Enumeration of Minimum Hamming Weight Polar Codewords with Sublinear Complexity

Fengyi Cheng, Aijun Liu, Jincheng Dai, Kai Niu, and Xiaohu Liang

Abstract

Polar code, with explicit construction and recursive structure, is the latest breakthrough in channel coding field for its low-complexity and theoretically capacity-achieving property. Since polar codes can approach the maximum likelihood performance under successive cancellation list decoding (SCLD), its decoding performance can be evaluated by Bonferroni-type bounds (e.g., union bound) in which the Hamming weight spectrum will be used. Especially, the polar codewords with minimum Hamming weight (PC-MHW) are the most important item in that bound because they make major contributions to the decoding error pattern particularly at high signal-to-noise-ratio. In this work, we propose an efficient strategy for enumerating the PC-MHW and its number. By reviewing the inherent reason that PC-MHW can be generated by SCLD, we obtain some common features of PC-MHW captured by SCLD. Using these features, we introduce a concept of zero-capacity bit-channel to obtain a tight upper bound for the number of PC-MHW, whose computing complexity is sublinear with code length. Furthermore, we prove that the proposed upper bound is really the exact number of PC-MHW in most cases. Guided by the bound and its theoretical analysis, we devise an efficient SCLD-based method to enumerate PC-MHW, which requires less than half of the list size compared with the existing methods.

Index Terms

Polar codes, Minimum Hamming weight, Successive cancellation list decoder.

This work is supported by the National Natural Science Foundation of China (No.61501508, No.61671476 and No.61901516), in part by the Natural Science Foundation of Jiangsu Province of China (No.BK20180578) and China Postdoctoral Science Foundation (No.2019M651648). Fengyi Cheng, Aijun Liu and Xiaohu Liang are with the department of Communication Engineering, Army Engineering University of PLA, Nanjing 210007, China (e-mail: tech_cfy@126.com, liuaj.cn@163.com, liangxiaohu688@163.com). Jincheng Dai and Kai Niu are with the Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications (BUPT), Beijing 100876, China (email: dai-jincheng@bupt.edu.cn, niukai@bupt.edu.cn)
I. INTRODUCTION

POLAR codes can achieve Shannon capacity under successive cancellation decoding (SCD) as the code length goes to infinity [1]. For the moderate code length, however, SC decoding (SCD) cannot provide satisfactory performance. To overcome this shortcoming, successive cancellation list decoding (SCLD) was proposed by [2]. Unlike SCD, SCLD can reserve $L$ most reliable decoding paths, where the reliability is evaluated by a path metric (PM). After that, some algorithms are proposed to improve the decoding latency, memory space and power overhead of SCLD [3]- [5]. Especially, assisted by cyclic redundancy check (CRC) [2] [6], the performance of SCLD can be further improved to make polar codes as the coding scheme for the control channel in the $5^{th}$ generation wireless communication standards [7].

Considering SCLD can approach the ML performance even for practical list size $L$ (e.g., $L=8$), its performance, especially at high signal to noise ratio (SNR), can be evaluated by the union bound where the number of polar codewords with minimum hamming weight (PC-MHW) is the most important item [8]. Moreover, [9] further verifies that the inherent reason that CRC can improve the performance of SCLD is preventing the erroneously decoded minimum Hamming distance codewords from passing the check. The first method to enumerate PC-MHW and its number is proposed in [10], where the authors verify that if the all-zero codeword is BPSK modulated and transmitted by the noiseless AWGN channel, then the information vector whose corresponding codeword has MHW will survive in the $L$ remaining paths on the competition of SCLD. Based on the one to one relationship between the codeword with its information vector, the PC-MHW can also be seen as the output of SCLD. However, this method requires large computation complexity and memory space. To solve this problem, [11] proposes a searching strategy which can divide the set of PC-MHW into several subsets for searching, so as to narrow down the required list size of SCLD. In practical, both the used list size in these two methods should be set larger than the actually required value to prevent omission. This extra list size, which can be regarded as unnecessary overhead, can be avoided if the number of PC-MHW can be approximately predicted before searching. In [12], a probabilistic computation method is proposed to evaluate the Hamming weight spectrum of polar codes with complexity $O(N^5)$, where $N$ is the code length. Thereafter, the accuracy of this evaluation is enhanced by [13] and the complexity can be reduced to $O(N^3)$. However, this method is tenable only at high code rate. Moreover, the complexity of these two methods are still too high to analyze the codes with
moderate or long code length.

In this paper, we propose an efficient strategy to enumerate the PC-MHW and its number. Based on the fact that the PC-MHW can be searched by SCLD, we analyze the characters of the PC-MHW searched by the SCLD. Then, guided by these characters, we propose a tight upper bound for evaluating the number of PC-MHW with sublinear complexity $O(\log_2 N)$. Based on such bound, we further propose an efficient strategy to search the PC-MHW by less than half of the list size required in the existing methods.

The highlights of our contributions are summarized as follows:

1) The essential reason that the PC-MHW can be generated by SCLD is reviewed. For each path of SCLD, there may exist some bits that does not be hard decided according to the corresponding decoding log-likelihood ratio (LLR). The set of the locations of such bits is referred to as reverse decision set (RDS). We prove that the set of PC-MHW can be divided into several subsets to ensure the information vectors of the PC-MHW in a same subset share the same RDS when they are taken as the output of SCLD-based searching.

2) We give a tight upper bound of the number of PC-MHW. For each subset of PC-MHW, divided according to their RDS obtained in SCLD-based searching, we can give an upper bound for its cardinality. By adding all such upper bounds, the bound for the total number of the PC-MHW can be derived. We further demonstrate that this upper bound, obtained with sublinear complexity $O(\log_2 N)$, is really the exact number of PC-MHW at most code rate or code length.

3) We propose an efficient strategy for enumerating PC-MHW. Guided by the proposed upper bound and the theoretical analysis of it, we can further divide the subset of PC-MHW into several smaller subsets. In once searching, we only need to search one smaller subset but instead of the whole PC-MHW. Thus, the required list size of SCLD used for searching can be further narrowed down to less than half of that used in the existing methods, so as to reduce the complexity and space memory for searching.

The reminder of this paper is organized as follows. Section II describes some basic conception associated with polar codes, which will be incurred in the following paper. In Section III, we will reveal how the information vectors of PC-MHW can be enumerated by SCLD so as to obtain their common features captured by SCLD. An upper bound of the number of PC-MHW is described in Section IV. In Section V, an efficient strategy for enumerating PC-MHW is proposed. Simulation results are given in Section VI, and conclusions are drawn in Section VII.
By necessity, this paper contains a fair amount of theoretical proof. Thus, on a first reading, the reader is advised to preview the Section III.A which will provide a high-level description of the proposed enumeration for the number of PC-MHW.

II. PRELIMINARIES

A. Notation Conventions

In this paper, we use lowercase letters, such as $x$, to denote scalars. $[x]$ is a ceiling function of a float value $x$. We write calligraphic characters (e.g., $\mathcal{X}$) to denote sets. $|\mathcal{X}|$ is cardinality of $\mathcal{X}$. $\mathcal{X}-\mathcal{Y}$ means the difference set between $\mathcal{X}$ and $\mathcal{Y}$. $\phi$ stands for null vector or null set. The notation $x_i^j$, with $i<j$, is used to denote a vector $[x_i,x_{i+1},...,x_j]$. If $i>j$, $x_i^j=\phi$. If $i=j$, $x_i^j=x_i$. When the dimension does not need to be emphasized, we also use bold lowercase letters or Greek letters, such as $\mathbf{x}$ or $\alpha$, to denote vectors. $\mathbf{0}$ and $\mathbf{1}_i$ stand for $i$-length all-zero and all-one vector, respectively. Note that when $i\leq 0$, $\mathbf{0}$ and $\mathbf{1}_i$ are both null vector. Let $[i,j]$ be the set of consecutive integer $\{i,i+1,...,j\}$. Given an index set $\mathcal{X}\subset[i,j]$, let $\mathbf{u}_\mathcal{X}$ denote the subvector of $\mathbf{u}_k$s with $k\in\mathcal{X}$. $w(\mathbf{u})$ denotes the Hamming weight of $\mathbf{u}$. Bold letters, such as $\mathbf{X}$, denote matrices. $\mathcal{B}$ and $\mathcal{Z}^*$ denote the set of binary $\{0,1\}$ and positive integer, respectively. $\otimes$ is the Kronecker product. $N(a,b)$ denotes Gaussian distribution with mean $a$ and variance $b$. $\text{sign}(\cdot)$ is the sign function. $\text{min}(\mathcal{X})$ is the minimum value in set $\mathcal{X}$.

Let $\mathbf{b}_n(i)=[b_{i,1},b_{i,2},...,b_{i,n}]$ be the vector of binary expansion of integer $i$, where the $b_{i,1}$ is the least significant bit. $\mathbf{b}_n(i)^k$ stands for vector $[b_{i,j},b_{i,j+1},...,b_{i,k}]$, with $1\leq j\leq k\leq n$.

For vector $\mathbf{x}$ that contains element $\alpha$, $\mathcal{P}_\alpha[\mathbf{x}]$ denotes the set of positions of $\alpha$ in $\mathbf{x}$. $\mathcal{P}_\alpha[\mathbf{x}]_j$ represents an element in $\mathcal{P}_\alpha[\mathbf{x}]$, $j=1,2,...,|\mathcal{P}_\alpha[\mathbf{x}]|$, and we assume $\mathcal{P}_\alpha[\mathbf{x}]_1<\mathcal{P}_\alpha[\mathbf{x}]_2<...<\mathcal{P}_\alpha[\mathbf{x}]_{|\mathcal{P}_\alpha[\mathbf{x}]|}$.

Example 1: For $\mathbf{x}=[1,1,0,1,0]$, $\mathcal{P}_0[\mathbf{x}]=\{3,5\}$, $\mathcal{P}_0[\mathbf{x}]_1=3$, $\mathcal{P}_0[\mathbf{x}]_2=5$; $\mathcal{P}_1[\mathbf{x}]=\{1,2,4\}$, $\mathcal{P}_1[\mathbf{x}]_1=1$, $\mathcal{P}_1[\mathbf{x}]_2=2$, $\mathcal{P}_1[\mathbf{x}]_3=4$.

B. Polar Codes

A polar code with message length $K$ and code length $N=2^n$, $n=1,2,...$, is determined by matrix $\mathbf{G}_N=\left[\begin{array}{c}1 & 0 \\ 0 & 1\end{array}\right]^\otimes n$ and the information set $\mathcal{A}\subset[1,N]$. Note that $|\mathcal{A}|=K$ and the code rate $R$ is equal to $K/N$. Let $\mathbf{u}_1^N$ be the information vector of polar code and $\mathbf{u}_\mathcal{A}$ be the vector of source message bits which are sent through the bit-channels with indices in $\mathcal{A}$. Let $\mathcal{A}^c=[1,N]-\mathcal{A}$. The bits sent by the bit-channels with indices in $\mathcal{A}^c$ are fixed to 0. Hence, the encoding process of polar code can be expressed as $\mathbf{c}_1^N=\mathbf{u}_1^N\mathbf{G}_N=\mathbf{u}_\mathcal{A}\mathbf{G}_N^{\mathcal{A}}$, where $\mathbf{c}_1^N$ is the codeword and $\mathbf{G}_N^{\mathcal{A}}$ is
the generator matrix which is composed of rows in $G_N$ with indices in $A$. The MHW of polar codes with $G_N^d$ is denoted by $d_m$. Throughout this paper, we assume that the codeword is BPSK modulated by \{0, 1\} → \{1, -1\}. Thus, the received sequence $y_1^N$ satisfies $y_i = (1 - 2c_i) + n_i$, where $n_i \sim N(0, \sigma^2)$ is an additive white Gaussian noise (AWGN).

C. Successive-Cancellation Decoding

The process of SC decoding can be depicted on a code tree as shown in Fig.1(a) [14]. The stage of the root node is the depth of the tree. For $N$-length polar code, the code tree is composed of $2N-1$ nodes and $n+1$ stages. The root node and the leaf nodes are at the stage $n$ and 0, respectively. Let $V_\lambda^{(j)}$ denote the $j$-th node (counting from the left) at stage $\lambda$, with $j \in \llbracket 1, 2^{N-\lambda} \rrbracket$. Each node has a LLR vector $\alpha[V_\lambda^{(j)}]$ and a codeword vector $\beta[V_\lambda^{(j)}]$. Both of these two message vectors are $2^\lambda$-length and they can be written as

$$
\alpha[V_\lambda^{(j)}] = \alpha[V_\lambda^{(j)}]_{1}^{2^\lambda} = \left[ \alpha[V_\lambda^{(j)}]_1, \alpha[V_\lambda^{(j)}]_2, \ldots, \alpha[V_\lambda^{(j)}]_{2^\lambda} \right]
$$

$$
\beta[V_\lambda^{(j)}] = \beta[V_\lambda^{(j)}]_{1}^{2^\lambda} = \left[ \beta[V_\lambda^{(j)}]_1, \beta[V_\lambda^{(j)}]_2, \ldots, \beta[V_\lambda^{(j)}]_{2^\lambda} \right]
$$

Actually, the SC decoding process is the process of calculating such two types of message vectors for each node. The LLR vectors are calculated from stage $n$ to stage 0 while the codeword vectors are updated from stage 0 to stage $n$.

**Definition 1 (decoding LLR, decoding bit and input vector):** For any leaf node $V_0^{(j)}$, $j \in [1, N]$, $\alpha[V_0^{(j)}]$ and $\beta[V_0^{(j)}]$ have only one element. $\alpha[V_0^{(j)}]$ is referred to as decoding LLR. $\beta[V_0^{(j)}]$ is called the decoding bit. The LLR vector of the root node is called input vector of decoder.

At stage $n$, the input vector of decoder is set by $\alpha[V_n^{(1)}] = y_1^N$.

For any $\lambda \in \llbracket 1, n-1 \rrbracket$ and $j \in \llbracket 1, 2^\lambda \rrbracket$, node $V_\lambda^{(j)}$ has a parent node ($V_{\lambda+1}^{(2j-1)}$), a left child node ($V_{\lambda-1}^{(2j)}$) and a right child node ($V_{\lambda-1}^{(2j)}$). As shown in Fig.1(b), $V_\lambda^{(j)}$ will participate in calculating the LLR vectors of its two child nodes (i.e., $\alpha[V_{\lambda+1}^{(2j)}]$ and $\alpha[V_{\lambda-1}^{(2j)}]$) and send itself codeword vector $\beta[V_\lambda^{(j)}]$ to its parent node. When $\alpha[V_\lambda^{(j)}]$ is determined, $V_\lambda^{(j)}$ is immediately activated to calculate $\alpha[V_{\lambda-1}^{(2j)}]$ by

$$
\alpha[V_{\lambda-1}^{(2j)}]_i = \alpha[V_\lambda^{(j)}]_1 \llcorner \alpha[V_\lambda^{(j)}]_{i+2^{\lambda-1}}, \quad i \in \llbracket 1, 2^{\lambda-1} \rrbracket
$$

with $x \llcorner y := \text{sign}(x) \text{sign}(y) \min(|x|, |y|)$. After receiving $\beta[V_{\lambda-1}^{(2j-1)}]$ from its child node, $V_\lambda^{(j)}$ will update $\alpha[V_{\lambda-1}^{(2j)}]$ by

$$
\alpha[V_{\lambda-1}^{(2j)}]_i = \left(1 - 2\beta[V_{\lambda-1}^{(2j-1)}]_i\right) \alpha[V_\lambda^{(j)}]_i + \alpha[V_\lambda^{(j)}]_{i+2^{\lambda-1}}
$$
with \( i \in \left[1, 2^{\lambda-1}\right] \). Then, \( V_{\lambda}^{(j)} \) waits until it receives \( \beta[V_{\lambda-1}^{(2j)}] \) and \( \beta[V_{\lambda-1}^{(2j-1)}] \) to update \( \beta[V_{\lambda}^{(j)}] \)

\[
\beta[V_{\lambda}^{(j)}]_i = \begin{cases}
\beta[V_{\lambda-1}^{(2j)}]_i \oplus \beta[V_{\lambda-1}^{(2j-1)}]_i, & i \leq 2^{\lambda-1} \\
\beta[V_{\lambda-1}^{(2j)}]_i, & i > 2^{\lambda-1}
\end{cases}
\]

(5)

If \( \lambda=0 \), \( \beta[V_{\lambda}^{(j)}] \) is the decoding bit \( \hat{u}_j \) and can be derived by making hard decision of the decoding LLR \( \alpha[V_0^{(j)}] \), i.e.,

\[
\hat{u}_j = \beta[V_0^{(j)}] = h(\alpha[V_0^{(j)}]) = \begin{cases}
0, & j \in \mathcal{A}^c \text{ or } \alpha[V_0^{(j)}] > 0 \\
1, & \text{others}
\end{cases}
\]

(6)

\( \beta[V_{\lambda}^{(j)}] \) can also be directly calculated by the decoding bits

\[
\beta[V_{\lambda}^{(j)}] = \hat{u}_i^{2^\lambda} \mathbf{G}_{2^\lambda}
\]

(7)

For any \( i \in \left[ (j-1)2^\lambda+1, j2^\lambda \right] \), we can say leaf node \( V_0^{(j)} \) or decoding bit \( \hat{u}_i \) participates in the calculating of \( \beta[V_{\lambda}^{(j)}] \). On completion of updating \( \beta[V_{\lambda}^{(j)}] \), the operations associated with \( V_{\lambda}^{(j)} \) are terminated and \( V_{\lambda}^{(j)} \) will never be activated.

### D. Successive-Cancellation List Decoding

SCLD will reserve more than one decoding candidates (or paths). For any position \( i \in \mathcal{A} \), SCLD splits every decoding paths into 2 threads to consider both the probability of the current bits being 0 or 1. Thus, the decoding bits in one path may not be determined according to the suggestion of its LLR. To avoid an exponentially growing complexity, at most \( L \) most reliable paths (or trajectories) could be reserved in the whole decoding process. Unlike SCD, when referring to a decoding trajectory of the SCLD, we should indicate the decoding step and the order of the path in the reserving list. The \( i \)-th decoding step means that when \( u_i \) is just decoded by SCLD. Let \( u_i[l]^2 = [u_i[l]_1, u_i[l]_2, ..., u_i[l]_j] \) denote the vector of the first \( j \) bits of the \( l \)-th decoding path at the
$i$-th decoding step, $i > j$. Note that $u_i[l_1^i]$ is not necessarily equal to $u_{i+1}[l_1^{i+1}]$. Especially, if $i < j$, $u_i[l_1^i] = \phi$. When all the bits are decoded, the most reliable path will be output as the decoding result. The reliability is evaluated by a path metric (PM). In this paper, we adopt the LLR-based SCLD proposed in [3] whose PM is calculated in logarithmic domain. Concretely, the PM of decoding paths in this paper is defined as follows:

**Definition 2 (PM and RDS):** For any $i \in [1, N]$, the PM of the $l$-th decoded path $\hat{u}_i[l_1^i]$, $l \in [1, L]$, is defined by:

$$ PM[l_1^i] = \sum_{j \in A_{l_1^i}} |L_i[l_1^i]_j|, $$

where $L_i[l_1^i]_j = \ln \frac{Pr(\hat{u}_i[l_1^i]_j = 0 | y_1^{N-1}, \hat{u}_i[l_1^i]_{\sim j} = 1)}{Pr(\hat{u}_i[l_1^i]_j = 1 | y_1^{N-1}, \hat{u}_i[l_1^i]_{\sim j} = 0)}$ is the decoding LLR for $\hat{u}_i[l_1^i]_j$, and $A_{l_1^i}$, called reverse decision set (RDS) of path $\hat{u}_i[l_1^i]$, is the set of positions at which $\hat{u}_i[l_1^i]$ does not make hard decision based on decoding LLR.

Note that RDS can also include the elements in $A_c$. In the following, we simply use $\overline{A}_N[l]$ to denote the RDS of $\hat{u}_N[l_1^N]$.

**E. Searching for PC-MHW**

For polar codes with $G_A^N$, let $\mathcal{U}_{N,m}$ be the set of information vector whose corresponding codeword has MHW. $d_m$ is equal to the minimum row weight of $G_A^N$ [15]. $\mathcal{U}_{N,m}$ can be searched by SCLD under the condition that all-0 polar codeword is BPSK modulated and sent by noiseless AWGN channel [10]. In the following paper, this condition will be equivalently defined as

$$ y_1^N = 1_N. $$

Under this condition, on completion of SCLD, if we first discard the all-0 path, then, from the remaining $L-1$ decoding paths, we can obtain $\mathcal{U}_{N,m}$ by selecting the $|\mathcal{U}_{N,m}|$ most reliable ones.

Further, to reduce the searching latency and the required memory space, [11] proposes a multi-level SCLD-based method by dividing $\mathcal{U}_{N,m}$ into $|A_m|$ subsets to search, where $A_m$ satisfies

$$ A_m = \{ i \in \mathcal{A} | w(g_{N}^{(i)}) = d_m \} $$

where $g_{N}^{(i)}$ is the $i$-th row vector of $G_N$. The division of $\mathcal{U}_{N,m}$ is expressed as

$$ \mathcal{U}_{N,m} = \bigcup_{i \in A_m} \mathcal{U}_{N,m}^{(i)} $$

where $\mathcal{U}_{N,m}^{(i)}$ is the set of vector $u_1^N$ that satisfies the following two conditions:

1) $u_1^{i-1} = 0_{i-1}$, $u_i = 1$, $u_{A_i} \in B^{\mid A_i \mid}$ and $u_{A^c_i} = 0_{\mid A^c_i \mid}$, with $A_i = \mathcal{A} \cap [i+1, N]$ and $A^c_i = \mathcal{A}^c \cap [i+1, N]$. 


2) \( w(u_1^N G_N) = w(g_N) \).

Based on the one to one relationship between the codeword with its information vector, when \( U_{N,m} \) is searched, the set of polar codewords with MHW is also derived.

### III. Features of PC-MHW as Output of Multi-Level SCLD

#### A. High Level Description of Enumeration

In the following, we shall use the expression of searched path to refer to the decoded path of SCLD under \( y_1^N = 1_N \). Now that the information vectors in \( U_{N,m}^{(i)} \) can be searched by the multi-level SCLD [11], they should have the same feature that can be captured by the decoder. Actually, [11] has proved that the PMs of all the searched paths in \( U_{N,m}^{(i)} \) are equal. In this paper, we further prove that for any searched path in \( U_{N,m}^{(i)} \), its RDS is fixed to \( \{i\} \) (Theorem 7). In general AWGN channel, where none of the decoding LLR is 0-valued, there can exist only one decoding path of SCLD whose RDS is \( \{i\} \). However, under \( y_1^N = 1_N \), there can be \( |U_{N,m}^{(i)}| \geq 1 \) such paths. This implies that for any searched path in \( U_{N,m}^{(i)} \), some decoding LLRs would be 0. In other words, the number of 0-valued decoding LLR should be associated with \( |U_{N,m}^{(i)}| \).

Guided by this conclusion, in Section IV we further prove that for all the searched paths in \( U_{N,m}^{(i)} \), the location sets of their 0-valued decoding LLR are identical (Theorem 16). Such set, denoted by \( \mathcal{I}_0^i \), can be determined based on a concept of zero-capacity bit-channel associated with \( i \). Using \( \mathcal{I}_0^i \), we can give the upper bound of \( |U_{N,m}^{(i)}| \) (Theorem 17). Since \( U_{N,m} = \bigcup_{i \in \mathcal{A}_m} U_{N,m}^{(i)} \), by adding up the upper bound of each subset, the upper bound of \( |U_{N,m}| \) follows.

Recalling that \( U_{N,m} \) can be divided into several subsets according to index set \( \mathcal{A}_m \), in Section V, we adopt this idea and further divide \( U_{N,m}^{(i)} \) into several small subsets according to index set \( \mathcal{I}_0^i \) [see equation (33)]. Based on this division, we propose a searching strategy for PC-MHW whose required list size is less than half of that required in the existing methods (Algorithm 1).

The theoretical analysis of this paper is based on three steps of simplification. First, by introducing a retracing SCLD, we can concentrate on one single searched path, but instead of all of them output by SCLD. Then, we divide the searching process of one single path into two phases and prove that the 0-valued decoding LLR can only be generated at the second phase. Thus, we only need to focus on the second phase when seeking the locations of 0-valued decoding LLRs in a searched path. Finally, the third simplification is the decomposition of the second searching phase. Note that this decomposition, based on the recursive structure of polar codes, is not a new idea: it was applied in [14] to simplify the process of SCD.
B. Retracing SCLD Depicted on Code Tree

For any decoding path $\hat{u}_N[l][N]$, its message updating rules in SCLD are identical with those used in SCD except for the final hard bit decision rule. Actually, if $\overline{A}_N[l] \cap \mathcal{A} = \phi$, $\hat{u}_N[l][N]$ can also be seen as the output of SCD. Based on this, we propose a re-tracing SCLD (RSCLD) which performs the same message updating operation as SCD but with a determined decoding output.

We will use the code tree to describe the re-tracing process. In this way, the re-tracing process is actually a process of updating the two kinds of message vectors for each node. The updating order is identical with SCD. To simplify notation, in the RSCLD we still use $\alpha[V_\lambda^((j))]$ and $\beta[V_\lambda^((j))]$ to respectively denote the LLR and codeword vector of node $V_\lambda^((j))$ like in SCD.

RSCLD outputs only one single path and its only difference from SCD is the process of hard bit decision. Using RSCLD, we just retrace the generating process of a known decoded path of SCLD. Thus, $\overline{A}_N[l]$ is assumed to be known. Concretely, on the code tree, the RSCLD starts with setting $\alpha[V_0(1)]=2y_i/\sigma^2$, $i \in [1,N]$. Then, it performs the same decoding procedure as SCD except for the leaf nodes. At stage 0, it should adopt the following decision rule

$$
\hat{u}_N[l][j]=\beta[V_0^((j))]=\begin{cases} 1-h(\alpha[V_0^((j))]), & j \in \overline{A}_N[l] \cap \mathcal{A} \\ h(\alpha[V_0^((j))]), & \text{others} \end{cases}
$$

Definition 3 (GAN): On the code tree, for any non-root node $V_\lambda^((j))$, there exists one edge that can connect $V_\lambda^((j))$ with the root node. The generalized ancestor nodes (GAN) of $V_\lambda^((j))$ are the nodes on such edges (including $V_\lambda^((j))$ itself). The set of the GAN of $V_\lambda^((j))$ is denoted by $\mathcal{G}[V_\lambda^((j))]$.

Example 2: On the 8-length code tree, shown in Fig.1(a), we use the filled cycles to signify the nodes in $\mathcal{G}[V_0^((3))]$. At any stage larger than $\lambda$, there is one and only one GAN of $V_\lambda^((j))$.

We can conclude some properties about GAN:

Property 1: For any node $V_\lambda^((j))$, $\lambda \in [0, n-1]$, $j \in [1, 2^{N-\lambda}]$, to determine $\alpha[V_\lambda^((j))]$, all the LLR vectors of its GAN should be calculated in advance.

Property 2: Given a leaf node $V_0^((i))$, $i \in [1, N]$, if it participates in updating $\beta[V_\lambda^((j))], V_\lambda^((j)) \in \mathcal{G}[V_0^((i))]$.

Property 3: A node is the GAN of both its two child nodes.

Property 4: Given 3 nodes $V_\lambda^((j))$, $V_{\lambda_2}^((j_2))$ and $V_{\lambda_3}^((j_3))$, $\lambda_1 > \lambda_2 > \lambda_3$, if $V_\lambda^((j_1)) \in \mathcal{G}[V_{\lambda_2}^((j_2))]$ and $V_{\lambda_2}^((j_2)) \in \mathcal{G}[V_{\lambda_3}^((j_3))]$, then $V_{\lambda_1}^((j_1)) \in \mathcal{G}[V_{\lambda_3}^((j_3))]$. 

C. Division of Searching Process

From [11], we can find that $\mathcal{U}_{N,m}$ can be divided by (11) to search. This implies that the information vectors included in $\mathcal{U}_{N,m}^{(i)}$, $i \in \mathcal{A}_m$, should share some common features when searched by SCLD. To explore such features, we shall first focus on a single searched path in $\mathcal{U}_{N,m}^{(i)}$.

For any searched path $\hat{u}_N[l]_1^N \in \mathcal{U}_{N,m}^{(i)}$, its decoding process can be divided into 2 phases, i.e., the decoding of $\hat{u}_N[l]_1^i = [0_{i-1}, 1]$ (the 1st phase) and the decoding of $\hat{u}_N[l]_{i+1}^N$ (the 2nd phase). In other words, the first phase starts with feeding the SCLD by $y_1^N$ and ends up with decoding $u_i$. The remaining part of decoding process is the second phase. Note that in the following when referring to the decoding phase, we only consider in a single path.

If we use RSCLD to retrace the decoding process of any searched path $\hat{u}_N[l]_1^N \in \mathcal{U}_{N,m}^{(i)}$, we can summarize the features of its first decoding phase in the following lemmas.

**Lemma 4:** For any nodes $V_{\lambda}^{(j)}$, if $\alpha[V_{\lambda}^{(j)}]$ is updated at the first decoding phase, all the LLRs in $\alpha[V_{\lambda}^{(j)}]$ are identical and positive, i.e., $\alpha[V_{\lambda}^{(j)}]_1 = \alpha[V_{\lambda}^{(j)}]_2 = \ldots = \alpha[V_{\lambda}^{(j)}]_{2\lambda} > 0$. If $\beta[V_{\lambda}^{(j)}]$ is updated at the first decoding phase, we have $\beta[V_{\lambda}^{(j)}] = 0_{2\lambda}$.

**Proof:** This can be easily verified by the message updating rules under $\alpha[V_n^{(1)}] = y_1^N = 1_N$. ■

**Lemma 5:** For any searched path $\hat{u}_N[l]_1^N \in \mathcal{U}_{N,m}^{(i)}$, its RDS should include $i$, i.e., $\overline{A}_N[l] \supseteq \{i\}$.

**Proof:** From Lemma 4, all the decoding LLRs updated at the first decoding phase should be positive. Considering $\hat{u}_N[l]_1^i = [0_{i-1}, 1]$, the theorem follows. ■

Let $\mathcal{U}_{N,m}^{(i)}$ be the set of vector $u_{i}^N$ that satisfies $u_1^{i-1} = 0_{i-1}$, $u_i = 1$, $u_{A_i} \in \mathcal{B}^{|A_i|}$, $u_{A_c} = 0_{|A_c|}$. Clearly, $\mathcal{U}_{N,m}^{(i)} \subseteq \mathcal{U}_{N,m}^{(i)}$. When using the multi-level SCLD in [11] to search $\mathcal{U}_{N,m}^{(i)}$, with $i \in \mathcal{A}_m$, the work the decoder actually do is to recognize the paths in $\mathcal{U}_{N,m}^{(i)}$ from all the paths in $\mathcal{U}_{N}^{(i)}$. Since all the vectors in $\mathcal{U}_{N}^{(i)}$ (including $\mathcal{U}_{N,m}^{(i)}$) share the same first $i$ bits, i.e., $[0_{i-1}, 1]$, the first decoding phase of all the paths in $\mathcal{U}_{N}^{(i)}$ are identical. That’s to say, the second decoding phase is the crux to search $\mathcal{U}_{N,m}^{(i)}$ by multi-SCLD and should be analyzed emphatically.

D. Characteristics of Searched Path in $\mathcal{U}_{N,m}^{(i)}$

In this part, we will focus on the second decoding phase to review how the PC-MHW can be searched by the multi-level SCLD.

**Theorem 6:** For any two different searched paths, if their corresponding codewords have the same weight, their PMs should be equal and vice versa.

**Proof:** This theorem can be easily proved by [6, lemma 3]. ■

**Theorem 7:** For any searched path $\hat{u}_N[l]_1^N$ that satisfies $\hat{u}_N[l]_1^N \in \mathcal{U}_{N,m}^{(i)}$, we have $\overline{A}_N[l] = \{i\}$. 
Proof: Please see Appendix A.

Theorem 8: For any searched path \( \hat{u}_N[l]^{N} \in \mathcal{U}_{N,m}^{(i)} \), if it has \( \overline{A}_N[l] = \{i\} \), \( w(\hat{u}_N[l]^{N}G_N) = w(g_N^{(i)}) \).

Proof: Since \( \overline{A}_N[l] = \{i\} \), from (8), the PM of \( \hat{u}_N[l]^{N} \) is \( \left| \alpha[V_0^{(i)}] \right| \). Based on Theorem 7, the PM of the searched path \( [0_{i-1}, 1, 0_{N-i}] \) is also \( \left| \alpha[V_0^{(i)}] \right| \). From Theorem 6, we have

\[
w(\hat{u}_N[l]^{N}G_N) = w([0_{i-1}, 1, 0_{N-i}]G_N) = w(g_N^{(i)}).
\] (13)

Therefore, the theorem is true.

Note that if the equation of PM is changed, e.g., using (10) in [3], Theorem 8 still holds. This is because such change will not effect the decoding result and the RDS of path will not change.

From the above, we can draw the following conclusions

- For any searched path \( \hat{u}_N[l]^{N} \in \mathcal{U}_{N,m}^{(i)} \), its decoding bits obtained at the second decoding phase, i.e., \( \hat{u}_N[l]^{N}_{i+1} \), are hard decided based on the decoding LLR.
- The necessary and sufficient condition for a searched path \( \hat{u}_N[l]^{N} \) belonging to \( \mathcal{U}_{N,m}^{(i)} \), \( i \in \mathcal{A}_m \), is \( \overline{A}_N[l] = \{i\} \), i.e.,

\[
\hat{u}_N[l]^{N} \in \mathcal{U}_{N,m}^{(i)} \iff \overline{A}_N[l] = \{i\}, \quad i \in \mathcal{A}_m \tag{14}
\]

- The number of \( \mathcal{U}_{N,m}^{(i)} \) equals to the number of the searched paths whose RDS is \( \{i\} \).

In general AWGN channel, where the noise can not be negligible and none of the decoding LLRs is 0-valued, there can only be one decoding path of SCLD whose RDS is \( \{i\} \). However, given \( y_N^{N} = 1_N \) and decoded trajectory \( \hat{u}_1^{N} = [0_{i-1}, 1] \), some decoding LLRs for \( u^{N}_{i+1} \) could be 0-valued and their corresponding decoding bits, whether decoded as 1 or 0, can be regarded to be hard decided by the suggestion of the LLR. Thus, at a position of 0-valued decoding LLR, the current decoding trajectory can be split into 2 threads with no change of the original RDS. This is essentially why there are multiple searched paths whose RDS is \( \{i\} \). This implies that given \( y_1^{N} \) and \( \hat{u}_1^{i} = [0_{i-1}, 1] \), the number of the zero-valued decoding LLR updated in the second decoding phase may determine \( \mathcal{U}_{N,m}^{(i)} \).

IV. ENUMERATOR OF PC-MHW

In previous, RSCLD made it possible to simplify the analysis of the entire searching process into the analysis of a single path in \( \mathcal{U}_{N,m}^{(i)} \). Since the path is chosen randomly, the analysis can reflect the commonality of all paths in \( \mathcal{U}_{N,m}^{(i)} \). We obtained that the locations of the 0-valued decoding LLRs would associate with \( \mathcal{U}_{N,m}^{(i)} \). To seek such locations, in this part, we will further simplify the analysis by decomposition of the second searching phase of any path in \( \mathcal{U}_{N,m}^{(i)} \).
A. Decomposition of Second Searching Process

When searching any path \(\hat{u}_N[i]_1^N \in U_{N,m}^{(i)}\), \(\hat{u}_N[i]_i=1\) is the only reason to cause some subsequent decoding LLRs to be 0-valued. Hence, the 0-valued decoding LLRs can only exist in the second decoding phase. To determine their positions, we will disassemble the second decoding phase.

The decomposition can be visualized on the code tree by dividing the nodes whose LLR vector is updated at the second phase into several subcode-trees. Since the code tree used to represent decoder is full binary, the subcode-trees obtained by decomposition can be uniquely identified by their leaf nodes. That’s to say, dividing all the nodes whose LLR vector is updated at the second phase is tantamount to dividing the \(N-i\) leaf nodes following \(V_0^{(i)}\), i.e., \(V_0^{(i+1)}\) to \(V_0^{(N)}\). We will introduce the dividing method and prove its rationality in Theorem 10. Before that, we first give a lemma to prove the existence of the decomposition for any \(i \in [1, 2^{n-1}]\).

Lemma 9: Given any integer \(i \in [1, 2^{n-1}]\), we have
\[
2^n-i = \gamma_i(n-w(b_n (i-1))) = \gamma_i(|P_0[b_n (i-1)]|)
\]
where
\[
\gamma_i(x) \triangleq \sum_{k=1}^{x} 2^{P_0[b_n(i-1)]}k-1.
\]

Proof: Since \(2^n-i = (2^n-1)-(i-1)\), this lemma can be easily proved by expanding \(2^n-1\) and \(i-1\) binary.

Lemma 9 implies that any positive integer \(2^n-i\) can be broken into \(|P_0[b_n (i-1)]|\) smaller positive integers. Thus, for any leaf node with index \(i \in [1, N-1]\) (i.e., \(V_0^{(i)}\)), we can divide its subsequent \(N-i\) ones into \(|P_0[b_n (i-1)]|\) parts. For the \(k\)-th part, \(k \in [1, |P_0[b_n (i-1)]|]\), let \(I_i(k)\) denote the index set of the leaf nodes in it. Specifically, \(I_i(k)\) has \(2^{P_0[b_n(i-1)]}k-1\) consecutive indices and is arrayed in the ascending order
\[
I_i(k) = \begin{cases} 
[i+1, i+\gamma_i(1)], & k=1 \\
[i+1+\gamma_i(k-1), i+\gamma_i(k)], & k \geq 2 
\end{cases}
\]

Theorem 10: The nodes whose LLR vector is calculated at the second phase can be divided into \(|P_0[b_n (i-1)]|\) subcode-trees. For the \(k\)-th subcode-tree, \(k \in [1, |P_0[b_n (i-1)]|]\), the index set of the leaf nodes is \(I_i(k)\). By doing this, we obtain that:

1) The \(k\)-th subcode-tree is rooted at node \(V_0^{(j(i,k))}\), where
\[
\lambda(i, k) = P_0[b_n (i-1)]_k-1, \quad j(i, k) = \frac{i + \gamma_i(k)}{2^{P_0[b_n(i-1)]}k-1}.
\]
2) For any node whose LLR vector is updated at the second phase, it must be involved in one and only one of the $|\mathcal{P}_0[b_n (i-1)]|$ subcode-trees.

Proof: For the first problem, from (17), we can find that the number of the leaf nodes involved in the $k$-th subcode-tree is $2^{\mathcal{P}_0[b_n (i-1)]} - 1$. Considering the subcode-tree is full binary, the depth of the subcode-tree is $\mathcal{P}_0[b_n (i-1)] k - 1$. Thus, $\lambda(i, k) = 2^{\mathcal{P}_0[b_n (i-1)]} - 1$. Similarly, since $i + \gamma_i(k)$ is the maximum index in $\mathcal{I}_i(k)$, we have $j(i, k) = \frac{i + \gamma_i(k)}{2^{\mathcal{P}_0[b_n (i-1)]} - 1}$.

Then, we will deal with the second problem. Assume that $\alpha[V_{\lambda}^{(j)}]$ is updated at the second phase. Clearly, node $V_{\lambda}^{(j)}$ should be GAN of at least one leaf node, denoted by $V_{0}^{(jo)}$. Since the subcode-trees are full binary, if $V_{\lambda}^{(j)}$ is excluded from the $|\mathcal{P}_0[b_n (i-1)]|$ subcode-trees obtained by decomposition, then the leaf node $V_{0}^{(jo)}$ should be also excluded. This contradicts with the fact obtained from Lemma 9, i.e., none of the leaf nodes is excluded from the obtained subcode-trees. Thus, all the nodes whose LLR vector is updated at the second phase are involved in by the decomposition. Similarly, if $V_{\lambda}^{(j)}$ belongs to two different subcode-trees simultaneously, leaf node $V_{0}^{(jo)}$ should also belong to two subcode-trees. This also contradicts with the fact. Thus, the theorem follows.

Note that $\lambda(i, k) = 0$ indicates that $i$ is odd and $k = 1$. Conversely, $k = 1$ does not necessarily lead to $\lambda(i, k) = 0$.

Example 3: In Fig.1(a), if $i = 1$, three subcode-trees obtained by the decomposition according to Theorem 10 are boxed out by the dashed line. The LLR vector of any excluded node is updated at the first phase. Note that if $\lambda(i, k) = 0$, the subcode-tree is a single leaf node $V_{0}^{(i+1)}$.

It is worth mentioning that this dividing is just a method to facilitate analysis and will not change or omit any process of the original searching. That’s to say, it would enable some procedures and factors in the second decoding phase to be ignored in the analysis, but they still exist in the actual decoding process. Specifically, in each local decoder (or, subcode-tree), we can deem that only the LLR vector of the root node is effected by its previous decoding while the other nodes are only effected by their root node. Therefore, once the LLR vector of the root node is determined, i.e., on activation of each local decoder, the impact of the previous decoding can be ignored in all its following message updating process in this local decoder.
B. Input Vector of Subcode-Tree

Based on the above decomposition, for any searched path in $\mathcal{U}_{N,m}^{(i)}$, determining its locations of the 0-valued decoding LLRs from the whole $N$ locations can be simplified by first determining them in any one of the $|\mathcal{P}_0|b_n(i-1)|$ local decoders. Then, in the rest $|\mathcal{P}_0|b_n(i-1)|-1$ ones, we can obtain them similarly. The crux of such simplification is to determine the input vector for each local decoder so that they can be regarded to be independent. We can use RSCLD to retrace the decoding process and draw some conclusions about the root node of the local decoder, which are given as follows.

**Theorem 11:** For any $i \in [1, N-1]$ and $k \in [1, |\mathcal{P}_0|b_n(i-1)|]$, $j(i, k)$ is even.

**Proof:** The binary expansion of $i+\gamma_i(k)$ is $b_n(i+\gamma_i(k))=[b_i+\gamma_i(k),1, b_i+\gamma_i(k),2, \ldots, b_i+\gamma_i(k),n]$.

Meanwhile, we have

$$b_n(i+\gamma_i(k)) = b_n(i-1) + b_n(\gamma_i(k)) + b_n(1)$$

(18)

Substituting (16) into (18), we have $b_i+\gamma_i(k),1, b_i+\gamma_i(k),2, \ldots, b_i+\gamma_i(k),|\mathcal{P}_0|b_n(i-1)| = 0_{|\mathcal{P}_0|b_n(i-1)|}$. This implies that $\frac{i+\gamma_i(k)}{2} \in \mathbb{Z}^+$. The theorem is proved.

**Theorem 11** indicates that when using RSCLD to retrace the searching process of any path $\hat{u}_N[l]_i \in \mathcal{U}_m^{(i)}$, $\alpha[V^{(j(i,k))}_{\lambda(i,k)}]$ can only be calculated by (4) in which $\alpha[V^{(j(i,k))}_{\lambda(i,k)+1}]$ and $\beta[V^{(j(i,k))-1}_{\lambda(i,k)}]$ will participate. By recalling the updating rule of $\beta[V^{(j(i,k))-1}_{\lambda(i,k)}]$, i.e., equation (7), we can easily obtain that

$$\beta[V^{(j(i,k))-1}_{\lambda(i,k)}] = \begin{cases} 1, & \lambda(i,k)=0 \\ \hat{u}_N[l]_{\text{min}(\mathcal{I}_i(k))-1}^{\text{min}(\mathcal{I}_i(k))}, & \lambda(i,k)>0 \end{cases}$$

(19)

It can be verified that

$$\text{min}(\mathcal{I}_i(k)) - 2^{\lambda(i,k)} \leq i \leq \text{min}(\mathcal{I}_i(k)) - 1.$$

Thus, $u_N[l]_i$ will participate in the calculation of $\beta[V^{(j(i,k))-1}_{\lambda(i,k)}]$. From **Property 2**, $V^{(j(i,k))-1}_{\lambda(i,k)} \in G(V^{(i)}_0)$.

As the parent node of $V^{(j(i,k))-1}_{\lambda(i,k)}$, $V^{(j(i,k))}_{\lambda(i,k)+1}$ also should be the GAN of $V^{(i)}_0$, and thus $\alpha[V^{(j(i,k))}_{\lambda(i,k)+1}]$ is updated at the first decoding phase. From **Lemma 4**, we can obtain that

$$\alpha[V^{(j(i,k))}_{\lambda(i,k)+1}] = x \cdot 1^{2^{\lambda(i,k)+1}}$$

(20)

We aim to give a theoretical prediction to determine the locations of zero-valued decoding LLRs, but instead of directly using the results of the decoder. Actually, in this section we will prove that for any searched path in $\mathcal{U}_m^{(i)}$, the locations of 0-valued decoding LLRs are fixed.
Substituting (19) and (20) into (4), we can conclude that
\[
\alpha[V^{(j(i,k))}_{\lambda(i,k)}] = \begin{cases} 
2^{x} \cdot (1_{2\lambda(i,k)} - \hat{u}_{N}[\min(I(k))-1]_{\min(I(k))-2\lambda(i,k)} G_{2\lambda(i,k)}), & \lambda(i,k) > 0 \\
0, & \lambda(i,k) = 0 
\end{cases}
\]  
(21)

We can further determine the value of constant \(x\). For the two updating formulae associated with LLR, i.e., (3) and (4), at the first decoding phase only (4) can change (or precisely, double) the LLR value of the input LLR vector. We have
\[
x = \begin{cases} 
2^{w(b_n(i-1))} & \lambda(i,k) + 1 < n \\
1, & \lambda(i,k) + 1 = n 
\end{cases}
\]  
(22)

where \(w(b_n(i-1))\) is equal to the times of using (4) to obtain \(V^{(j(i,k))/2}_{\lambda(i,k)+1}\), updating from root node \(V^{(1)}_{0}\) through the GANs of \(V^{(j(i,k))/2}_{\lambda(i,k)+1}\).

C. Zero-Capacity Bit-Channel

Let \(g^{(i)}_N(x)\) denote the vector of the first \(2^x\) elements in \(g^{(i)}_N\). Recalling the input LLR vector of the decomposed local decoder that has been determined in (21), the local decoders can be seen to be independent with each other. We can first focus on any one of the local decoders. For any \(i \in [1,N-1]\), we will define the zero-capacity bit-channels (ZCBC) as follows

**Definition 12:** For any \(k \in [1,|P_0[|b_n(i-1)|]|]\), if we use \(I^0_i(k)\) to denote the set of the \(h\)-th element in \(I_i(k)\), where
\[
h \in P_1[g^{(i)}_N(P_0[|b_n(i-1)|]_k-1)],
\]  
(23)

then the bit-channels with indices in \(I^0_i(k)\) are defined as zero-capacity bit-channels associated with \(i\) (ZCBC-i).

Note that all the \(|P_0[|b_n(i-1)|]|\) local decoders have their own respective ZCBC. Thus, the set of index of ZCBC-i, denoted by \(I^0_i\), can be expressed as
\[
I^0_i = \bigcup_{k=1}^{|P_0[|b_n(i-1)|]|} I^0_i(k) 
\]  
(24)

**Example 4:** For 8-length polar codes (i.e., \(n=3\)), if \(i=2\), we have \(P_0[|b_n(i-1)|] = \{2,3\}\). Thus, the following 6 bit-channels can be divided into \(|P_0[|b_n(i-1)|]| = 2\) parts. Concretely, \(I_i(1) = \{3,4\}\) and \(I_i(2) = \{5,6,7,8\}\). Since \(g^{(i)}_N(P_0[|b_n(i-1)|]_1) - 1 = g^{(i)}_N(2) - 1 = [1,1]\) and \(P_1[g^{(i)}_N(1)] = \{1,2\}\), both the two bit-channels in \(I_i(1)\) are ZCBC-i. Similarly, since \(g^{(i)}_N(P_0[|b_n(i-1)|]_2 - 1) = g^{(i)}_N(3) - 1 = [1,1,0,0]\) and \(P_1[g^{(i)}_N(2)] = \{1,2\}\), then the first two bit-channels in \(I_i(2)\) are ZCBC-i. Thus, \(I^0_i = I^0_i(1) \cup I^0_i(2) = \{3,4\} \cup \{5,6\} = \{3,4,5,6\}\).
D. Upper Bound for Number of PC-MHW

For bit-channel with index in $T_i^0$, we call it zero-capacity because we will prove that for any searched path $\hat{u}_N[i]_1^N \in U_{N,m}^{(i)}$, the set of the locations of the 0-valued decoding LLRs is $T_i^0$. Before that, we first give three lemmas which will be incurred in the following discussion.

**Lemma 13:** Given decoded path of SCD $\hat{u}_N^i \in U_{N,m}^{(i)}$, i.e., $[1, N]$, on the code tree, for any $\lambda \in [0, n-1]$ and $j_\lambda \in [1, 2^{n-\lambda}]$, if $V_\lambda^{(j_\lambda)} \in G[V_0^{(j)}]$, then we have

$$w(\beta[V_\lambda^{(j_\lambda)}]) = w(g(i)^{(i)})/2 |P_1[b_1(i-1)^{n+1}]]$$  (25)

**Proof:** When $\lambda=0$, we have

$$w(\beta[V_0^{(0)}]) = w(V_0^{(0)}) = w([\hat{u}_i]) = 1 = w(g(i)^{(i)})/2 |P_1[b_1(i-1)]$$  (26)

The lemma is true. If $\lambda=1$, it can be easily verified that

$$w(\beta[V_1^{(j_1)}]) = \begin{cases} 2 = w(g(i)^{(i)})/2 |P_1[b_1(i-1)]|^{-1}, & \text{if } i \text{ is even} \\ 1 = w(g(i)^{(i)})/2 |P_1[b_1(i-1)]|, & \text{if } i \text{ is odd} \end{cases}$$

Thus, it is also true for $\lambda=1$. Due to the recursive structure, when $\lambda=2$, we can regard the $N/2$ code-tree nodes at the stage 1 as the leaf nodes and ignore the nodes at stage 0. By doing this, the weight of the new leaf nodes is 2 (if its index is even) or 1 (if its index is odd). We can verify that this lemma is still true at stage 2. By an induction, the lemma follows. ■

**Lemma 14:** For any $\lambda \in [0, n-1]$

$$w(g(i)^{(i)}) = w(g(i)^{(i)})/2 |P_1[b_1(i-1)]|^{n+1}$$  (27)

**Proof:** If we regard this lemma as a special case of Lemma 13, i.e., the decoded path $\hat{u}_N^1$ is $[0_{1-1}, 1, 0_{N-1}]$, it follows. ■

**Lemma 15:** Under SCD, if the input vector $y_1^N$ satisfies

$$y_1^N = a \cdot (1_N - u_1^N G_N)$$  (28)

where $u_1^N$ is any vector in $U_m^{(i)}$, with $i \in [1, N-1]$, and $a$ is any positive constant, then the location set of the 0-valued decoding LLR is $P_1[g(i)^{(i)}]$. ■

**Proof:** Please see Appendix B.
**Theorem 16:** For any searched path \( \hat{u}_N[l]_T \in \mathcal{U}_{N,m}^{(i)}, \) the set of the locations of the 0-valued decoding LLRs is \( \mathcal{T}_i^0. \)

**Proof:** Since we only consider one single path, we can use RSCLD to analyze it generating process. The 0-valued decoding LLRs can only be generated at the second decoding phase which can be divided into \( |\mathcal{P}_0[b_n(i-1)]| \) local decoders according to **Theorem 10.**

Without loss of generality, we first focus on the \( k \)-th local decoder, \( k \in \{1, |\mathcal{P}_0[b_n(i-1)]| \}. \) From previous analysis, the bits decoded by the \( k \)-th local decoder, i.e., the bits with indices in \( \mathcal{T}_i(k) \) of path \( \hat{u}_N[l]_T \), denoted as \( \hat{u}_N[l]_{\mathcal{T}_i(k)} \), are made hard decision according to their decoding LLRs. Thus, for \( \hat{u}_N[l]_{\mathcal{T}_i(k)} \), the \( k \)-th local decoder can be seen as an SCD with input \( \alpha[V_{\lambda(i,k)}^{(j(i,k),1)}] \).

If \( \lambda(i,k) > 0 \), \( \hat{u}_N[l] \), is the \( (i-\min(\mathcal{T}_i(k))+2\lambda(i,k)+1) \)-th bit in \( \hat{u}_N[l]^{\min(\mathcal{T}_i(k))-1} \). Meanwhile, the previous analysis has proved that \( V_{\lambda(i,k)}^{(j(i,k),1)} \in \mathcal{G}(V_0^{(i)}) \). Based on this two facts, we have

\[
\begin{align*}
(29) & \quad w(g_{2\lambda(i,k)}^{(i-\min(\mathcal{T}_i(k))+2\lambda(i,k)+1)}) \overset{(a)}{=} w(g_{N}^{(i)}(\lambda(i,k))) \overset{(b)}{=} w(g_{N}^{(i)})/2^{\mathcal{P}_1[b_n(i-1)]_{\alpha(i,k)+1}} \overset{(c)}{=} w(\beta[V_{\lambda(i,k)}^{(j(i,k),1)}]) \overset{(d)}{=} w(\hat{u}_N[l]^{\min(\mathcal{T}_i(k))-1})_{\alpha(i,k)+1} = 0
\end{align*}
\]

where the step (a) is based on the structure of \( G_N, \) step (b) comes from **Lemma 14** based on the fact of \( \lambda \), step (c) is obtained by **Lemma 13** and step (d) is based on equation (19).

Moreover, since \( \hat{u}_N[l]_T \in \mathcal{U}_{N,m}^{(i)}, \) it follows that

\[
\hat{u}_N[l]^{i-1} = 0_{i-\min(\mathcal{T}_i(k))+2\lambda(i,k)} \quad (30)
\]

Combining (29) and (30), we have \( \hat{u}_N[l]^{\min(\mathcal{T}_i(k))-1} \in \mathcal{U}_{2\lambda(i,k),m}^{(i-\min(\mathcal{T}_i(k))+2\lambda(i,k)+1)}. \) Recalling **Lemma 15**, the location set of zero-valued decoding LLRs under the \( k \)-th local decoder is

\[
\mathcal{P}_1[g_{2\lambda(i,k)}^{(i-\min(\mathcal{T}_i(k))+2\lambda(i,k)+1)}] = \mathcal{P}_1[g_{N}^{(i)}(\lambda(i,k))] = \mathcal{P}_1[g_{N}^{(i)}(\mathcal{P}_0[b_n(i-1)])_{k-1}] \quad (31)
\]

Based on the definition of ZCBC-\( i \), the theorem is true in the \( k \)-th local decoder.

In the case of \( \lambda(i,k) = 0 \), we have \( k = 1 \). It follows that \( j(i,k) = i+1 \). Meanwhile, in this case, it can be easily obtained that \( \mathcal{T}_i^0(k) = \{i+1\} \). From (21) we can directly derive: \( \alpha[V_0^{(i+1)}] = 0 \). Thus, this theorem is also true in the case of \( \lambda(i,k) = 0 \).

In any other local decoder, this conclusion can be derived by the same method. Overall, the location set of 0-valued decoding LLRs generated at the whole decoding process is \( \mathcal{T}_i^0 \). The theorem follows.

---

2It is worth noting that SCD can only reserve one path, however, all the paths whose RDS is none can be regarded as the valid output of it. When none of the decoding LLR is 0, there is only one valid output.
Theorem 17: For polar codes with generator matrix $G_n^A$, the number of PC-MHW is upper bounded by $\sum_{i \in \mathcal{A}_m} 2^{[I^0_i \cap \mathcal{A}]}$, i.e., $|U_{N,m}| \leq \sum_{i \in \mathcal{A}_m} 2^{[I^0_i \cap \mathcal{A}]}$.

Proof: We can first consider the upper bound of $|U_{N,m}^{(i)}|, i \in \mathcal{A}_m$. When searching $U_{N,m}^{(i)}$, at any location $j \in [1, i-1]$, we call a decoding trajectory $\hat{u}_j[l]^j_{i-1}$ is valid if its RSD is $\phi$. Meanwhile, at any $j \in [i, N]$, a trajectory $\hat{u}_j[l]^j_{i}$ is valid if its RSD is $\{i\}$. Based on Theorem 7-8, when all the $N$ bits are decoded by SCLD, the set of valid trajectory is just $U_{N,m}^{(i)}$. Therefore, this theorem can be proved by focusing on how the number of valid trajectories changes at each location.

For any location $j \in [1, i-1]$, only one decoding trajectory, i.e., $0_j$, is valid. Moreover, at location $i$, the valid trajectory is just $[0_{i-1}, 1]$. When $j \in [i+1, N]$, for any input valid trajectory, e.g., $\hat{u}_{j-1}[l]_1^{j-1}$, we can consider the following 4 cases

1) If $j \in [i+1, N] \cap \mathcal{A}$ and the decoding LLR of $u_j$ is zero, then $\hat{u}_{j-1}[l]_1^{j-1}$ will be split into two valid trajectories, i.e., $[\hat{u}_{j-1}[l]_1^{j-1}, 0]$ and $[\hat{u}_{j-1}[l]_1^{j-1}, 1]$.

2) If $j \in [i+1, N] \cap \mathcal{A}$ and the decoding LLR of $u_j$ is non-zero, then $\hat{u}_{j-1}[l]_1^{j-1}$ can educe only one valid path.

3) If $j \in [i+1, N] \cap \mathcal{A}^c$ and the decoding LLR of $u_j$ is negative, then $\hat{u}_{j-1}[l]_1^{j-1}$ can not educe any valid path in all the subsequent decoding steps in that its RDS will become $\{i, j\}$.

4) If $j \in [i+1, N] \cap \mathcal{A}^c$ and the decoding LLR of $u_j$ is non-negative, then $\hat{u}_{j-1}[l]_1^{j-1}$ will educe only one valid path, i.e., $[\hat{u}_{j-1}[l]_1^{j-1}, 0]$.

In Theorem 16 we proved that the location set of zero-valued decoding LLRs of any searching path in $U_{N,m}^{(i)}$ should be fixed to $\mathcal{I}^0_i$. Thus, for the 4 cases listed above, only in case 1), i.e., at any location in $\mathcal{I}^0_i \cap \mathcal{A}$, the number of the valid trajectories should be doubled. Meanwhile, at the other locations such number will stay the same [in case 2) or 4)] or decrease [in case 3)] compared with the last location.

Therefore, we can conclude that $|U_{N,m}^{(i)}| \leq 2^{[I^0_i \cap \mathcal{A}]}$. Recalling that $U_{N,m} = \bigcup_{i \in \mathcal{A}_m} U_{N,m}^{(i)}$, we have

$$|U_{N,m}| = \sum_{i \in \mathcal{A}_m} |U_{N,m}^{(i)}| \leq \sum_{i \in \mathcal{A}_m} 2^{[I^0_i \cap \mathcal{A}]}$$

The theorem follows.

Actually, $\sum_{i \in \mathcal{A}_m} 2^{[I^0_i \cap \mathcal{A}]}$ is really the exact value of $|U_{N,m}|$ in most cases.

E. Complexity

The complexity of the proposed method to estimate the number of PC-MHW is equal to that of calculating the number of ZCBC-i, for all $i \in \mathcal{A}_m$. For each $i \in \mathcal{A}_m$, the complexity of calculating
ZCBC-i is $O(\log_2 N)$. Thus, the complexity of the proposed estimation is calculated as

$$O(\sum_{i \in A_m} 2^{|T_i \cap A|}) \leq O(|A_m| \log_2 N) \quad (32)$$

Unlike the methods in [12] and [13], the complexity is not much affected by $N$.

V. EFFICIENT ENUMERATOR STRATEGY FOR PC-MHW

Actually, in previous part we not only provided the upper bound of the number of PC-MHW, but also given that for each $|U_{N,m}^{(i)}|$, with $i \in A_m$. In this part, we will propose an efficient strategy to search specific PC-MHW. We can array $A_m$ and $I_0^i$ according to the ascending order, i.e., $A_m = \{A_{m,1}, A_{m,2}, ..., A_{m,|A_m|}\}$ and $T_i^0 = \{T_i^0, T_i^1, ..., T_i^{|T_i|}\}$. Based on this, we will propose a double multi-level SCLD-based searching strategy which divides $U_{N,m}^{(i)}$ into $|T_i^0 \cap A| + 1$ parts to search, i.e.,

$$U_{N,m}^{(i)} - \{g_N^{(i)}\} = \bigcup_{j \in T_i^0 \cap A} U_{N,m}^{(i,j)} \quad (33)$$

where $U_{N,m}^{(i,j)}$ is the set of vector $u_N^1$ which satisfies the following two conditions:

1) $u_{i-1}^1 = 0, u_i = 1, u_{i+1}^j = 0, u_j = 1, \quad u_{A_j} \in B^{A_j}, \quad u_{A_c} = 0$,

2) $w(u_N^1 G_N) = w(g_N^{(i)})$.

where $A_j = A \cap [j+1, N]$ and $A_c = A \cap [j+1, N]$. Note that when $j = i+1$, $u_{i+1}^j$ is null vector.

This strategy can further reduce the list size, so as to reduce the memory space and the sorting complexity required for searching. In Algorithm 1, we elaborate the process of this strategy. We can find that in the proposed searching strategy, the maximum required list size is $2^{|T_i^0 \cap A| - 1}$, which is less than half of that required for method in [11] ($|U_{N,m}^{(i)}|$) and the method in [10] ($|U_{N,m}| + 1$). It is worth noting that the methods in [11] and [10] cannot estimate the number of PC-MHW, in practice the used list size should be set larger than the actually required value.

VI. SIMULATION RESULTS

A. Accuracy of Estimation

Actually, we can directly take the upper bound, i.e., $\sum_{i \in A_m} 2^{|T_i \cap A|}$, as the estimated value of the number of PC-MHW. In Fig.2-3, we compare this estimated value between the exact value $|U_{N,m}|$ which comes from exhaustive searching. We offer the fine-granularity simulation for 99 code rates that arrange from 0.01 to 0.99 with 0.01 step. In Fig.2, the polar codes are constructed by gaussian approximation (GA) algorithm [16] and the constructing $E_b/N_0$ is set 0
Algorithm 1: Double Multi-level SCLD-based Enumerator for PC-MHW

Input: \(y_1^N = 1_N, A_m\)

Output: \(U_{N,m}\)

for \(k = 1, 2, \cdots, |A_m|\) do

\(i \leftarrow A_m,k\)

Obtain \(I_0^i\) by Definition 12;

\(cnt \leftarrow 0;\)

\(U_t \leftarrow \phi; \quad \text{// A temporary set to reserve searched paths}\)

for \(j = 1, 2, \cdots, |I_0^i|\) do

if \(I_0^i,j \in A\) then

\(cnt \leftarrow cnt + 1;\)

Using SCLD to obtain \(U_{N,m}^{(i, I_0^i,j)}\) with list size \(L = 2^{|I_0^i \cap A| - cnt}\) under 2 conditions:

1. \(y_1^N = 1_N;\)
2. \([\hat{u}_{i-1} = 0, i-1, \hat{u}_i = 1, \hat{u}_{i+1} = 0, i-1, \hat{u}_{I_0^i,j} = 1, \hat{u}_{I_0^i,j} = 1].\)

\(U_t = U_t \cup U_{N,m}^{(i, I_0^i,j)};\)

\(U_{N,m}^{(i)} = U_t \cup \{S_N^{(i)}\};\)

\(U_{N,m} = \bigcup_{i \in A_m} U_{N,m}^{(i)};\)

Return \(U_{N,m};\)

Fig. 2. Number of PC-MHW obtained by exhaustive searching and the proposed evaluating method when using GA algorithm [16] for construction.

dB and 2 dB. The code length 1024 and 256 are considered. We can find that for almost all the rates expect some high ones, the evaluated value equals the exact one. In Fig.3, the constructing method is changed to polarization weight (PW) algorithm [17]. We can find that for all the rates
Fig. 3. Number of PC-MHW obtained by exhaustive searching and the proposed evaluating method when using PW algorithm [17] for construction.

| Scheme | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 128, [12]| 1.2 | 3.1 | 2.3 | 4.7 | 6.8 | 8.4 | 13.7 | 15.4 | 20.7 |
| 128, [13]| 0   | 0   | 0.2 | 1.2 | 0   | 0   | 0   | 0   | 0   |
| 128.P. | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 256, [12]| 2.3 | 3.1 | 5.2 | 7.1 | 12.7| 18.4| 29.2| 38.4| 48.1 |
| 256, [13]| 2.1 | 0.8 | 0.1 | 0.5 | 0.1 | 0   | 0   | 0   | 0   |
| 256.P. | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

the evaluated value is equal to the exact one, irrespective of the code length.

In Table I, we also compare the proposed estimation, i.e., \( \sum_{i \in A} 2^{|I_0 \cap A|} \), with those proposed in [12] [13]. We consider the code length of 128 and 256. For each length, we adopt 9 code rates (from 0.1 to 0.9 with step 0.1). The estimated number of PC-MHW is denoted as \( \hat{|U_{N,m}|} \). The value for each scheme is calculated by \( |\ln(|\hat{U_{N,m}}|/|U_{N,m}|)| \). The polar codes are designed by GA algorithm with \( \sigma^2=0.6309 \). It can be find that the proposed algorithm (denoted by P.) can evaluate the exact value in the entire rate range and is obviously more accurate than the both existing methods.
Fig. 4. The FER and FER bounds give in (35) of polar code with $N = \{256, 512\}$, code rate $R = 0.3$ and list size $L = 8$.  

Fig. 5. Comparison of the actual performance of SCLD with the bound given in (35) on the order of $10^{-3}$, $10^{-4}$ and $10^{-5}$ FER. The polar code with $N = 1024$ is decoded by SCLD with $L = 8$.

B. Estimation for Performance of SCLD

When polar codewords are transmitted through AWGN channel, it can approach the ML performance under SCLD. Thus, the performance can be upper bounded by union bound \[8\]

$$P_{un} \leq \sum_d A_d Q\left(\frac{\sqrt{d}}{\sigma}\right)$$ \hspace{1cm} (34)

where $A_d$ is the number of polar codewords with weight $d$. At high SNR, the ML performance is dominated by the item with MHW. Based on Theorem 17, at high SNR, the frame error rate (FER) of polar codes under SCLD can be estimated by

$$P_{SCL} \approx A_{d_m} Q\left(\frac{\sqrt{d_m}}{\sigma}\right) \leq \sum_{i \in A_m} 2^{\left| I_0^i \cap A \right|} \cdot Q\left(\frac{\sqrt{d_m}}{\sigma}\right)$$ \hspace{1cm} (35)

In Fig.4, we give the comparison between the actual performance of SCLD with the proposed performance bound given in (35). The considered code lengths are in $\{256, 512\}$. For each code
C. Comparison with Existing Methods

In Table II, we compare the proposed enumeration (including PC-MHW and its number) with the existing methods. Among all the methods for evaluating $|U_{N,m}|$, the complexity of the proposed method is minimum. Moreover, it is suitable for all the code rates. As for enumerating PC-MHW, [12] and [13] do not provide solution. The proposed method needs less than half of the list size required in [10] and [11].

VII. Conclusion

In this paper, we proposed an efficient method to enumerate the PC-MHW and its number. First, we revealed how the PC-MHW can be enumerated by the existing multi-level SCLD-based schemes [11], with $y_1^N=1_N$, and obtained the necessary and sufficient condition for a searched path $\hat{u}_N[i_1]^N$ in $U_{N,m}^{(i)}$, that is, its RDS equals to $\{i\}$. Subsequently, we introduced a concept of ZCBC-i which will be used to given an upper bound for $|U_{N,m}^{(i)}|$ and further to derive the upper bound of the number of PC-MHW, i.e., $|\sum_{i\in A_m}U_{N,m}^{(i)}|\leq\sum_{i\in A_m}2^{2^{|\mathbb{Z}^{(i)}\cap A|-1}}$. Guided by the previous analysis, we proposed a double multi-level SCLD-based searching strategy to enumerate all the
PC-MHW, which can further divide $U_{N,m}^{(i)}$ into several subsets to search. The maximum required list size for the proposed searching strategy is much less than that for the existing methods, so as to reduce the complexity and memory space required for searching.

**Appendix A**

**Proof of Theorem 7**: We first consider the case of $u_N[l]^N=g_N^{(i)}\in U_{N,m}^{(i)}$. Based on the hard decision rule given in (12), proving $\mathcal{A}_N[l]=\{i\}$ is equivalently to proving for any $j=1,2,...,N$, $L_N[l]_{j}\geq 0$ and $L_N[l]_{i}>0$. Note that when decoding LLR is 0, the corresponding decoding bit, whether decoded to be 0 or 1, can be seen to be decided based on the decoding LLR.

Based on the condition: $y_1^N=1_N$ and $\hat{u}_N[l]_{j}^0=0_{i-1}, L_N[l]_{i}>0$ can be easily proved. Then, we will use contradiction method to prove that none of the decoding LLRs of $\hat{u}_N[l]_{j}^N$ is negative.

We assume that $\mathcal{P}_{nLLR}$ is the set of positions at which the decoding LLR of $\hat{u}_N[l]_{j}^N$ is negative. We will determine $\mathcal{P}_{nLLR}$ by retrace the searching process of $\hat{u}_N[l]_{j}^N$. Actually, we only need to discuss the existence of the first position where the negative decoding LLR appears, i.e., the minimum value in $\mathcal{P}_{nLLR}$, denoted by $j_0$. If the conclusion of the retracing is that $j_0$ does not exist, then $\mathcal{P}_{nLLR}=\phi$, and it follows that $\mathcal{A}_N[l]=\{i\}$. Since $\hat{u}_N[l]_{i}=1$ is the fundamental reason to cause $L_N[l]_{j_0}<0$, we can easily obtain that $j_0>i$.

Now we can use RSCLD to explore how these negative LLRs were generated in the searching. On the code tree, to obtain $\alpha[V_0^{(jo)}]<0$, the LLR vector of all the GAN of $V_0^{(jo)}$ should be calculated in advance (from Property 1). Thus, at least one of the GAN of $V_0^{(jo)}$ has negative LLR in its LLR vector. We can write $\mathcal{G}[V_0^{(jo)}]$ as follows

$$\mathcal{G}[V_0^{(jo)}] = \{V_0^{(jo)}, V_1^{(j1)}, ..., V_N^{(jN)}\} \quad (36)$$

In $\mathcal{G}[V_0^{(jo)}]$, we assume that $V_f^{(jf)}$ is the first node whose LLR vector has negative LLR in the searching process, with $0\leq f<N$. Since the LLR message is updated from the root node to the leaf node, this means that $\alpha[V_f^{(jf)}]$ has negative LLR and for any $k$ that satisfies $f<k\leq N$, $\alpha[V_k^{(jk)}]$ has no negative element. Based on this assumption, the LLR vector of the parent node of $V_f^{(jf)}$, i.e., $\alpha[V_{f+1}^{(jf+1/2)}]$, has no negative element. This implies that the negative LLR in $\alpha[V_f^{(jf)}]$ can not be introduced by (3). In other words, $\alpha[V_f^{(jf)}]$ can only be calculated by (4) in which $\alpha[V_{f+1}^{(jf+1/2)}]$ and $\beta[V_f^{(jf-1)}]$ will be used. Concretely,

$$\alpha[V_f^{(jf)}]_{i}=(1-2\beta[V_{f+1}^{(jf-1)}]_{i})\alpha[V_{f+1}^{(jf+1/2)}]_{i}+\alpha[V_{f+1}^{(jf+1/2)}]_{i+2f} \quad (37)$$
Thus, $\alpha[V_{f+1}^{(j_f/2)}]$ has no negative element, a necessary condition of generating negative LLR in $\alpha[V_f^{(j_f)}]$ should be

$$\beta[V_f^{(j_f-1)}] \neq 0_2^j.$$  \hspace{1cm} (38)

Since $\hat{u}_N[l]_1^N=[0_{i-1}, 1, 0_{N-1}]$, to satisfy (38), $\beta[V_0^{(i)}]=\hat{u}_N[l]_i$ should participate in the calculation of $\beta[V_f^{(j_f-1)}]$ by using (7). From Property 2, $V_f^{(j_f-1)}$ is the GAN of $V_0^{(i)}$. Further, since the parent node $V_{f+1}^{(j_f/2)}$ is the GAN of $V_f^{(j_f-1)}$ (Property 3), according to Property 4 we can obtain that

$$V_{f+1}^{(j_f/2)} \in G[V_0^{(i)}].$$ \hspace{1cm} (39)

This means that $\alpha[V_{f+1}^{(j_f/2)}]$ is already updated to obtain $\alpha[V_0^{(i)}]$. Thus, $\alpha[V_{f+1}^{(j_f/2)}]$ is calculated at the first decoding phase. From Lemma 4, all the LLRs in $\alpha[V_{f+1}^{(j_f/2)}]$ should be positive and identical. Using (37), for any $i \in [1, 2^f]$, we have

$$\alpha[V_f^{(j_f)}]_i = \begin{cases} 
0, & \text{if } \beta[V_f^{(j_f-1)}]_i = 1 \\
2\alpha[V_{f+1}^{(j_f/2)}]_i, & \text{if } \beta[V_f^{(j_f-1)}]_i = 0
\end{cases} \hspace{1cm} (40)$$

Thus, $\alpha[V_f^{(j_f)}]$ has no negative LLR. This is contradicted with the assumption. It follows that $j_0$ does not exist and $P_{nLLR} = \phi$. Therefore, for searched path $\hat{u}_N[l]_1^N=[0_{i-1}, 1, 0_{N-1}]$, we can obtain that the all its decoding LLRs are non-negative and $\alpha[V_0^{(i)}] > 0$. It indicates that $\mathcal{A}_N[l] = \{i\}$. Further, using (8), we can obtain that the PM of such searched path is $|\alpha[V_0^{(i)}]|$.

Next, let us consider the case of $\hat{u}_N[l]_1^N \in \mathcal{U}_{N,m}^{(i)}$. Since $w(\hat{u}_N[l]_1^N)G_N = w(g_N^{(i)})$, from Theorem 6, the PM of $\hat{u}_N[l]_1^N$ is equal to that of the searched path $[0_{i-1}, 1, 0_{N-1}]$. Recalling we already obtained that the PM of the searched path $[0_{i-1}, 1, 0_{N-1}]$ is $|\alpha[V_0^{(i)}]|$ in the above, then based on (8) we have

$$PM[l]_N = \sum_{j \in \mathcal{A}_N[l]} |\alpha[V_0^{(j)}]| = |\alpha[V_0^{(i)}]|.$$ \hspace{1cm} (41)

Meanwhile, from Lemma 5, $\hat{u}_N[l]_1^N$ has $\mathcal{A}_N[l] \supseteq \{i\}$. Thus, $\mathcal{A}_N[l]$ can only be $\{i\}$.

According to the proof of the two cases, the theorem is true for any searched path $\hat{u}_N[l]_1^N \in \mathcal{U}_{N,m}^{(i)}$.

**APPENDIX B**

**Proof of Lemma 15:** Decoding $y_1^N$ is equivalent to decoding a punctured polar code [18]. On the code tree, each stage has $N$ LLRs, even if such LLRs may belong to different nodes. For the $\lambda$-th stage, $\lambda \in [0, n-1]$, we can array them in one single vector, i.e.,

$$\Gamma_\lambda = [\alpha[V^{(1)}], \alpha[V^{(2)}], ..., \alpha[V^{(2^n-\lambda)}]].$$ \hspace{1cm} (42)
Note that we only care about the locations of the zero-valued LLRs at each stage, i.e., \( \mathcal{P}_0[\Gamma_\lambda] \).

This lemma can be proved by mathematical induction. We first consider the \( n \)-th stage. LLR vector \( \alpha[V_n^{(1)}] \) is divided into \( 2^{n-1} \) combinations to update the \( N \) LLRs at the \( n-1 \) stage (i.e., \( \alpha[V_{n-1}^{(1)}] \) and \( \alpha[V_{n-1}^{(2)}] \)). The combination is expressed as \((\alpha[V_n^{(1)}]_j, \alpha[V_n^{(1)}]_{j+2^{n-1}})\), \( j \in [1, 2^{n-1}] \), and its corresponding output LLR combination at the \( n-1 \)-th stage can be expressed as \((\alpha[V_{n-1}^{(1)}]_j, \alpha[V_{n-1}^{(2)}]_j)\). From [18], in SCD, we can obtain the following relations:

\[
\begin{align*}
(\alpha[V_n^{(1)}]_j=0, \alpha[V_n^{(1)}]_{j+2^{n-1}}=0) & \Rightarrow (\alpha[V_{n-1}^{(1)}]_j=0, \alpha[V_{n-1}^{(2)}]_j=0) \\
(\alpha[V_n^{(1)}]_j \neq 0, \alpha[V_n^{(1)}]_{j+2^{n-1}}=0) & \Rightarrow (\alpha[V_{n-1}^{(1)}]_j=0, \alpha[V_{n-1}^{(2)}]_j \neq 0) \\
(\alpha[V_n^{(1)}]_j=0, \alpha[V_n^{(1)}]_{j+2^{n-1}} \neq 0) & \Rightarrow (\alpha[V_{n-1}^{(1)}]_j \neq 0, \alpha[V_{n-1}^{(2)}]_j=0) \\
(\alpha[V_n^{(1)}]_j \neq 0, \alpha[V_n^{(1)}]_{j+2^{n-1}} \neq 0) & \Rightarrow (\alpha[V_{n-1}^{(1)}]_j \neq 0, \alpha[V_{n-1}^{(2)}]_j \neq 0)
\end{align*}
\]  

where the notation \( \Rightarrow \) means generating by (3) and (4).

Let \( c_1^N = u_1^N G_N \) and it can be expressed as

\[
c_1^N = \left[u_{1/2}^N G_{N/2} \oplus u_{N/2+1}^N G_{N/2}, u_{N/2+1}^N G_{N/2}\right].
\]  

Then, we have

\[
w(c_1^N) = w(u_{1/2}^N G_{N/2}) + 2w(c_{N/2+1}^N) - 2v,
\]  

where \( v \) is the number of the bits in \( u_{1/2}^N G_{N/2} \) and \( c_{N/2+1}^N \) equal to 1 at the same position. Obviously, one has

\[
y_1^N = a \cdot (1_N - c_1^N) = \alpha[V_{n-1}^{(1)}]
\]  

If \( i > 2^{n-1} \) (or \( b_{i-1,n-1} = 1 \)), then \( u_{1/2}^N G_{N/2} = 0 \). Using (44), we have

\[
c_{2^{n-1}}^N = c_{1+2^{n-1}}^N.
\]  

Substituting (47) into (46), we can obtain that in vector \( \alpha[V_{n}^{(1)}] \) there can only exist the combination of \((\alpha[V_{n}^{(1)}]_j=0, \alpha[V_{n}^{(1)}]_{j+2^{n-1}}=0)\), and \((\alpha[V_{n}^{(1)}]_j > 0, \alpha[V_{n}^{(1)}]_{j+2^{n-1}}=0)\) or \((\alpha[V_{n}^{(1)}]_j=0, \alpha[V_{n}^{(1)}]_{j+2^{n-1}} = 0)\) does not exist. In this case, from (43), \( \mathcal{P}_0[\Gamma_{n-1}] \) can be obtained by:

\[
\mathcal{P}_0[\Gamma_{n-1}] = \mathcal{P}_0[\alpha[V_{n}^{(1)}]] = \mathcal{P}_1[c_1^N]
\]  

Otherwise, we consider the case of \( i \leq 2^{n-1} \) (or \( b_{i-1,n} = 0 \)). Since \( u_1^N \in \mathcal{U}_{N,m}^{(i)} \), it follows that \( w(c_1^N) = w(g_1^{(i)}) \). Meanwhile, from Lemma 2 in [11], we have

\[
w(u_{1/2}^N G_{N/2}) \geq w(g_1^{(i)}) = w(c_1^N).
\]
If \( w(u_1^{N/2}G_{N/2}) \geq \frac{w(c_1^N)}{2} \), (45) implies that \( w(c_{N/2+1}^N) < v \). This contradicts with the definition of \( v \). Therefore, it can only be \( w(u_1^{N/2}G_{N/2}) = w(c_1^N) \). From (45), we can further obtain that \( w(c_{N/2+1}^N) = v \). This means that for any \( j \in [1, 2^{n-1}] \), \( c_j \) and \( c_{j+2^{n-1}} \) can not be 1 simultaneously and the combination \( \alpha[V_n^{(1)}]_j = 0, \alpha[V_n^{(1)}]_{j+2^{n-1}} = 0 \) does not exist. This lead to

\[
P_0[\Gamma_{n-1}] = P_1[c_1^N] \cup P_1[c_{N/2+1}^N].
\]  

(50)

Then, at the \( n-1 \)-th stage, the code tree can be divided into two subcode-trees. The first one is rooted at \( V_n^{(1)} \) and the second one is rooted at \( V_n^{(2)} \). From the above, we can obtain that if \( b_{i-1,n} = 1 \), it has \( P_0[\alpha[V_{n-1}^{(1)}]] = P_0[\alpha[V_{n-1}^{(2)}]] \). Since the two subcode-trees performs independently, in their respective following decoding stage, the set of the locations of zero-valued LLRs will be identical with each other. Meanwhile, if \( b_{i-1,n} = 0 \), the second subcode-tree can not generate any 0-valued LLR in its descendant nodes.

From the above analysis, for any stage \( \lambda \in [0, n-1] \), we can conclude that

\[
P_0[\Gamma_{\lambda}] = \begin{cases} P_0[\Gamma_{\lambda}^+], & b_{i-1,n} = 1 \\ \phi, & b_{i-1,n} = 0 \end{cases}
\]  

(51)

where \( \Gamma_{\lambda}^+ \) and \( \Gamma_{\lambda}^- \) are the subvectors of \( \Gamma_{\lambda} \) with its first and last \( \frac{N}{2} \) elements, respectively. That’s to say, to obtain \( P_0[\Gamma_{n-2}] \), it is enough to focus on the first subcode at the \( n-1 \) stage, but instead of considering the both subcodes.

To facilitate induction, based on (48) and (50), we can write \( P_0[\Gamma_{n-1}] \) as:

\[
P_0[\Gamma_{n-1}] = \begin{cases} P_1[u_1^N G_N], & b_{i-1,n} = 1 \\ P_1[[u_1^{N-2^{n-1}}, 0_{2^{n-1}}] G_N], & b_{i-1,n} = 0 \end{cases}
\]  

(52)

Similarly, for the \( \lambda \)-th stage, \( \lambda \in [1, n-2] \), we assume that

\[
P_0[\Gamma_{\lambda}] = \begin{cases} P_1[u_1^N G_N], & [b_{i-1,\lambda+1}, ..., b_{i-1,n}] = 1_{n-\lambda} \\ P_1[[u_1^{N-k_{\lambda}}, 0_{k_{\lambda}}] G_N], & \text{otherwise} \end{cases}
\]  

(53)

where \( k_{\lambda} = \sum_{j=\lambda+1}^{n} (1-b_{i-1,j})2^{j-1} \).

To calculate \( P_0[\Gamma_{\lambda-1}] \), we can also divide the code-tree nodes from stage 0 to stage \( \lambda \) into \( 2^{n-\lambda} \) subcode-trees. From (51), some of the LLR vectors of the \( 2^{n-\lambda} \) root nodes have no zero-valued LLR and thus their descendant nodes also have no 0-valued LLR. We can determine such root nodes to simplify the computation. Before that, we first define \( S_n^i \) as the set of integer \( i \in [0, n-1] \) whose binary expansion satisfies \( b_{i,j} = 1 \), with \( j \in [1, n-1] \). Obviously, \( |S_n^i| = 2^{n-1} \).
Example 5: \( S_1^1 = \{1\}, S_1^2 = \{1, 3\}, S_2^3 = \{2, 3\}, S_3^3 = \{1, 3, 5, 7\}, S_3^2 = \{2, 3, 6, 7\}, S_3^3 = \{4, 5, 6, 7\} \).

Using equation (51) recursively from stage \( n - 1 \) to stage \( \lambda \), for any \( k \in [0, 2^{n-\lambda}-1] \) we have

\[
\mathcal{P}_0[\alpha[V_\lambda^{(k+1)}]] = \begin{cases} 
\phi, & k \in \mathcal{S}, \\
\mathcal{P}_0[\alpha[V_\lambda^{(1)}]], & \text{otherwise,}
\end{cases}
\]

(54)

where \( \mathcal{S} = \bigcup_{j \in \mathcal{P}_0[b_n(i-1)]} S_{n-\lambda}^j \). Note that if \( \mathcal{P}_0[b_n(i-1)] = \phi \), then \( \mathcal{S} = \phi \).

Obviously, since \( 0 \notin \bigcup_{j=1}^{n-\lambda} S_{n-\lambda}^j \), then \( \mathcal{P}_0[\alpha[V_\lambda^{(1)}]] \) is bound to a non-empty set. That’s to say, for each subcode-tree divided at the \( \lambda \)-th stage, the location set of zero-valued LLR in its input vector can only be null set or equal to \( \mathcal{P}_0[\alpha[V_\lambda^{(1)}]] \).

Actually, to obtain \( \mathcal{P}_0[\Gamma_{\lambda-1}] \), we do not have to calculate \( \mathcal{P}_0[\alpha[V_\lambda^{(1)}]] \). Instead, we just need to calculate in any subcode-tree (divided at stage \( \lambda \)) whose LLR vector of its root node has 0-valued LLR.

From (53), if \( [b_{i-1, \lambda+1}, \ldots, b_{i-1,n}] = 1_{n-\lambda} \), then \( k_\lambda = 0 \), and thus \( u_1^N = [0_{N-2 \lambda}, u_{N-2 \lambda+1}^N] \), where \( u_{N-2 \lambda+1}^N \neq 0_{2 \lambda} \); if \( [b_{i-1, \lambda+1}, \ldots, b_{i-1,n}] \neq 1_{n-\lambda} \), then \( [u_1^{N-k_\lambda}, 0_{k_\lambda}] = [0_{N-k_\lambda-2 \lambda}, u_{N-k_\lambda-2 \lambda+1}^N \neq 0_{2 \lambda}, 0_{k_\lambda}] \).

Based on this, for the subcode-tree rooted at \( V_\lambda^{(N-k_\lambda/2^{\lambda})} \) (whose leaf nodes are \( V_0^{(N-k_\lambda-2 \lambda+1)}, V_0^{(N-k_\lambda-2 \lambda+2)}, \ldots, V_0^{(N-k_\lambda)} \)), we can obtain the following equation

\[
\mathcal{P}_0[\alpha[V_\lambda^{(N-k_\lambda/2^{\lambda})}]] = \mathcal{P}_1[u_{N-k_\lambda-2 \lambda+1}^N G_{2 \lambda}]
\]

(55)

It can be easily obtained that \( N-k_\lambda \geq i = N-k_0 \geq N-k_\lambda-2 \lambda+1 \). From \( u_i = 1 \), one has \( \mathcal{P}_0[\alpha[V_\lambda^{(N-k_\lambda/2^{\lambda})}]] \neq \phi \). Thus, (54) can be rewritten as

\[
\mathcal{P}_0[\alpha[V_\lambda^{(k+1)}]] = \begin{cases} 
\phi, & k \in \mathcal{S}, \\
\mathcal{P}_0[\alpha[V_\lambda^{(N-k_\lambda/2^{\lambda})}]], & k \in [0, 2^{n-\lambda}-1] - \mathcal{S},
\end{cases}
\]

(56)

Since \( \alpha[V_\lambda^{(N-k_\lambda/2^{\lambda})}] \) generates \( \alpha[V_{\lambda-1}^{(N-k_\lambda/2^{\lambda})-1}], \alpha[V_{\lambda-1}^{(N-k_\lambda/2^{\lambda})+1}] \), (56) implies that we can determine \( \mathcal{P}_0[\Gamma_{\lambda-1}] \) by calculating \( \mathcal{P}_0[\alpha[V_{\lambda-1}^{(N-k_\lambda/2^{\lambda})+1}]] \) first.

Note that \( u_i \) is the \( (i - N + k_\lambda + 2 \lambda) \)-th bit in \( u_{N-k_\lambda-2 \lambda+1}^N \). From Property 2, we have \( V_{\lambda}^{(N-k_\lambda/2^{\lambda})} \in \mathcal{G}[V_0^{(i)}] \). Meanwhile, since \( u_i \in U_{N,m}^{(i)} \), we can obtain the following equations

\[
w(u_{N-k_\lambda-2 \lambda+1}^N G_{2 \lambda}) = w(\beta[V_{\lambda}^{(N-k_\lambda/2^{\lambda})}]) = \frac{w({g}_N^{(i)})}{2^{|\mathcal{P}_1[b_n(i-1)]_{\lambda+1}|}}
\]

(57)
where step (a) is obtained by Lemma 13, step (b) is based on Lemma 14 and step (c) is based on the structure of $G_N$. Thus, we can obtain that $u_{N-k\lambda}^{N-k\lambda} \in \mathcal{U}_{2\lambda,m}^{i-N+k\lambda+2\lambda}$. Under this condition, we can refer to the method of calculating $\mathcal{P}_0[\Gamma_{n-1}]$ and obtain the following equation

$$
\mathcal{P}_0[\alpha[V_{\lambda-1}^{\frac{N-k\lambda}{2\lambda}-1}], \alpha[V_{\lambda-1}^{\frac{N-k\lambda}{2\lambda}+1}]] = \begin{cases} 
\mathcal{P}_1[u_{i}^{N-k\lambda}G_N], & i-N+k\lambda+2\lambda \geq 2\lambda-1 \\
\mathcal{P}_1[[u_{i}^{N-k\lambda-2\lambda+1}, 0_{2\lambda-1}]G_{2\lambda}], & i-N+k\lambda+2\lambda < 2\lambda-1 
\end{cases}
$$

Further, using the fact that $i=N-k_0$, it can be easily verified that

$$i-N+k\lambda+2\lambda \geq 2\lambda-1 \iff b_{i-1,\lambda}=1, \quad i-N+k\lambda+2\lambda < 2\lambda-1 \iff b_{i-1,\lambda}=0
$$

From (56) and (58), we can further obtain that

$$
\mathcal{P}_0[\Gamma_{n-1}]= \begin{cases} 
\mathcal{P}_1[u_{i}^N G_N], & [b_{i-1,\lambda}, \ldots, b_{i-1,n}]=1_{n-\lambda+1} \\
\mathcal{P}_1[[u_{i}^{N-k\lambda-1}, 0_{k\lambda-1}]G_N], & \text{otherwise}
\end{cases}
$$

By simply induction, at stage 0 we have

$$
\mathcal{P}_0[\Gamma_0]= \begin{cases} 
\mathcal{P}_1[u_{i}^N G_N]=\mathcal{P}_1[g_N^{(N)}], & [b_{i-1,1}, \ldots, b_{i-1,n}]=1_n \\
\mathcal{P}_1[[u_{i}^{N-k_0}, 0_{k_0}]G_N], & \text{otherwise}
\end{cases}
$$

Since $i=N-k_0$ and $u_{i}^N \in \mathcal{U}_{N,m}^{i(i)}$, then $\mathcal{P}_0[\Gamma_0] = \mathcal{P}_1[g_N^{(i)}]$. The lemma is proved.

REFERENCES

[1] E. Arzian, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” IEEE Trans. Inf. Theory, vol. 55, no. 7, pp. 3051-3073, Jul. 2009.

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

[3] A. Balatsoukas-Stimming, M. B. Parizi and A. Burg, “LLR-based successive cancellation list decoding of polar codes,” IEEE Trans. Signal Process., vol. 63, no. 19, pp. 5165-5179, October 2015.

[4] S. A. Hashemi, C. Condo, and W. J. Gross, “Fast simplified successive-cancellation list decoding of polar codes,” in 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Mar. 2017, pp. 1-6.

[5] S. A. Hashemi, C. Condo, and W. J. Gross, “Fast and flexible successive-cancellation list decoders for polar codes,” IEEE Trans. Sign. Proc., vol. 65, no. 21, pp. 5756-5769, Nov. 2017.

[6] K. Niu and K. Chen, “CRC-aided decoding of polar codes,” IEEE Commun. Lett, vol. 16, no. 10, pp. 1668-1671, 2012.

[7] 3rd Generation Partnership Project (3GPP) TS 38.212, “Multiplexing and channel coding,” V.15.1.0, 2018.

[8] X. Ma, J. Liu and B. Bai, “New techniques for upper-bounding the ML decoding performance of binary linear codes,” IEEE Trans. Commun., vol. 61, no. 3, pp. 842-851, Mar. 2013.

[9] B. Li, H. Shen, D. Tse, “An Adaptive Successive Cancellation List Decoder for Polar Codes with Cyclic Redundancy Check”, IEEE Commun. Lett., vol.16, no. 12, pp. 2044-2047, Dec. 2012.

[10] Z. Z. Liu, K. chen, J. Niu and Z. Q. He, “Distance spectrum analysis of polar codes,” IEEE WCNC, Apr. 2014, pp. 6-9.
[11] Q. S. Zhang, A. J. Liu, X. F. Pan and K. G. Pan, “CRC code design for list decoding of polar codes,” *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1229-1232, Jun. 2017.

[12] M. Valipour and S. Yousefi, “On probability weight distribution of polar codes,” *IEEE Commun. Lett.*, vol. 17, no. 11, pp. 2120-2123, Nov. 2013.

[13] Q. S. Zhang, A. J. Liu and X. F. Pan, “An enhanced probabilistic computation method for the weight distribution of polar codes,” *IEEE Commun. Lett.*, vol. 21, no. 12, pp. 2562-2565, Dec. 2017.

[14] A. Alamdar-Yazdi and F. R. Kschischang, “A simplified successive-cancellation decoder for polar codes,” *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378-1381, Dec. 2011.

[15] A. Eslami and H. Pishro-Nik, “On finite-length performance of polar codes: stopping sets, error floor and concatenated design,” *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 919-929, Mar. 2013.

[16] P. Trifonov, “Efficient design and decoding of polar codes,” *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221-3227, Nov. 2012.

[17] G. He, J. C. Belfiore, X. Liu, Y. Ge, R. Zhang, I. Land, Y. Chen, R. Li, J. Wang, G. Yang, and T. Wen, “β-expansion: A theoretical framework for fast and recursive construction of polar codes,” in *IEEE GLOBECOM*, Dec. 2017, pp. 1-6.

[18] L. Zhang, Z. Y. Zhang, X. B. Wang, Q. L. Yu and Yan Chen, “On the puncturing patterns for punctured polar codes,” in *IEEE ISIT*, Jul. 2014, pp. 121-125.