List Decoding of Polar Codes: How Large Should the List Be to Achieve ML Decoding?

Arman Fazeli, Alexander Vardy, and **Hanwen Yao**
University of California San Diego, La Jolla, CA 92093, USA
{afazelic,avardy,hwyao}@ucsd.edu

***Abstract*—Successive-cancellation list (SCL) decoding is a widely used and studied decoding algorithm for polar codes. For short blocklengths, empirical evidence shows that SCL decoding with moderate list sizes (say, $L \leqslant 32$) closely matches the performance of maximum-likelihood (ML) decoding. Hashemi *et al.* proved that on the binary erasure channel (BEC), SCL decoding actually coincides with ML decoding for list sizes $L \geqslant 2^\gamma$, where γ is a new parameter we call the *mixing factor*. Loosely speaking, the mixing factor counts the number of information bits mixed-in among the frozen bits; more precisely $\gamma = \left|\left\{i \in \mathcal{F}^c : i \leqslant \max\{\mathcal{F}\}\right\}\right|$, where $\mathcal{F} \subset [n]$ denotes the set of frozen indices.**

Herein, we extend the aforementioned result of Hashemi *et al.* from the BEC to *arbitrary* binary-input memoryless symmetric channels. Our proof is based on capturing all 2^γ decoding paths that correspond to the γ information bits appearing before the last frozen bit, and then finding the most-likely extension for each of these paths efficiently using a *nearest coset decoding* algorithm introduced herein. Furthermore, we present a hybrid successive-cancellation list (H-SCL) decoding algorithm, which is a hybrid between conventional SCL decoding and nearest coset decoding. We believe that the hybrid algorithm can outperform the conventional SCL decoder with lower decoding complexity.

***Index Terms*—coding theory, polar codes, successive-cancellation list decoding, maximum-likelihood decoding, decoding complexity**

I. Introduction

Since their invention by Arıkan in 2008, polar codes have been a subject of intensive research. Successive-cancellation list decoding [12] made polar codes suitable for a wide range of applications in practice. In particular, after nearly a decade of improvements, polar codes have been selected by 3GPP [1] as the coding method of choice for eMBB control channels in the fifth generation (5G) of the wireless technology standard.

Today, the literature contains numerous efficient decoding methods for polar codes, many of which are based on successive-cancellation list decoding [12]. The computation complexity of SCL scales linearly with the list size, denoted by L. For small code-lengths, simulations show that it is possible to use SCL with moderate list sizes, such as $L = 16$ or $L = 32$, to achieve error probabilities nearly as small as that of maximum-likelihood (ML) decoding. However, numerical observations also show that one has to increase the list size as the blocklength increases in order to maintain the same near-optimal performance. The increased decoding complexity associated with larger values of L makes polar codes less attractive for practical purposes. Unfortunately, theory currently falls short of predicting how large the value of L should be, which in turn requires practitioners to derive these numbers experimentally.

The algorithmic implementation of successive-cancellation decoding in [2] allows the decoder to efficiently follow a single decoding path on the polar decoding tree. SCL decoding of polar codes allows one to pursue *multiple paths* on the polar decoding tree. For an (n, k) polar code, there are k layers in the decoding tree where the paths branch off. These layers corresponds to the positions of the information bits in the uncoded length-n vector $\mathbf{u}$. Therefore, by setting the list size as $L = 2^k$, the SCL decoding method explores all 2^k valid polar codewords and becomes equivalent to ML decoding.

Hashemi, Mondelli, Hassani, Urbanke, and Gross [7] proved that for the class of *binary erasure channels* it is possible to retain the optimal performance of ML decoding by setting $L = 2^{k-\tau}$, where τ is the number of information bits that appear *after* the last frozen index in $\mathbf{u}$. For simplicity, let us define the *mixing factor* of the polar code as the number of information bits that appear *before* the last frozen bit in the uncoded vector $\mathbf{u}$ and denote it by γ. We extend this result to *arbitrary binary-input memoyless symmetric* (BMS) channels in the following theorem.

Theorem 1 (Main theorem)**.** *Consider transmission over a binary-input memoryless symmetric channel W using an (n, k) polar code whose mixing factor is γ. Then successive-cancellation list decoding with list size $L = 2^\gamma$ achieves maximum-likelihood decoding.*

In order for the SCL decoding algorithm to achieve ML decoding, we modify the hard decision rule once the decoder reaches the last frozen index. The proposed rule is based on comparing the minimum distances between the received vector and two polar cosets, which allows the decoder to follow the most likely path on the decoding tree from this index onward. The decoding complexity of the decoder in Theorem 1 is bounded by $O(2^\gamma n \log n)$, which can be significantly smaller than $O(2^k n \log n)$ depending on the location of frozen bits. In simulations, we observe that it usually suffices to pick much smaller values of L in order to closely match the performance of the ML decoder. Furthermore, empirical evidence shows that in order to obtain near-ML performance, the required list size should be larger for polar codes with larger mixing factors. This explains why code dimension alone is not enough to predict the required list size.

A. Our contributions

This work may be regarded as an attempt to shed some light on how large the list should be in SCL decoding of polar codes, thereby clarifying why SCL decoding of some polar codes requires larger list sizes than the others.

We define certain polar cosets that correspond to polar codewords that agree with the so-far-decoded bits during the decoding process. We then use the recursive structure of polar codes to efficiently compute the minimum distance between any given vector and any such coset. Based upon these results, a new hard-decision decoding rule is proposed, which differs from the conventional rule used in successive-cancellation (or SCL) decoding. The new rule is based on comparing the minimum distances between the received vector and some of the aforementioned polar cosets. We prove that once the decoder passes the last frozen index, it can switch to the alternative hard-decision rule, and efficiently obtain the maximum-likelihood codeword. This leads to the proof of the upper bound on the required list size in Theorem 1.

Lastly, we introduce a hybrid SCL decoding algorithm, in which the decoder switches the hard-decision rule half-way through the decoding process. We explain why the new method is likely to outperform the conventional SCL decoding.

B. Related work

The list size in the SCL decoding algorithm has been a subject of research in multiple prior works. The fact that one can achieve near-ML performance with reasonably small list sizes was first discovered through numerical simulations in [12]. However, the numerical simulations also showed that as the block-length increases, one has to increase the list size in order to maintain this property. In fact, in [10], it was shown that the scaling exponent of polar codes under SCL decoding with a fixed list size is the same as that of SC decoding. The recent works of [4] and [5] provided the first theoretically-proven bounds on the list size, which are rooted in information-theoretical quantities and are easy to compute.

Nearly all of our results in this paper also extend to "polar-like" codes. One example would be the RM-polar codes introduced in [8]. Polar codes with dynamic frozen constraints are another example. Such codes were first introduced in [14], and have been subject to extensive research since. Polarization-adjusted convolutional (PAC) codes of Arıkan [3] can be regarded as polar codes with dynamic frozen constraints [16]. In fact, it was recently shown in [9] and [6] that *any* linear code can be encoded as a polar code with dynamic frozen constraints. SCL decoding can be used effectively to decode all of these codes, which further highlights the importance of improving the bounds on its list size.

C. Paper outline

In the next section, we give a brief overview of successive-cancellation list decoding. In Section III, we define the mixing factor of polar codes and introduce an alternative way to make hard decisions throughout the decoding process. We then provide a proof for Theorem 1. The hybrid SCL decoding algorithm is presented in Section IV. Finally, we conclude the paper in Section V with a few remarks and a conjecture.

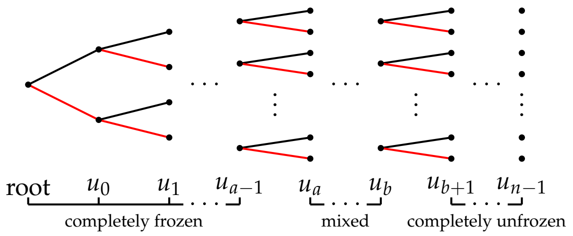

Fig. 1. The binary tree associated with decoding an (n,k) polar code. There are n steps in the decoding and 2^n paths, out of which only 2^k represent valid polar codewords. The red nodes/edges correspond to $u_i = 1$ decisions. u_a is the first unfrozen bit while u_b is the last frozen bit in $\mathbf{u}$. The mixing factor γ is the number of unfrozen bits between u_a and u_b.

II. An Overview of SCL Decoding of Polar Codes

Let us begin by recalling that many families of codes can be viewed as polar codes with *row selection rules* different than that of conventional polar codes. Reed-Muller (RM) codes are one such family. Although these codes and their decoding algorithms have been studied extensively, most practitioners agree that for short-length codes and under reasonable decoding complexity limitations, SCL decoding of polar codes achieves the lowest error probability. Polarization-adjusted convolutional codes, recently introduced by Arıkan in [3], are another example of polar codes, in which not only the row selection rule differs from that of the conventional polar code but also it utilizes a set of dynamic frozen constraints to determine the values of frozen bits. Simulations show that the required lists size for the SCL decoding to obtain near-optimal performance of RM codes and PAC codes are significantly larger than that of polar codes at the same code-lengths [16].

Let $n = 2^m$ denote the code length. Assume $G = A_2^{\otimes m}$ is the $n \times n$ generating matrix of polar codes at length n, and A_2 is the 2×2 polarization kernel. The encoding relation is given by

$$\mathbf{u}G = \mathbf{x}, \tag{1}$$

where $\mathbf{u} = u_0^{n-1}$ and $\mathbf{x} = x_0^{n-1}$ are the information (uncoded) vector and the polar codeword respectively. Let k denote the code dimension. The construction algorithms of polar codes, such as that in [11], designate k coordinates on the vector $\mathbf{u}$ to carry raw information bits while the other $n-k$ bits are frozen according to some pre-determined values that are known to the decoder. Both the SC and and the SCL decoding algorithms are based on decoding the values of u_i sequentially for $i = 0, 1, \ldots, n-1$. This process can be viewed as pursuing one path in the SC decoder, or L paths in the SCL decoder with list size L, in a binary decoding tree with depth n such as that in Figure 1.

For any frozen coordinate such as u_i, its value can be fixed to some number in $\{0,1\}$ or dynamically set according to u_j for $j = 0, 1, \ldots, i-1$ since the values of u_j for $j = 0, 1, \ldots, i-1$ are known to the decoder at the time it is decoding u_i. The latter assignment of frozen bits is known as *dynamic frozen constraints* and was first introduced in [14]. Recently, it has been shown in [9], [16] that any linear code can be encoded as polar codes with dynamic frozen constraints. The main question of interest is that whether the existing decoding algorithms of polar codes can also be modified to decode such codes. In [16], it is showed that the conventional *successive-cancellation* decoding of polar codes cannot be used to decode these arbitrary linear codes. In fact, it is known that the ML decoding of arbitrary codes is an NP-hard task in general. On the other hand, *successive-cancellation list* (SCL) decoding of polar codes is shown to be able to attain probabilities of error nearly as good as that of the ML decoder with small list sizes. An interesting question pops up: can we use the SCL decoding algorithm of polar codes with *small* list sizes to decode *any* linear code nearly as good as the ML decoder?

The SC decoding algorithm is based on decoding u_i's for $i = 0, 1, \ldots, n-1$ sequentially. At each step, it estimates the value of $\hat{u}_i$ according to the following rule:

$$\hat{u}_i = \begin{cases} f_i & \text{if } u_i \text{ is a frozen bit} \\ 0 & \text{if not frozen and } \frac{W(\mathbf{y}, u_0^{i-1}|u_i=0)}{W(\mathbf{y}, u_0^{i-1}|u_i=1)} > 1, \\ 1 & \text{if not frozen and } \frac{W(\mathbf{y}, u_0^{i-1}|u_i=0)}{W(\mathbf{y}, u_0^{i-1}|u_i=1)} < 1, \\ Ber(\frac{1}{2}) & \text{otherwise,} \end{cases} \tag{2}$$

where f_i is the frozen value for u_i and $W(\mathbf{y}, u_0^{i-1}|u_i = t)$ is the bit-channel transition probability defined as

$$W(\mathbf{y}, u_0^{i-1}|u_i = t) = \sum_{u_{i+1}^{n-1} \in \{0,1\}^{n-i-1}} \frac{1}{2^{n-1}} W_n(\mathbf{y}|\mathbf{u}G), \tag{3}$$

in which $\mathbf{y}$ is the received vector after transmitting $\mathbf{x}$ over n i.i.d. copies of a BMS channel W and

$$W_n(\mathbf{y}|\mathbf{u}G) = \prod_{j=0}^{n-1} W(y_j|x_j). \tag{4}$$

Despite that there are exponentially many terms in the transition probabilities defined in (3), the butterfly-like structure of polar codes allows the decoder to compute these probabilities with an average computation complexity of $O(\log n)$. Thus, the decoding complexity of SC decoder and SCL decoder with list size L can be given by $O(n \log n)$ and $O(Ln \log n)$ respectively.

III. Mixing Factor and Nearest Coset Decoding

Given that there are only k unfrozen bits among u_i's, there could be at most 2^k branch-offs during the decoding process over the decoding tree that correspond to the valid polar codewords. Thus, the maximum list size required to achieve ML decoding can not be larger than 2^k. In this section, we show how one can improve this upper bound to 2^γ, where γ is the mixing factor of polar code and is defined as below.

Definition 1 (Mixing Factor)**.** *Let $\mathcal{C}$ be an (n, k) polar code, where $\mathcal{F} \subset [n]$ and $\mathcal{A} = [n] \setminus \mathcal{F}$ denote the set of frozen and unfrozen indices respectively. We denote the mixing factor of this code by γ and define it as*

$$\gamma \triangleq |\{t \in \mathcal{A} \mid t \leqslant \max(\mathcal{F})\}|. \tag{5}$$

Equivalently, we can denote the uncoded polar vector by $\mathbf{u}$ and then, the mixing factor will be the number of unfrozen bits in $\{u_a, u_{a+1}, \ldots, u_{b-1}\}$, where $a = \min(\mathcal{A})$ and $b = \max(\mathcal{F})$. This can also formulated as

$$\gamma = k - (n - 1 - b). \tag{6}$$

The proof of Theorem 1 is based on following *all* valid decoding paths on the decoding tree until the decoder arrives at the last frozen bit, *i.e.* u_b. Given that there are γ unfrozen bits before u_b, the list size has to be at least 2^γ to capture all of these paths. Then, we utilize a hard decision rule different to that in (2) to extend each of these 2^γ paths to the end. The ML codeword corresponds the most-likely path among these length-n completed paths.

Let $i \geqslant b$. Consider one of the paths that the decoder pursued until it reached u_i. Therefore, the decoder is provided with the values of u_j for $j = 0, 1, \ldots, i-1$ when it approaches u_i. Given that there are no future frozen bits after u_i, all of the 2^{n-i} path extensions represent valid polar codewords. These codewords together form a polar coset, which can be defined as

$$\forall t \in \{0, 1\} : \\ \mathcal{C}_t^{(i)}(u_0^{i-1}) \triangleq \sum_{j=0}^{i-1} u_j g_j + t.g_i + \langle g_{i+1}, g_{i+2}, \ldots, g_{n-1} \rangle, \tag{7}$$

where g_j is the j'th row in the generating matrix of length-n polar code, G. Given the definition of transition probabilities in (3), we can see that the hard decision rule in (2) is equivalent to

$$\sum_{\mathbf{x} \in \mathcal{C}_0^{(i)}(u_0^{i-1})} W_n(\mathbf{y}|\mathbf{x}) \quad \gtreqless \quad \sum_{\mathbf{x} \in \mathcal{C}_1^{(i)}(u_0^{i-1})} W_n(\mathbf{y}|\mathbf{x}). \tag{8}$$

In the following, we assume that the underlying channel is a binary symmetric channel (BSC) with some flip probability $p < 1/2$. We recall that maximum-likelihood decoding for BSC is equivalent to minimum (Hamming) distance decoding. Everything can be extended to arbitrary binary memoryless symmetric (BMS) channels by replacing the Hamming distance with a proper distance that is defined according to the channel transition probabilities. We refer the interested reader to Section III in [15] for more details on this topic. Thus, to find the most likely path extension, it suffices to find which of the two cosets above are closer to the received vector $\mathbf{y}$ in Hamming distance, *i.e.*

$$\min_{\mathbf{x} \in \mathcal{C}_0^{(i)}(u_0^{i-1})} d_H(\mathbf{y}, \mathbf{x}) \quad \gtreqless \quad \min_{\mathbf{x} \in \mathcal{C}_1^{(i)}(u_0^{i-1})} d_H(\mathbf{y}, \mathbf{x}), \tag{9}$$

which is also equivalent to

$$\max_{\mathbf{x} \in \mathcal{C}_0^{(i)}(u_0^{i-1})} W_n(\mathbf{y}|\mathbf{x}) \quad \gtreqless \quad \max_{\mathbf{x} \in \mathcal{C}_1^{(i)}(u_0^{i-1})} W_n(\mathbf{y}|\mathbf{x}). \tag{10}$$

For the technique of utilizing the $\max$ function instead of the summation in decoder, see also [13], although it was used for a different purpose. This technique was also used in [6] to establish the proof of NP-hardness for SC decoding of arbitrary linear codes. In the following, we present two techniques to *efficiently* obtain the nearest coset.

Method 1. Let us re-write the summations in (8) as

$$\begin{aligned}&\sum_{\mathbf{x}\in\mathcal{C}_t^{(i)}(u_0^{i-1})} W_n(\mathbf{y}|\mathbf{x})\\ &=\sum_{\mathbf{x}\in\mathcal{C}_t^{(i)}(u_0^{i-1})} p^{d_H(\mathbf{y},\mathbf{x})}(1-p)^{n-d_H(\mathbf{y},\mathbf{x})}\\ &=\sum_{j=0}^{n} p^j(1-p)^{n-j}\times|\{\mathbf{x}\in\mathcal{C}_t^{(i)}(u_0^{i-1})|d_H(\mathbf{y},\mathbf{x})=j\}|.\end{aligned} \tag{11}$$

The efficient implementation of SC decoding in [2] enables us to calculate the summation in (11) for all values of $0<p<1/2$ with $O(\log n)$ computation complexity on average. Here, we can utilize a technique from [6] and set p to be a *very* small number such that the closest codeword $\mathbf{x}$ in $\mathcal{C}_t^{(i)}$ to $\mathbf{x}$ becomes the dominant term in (11). Recalling the discussion in Section II.B of [6], we deduce that it suffices to set $p=2^{-2n}$ for the two hard decision rules in (8) and (10) to become equivalent.

Method 2. While the first method is perfectly capable of identifying the closest polar coset to the received vector, it requires the decoder to perform mathematical operations with very high precision due to the fact that p is extremely small. This can become problematic when n is large or when the chip space is limited.

An alternative method is to use the existing recursive structure of the SC decoder to compute the distances to any such coset. Let $\mathcal{C}_t^{*(i)}(\cdot)$ denote the polar cosets defined in (7) but for length $n^*=n/2$. Assume $i \mod 2=0$. It is possible to show that

$$\begin{aligned}&\mathcal{C}_t^{(i)}(u_0^{i-1})=\\ &\quad\big\{(c_1|c_2)|c_1\in\mathcal{C}_0^{*(\frac{i}{2})}(u_{0,even}^{i-1}\oplus u_{0,odd}^{i-1}),\\ &\quad c_2\in\mathcal{C}_t^{*(\frac{i}{2})}(u_{0,odd}^{i-1})\big\}\quad\cup\\ &\quad\big\{(c_1|c_2)|c_1\in\mathcal{C}_1^{*(\frac{i}{2})}(u_{0,even}^{i-1}\oplus u_{0,odd}^{i-1}),\\ &\quad c_2\in\mathcal{C}_{1-t}^{*(\frac{i}{2})}(u_{0,odd}^{i-1})\big\},\end{aligned} \tag{12}$$

where $u_{0,odd}^{i-1}$ and $u_{0,even}^{i-1}$ denote the subvectors of u_0^{i-1} that consist of *only* odd and even indices respectively. A similar expression holds true for when $i \mod 2=1$. The problem of finding $d_H\big(y_0^{n-1},\mathcal{C}_0^{(i)}(u_0^{i-1})\big)$ can now be simplified to finding the Hamming distance between two halves of the received vector $y_0^{n/2-1}$ and $y_{n/2}^{n-1}$ and the corresponding polar cosets at length n^* in (12). Algorithm 1 performs this task. The proof is based on a mathematical induction on the number of polarization levels, $m=\log n$. Due to lack of space, we leave the proof for the extended version of this paper.

Algorithm 1: CalcD(n, u_0^{i-1}, y_0^{n-1}),

```
Input: block length $n$, vector $u_0^{i-1}$, and received
       vector $y_0^{n-1}$
Output: a pair of distances
       $\big(d_H\big(y_0^{n-1},\mathcal{C}_0^{(i)}(u_0^{i-1})\big), d_H\big(y_0^{n-1},\mathcal{C}_1^{(i)}(u_0^{i-1})\big)\big)$
if $n=1$ then                    // Stopping condition
    return $d_H(u_0,y_0)$
else
    if $i \mod 2=0$ then
        $(a_0,a_1)\leftarrow$ CalcD($\frac{n}{2}, u_{0,even}^{i-1}\oplus u_{0,odd}^{i-1}, y_0^{\frac{n}{2}-1}$)
        $(b_0,b_1)\leftarrow$ CalcD($\frac{n}{2}, u_{0,odd}^{i-1}, y_{\frac{n}{2}}^{n-1}$)
        $d_0\leftarrow\min(a_0+b_0,a_1+b_1)$
        $d_1\leftarrow\min(a_0+b_1,a_1+b_0)$
        return $(d_0,d_1)$
    else
        $(a_0,a_1)\leftarrow$ CalcD($\frac{n}{2}, u_{0,even}^{i-2}\oplus u_{0,odd}^{i-2}, y_0^{\frac{n}{2}-1}$)
        $(b_0,b_1)\leftarrow$ CalcD($\frac{n}{2}, u_{0,odd}^{i-2}, y_{\frac{n}{2}}^{n-1}$ )
        if $u_{i-1}=0$ then
            $(d_0,d_1)\leftarrow(a_0+b_0,a_1+b_1)$
        else
            $(d_0,d_1)\leftarrow(a_0+b_1,a_1+b_0)$
        return $(d_0,d_1)$
```

Proof of Theorem 1. Let $\mathcal{C}$, $\mathcal{F}$, and $\mathcal{A}$ denote the (n,k) polar code, the set of its frozen indices, and the set of unfrozen indices respectively. Further, let $\{i_1,i_2,\ldots,i_\gamma\}=\{t\in\mathcal{A}\mid t\leqslant\max(\mathcal{F})\}$ be the set of unfrozen indices that appear before the last frozen index. Then, $\mathcal{C}$ can be represented as the disjoint union of the following 2^γ polar cosets:

$$\mathcal{C}_{u_{i_1},u_{i_2},\ldots,u_{i_\gamma}}=\sum_{j=0}^{\max(\mathcal{F})}u_jg_j+\langle g_{b+1},g_{b+2},\ldots,g_{n-1}\rangle, \tag{13}$$

where $\{u_{i_1},u_{i_2},\ldots,u_{i_\gamma}\}\in\{0,1\}^\gamma$. By setting the list size to $L=2^\gamma$, the decoder captures all 2^γ possible paths until the last frozen bit, which corresponds to all of the 2^γ affine shifts

$$\sum_{j=0}^{\max(\mathcal{F})}u_jg_j\quad\text{for all }\{u_{i_1},u_{i_2},\ldots,u_{i_\gamma}\}\in\{0,1\}^\gamma. \tag{14}$$

The decoder can then use the nearest coset decoding algorithm to efficiently compute the closest codeword to $\mathbf{y}$ in each of these cosests, which is equivalent to extending each path in an ML fashion according to the decision rule in (9). The extended path with the maximum conditional probability, *i.e.* $W_n(\mathbf{y}|\mathbf{u}G)=\prod_{j=0}^{n-1}W(y_j|x_j)$ corresponds to the ML codeword. $\square$

We also point out that a similar statement was given in [17]. The method above does not rely on using the soft information throughout the decoding in contrast to that of [17], which makes the proposed method more suitable for practical implementations.

IV. Hybrid Successive-Cancellation List Decoding Algorithm

In the proof of Theorem 1, we described a modified SCL decoding algorithm, where the list decoder captures all 2^γ available paths until it reaches the last frozen bit, and then extends each path in an ML fashion. We point out that there is indeed no need to extend all of the 2^γ decoding paths. This is because Algorithm 1 allows the decoder to compute the distance between the received vector $\mathbf{y}$ and all of the 2^γ polar cosets that corresponds to these paths. The decoder can essentially drop all those paths except the one that corresponds to the coset with minimum distance and simply extend the remaining path to the end. This can significantly reduce the overall decoding complexity, particularly when γ is small.

It is also evident that 2^γ could be significantly large for the practical implementation of the SCL decoder if one wishes to capture all $L = 2^\gamma$ paths. However, the numerical simulations show that one can achieve *near-ML* performance even by setting L to much smaller values. In this section, we present the Hybrid Successive-Cancellation List (H-SCL) decoding algorithm, which is a hybrid between the SCL decoding algorithm and the nearest coset algorithm presented earlier. As an input, it takes the list size L and a switching index t, where $t \leqslant \tau = k-\gamma$. The algorithm works in two main stages:

Stage 1. We use the conventional SCL decoding algorithm to capture L decoding paths until the decoder reaches u_t.

Stage 2. The nearest coset decoding algorithm is used to determine the distances between the received vector $\mathbf{y}$ and the polar cosets that correspond to these L paths. We pick the closest coset and find the most-likely path extension for this coset by using the nearest coset decoding algorithm in the remaining $n - t$ decoding steps.

A high-level description of the main loop in H-SCL decoding is given in Algorithm 2. The notations are extracted from [12]. The main differences are the additions of `RecursivelyCalcD` and `continue_ML_path` functions. They both require $O(n \log n)$ computational complexity, which proves that the overall decoding complexity of H-SCL is at most $O(Ln \log n)$ asymptotically. However, the numerical simulations show a significant improvement in the constant.

Algorithm 2: H-SCL decoder, main loop

```
Input: the received vector $\mathbf{y}$, a switching index $t$
       where $t \leqslant \tau = k - \gamma$ , and a list size $L$
Output: a decoded codeword $\hat{\mathbf{c}}$
// Initialization
1  initializeDataStructures()
2  $\ell \leftarrow$ assignInitialPath()
3  $P_0 \leftarrow$ getArrayPointer_P$(0, \ell)$
4  for $\beta = 0, 1, \ldots, n-1$ do
5      set $P_0[\beta][0] \leftarrow W(\mathsf{y}_\beta|0)$, $P_0[\beta][1] \leftarrow W(\mathsf{y}_\beta|1)$
// Main SCL loop
6  for $\varphi = 0, 1, \ldots, t$ do
7      recursivelyCalcP$(m, \varphi)$
8      if $\mathsf{u}_\varphi$ is frozen then
9          for $\ell = 0, 1, \ldots, L-1$ do
10             if activePath$[\ell]$ = false then continue
11             $C_m \leftarrow$ getArrayPointer_C$(m, \ell)$
12             set $C_m[0][\varphi \mod 2]$ to the frozen value of
               $\mathsf{u}_\varphi$
13     else
14         continuePaths_UnfrozenBit$(\varphi)$
15     if $\varphi \mod 2 = 1$ then
16         recursivelyUpdateC $(m, \varphi)$
// Find the nearest coset
17 $\ell' \leftarrow 0$, $d' \leftarrow \infty$
18 for $\ell = 0, 1, \ldots, L-1$ do
19     if activePath$[\ell]$ = false then continue
20     $C_m \leftarrow$ getArrayPointer_C$(m, \ell)$
21     $U_t \leftarrow$ getArrayPointer_U$(m, \ell)$
22     $d \leftarrow$ RecursivelyCalcD$(n, U_t)$
23     if $d < d'$ then
24         $\ell' \leftarrow \ell$, $d' \leftarrow d$
// Extent the ML path
25 for $\varphi = t+1, t+2, \ldots, n-1$ do
26     continue_ML_path$(\ell', \varphi)$
// Return the completed codeword
27 set $C_0 \leftarrow$ getArrayPointer_C$(0, \ell')$
28 return $\hat{\mathbf{c}} = (C_0[\beta][0])_{\beta=0}^{n-1}$
```

V. Concluding Remarks

In this paper, we introduced the mixing factor of polar codes, denoted by γ. The SCL decoding of polar codes was shown to achieve ML decoding when the list size is $L = 2^\gamma$ or larger. Thus, the computational complexity of ML decoding of polar codes can be upper bounded by $O(2^\gamma n \log n)$, where n is the block-length. This fact sheds some lights on why some polar codes require larger list sizes than the others for the SCL decoding algorithm to obtain near-ML performances. In fact, based on our numerical calculations, Reed-Muller codes tend to have the largest mixing factors among codes with the same lengths and dimensions, which coincides with the observations in [16], in which PAC codes with RM rate profilers required larger list sizes to achieve ML performance than the conventional polar codes. We loosely conjecture that polar codes with larger mixing factors require larger list sizes for the SCL decoding algorithm to obtain near-ML performance. Although, the precise definition of *near-ML* is a subject of another discussion, which is left for a future work.

We also introduced the hybrid successive-cancellation list decoding of polar codes, which is based on switching from the SCL decoding algorithm to the nearest coset decoding once the decoder reaches the last frozen bit. This technique reduces the overall decoding complexity and further improve the error probability of the decoder as it prevent the decoder from exploring non-ML paths.

References

[1] 3GPP Working Group Meeting RAN1 #87, Contribution R1-1611108, *Evaluation on channel coding candidates for URLLC and mMTC*, Reno, Nevada, USA, November 2016.

[2] E. Arıkan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–73, June 2009.

[3] E. Arıkan, “From sequential decoding to channel polarization and back again,” arXiv preprint arXiv:1908.09594, Auguest 2019.

[4] M. C. Coskun and H. D. Pfister, “Bounds on the list size of successive cancellation list decoding,” *IEEE International Conference on Signal Processing and Communications (SPCOM)*, pp. 1–5, July 2020.

[5] M. C. Coskun and H. D. Pfister, “An information-theoretic perspective on successive cancellation list decoding and polar code design,” arXiv preprint arXiv:2103.16680.

[6] A. Fazeli, A. Vardy, and H. Yao, “Hardness of successive-cancellation decoding of linear codes,” *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, pp. 455–460, June 2020.

[7] S.A. Hashemi, M. Mondelli, S.H. Hassani, R.L. Urbanke, and W.J. Gross, “Partitioned list decoding of polar codes: Analysis and improvement of finite length performance,” pp. 1–7, *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Singapore, 2017, also available as arXiv preprint arXiv:1705.05497, May 2017.

[8] B. Li, H. Shen, and D. Tse, “A RM-polar codes,” arXiv preprint arXiv:1407.5483. 2014 Jul 21.

[9] B. Li, H. Zhang, and J. Gu, “On pre-transformed polar codes,” arXiv preprint arXiv:1912.06359, December 2019.

[10] M. Mondelli, S. H. Hassani, and R. L. Urbanke, “Scaling exponent of list decoders with applications to polar codes,” *IEEE Transactions on Information Theory*, no. 61, vol. 09, pp. 4838–51, August 2015.

[11] I. Tal and A. Vardy, “How to construct polar codes,” *IEEE Transactions on Information Theory*, no. 59, vol. 10, pp. 6562–82, July 2013.

[12] I. Tal and A. Vardy, “List decoding of polar codes,” *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–26, March 2015.

[13] P. Trifonov, “A score function for sequential decoding of polar codes,” *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, pp. 1470–1474, June 2018.

[14] P. Trifonov and V. Miloslavskaya, “Polar codes with dynamic frozen symbols and their decoding by directed search,” *Proceedings of the IEEE Information Theory Workshop (ITW)*, pp. 1–5, September 2013.

[15] A. Vardy and Y. Be’ery, “More efficient soft decoding of the Golay codes,” *IEEE Transactions on Information Theory*, no. 37, vol. 3, pp. 667–72, May 1991.

[16] H. Yao, A. Fazeli, and A. Vardy, “List decoding of Arıkan’s PAC codes,” arXiv preprint arXiv:2005.13711, May 2020.

[17] Z. Zhang et al, “A split-reduced successive cancellation list decoder for polar codes,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 292–302, November 2015.