[l]|u_i),
\]

which means operation of the paths splits can be replaced by SC algorithm in each thread at \( i \)th ideal sub-channel. To get the optimal set \( \Omega \), we should try \( \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{K} = 2^K \) probabilities. It is impossible to realize when \( K \) is large enough. Therefore, we construct \( \Omega \) approximately with the GA method [14]. However, considering that \( \Phi^{-1}(\alpha) \) in [14] is hard to be computed, so \( \Phi(\alpha) \) can be approximated by the following piecewise function [15]:

\[
\Phi^{-1}(\alpha) = \begin{cases} 
\frac{e^{-0.4527x+0.86}+0.0218}{\sqrt{\pi x^{-2}}} \left( 1 - \frac{10}{7x} \right), & 0 < x < 10 \\
\sqrt{\frac{x}{\pi}} - \frac{2}{3} \left( 1 - \frac{10}{7x} \right), & x \geq 10.
\end{cases}
\]

With the help of the GA, we can obtain the expectation of the error probability for every information bits under the designed SNR. Then we take \( \omega \) smallest of them into \( \Omega, \Omega = \{i_1, \ldots, i_\omega\} \subset \mathcal{A} \) and \( \omega \) is the size of splits set. \( \omega \) can be set according to communication requirements, it is fixed with the size of critical set for comparison. To further reduce the splits, a splitting threshold \( \theta_{\text{split}} \) is considered in our work. For \( i \in \Omega \), we splits the path at \( i \)th position if the average LLR value of \( |\mathcal{L}| \) paths satisfies

\[
\frac{1}{|\mathcal{L}|} \sum_{l=1}^{\mathcal{L}} \ln \frac{\Pr(\hat{x}_1[l] = 0)}{\Pr(\hat{x}_1[l] = 1)} < \theta_{\text{split}}.
\]

Based on the observation above, \( \theta_{\text{split}} \) can directly influence the performance of the path splitting criterion. It should be set properly, for an undersize value will lead to dramatically performance deterioration and an oversize one will result in unnecessary splitting. In our work, \( \theta_{\text{split}} \) is defined as

\[
\theta_{\text{split}} = E[I_{\mathcal{L}}^{(i)}|\min] + \theta, \; i \in \mathcal{A},
\]

where \( E[I_{\mathcal{L}}^{(i)}|\min] \) is smallest mean LLR value of information bit computed by the GA and \( \theta \) is given by experience as margin. The bit whose average LLR value is lower than the minimum theoretical value has a high probability to be erroneously decoded by SC algorithm, which means that path splitting operation should be provided to improve the error correction ability. In this way, we slow down the rapid expansion of the list sizes, reducing the operation of early paths replication, sorting and LLR calculations, which can significantly rise down the complexity of CA-SCL decoder with almost no performance loss.

| Table 1 Evaluation of CA-SCL algorithm under Gaussian approximate |
| --- |
| \( N = 512, K = 256, L = 16, \) number of simulation blocks \( T = 10^7 \) |
| \( E_b/N_0(\text{dB}) \) | 1 | 1.5 | 2 | 2.5 |
| CRC passing | 8,928,231 | 9,869,746 | 10,040,541 | 10,047,665 |
| Correct path existing | 8,911,653 | 9,851,793 | 9,989,436 | 9,999,593 |
| False alarm | 835 | 241 | 80 | 5 |
3.3 Path contraction mechanism

Suppressing the rapid growth of the previous paths is not enough to greatly reduce the computational complexity of the CA-SCL algorithm. Since the number of the paths double when performing the paths split, the size of the set $|L|$ quickly increases to the designed list size $L$. Moreover, most of the information bits with the high channel index are transmitted over the sufficient polarized sub-channels according to the construction of polar code, which means that they have a high probability to be decoded correctly by SC algorithm. Therefore, an extra paths deletion operation should be considered for CA-SCL decoder.

To realize the path contraction mechanism, we first analyze which path can be eliminated in CA-SCL decoding. Obviously, if path $l$ satisfies

$$\Pr(\hat{y}_i[l] = u_i) < \Pr(\hat{y}_b[l] = u_i),$$ (12)

where $\Pr(\hat{y}_i[l] = u_i)$ represent the correct probability of the correct path in CA-SCL decoder, the path $l$ should be deleted to reduce the computational complexity. However, the correct path $l^c$ is unclear when processing CA-SCL algorithm. Therefore, instead of the correct probability of the correct path, we adopt the probability that the correct path $l^c$ exists in the list to approximate it in the path contraction operation. In [16], Path Metric Range (PMR) was proposed to measure the probability of that the correct path exists in the list for the LLR-based SC-List algorithm [11]. In this work, the concept of PMR is applied in the probability-based SC-List decoder.

**Definition 1.** To measure the probability of the correct path existing in the list, we defined the like logarithm probability ratio ($LPR$) of path $l$ at bit $u_i$ for probability-based SC-List decoder as follow

$$LPR_l^{(i)} = \ln \frac{\Pr(\hat{y}_i[l] = u_i)}{\Pr(\hat{y}_b[l] = u_i)},$$ (13)

where $l^c$ is the first ranking candidate in the list and it has the largest probability $\Pr(\hat{y}_i[l^c] = u_i)$ to be the correct path. Ranking represents the location of the corresponding path in the list. According to structure of CA-SCL decoder, a lower ranking path correspond to a higher $LPR$ value.

An intuitive understanding is that when $LPR_{l^c}^{(i)}$ for all $l$ are close to 0, all candidate paths have the similar probability value, so the correct path cannot be identified clearly and it is easy to be deleted after the paths competition. In contrast, when $LPR_{l^c}^{(i)}$ values for ranking $2 \leq l \leq L$ go infinity, the path with maximum probability is likely to be the correct one.

**Definition 2.** During the CA-SCL decoding process, if the absolute values of LLR are large enough for all candidate paths at an information bit $u_i$, the value of $LPR_{l^c}^{(i)}$ will tend to decrease compared to the bit $u_{i-1}$. On the contrary, if $u_i$ is a frozen bit and the absolute values of LLR are close to zero, a rising trend will appear on the $LPR_{l^c}^{(i)}$.

To verify the conclusion above, suppose that the probabilities of alive paths at $u_{i-1}$ satisfy

$$P_1^{(i-1)} \leq \cdots \leq P_b^{(i-1)} \leq \cdots \leq P_L^{(i-1)},$$ (14)

where $P_l^{(i-1)}$ is a simplification of $\Pr(\hat{y}_l^{(i-1)} = u_{i-1})$. We analyse the $LPR$ characteristics of the last ranking candidate, referring to $LPR_{l^c}^{(i)}$, by Monte-Carlo simulation and the result is shown in Figure 3. It is equally applicable to the other ranking $l$ in the list. The $LPR_{l^c}^{(i)}$ of frozen bits and information bits are marked in red line and blue line, respectively.

(1) Assuming that $u_i$ is an information bit and the absolute LLR values for all paths are large enough. After paths splitting operation, the correct probabilities of two extensions corresponding to path $l$ satisfy the relationship as follows:

$$P_l^{(i)} + P_l^{(i)} e^{\lambda_{l^c}^{(i)}} = P_l^{(i-1)}, 1 \leq l \leq L,$$ (15)

where $\lambda_{l^c}^{(i)}$ is LLR value of $l$th path at $u_i$. Because of the great LLR values, it can be easily observed that $P_l^{(i)} e^{\lambda_{l^c}^{(i)}}$ is largely outweigh $P_l^{(i)}$, which means that only the extension $P_l^{(i)} e^{\lambda_{l^c}^{(i)}}$ can be kept in the list. Without loss of generality, suppose that $P_b^{(i)} e^{\lambda_{l^c}^{(i)}}$ and $P_l^{(i)} e^{\lambda_{b}^{(i)}}$ are the maximum and the minimum correct probability in the list after the paths competition, respectively. Hence, the $LPR_{l^c}^{(i)}$ can be calculated as

$$LPR_{l^c}^{(i)} = \ln \frac{P_b^{(i)} e^{\lambda_{b}^{(i)}}}{P_b^{(i)} e^{\lambda_{b}^{(i)}}} = \ln \left( \frac{P_b^{(i-1)} e^{\lambda_{b}^{(i-1)}}}{P_b^{(i-1)} e^{\lambda_{b}^{(i-1)}}} + 1 + e^{\lambda_{b}^{(i-1)}} \right)$$ (16)

$$= \ln \frac{P_l^{(i-1)}}{P_l^{(i-1)}} + \ln \left( \frac{1 + e^{\lambda_{l^c}^{(i)}}}{1 + e^{\lambda_{b}^{(i)}}} \right).$$

Since $\lambda_{l^c}^{(i)}, \lambda_{b}^{(i)}$ are large enough, the second term of the equality approaches to zero, we can obtain

$$LPR_{l^c}^{(i)} = \ln \frac{P_l^{(i-1)}}{P_l^{(i-1)}} \leq \ln \frac{P_b^{(i-1)}}{P_b^{(i-1)}} = LPR_{l^c}^{(i-1)}.$$ (17)

The equation holds if and only if $\lambda_{l^c}^{(i)}$ goes infinity, which make $P_l^{(i-1)}$ close to $P_b^{(i-1)}$ and $P_b^{(i-1)}$ approach to $P_l^{(i-1)}$. Thus, while $u_i$ is an information bit and $\lambda_{l^c}^{(i)}$ is large enough, we can notice that $LPR_{l^c}^{(i)}$ is smaller than $LPR_{l^c}^{(i-1)}$.

(2) If $u_i$ is a frozen bit and the absolute LLR values are close to zero for all survival paths, the characteristics of the sequence will play an important role. Since frozen bits are forced to zero,
We can obtain
\[
\Pr(\hat{u}_i[l] = 0) \geq \Pr(\hat{u}_i^{-1}[l] = 0)
\]
we can obtain
\[
\frac{\Pr(\hat{u}_i[l] = 0)}{\Pr(\hat{u}_i^{-1}[l] = 0)} \leq \frac{\Pr(\hat{u}_i[l] = 0)}{\Pr(\hat{u}_i^{-1}[l] = 0)}.
\]
Equation (19) refers that the probability of a higher ranking candidate will decrease slowly than the lower one at frozen bits. Hence,
\[
LPR^{(i-1)}_1 = \ln \frac{p_i^{(i-1)}}{p_i^{-1}} \leq \ln \frac{\Pr(\hat{u}_i[l] = 0)}{\Pr(\hat{u}_i[l] = 0)} = LPR^{(i)}_1.
\]
Thus, while \(u_i\) is a frozen bit and \(A^{(i)}\) is close to zero, we can notice that \(LPR^{(i)}_1\) is larger than \(LPR^{(i-1)}_1\).

The above analysis illustrates that if the information bits \(u_i, i \in A\) are reliable enough and the frozen bits \(u_i, i \in A^c\) are unreliable, the \(LPR\) values will act like description in Definition 2. Usually, for the bit rate \(R \leq 0.5\), information bits are reliable enough and for the bit rate \(R \geq 0.5\), frozen bits are unreliable enough contrary. As a consequence, when bit rate \(R = 0.5\), such property is easy to be observed, just as shown in Figure 3.

In order to eliminate the unnecessary decoding paths, we investigated the variation trend of the correct path location in CA-SCL decoding. Numerical result of the correct path ranking for polar code \(P(512, 256)\) with list size \(L = 16\) is shown in Figure 4 as an example. Frozen bits and information bits are marked in red line and blue line, respectively.
According to Figure 4, we can clearly observe that the survival probability of the correct path will decline when it meets the information bits and will increase when it meets the frozen bits, which shows the same property as the variation trend of $LPR^{(i)}$ introduced in Definition 2. However, some special bits such as {338, 390, 391, 417} have large LLR values [6.9462, 7.0041, 8.6048, 6.3288] relative to other frozen bits because of the insufficient polarization, which may act like information bits, leading to a decline in the ranking of the correct path.

Based on above analysis, we conjecture that the $LPR^{(i)}$ value for the probability-based SC-List decoder can represent the ranking of the correct path approximately. A larger $LPR^{(i)}$ represents a higher ranking of the correct path. Since $LPR^{(i)}$ for $1 \leq i < L$ has the same property, this conjecture is also suitable for $LPR^{(i)}$, $2 \leq i < L$. Therefore, $LPR^{(i)}$ for $1 \leq i < L$ can be employed to determine the ranking of the correct path $R_i$, referring to

$$R_i \propto LPR^{(i)}.$$ \hspace{1cm} (21)

The formula (21) shows the relative position of the correct path in $\{l, l + 1, ..., L\}$ in order of increasing probability. From another perspective, we can judge that whether the $l$th path is the correct one with metric $LPR^{(i)}$, $1 \leq i \leq L$. When $LPR^{(i)}$, $1 \leq i \leq L$ is large, we can infer that the relative position of the correct path is high so that the path $l$ may not be the correct one so we can delete it in the decoding process. Therefore, we use a threshold to judge whether the path $l$ should be eliminated and it is defined in Definition 3.

**Definition 3.** While performing SC-List algorithm, the minimum $LPR^{(i)}$ value is defined as the decisive $LPR$ value, denoted by $LPR_{\text{min}}$, for it approximately characterizes the survival probability of the correct path in whole decoding process. If $LPR^{(i)} > LPR_{\text{min}}$, for any $1 \leq l < L$, $1 \leq i \leq N$, the path $l$ at $u_i$ can be deleted.

In order to reduce the computational complexity of CA-SCL algorithm with no performance loss, the $l$th path with

$$LPR^{(i)} > \left( LPR^{(i)} \right)_{\text{max}}$$ \hspace{1cm} (22)

should be eliminated. However, we cannot calculate the $(LPR^{(i)})_{\text{max}}$ accurately. Hence, we replace it by $LPR_{1\text{min}}$, it can dynamically update during the decoding process.

As shown in Figure 5, we take an example of our proposed pruning process. Since the $LPR_{\text{min}}$ characterizes the survival probability of the correct path in the whole decoding process, if the distance between path $l$ and path $L$ at $u_i$ is larger than decisive $LPR$, we have reason to believe that the $l$th path cannot be the correct one. Therefore, for $l$th path at $u_i$, if

$$LPR^{(i)} > LPR_{\text{min}},$$ \hspace{1cm} (23)

the $l$th path should be deleted. Otherwise, the $l$th should be hold.

Actually, $LPR_{\text{min}}$ will changes with different SNRs, block lengths and list sizes. To determine the $LPR_{\text{min}}$ for CA-SCL decoding dynamically, a variable $\rho$ is adopted to record the $LPR_{\text{min}}$ in the whole decoding process for each individual block. It is first initialized to $\theta_{\text{split}}$ we mentioned in Subsection 3.2. After processing the path eliminations at $u_i$, the decoder will update the threshold, $\rho$, with rule as follow

$$\rho = \min \left( \rho, \max_{l \in L} \left( LPR^{(i)} \right) \right).$$ \hspace{1cm} (24)

It is because while $\rho$ is larger than the maximum of $LPR^{(i)}$ at bit $u_i$, the worst case of the correct path ranking after path competition is better than the ranking corresponding to the deleting threshold before $u_i$, which means that the contraction will
be insufficient in the following decoding process without narrowing $\rho$. Based on above analysis, we can dynamically updating the deleting threshold $\rho$ in different construction of polar codes and transmission conditions, making our contraction mechanism have a better robustness.

Moreover, to further reduce the average list size, it has been proven in [9] that when performing SC algorithm after certain bit $\varphi_{sc}$, it keeps the same performance with SC-List algorithm because of the channels polarization. We adopt this improvement to our decoder by early checking CRC codes and simulation results show that this property is also suitable for CA-SCL algorithm. To determine the position $\varphi_{sc}$, representing the beginning bit for SC-only decoding, we consider the last bit in critical set [13], $u_\tau$. Because the bits in the critical set has a relatively high error probability, after $u_\tau$, the sub-channels will tend to be noiseless channels and SC decoding can achieve the performance of CA-SCL decoding. Meanwhile, considering that the $LPR$ values decline at information bits, the minimum $LPR$ value may occur in those noiseless sub-channels. Therefore, we leave a bit position to record the minimum $LPR$. The bit position $\varphi_{sc}+1$ is used to reduce the computational complexity. After $\varphi_{sc}$, only the path which can passes the CRC checking can continue to decode using SC algorithm and the computation complexity will decrease further according to our proposed path contraction mechanism.

In summary, the selective PSS and the path contraction mechanism we proposed are materialized in an improved path splitting strategy on SC-List (IPSS-SCL) decoding algorithm. The details are described in Algorithm 1.

### Algorithm 1: IPSS-SCL Decoding Algorithm

**Input:** $E_b N_0$, $N$, $L$, $u^N_1$, $\theta_f$

**Output:** $u^N$  

1. Initialize $\Omega, \varphi_{sc}, \rho$ with $E_b N_0$;
2. for $\varphi = 0, 1, 2, ..., N - 1$ do
3.  Calculate the probabilities of all alive paths $l, l \in L$;
4.  if $u_{\varphi} \in \mathbb{A}^*$ then
5.      $u_{\varphi}[l] = 0, l \in L$
6.  else
7.      if $\varphi \in \Omega$ then
8.          Calculate the mean LLR of the list;
9.          if $\mu < E[L_i^{(\varphi)}]_{\text{min}} + \theta_f$ then
10.             Split the paths
11.          else
12.             Make hard decision at $u_{\varphi}[l]$ for all $l$;
13.          end
14.      end
15.  end
16. for $l = 0, 1, 2, ..., L - 1$ do
17.  Calculate the $LPR_l^{(\varphi)}$ of all alive paths $l, l \in L$;
18.  if $LPR_l^{(\varphi)} > \rho$ then
19.      Kill the path $l$
20.  else
21.      Update the threshold $\rho$
22.  end
23. if $\varphi = \varphi_{sc} + 1$ then
24.      kill the path that can’t pass CRC
25. end
26. end

---

In this section, we evaluate the BLER performance and computational complexity of the IPSS-SCL decoder. Since polar codes are selected as the short code standard in the 5G control channel, we focus on the polar codes with shorter code lengths. Therefore, we take polar codes (512, 256) and (256, 128) transmissions using BPSK modulation over AWGN channel. List sizes will be selected as $L = 4$ and $L = 16$ unless otherwise noted. Polar codes are constructed using the GA [14]. Meanwhile, the information bits are concatenated with a 16-bits CRC with generator polynomial $g(D) = D^{16} + D^5 + D^2 + 1$ and coding rate for polar codes is fixed to 0.5 while the effective information rate is $(K - 16)/N$. The threshold $\theta_f = 3$ is given in this work. The number of simulation blocks is $10^6$.

### 4.1 BLER performance comparisons

In this section, BLER performance of several decoding algorithms are compared under the same effective code rate in the simulations, including the proposed IPSS-SCL decoder, the conventional CA-SCL decoder and the exiting path splitting selecting (PSS-SCL) decoder [10].

In Figure 6, we compare the performance of polar codes with different code lengths in list size $L = 16$. We can observe that the proposed decoder performs no more than 0.05 dB performance loss compared with CA-SCL decoder in the BLER ranges from $10^{-3}$ to $10^{-4}$. Moreover, as the code length grows, the performance loss will become more negligible. This is because the main reason leading to the performance loss in our proposed decoder is the paths deletion operation which using the estimated location of the correct path. As we discuss in Section 3, our approximation based on $LPR$ values need to satisfy the condition of a large enough LLR value.
for every candidate paths at information bits, which means that a smaller LLR value will influence the accuracy of the location significantly. Thus, while the code length goes larger, a larger LLR value will be calculated in CA-SCL decoder due to the higher degree of channel polarization, corresponding to a more accurate approximation for correct path. In a similar way, a higher SNR means a higher LLR value, resulting in a less performance degradation. In addition, compared with PSS-SCL decoder [10], our proposed decoder can provide a better error correction performance as the transmission conditions become better. This is because the location estimation of the correct path is more reliable although a large number of paths are deleted.

Figures 6 and 7 indicate that the performance loss of our proposed IPSS-SCL decoder will become larger as the list size increasing compared to conventional CA-SCL decoder. We can clearly notice that our decoder has almost no performance loss in a low list size. It is because if the list size is large enough (such as L = 16), our proposed decoder cannot fill in the list due to the path contraction mechanism. A numerical result will be shown in Section 4.2 to verify this conclusion. Therefore, the list size has a less effect on our algorithm especially in the high ranges, which makes the performance curve larger since the list size can enhance the error correction ability of CA-SCL decoder significantly. At the same time, our proposed algorithm can provide a competitive error correction performance in middle to high SNR ranges with different list size compared to PSS-SCL decoder [10].

4.2 Average computational complexity

The main purpose of our proposed decoder is to reduce the computational complexity without a significant performance loss. In order to analyze the computational complexity theoretically, we use running time in simulation to represent the complexity and the running time of the conventional CA-SCL decoder is \(O(LN \log N)\) [2].

**Definition 4.** The running time of the IPSS-SCL decoder is \(O(\overline{E}_L \cdot N \log N)\), where \(\overline{E}_L\) represents the weighted average list size of the decoder and the weight is determined by polar codes construction.

To verify this conclusion, the running time of additional operations based on the probability SC-List decoder [2] is shown in Table 2. The value \(\overline{L}\) and \(L^{[0]}\) represents the average list size with weight 1 and the actual list size of bit \(u_i\), respectively. In order to obtain the splitting threshold \(\overline{\theta}_{\text{split}}\), we should calculate the \(E[L^{[0]}]_{\min}\) by the GA at first. The complexity of calculating the expectation via the GA method is \(O(N)\) [17]. Then, to determine whether the path need to be extended at an information bit \(u_i\), the comparison between the LLR values and \(\overline{\theta}_{\text{split}}\) costs \(O(L^{[0]})\) complexity per bit. After splitting operation, the value of \(LPR\) will be calculated for path contraction mechanism using the LLR values, which leads to \(L^{[0]} - 1\) time steps per bit. Then, paths deletion will be performed, which result in another comparison complexity \(O(L^{[0]})\) between \(LPR\) of each candidates and threshold \(\rho\). Finally, threshold \(\rho\) has to be updated according to Section 3.5 after the path competition. To obtain the maximum \(LPR^{[0]}\), \(O(L^{[0]})\) need to be introduced for searching.

To update the data recursively, it has been proven in [2] that the total running time of an entire block updating process with a constant list size satisfies the relationship as follow:

\[
f(\lambda) = 2^\lambda \cdot O(L \cdot 2^{m-\lambda}) + f(\lambda - 1), \lambda > 0,
\]  

where \(m = \log N\) and \(\lambda\) is initialized to \(m\). However, while the list size becomes alterable during the decoding process, Equation (25) should be changed into

\[
f(\lambda) = \sum_{i=1}^{N} O(L^{[0]} \cdot 2^{m-\lambda}) + f(\lambda - 1), \lambda > 0
\]  

![FIGURE 7 BLER performance of several algorithms for polar codes (512, 256) and (256, 128) with list size \(L = 4\)](image)

| Operation | Running time |
|-----------|--------------|
| Initialization calculating \(\overline{\theta}_{\text{split}}\) | \(O(N)\) |
| Updating splitting or not | \(\sum_{i=1}^{K} O(L^{[0]}(i)) = O(KT)\) |
| calculating \(LPR\) | \(\sum_{i=1}^{N} O(L^{[0]}(i)) = O(NT)\) |
| deleting or not | \(\sum_{i=1}^{N} O(L^{[0]}(i)) = O(NT)\) |
| updating data | \(O(\overline{E}_L N \log N)\) |
| updating threshold \(\rho\) | \(\sum_{i=1}^{N} O(L^{[0]}(i)) = O(NT)\) |
| Overall | \(O(\overline{E}_L N \log N)\) |

**TABLE 2 Running time of additional operations in IPSS-SCL Decoder**
FIGURE 8 The change of average list size in several algorithms for polar code (512, 256) with different $E_b/N_0$ and list size $L = 16$

due to the updating mechanism of CA-SCL decoder. Thus, expanding the recursion (26), we can obtain

$$f(m) = \sum_{i=1}^{N} O(n(i)L(i)) = O\left(\sum_{i=1}^{N} n(i)L(i) \cdot N \log N\right) = O\left(E_bL \log N\right),$$

(27)

where the weight $n(i)$ of bit $u_j$ can be calculated by

$$n(i) = 1 + \sum_{j=1}^{S} 2^j,$$

(28)

$S$ is the terminal condition that $j \leq m - 1$ as well as $\left[\frac{j - 1}{2^{m-1}}\right] \mod 2 = 0$.

Thus, the overall computational complexity of the IPSS-SCL decoder can be defined as $O(E_bL \log N)$ according to the analysis above. We notice that the average list size $E_bL$ can directly reflect the complexity of our algorithm, so we investigate the change of the list size in polar code (512, 256) and the result is shown in Figure 8. The PSS-SCL algorithm [10] splits the paths into two threads when the maximum LLR value of the paths is lower than the constant threshold. Therefore, the early growth of the number of alive paths will slow down. But for the IPSS-SCL decoder, in addition to slowing down the path growth rate, a reasonable pruning operation is also considered. Therefore, the list size will be limited in a low range for the entire block so that the complexity can be reduced significantly. However, we can observe in Figure 8 that the largest list size of a $L = 16$ decoder is no more than 6, which means that the performance of our decoder may not increase if the list size changes from $L = 8$ to $L = 16$. This results in the expansion of the performance gap between our proposed decoder and conventional CA-SCL with a large list size we mentioned in Section 4.1.

Figure 9 compares the relationship between the weighted average list size of different algorithms and $E_b/N_0$ in different polar codes. We take the frozen bits into account so the actual list size of the conventional CA-SCL decoding is smaller than the designed value due to the unfilled list at the beginning frozen bits. Note that our proposed algorithm has a desirable decline in average list size comparing to the CA-SCL decoder and the PSS-SCL decoder [10], for more than 75% decline in complexity comparing to CA-SCL algorithm with list size $L = 16$. Moreover, while the list size goes larger, the complexity of the IPSS-SCL decoder growing much slower than CA-SCL decoder, which means that the reducing of complexity will be more significantly under a higher list size. It is because most of the additional paths are redundant for decoding in SC-List-based decoder especially under a high list size, the IPSS-SCL algorithm can reduce these invalid decoding paths by eliminating the candidates which have a high probability to be error-decoded.

Figures 10 and 11 show the numerical results of the normalized running time with different list sizes. Note that compared to CA-SCL decoder and the PSS-SCL decoder, our proposed decoder can significantly reduce the complexity for about 25% in a low list size $L = 4$. Moreover, as the list size growing, our decoder can reduce more decoding complexity, for over 70% decline compared to CA-SCL decoder with list size $L = 16$, due to the variation property of weighted average list size we discuss above.

In summary, our proposed decoder can achieve a desirable complexity reduction compared to exiting path splitting CA-SCL decoder especially in high list sizes without a significant performance loss. This effect will be more obvious if the IPSS is applied to the decoder with a shorter code length or a larger code rate. It is because if the number of the code length
FIGURE 10  The normalized running time of different polar codes in several algorithms with $L = 4$

FIGURE 11  The normalized running time of different polar codes in several algorithms with $L = 16$

decreases or code rate becomes larger, the threshold $\rho$ will be lower and the computational complexity will be further reduced while ensuring performance. It is also useful for polar codes with a larger code length due to the advanced CRC check.

5  |  CONCLUSION AND FUTURE WORK

The properties of the CA-SCL algorithm were analysed in the work. To alleviate the complexity problem, we proposed the IPSS-SCL algorithm for polar codes. Based on the path splitting selecting strategy, the proposed decoder can prevent the list size from growing too fast. Moreover, the path contraction mechanism can delete the path which seems to be erroneously decoded in order to reduce the list sizes further. Simulation results revealed that the proposed algorithm performed better than the existing paths splitting algorithm. Based on the critical set not only in reducing complexity, but also in the error correction performance at the area of medium to high SNR. The reason is that the proposed decoding algorithm can keep the correct path surviving longer during iterative decoding. At the same time, the polarization characteristics of the polar codes were taken into consideration. With aid of the characteristics, the splitting path can be shrunk to single one and almost without causing the error correction performance loss.

The future work is to apply this path splitting strategy to the simplified SC algorithm. Simplified SC algorithm can reduce the complexity as well as decoding latency of the original SC algorithm. Most literatures have proposed SC-List decoding based on simplified SC algorithm, which reduce SC-List decoding latency. To further reduce the complexity of SC-List algorithm, the proposed path splitting strategy may be used while ensuring the performance.

ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61873290, 61972431, 61572534, and in part by the Science and Technology Program of Guangzhou, China under Grant No. 202002030470.

ORCID

Xingcheng Liu https://orcid.org/0000-0003-1836-2205

REFERENCES

1. Arikan, E.: Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. IEEE Trans. Inf. Theory 55(7), 3051–3073 (2009)
2. Tal, I., Vardy, A.: List decoding of polar codes. IEEE Trans. Inf. Theory 61(5), 2213–2226 (2015)
3. Bin, L., Hui, S., David, T.: An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check. IEEE Commun. Lett. 16(12), 2044–2047 (2012)
4. Zhou, H., et al.: Segmented CRC-aided SC list polar decoding. In: IEEE 83rd Vehicular Technology Conference (VTC Spring), pp. 1–5 (2016)
5. Niu, K., Chen, K.: Stack decoding of polar codes. Electron. Lett. 48(12), 695–696 (2012)
6. Afisiadis, O., Balatsoukas-Stimming, A., Burg, A.: A low-complexity improved successive cancellation decoder for polar codes. In: 48th Asilomar Conference on Signals, Systems and Computers, pp. 2116–2120 (2014)
7. Alamdar-Yazdi, A., Kschischang, F.R.: A simplified successive-cancellation decoder for polar codes. IEEE Commun. Lett. 15(12), 1378–1380 (2011)
8. Ardakani, M.H., et al.: Fast successive-cancellation-based decoders of polar codes. IEEE Trans. Commun. 67(7), 4562–4574 (2019)
9. Zhang, Z., et al.: A split-reduced successive cancellation list decoder for polar codes. IEEE J. Sel. Areas Commun. 34(2), 292–302 (2016)
10. Gao, C., et al.: Path splitting selecting strategy-aided successive cancellation list algorithm for polar codes. IEEE Commun. Lett. 23(3), 422–425 (2019)
11. Balatsoukas-Stimming, A., Parizi, M.B., Burg, A.: LLR-based successive cancellation list decoding of polar codes. IEEE Trans. Signal Process. 63(19), 5165–5179 (2015)
12. Niu, K., Chen, K.: CRC-aided decoding of polar codes. IEEE Commun. Lett. 16(10), 1668–1671 (2012)
13. Zhang, Z., et al.: Progressive Bit-Flipping decoding of polar codes over layered critical sets. In: GLOBECOM 2017-2017 IEEE Global Communications Conference, pp. 1–6 (2017)
14. Trifonov, P.: Efficient design and decoding of polar codes. IEEE Trans. Commun. 60(11), 3221–3227 (2012)
15. Chung, S.-Y., Richardson, T.J., Urbanke, R.L.: Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. IEEE Trans. Inf. Theory 47(2), 657–670 (2001)
16. Rowshan, M., Viterbo, E.: Stepped list decoding for polar codes. In: IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC), pp. 1–5 (2018)
17. 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. 18(7), 1099–1102 (2014)

How to cite this article: Peng Y, Bao J, Liu X. An improved path splitting strategy on successive cancellation list decoder for polar codes. IET Commun. 2021;15:1198–1209.
https://doi.org/10.1049/cmu2.12153