Polar List Decoding for Large Polarization Kernels

Bhaskar Gupta,* **Hanwen Yao**,† **Arman Fazeli**,† and **Alexander Vardy**†
*Indian Institute of Technology Bombay, Mumbai, Maharashtra 400076, India
†University of California San Diego, La Jolla, CA 92093, USA
bhaskargupta@cse.iitb.ac.in, and **{hwyao, afazelic, avardy}@ucsd.edu**

***Abstract*—Polar codes constructed with large polarization kernels were recently proven to be able to achieve optimal finite-length scaling properties. However, straightforward decoding for large kernel polar codes introduces a complexity coefficient that is exponential to the kernel sizes, which makes such codes generally believed to be impractical. In this paper, we present a new method that decodes large kernel polar codes with a complexity coefficient that is polynomial to the kernel sizes. This could facilitate the implementation of large kernel polar codes for practical use in the future.**

Successive cancellation decoding for large kernel polar codes requires calculation on the probabilities of its bit channels. Similar to conventional polar codes, those bit channels follow a recursive relation, which make this calculation boil down to computing the probabilities for bit channels of a single polarization kernel. This kernel-level computation can be shown equivalent to soft-output maximum-likelihood (ML) decoding on a single bit of a linear block code. In our proposed method, we first use linear operations to represent the considered linear block code as a polar code with dynamically frozen bits, and then use a modified polar list decoder to get an approximate value on the soft-output of the desired bit. This method is motivated by the observation that at short block lengths, polar list decoding with a large enough list size can well-approximate ML decoding. The proposed low-complexity method allows us to decode polar codes constructed with a 64×64 polarization kernel with scaling exponent $\mu \approx 2.87$ for the first time.

***Index Terms*—Polar codes, large kernels, list decoding, successive cancellation decoding.**

I. Introduction

Polar codes, pioneered by Arıkan [1], give rise to the first explicit family of codes that provably achieve capacity for a wide range of channels. However, the performance of polar codes at finite block lengths turn out to be mediocre. One reason is that polar codes approach channel capacity at a rather slow speed. This is reflected by its finite-length scaling properties. For a family of capacity-achieving codes, scaling exponent describes how the gap between the code rate and the channel capacity vanishes as a function of the block length. The smaller the scaling exponent, the faster this family of codes approaches channel capacity. It has been shown by a series of papers [2]–[5] that polar codes have scaling exponent $\mu = 3.627$ for binary erasure channels (BEC), and scaling exponent $3.579 \leqslant \mu \leqslant 4.714$ for general binary memoryless symmetric (BMS) channels. These numbers are far from the optimal scaling exponent $\mu = 2$ [6], which can be achieved by random linear codes [7], [8].

Arıkan's polar codes are constructed based on the Kronecker product of a 2×2 binary matrix. One way to improve the scaling exponent for polar codes, and thus improving its finite-length performances, is by replacing Arıkan's size 2 matrix with some larger binary square matrices. Those large square matrices are called *polarization kernels*. Polar codes with large kernels were shown to provide asymptotically optimal scaling exponents [9]. Recently, plenty of polarization kernels of size 16, size 32, and size 64 have been proposed with good scaling properties [10]–[14]. In particular in [14], a 64×64 polarization kernel is constructed with $\mu \approx 2.87$. This kernel gives us the first explicit family of codes with scaling exponent under 3. However, decoding large kernel polar codes is generally believed to be impractical due to its high computational complexity. Conventional polar codes of length n admit successive cancellation (SC) decoding with complexity $O(n \log n)$. For a length-n polar code constructed with a size ℓ kernel, straightforward SC decoding requires $O(2^\ell n \log_\ell n)$ computational complexity. This means by employing a size ℓ polarization kernel, we can reduce the scaling exponent, but at the same time also introduce an extra 2^ℓ complexity coefficient on the decoding process. In the asymptotic regime, the coefficient 2^ℓ is just a constant. But in the finite-length regime, this coefficient turns out to be enormous for kernels of relative large sizes.

A. Related Prior Works

Reducing the decoding complexity for large kernel polar codes have been a subject of investigation in multiple prior works. In most cases, specific polarization kernels are identified for which low-complexity decoding is possible. P. Trifonov first proposed the window processing algorithm and used it to decode polar codes constructed with non-binary Reed-Solomon (RS) kernels [15]. Later in [13], [16], with further complexity reduction, the window processing algorithm was used to decode other selected polarization kernels of size 16 and 32. Recently in [17], the authors propose to perform column permutation on the polarization kernels to additionally reduce the complexity for the window processing algorithm. Independently, a class of polarization kernels called *permuted* kernels were

introduced in [12], [18]. Decoding permuted kernels can be viewed as SC decoding of conventional polar codes with look-aheads. Also, there are other decoding algorithms for large kernel polar codes based on trellises proposed in [19], [20]. But in general, the complexity of those algorithms stays high for arbitrary kernels of size 32 and size 64.

In our proposed algorithm, we employ a linear transformation technique that allows one to view *any* linear code as a polar code with dynamically frozen bits. Polar codes with dynamically frozen bits were first introduced in [21]. It has been shown in [21]–[23] that any linear code can be represented as a polar code with dynamically frozen bits. This allows one to use successive cancellation list (SCL) decoding for polar codes to perform *approximate* maximum likelihood (ML) decoding for the linear code. The required list size varies depending on the given linear code.

B. Our Contribution

In this paper, we propose a new method to perform SC decoding for large kernel polar codes. Our method is motivated by the observation that, by representing any linear code as a polar code with dynamically frozen bits, one can employ polar list decoding with large enough list size to get a *good approximate* to ML decoding.

SC decoding for large kernel polar codes requires calculation on the probabilities of its bit channels. Similar to conventional polar codes, those bit channels follow a recursive relation, so this calculation boils down to computing the probabilities for bit channels of a single polarization kernel. This kernel-level computation can be shown equivalent to soft-output ML decoding on a single bit of a linear block code. In our proposed method, we first employ linear transformation techniques to represent the considered linear block code as a polar code with dynamically frozen bits, and then use a modified polar list decoder with a large enough list size to get an approximate value on the soft-output of the desired bit.

Assuming we are using a size ℓ kernel, and list size L for approximation, the complexity of our proposed approach for kernel-level computation is the same as the list decoding complexity for length-ℓ polar codes, which is $O(L\ell \log_2 \ell)$. This complexity depends on the list size L, but it's polynomial in the kernel size ℓ. Our simulation results show that one can obtain good approximations with relatively small list sizes even for kernels of size 64. The proposed method enables us to decoding polar codes constructed with arbitrary polarization kernels of size 32 and 64. In particular for the first time, we are able to decode polar codes constructed with the 64×64 kernel in [14], which has a scaling exponent $\mu \approx 2.87$.

C. Paper Outline

In Section II, We begin by giving some preliminary discussions on large kernel polar codes and the SC decoding algorithm. In Section III, we introduce our SCL-Approximation Decoding Algorithm for kernel-level computation. In Section IV, we present numerical results of our algorithm on polarization kernels of size 32 and size 64. In Section V, we give a brief conclusion for our work.

II. Preliminaries

We use the following notations throughout this paper. We use bold letters like $\boldsymbol{u}$ to denote vectors, and non-bold letters like u_i to denote symbols within that vector. We let the indices for the symbols start from zero. For $\boldsymbol{u} = (u_0, u_1, \cdots, u_{n-1})$, we denote its subvector consists of symbols with indices from a to b as $\boldsymbol{u}_a^b = (u_a, u_{a+1}, \cdots, u_b)$. And we denote the concatenation of vector $\boldsymbol{u}$ and vector $\boldsymbol{v}$ as $(\boldsymbol{u}, \boldsymbol{v})$.

A. Large Kernel Polar Codes

Assuming $n = \ell^m$, an (n,k) large kernel polar code is a binary linear block code generated by k rows of the polar transform matrix $G = D_\ell K^{\otimes m}$, where $K^{\otimes m}$ is an m-fold Kronecker product of an $\ell \times \ell$ binary matrix K with itself, and D_ℓ is a base-ℓ digit-reversal permutation matrix. For Arıkan's conventional polar codes, K would be the 2×2 matrix

$$K_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

and D_ℓ would be the bit-reversal permutation matrix.

Consider a BMS channel $W : \mathcal{X} \to \mathcal{Y}$ as base channel with input alphabet $\mathcal{X} = \{0,1\}$ and output alphabet $\mathcal{Y}$, characterized by its transition probabilities $W(y|x)$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$. For a size ℓ polarization kernel K which is not upper triangular under any column permutation, it's shown in [24] that the polar transform matrix $G = D_\ell K^{\otimes m}$ gives rise to bit channels $W_m^{(i)}(\boldsymbol{y}_0^{n-1}, \boldsymbol{u}_0^{i-1}|u_i)$ with capacities approaching 0 or 1. The definition for the bit channels is given by

$$W_m^{(i)}(\boldsymbol{y}_0^{n-1}, \boldsymbol{u}_0^{i-1}|u_i) = \frac{1}{2^{n-1}} \sum_{\boldsymbol{u}_{i+1}^{n-1}} W^n(\boldsymbol{y}_0^{n-1}|(\boldsymbol{u}_0^{i-1}, u_i, \boldsymbol{u}_{i+1}^{n-1})G), \quad (1)$$

where W^n denotes n independent uses of channel W.

The encoding scheme for large kernel polar codes is given by $\boldsymbol{c} = \boldsymbol{u}G$, where just like conventional polar codes, $\boldsymbol{u}$ is a length-n binary input vector carrying k information bits, and $\boldsymbol{c}$ is the codeword for transmission. The positions of those k information bits in $\boldsymbol{u}$ are specified by an index set $\mathcal{A} \subseteq \{0,1,\cdots,n-1\}$. The index set $\mathcal{A}$ is chosen such that the bit channels $W_m^{(i)}$'s for $i \in \mathcal{A}$ are the k bit channels with the largest channel capacities. The rest of the $n-k$ bits in $\boldsymbol{u}$ are frozen to certain fixed values, usually zeros.

B. SC Decoding of Large Kernel Polar Codes

On the receiver side, for i goes from 0 to $n-1$, the SC decoding algorithm successively estimate bit u_i, and decode it

as $\widehat{u}_i$ based on the earlier decoded path $\widehat{\boldsymbol{u}}_0^{i-1}$ in the following way:

$$\widehat{u}_i = \begin{cases} \arg\max_{u_i \in \{0,1\}} W_m^{(i)}(\boldsymbol{y}_0^{n-1}, \widehat{\boldsymbol{u}}_0^{i-1} | u_i) & i \in \mathcal{A} \\ \text{frozen value} & i \notin \mathcal{A} \end{cases} \tag{2}$$

At the end of this process, the SC decoder returns the length-n estimated vector $\widehat{\boldsymbol{u}}$ as the decoded vector for $\boldsymbol{u}$.

To perform SC decoding, one has to calculate the probability $W_m^{(i)}(\boldsymbol{y}_0^{n-1}, \boldsymbol{u}_0^{i-1} | u_i)$ with a given channel output vector $\boldsymbol{y}_0^{n-1}$. Due to the Kronecker product structure for the polar transform matrix G, this probability can be calculated recursively. Assuming $i \mod \ell = \phi$, and $i = s\ell + \phi$, this probability for the bit channel can be calculated as

$$W_m^{(i)}(\boldsymbol{y}_0^{n-1}, \boldsymbol{u}_0^{i-1} | u_i) = W_1^{(\phi)}(\boldsymbol{z}_0^{\ell-1}, \boldsymbol{u}_{s\ell}^{s\ell+\phi-1} | u_i) \tag{3}$$

Here for the expression $W_1^{(\phi)}(\boldsymbol{z}_0^{\ell-1}, \boldsymbol{u}_{s\ell}^{s\ell+\phi-1} | u_i)$ in (3), we are considering the bit channel for a single polarization kernel K, and we let $\boldsymbol{z}_0^{\ell-1}$ denotes a length-ℓ channel output vector for this single kernel. For the symbols in $\boldsymbol{z}_0^{\ell-1}$, the transition probabilities of the base channels are given by

$$Pr(z_j | u) = W_{m-1}^{(s)}(\boldsymbol{y}_{(j+1)(n/\ell)-1}^{j(n/\ell)}, \boldsymbol{v}(j)_0^{s-1} | u), \quad u \in \{0,1\} \tag{4}$$

where $\boldsymbol{v}(j)_0^{s-1}$ is a length-s binary vector with

$$\boldsymbol{v}(j)_k = (\boldsymbol{u}_{k\ell}^{(k+1)\ell-1} K)_j$$

Therefore, with the recursive relation in (3), calculating the bit channel probabilities for large kernel polar codes boils down to computing the probabilities $W_1^{(i)}(\boldsymbol{y}_0^{\ell-1}, \boldsymbol{u}_0^{i-1} | u_i)$ for a single polarization kernel K. Moreover, for SC decoding in the LLR domain, it suffices to compute the ratio

$$R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1}) \stackrel{\text{def}}{=} \frac{W_1^{(i)}(\boldsymbol{y}_0^{\ell-1}, \boldsymbol{u}_0^{i-1} | u_i = 0)}{W_1^{(i)}(\boldsymbol{y}_0^{\ell-1}, \boldsymbol{u}_0^{i-1} | u_i = 1)} \tag{5}$$

In the context of large kernel polar codes, instead of considering the probability $W_m^{(i)}(\boldsymbol{y}_0^{n-1}, \boldsymbol{u}_0^{i-1} | u_i)$, it's convenient to consider

$$W_m^{(i)}((\boldsymbol{u}_0^{i-1}, u_i) | \boldsymbol{y}_0^{n-1}) = \frac{W_m^{(i)}(\boldsymbol{y}_0^{n-1}, \boldsymbol{u}_0^{i-1} | u_i)}{2W(\boldsymbol{y}_0^{n-1})} = \sum_{\boldsymbol{u}_{i+1}^{n-1} \in \{0,1\}^{\ell-i-1}} W^n((\boldsymbol{u}_0^i, \boldsymbol{u}_{i+1}^{n-1})G | \boldsymbol{y}_0^{n-1}), \tag{6}$$

which is the probability for path $\boldsymbol{u}_0^i$ given the channel output $\boldsymbol{y}_0^{n-1}$. The ratio in (5) can also be derived using those probabilities for the paths as

$$R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1}) = \frac{W_1^{(i)}((\boldsymbol{u}_0^{i-1}, 0) | \boldsymbol{y}_0^{\ell-1})}{W_1^{(i)}((\boldsymbol{u}_0^{i-1}, 1) | \boldsymbol{y}_0^{\ell-1})} \tag{7}$$

In this way, the task for computing $W_1^{(i)}(\boldsymbol{y}_0^{\ell-1}, \boldsymbol{u}_0^{i-1} | u_i)$ equivalently becomes the task for computing the ratio $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$ for a single kernel K. This task is referred as kernel processing [25], or kernel marginalization [26]. If this kernel-level computation task takes complexity $O(T)$, then the overall SC decoding complexity for length-n large kernel polar codes will be $O(T \cdot n \log_\ell n)$.

Straightforward calculation for $W_1^{(i)}(\boldsymbol{u}_0^i | \boldsymbol{y}_0^{\ell-1})$ as in (6) takes exponential complexity $O(2^\ell)$. This complexity can be slightly reduced by methods in [27] or by methods in [26]. But in general, direct calculation stays prohibitive for kernels with relative large sizes. More efficient algorithms such as window processing [15] and recursive trellis processing [19] have been proposed for computing $W_1^{(i)}(\boldsymbol{u}_0^i | \boldsymbol{y}_0^{\ell-1})$. Those algorithms are still exponential with respect to the kernel size ℓ, but with various improvements [16], [17], [25], they are efficient enough to process kernels of size 16 and some designed low-complexity kernels of size 32.

III. Successive Cancellation List Approximation Algorithm for Polarization Kernels

In this section, we propose a new algorithm that uses successive cancellation list (SCL) decoders for polar codes [28] to estimate the ratio in (7). We call it SCL-Approximation Algorithm.

First, we show the problem of computing the ratio $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$ can be transformed into the problem of soft-output ML decoding on a single bit of a linear block code.

A. Cancelling the Effect of the Preceding Bits

Let K be an $\ell \times \ell$ polarization kernel with $\ell = 2^t$ being a power of 2, and let

$$K = \begin{bmatrix} A^{(i-1)} \\ B^{(i)} \end{bmatrix}$$

where $A^{(i-1)}$ is defined to be the submatrix of K consisting of its rows with indices from 0 to $i-1$, and $B^{(i)}$ is defined to be the submatrix of K consisting of its rows with indices from i to $\ell - 1$. We start by expressing $W_1^{(i)}(\boldsymbol{u}_0^i | \boldsymbol{y}_0^{\ell-1})$ as

$$\begin{aligned} & W_1^{(i)}((\boldsymbol{u}_0^{i-1}, u_i) | \boldsymbol{y}_0^{\ell-1}) \\ &= \sum_{\boldsymbol{u}_{i+1}^{\ell-1}} W^\ell((\boldsymbol{u}_0^{i-1}, u_i, \boldsymbol{u}_{i+1}^{\ell-1})K | \boldsymbol{y}_0^{\ell-1}) \\ &= \sum_{\boldsymbol{u}_{i+1}^{\ell-1}} W^\ell(\boldsymbol{u}_0^{i-1}A^{(i-1)} + (u_i, \boldsymbol{u}_{i+1}^{\ell-1})B^{(i)} | \boldsymbol{y}_0^{\ell-1}) \end{aligned}$$

Since the base channel W is a BMS channel, for any $y \in \mathcal{Y}$ with $W(0|y) = p_1$ and $W(1|y) = p_2$, there exists a $\bar{y} \in \mathcal{Y}$ such that $W(0|\bar{y}) = p_2$ and $W(1|\bar{y}) = p_1$. So in the above expression for $W_1^{(i)}(\boldsymbol{u}_0^i | \boldsymbol{y}_0^{\ell-1})$, we can cancel out the impact of the earlier path $\boldsymbol{u}_0^{i-1}$ on $\boldsymbol{y}_0^{\ell-1}$ by replacing $\boldsymbol{y}_0^{\ell-1}$ with a new channel output vector $\boldsymbol{z}_0^{\ell-1}$, such that

$$z_j = \begin{cases} y_j & (\boldsymbol{u}_0^{i-1}A^{(i-1)})_j = 0 \\ \bar{y}_j & (\boldsymbol{u}_0^{i-1}A^{(i-1)})_j = 1 \end{cases}$$

We remark that this $\boldsymbol{z}_0^{\ell-1}$ is a new defined output vector different from the one in (3). By defining this new vector $\boldsymbol{z}_0^{\ell-1}$, we have

$$W_1^{(i)}((\boldsymbol{u}_0^{i-1}, u_i)|\boldsymbol{y}_0^{\ell-1}) = \sum_{\boldsymbol{u}_{i+1}^{\ell-1}} W^{\ell}((u_i, \boldsymbol{u}_{i+1}^{\ell-1})B^{(i)}|\boldsymbol{z}_0^{\ell-1})$$

So the ratio in (7) has the expression

$$R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1}) = \frac{\sum_{\boldsymbol{u}_{i+1}^{\ell-1}} W^{\ell}((0, \boldsymbol{u}_{i+1}^{\ell-1})B^{(i)}|\boldsymbol{z}_0^{\ell-1})}{\sum_{\boldsymbol{u}_{i+1}^{\ell-1}} W^{\ell}((1, \boldsymbol{u}_{i+1}^{\ell-1})B^{(i)}|\boldsymbol{z}_0^{\ell-1})} \tag{8}$$

If we denote the linear code generated by $B^{(i)}$ as $\mathbb{C}(B^{(i)})$, and view $\boldsymbol{z}_0^{\ell-1}$ as the channel output after transmitting a codeword in $\mathbb{C}(B^{(i)})$ through the base channels, then for the expression in (8), we basically have $Pr(u_i = 0|\boldsymbol{z}_0^{\ell-1})$ in the numerator, and $Pr(u_i = 1|\boldsymbol{z}_0^{\ell-1})$ in the denominator. Therefore, computing the ratio $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$ can be achieved by soft-output ML decoding on the first bit of the code $\mathbb{C}(B^{(i)})$ generated by $B^{(i)}$.

Since the polarization kernel K has full-rank, its submatrix $B^{(i)}$ has rank $\ell - i$, and $\mathbb{C}(B^{(i)})$ is an $(\ell, \ell - i)$ linear block code. To compute this ratio $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$ directly, one needs to check the probabilities for all the codewords in $\mathbb{C}(B^{(i)})$. This straightforward approach requires complexity $O(2^{\ell})$, which is exponential in the kernel size ℓ. In our algorithm, we propose to perform polar list decoding for $\mathbb{C}(B^{(i)})$, and approximate the ratio $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$ by only checking the probabilities for the codewords in the list. To perform polar list decoding, we first need to represent $\mathbb{C}(B^{(i)})$ as a polar code with dynamically frozen bits.

B. Representation as Polar Codes with Dynamic Freezing

Polar codes with dynamically frozen bits, first introduced in [21], are polar codes where each of the frozen bit u_j is not fixed to be zero, but set to be a linear function of its preceding bits as $u_j = f_j(u_0, u_1, \cdots, u_{j-1})$. It has been shown [21]–[23] that with linear operation techniques, *any* linear code can be encoded as a polar code with dynamically frozen bits.

We now represent $\mathbb{C}(B^{(i)})$ as a polar code with dynamically frozen bits, so that we can perform polar list decoding on top of it. Let $K_{\mathrm{A}} = BK_2^{\otimes t}$ be the Arıkan's polar transform matrix of size ℓ. we first define an $(\ell - i) \times \ell$ precoder matrix M as

$$M = B^{(i)}K_{\mathrm{A}} \tag{9}$$

Observe that K_{A} is invertible with $(K_{\mathrm{A}})^{-1} = K_{\mathrm{A}}$. So

$$B^{(i)} = MK_{\mathrm{A}} \tag{10}$$

Then with elementary row operations, we can transform M into a matrix M^* in reduced row echelon form (RREF). The relation between M and M^* is given by $M^* = TM$, where T is some $(\ell - i) \times (\ell - i)$ invertible matrix. By mutiplying T on both sides of (10), we get

$$TB^{(i)} = M^*K_{\mathrm{A}} \tag{11}$$

Denote $B^{(i)*} = TB^{(i)}$ as a new generator matrix. Since row operations preserve the linear space spanned by $B^{(i)}$, $B^{(i)*}$ generates the same code as $B^{(i)}$. Let $\boldsymbol{v}$ be a length-$(\ell - i)$ binary vectors with $\ell - i$ information bits. The encoding of $\boldsymbol{v}$ with the generator matrix $B^{(i)*}$ is given by

$$\boldsymbol{v}B^{(i)*} = \underbrace{\boldsymbol{v}M^*}_{\boldsymbol{w}}K_{\mathrm{A}} \tag{12}$$

Here M^* is in RREF, so vector $\boldsymbol{w} = \boldsymbol{v}M^*$ is a length-ℓ vector with $\ell - i$ information bits, and i dynamically frozen bits. Therefore, the encoding $\boldsymbol{v}B^{(i)*}$ can also be achieved by multiplying $\boldsymbol{w}$ with dynamically frozen bits with the Arıkan's polar transform matrix K_{A}. In this way, we represent $\mathbb{C}(B^{(i)})$ as a polar code with dynamically frozen bits.

The linear code $\mathbb{C}(B^{(i)})$ can be encoded either by $\boldsymbol{u}_i^{\ell-1}B^{(i)}$, or by $\boldsymbol{v}B^{(i)*}$. For the same codeword, the relation between $\boldsymbol{u}_i^{\ell-1}$ and $\boldsymbol{v}$ can be established by

$$\boldsymbol{u}_i^{\ell-1}B^{(i)} = \boldsymbol{v}B^{(i)*} \quad \Rightarrow \quad \boldsymbol{u}_i^{\ell-1} = \boldsymbol{v}T \tag{13}$$

This shows u_i equals to the first bit of $\boldsymbol{v}T$. In other words, $u_i = (\boldsymbol{v}T)_0$.

C. Ratio Estimation via Polar List Decoding

By viewing $\mathbb{C}(B^{(i)})$ as a polar code with dynamically frozen bits, we can perform SCL decoding in [28] for $\mathbb{C}(B^{(i)})$ to decode the vector $\boldsymbol{v}$. Assuming the list size we are using is L, at the end of the SCL decoding process, we will get a list of L different paths for $\boldsymbol{v}$. We denoted the set of those L paths as $P = \{\boldsymbol{v}^{[1]}, \boldsymbol{v}^{[2]}, \cdots, \boldsymbol{v}^{[L]}\}$.

Let's re-examine the expression in (8) for the ratio $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$. To compute this ratio directly, we need to divide all codewords in $\mathbb{C}(B^{(i)})$ into two sets. The first set contains codewords encoded by $\boldsymbol{u}_i^{\ell-1}$ with $u_i = 0$, and the second set contains codewords encoded by $\boldsymbol{u}_i^{\ell-1}$ with $u_i = 1$. Then $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$ can be computed as the ratio of the sums of the probabilities for codewords in those two sets.

In our SCL-Approximation Algorithm, instead of checking all the codewords in $\mathbb{C}(B^{(i)})$, after the SCL decoding process, we propose to only check those L codewords generated by paths in the list. We divide those L codewords into two sets depending on the values of u_i, and get an approximate value for the ratio as

$$\widehat{R}^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1}) = \frac{\sum_{\boldsymbol{v}\in P:(\boldsymbol{v}T)_0=0} W^{\ell}(\boldsymbol{v}B^{(i)*}|\boldsymbol{z}_0^{\ell-1})}{\sum_{\boldsymbol{v}\in P:(\boldsymbol{v}T)_0=1} W^{\ell}(\boldsymbol{v}B^{(i)*}|\boldsymbol{z}_0^{\ell-1})} \tag{14}$$

With a large enough list size L, empirically the polar list decoder is a good approximation for the ML decoder. Therefore, it is reasonable to expect that those L codewords in the list will capture majority of the probabilities, and thus $\widehat{R}^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$ in (14) will give us a precise enough approximation for $R^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$.

In this approach, the computation for the precoder matrix M^*, and the new generator matrix $B^{(i)*}$ can all be performed offline. So the complexity of this kernel-level computation is

Algorithm 1: SCL-Approximation Algorithm

Input: size ℓ kernel $K = \begin{bmatrix} A^{(i-1)} \\ B^{(i)} \end{bmatrix}$, index i, base channel W, and channel output vector $\boldsymbol{y}_0^{\ell-1}$

Output: $\widehat{R}^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1})$

set $M \leftarrow B^{(i)} K_{\mathrm{A}}$, where K_{A} is Arıkan's polar transform matrix of size ℓ
transform M into M^* in RREF by $M^* = TM$
set $B^{(i)*} \leftarrow M^* K_{\mathrm{A}}$
// The above part of the algorithm can be performed offline
$\boldsymbol{x} \leftarrow \boldsymbol{u}_0^{i-1} A^{(i-1)}$
$\boldsymbol{z} \leftarrow (z_0, z_1, \cdots, z_{\ell-1})$
for $i = 0, 1, ..., \ell - 1$ **do**
    if $x_i = 1$ **then**
        $z_i = y_i$
    else
        $z_i = \bar{y}_i$
perform SCL decoding with list size L on $(\boldsymbol{v}M^*)K_{\mathrm{A}}$ with channel output $\boldsymbol{z}$ to get the set of L paths $\{\boldsymbol{v}^{[1]}, \boldsymbol{v}^{[2]}, \cdots, \boldsymbol{v}^{[L]}\}$
$p_0 \leftarrow 0$
$p_1 \leftarrow 0$
for $i = 1, ..., L$ **do**
    if $(\boldsymbol{v}^{[i]}T)_0 = 0$ **then**
        $p_0 \leftarrow p_0 + Pr(\boldsymbol{v}^{[i]} B^{(i)*} | \boldsymbol{z})$
    else
        $p_1 \leftarrow p_1 + Pr(\boldsymbol{v}^{[i]} B^{(i)*} | \boldsymbol{z})$
return $\widehat{R}^{(i)}(\boldsymbol{u}_0^{i-1}, \boldsymbol{y}_0^{\ell-1}) = p_0/p_1$

$O(L\ell \log_2 \ell)$, the same as the complexity of polar list decoding for length-ℓ polar codes. This complexity is polynomial in the kernel size ℓ. If we apply the SCL-Approximation Algorithm, the overall complexity of SC decoding for length-n large kernel polar polar codes will be $O(L\ell \log_2 \ell \cdot n \log_\ell n)$. The pseudo code summarizing our SCL-Approximation Algorithm is given in Algorithm 1.

IV. Simulation Results

In this section, we show some simulation results of SC decoding with SCL-Approximation Algorithm for large kernel polar codes. We consider two polarization kernels K_{32} and K_{64} from [14]. The 32×32 kernel K_{32} has scaling exponent $\mu(K_{32}) = 3.122$, and the 64×64 kernel K_{64} has scaling exponent $\mu(K_{64}) \approx 2.87$. The large kernel polar codes considered here are all constructed with Monte-Carlo simulations.

Figure 1 shows the simulation results of SC decoding for $(1024, 512)$ polar codes on AWGN channels. The black line shows the SC decoding performance for conventional polar codes, and the color lines show the SC decoding performance with SCL-Approximation Algorithm for large kernel polar codes constructed with K_{32}. We can observe that as we increase the list size L for the SCL-Approximation Algorithm, the overall SC decoding performance gets better as expected. Since larger L is expected to gives us better approximation for the ratio in (7) for kernel-level computations. For K_{32}, our simulation results show that $L = 32$ is large enough to get us close to the performance limit of our approach. Thus it is reasonable to believe that for K_{32}, the SCL-Approximation Algorithm with $L = 32$ gives us a pretty precise approximation for the ratio in (7).

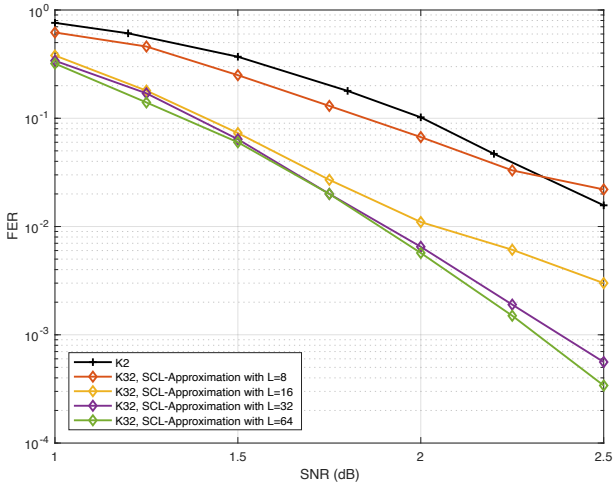

Fig. 1: SC decoding performance for (1024,512) polar codes

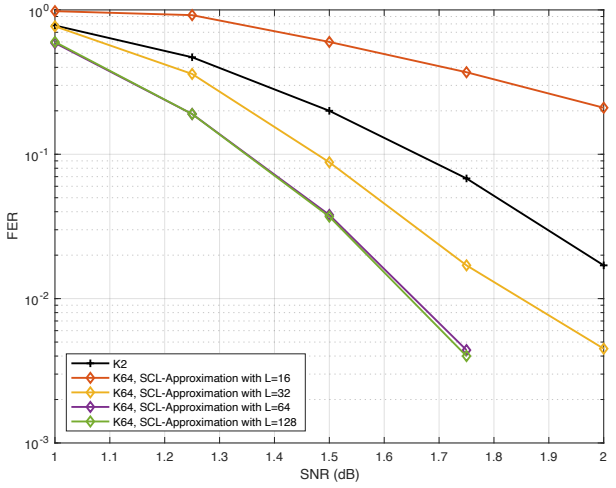

Fig. 2: SC decoding performance for (4096,2048) polar codes

Figure 2 shows the simulation results of SC decoding for $(4096, 2048)$ polar codes on AWGN channels. Similarly, the black line shows the SC decoding performance for conventional polar codes, and the color lines show the SC decoding performance with SCL-Approximation Algorithm for large kernel polar codes constructed with K_{64}. For K_{64}, $L = 64$ is large enough to get us close to the performance limit of

our approach. This gives the evidence that for K_{64}, the SCL-Approximation Algorithm with $L = 64$ is likely to give us a pretty precise approximation for the ratio in (7).

V. Conclusion

In this paper, we propose the SCL-Approximation Algorithm to perform kernel-level computation for SC decoding on large kernel polar codes. The SCL-Approximation Algorithm exploits the idea that polar list decoding with large enough list size can well-approximate ML decoding. This algorithm has computational complexity polynomial in the kernel size. With this low-complexity approach, we are able to SC decode polar codes constructed with a size 64 kernel for the first time.

References

[1] E. Arikan, "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. 3051–3073, 2009.

[2] D. Goldin and D. Burshtein, "Improved bounds on the finite length scaling of polar codes," *IEEE Transactions on Information Theory*, vol. 60, no. 11, pp. 6966–6978, 2014.

[3] S. H. Hassani, K. Alishahi, and R. L. Urbanke, "Finite-length scaling for polar codes," *IEEE Transactions on Information Theory*, vol. 60, no. 10, pp. 5875–5898, 2014.

[4] S. B. Korada, A. Montanari, E. Telatar, and R. Urbanke, "An empirical scaling law for polar codes," in *2010 IEEE International Symposium on Information Theory*. IEEE, 2010, pp. 884–888.

[5] M. Mondelli, S. H. Hassani, and R. L. Urbanke, "Unified scaling of polar codes: Error exponent, scaling exponent, moderate deviations, and error floors," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 6698–6712, 2016.

[6] V. Strassen, "Asymptotische abschatzugen in shannon's informationstheorie," in *Transactions of the Third Prague Conference on Information Theory etc, 1962. Czechoslovak Academy of Sciences, Prague*, 1962, pp. 689–723.

[7] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[8] M. Hayashi, "Information spectrum approach to second-order coding rate in channel coding," *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 4947–4966, 2009.

[9] A. Fazeli, H. Hassani, M. Mondelli, and A. Vardy, "Binary linear codes with optimal scaling: Polar codes with large kernels," *IEEE Transactions on Information Theory*, 2020.

[10] A. Fazeli and A. Vardy, "On the scaling exponent of binary polarization kernels," in *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2014, pp. 797–804.

[11] N. Presman, O. Shapira, S. Litsyn, T. Etzion, and A. Vardy, "Binary polarization kernels from code decompositions," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2227–2239, 2015.

[12] S. Buzaglo, A. Fazeli, P. H. Siegel, V. Taranalli, and A. Vardy, "On efficient decoding of polar codes with large kernels," in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2017, pp. 1–6.

[13] G. Trofimiuk and P. Trifonov, "Efficient decoding of polar codes with some 16× 16 kernels," in *2018 IEEE Information Theory Workshop (ITW)*. IEEE, 2018, pp. 1–5.

[14] H. Yao, A. Fazeli, and A. Vardy, "Explicit polar codes with small scaling exponent," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 1757–1761.

[15] P. Trifonov, "Binary successive cancellation decoding of polar codes with reed-solomon kernel," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 2972–2976.

[16] G. Trofimiuk and P. Trifonov, "Reduced complexity window processing of binary polarization kernels," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 1412–1416.

[17] F. Abbasi and E. Viterbo, "Large kernel polar codes with efficient window decoding," *IEEE Transactions On Vehicular Technology*, vol. 69, no. 11, pp. 14 031–14 036, 2020.

[18] S. Buzaglo, A. Fazeli, P. H. Siegel, V. Taranalli, and A. Vardy, "Permuted successive cancellation decoding for polar codes," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2618–2622.

[19] P. Trifonov, "Trellis-based decoding techniques for polar codes with large kernels," in *2019 IEEE Information Theory Workshop (ITW)*. IEEE, 2019, pp. 1–5.

[20] E. Moskovskaya and P. Trifonov, "Design of bch polarization kernels with reduced processing complexity," *IEEE Communications Letters*, vol. 24, no. 7, pp. 1383–1386, 2020.

[21] P. Trifonov and V. Miloslavskaya, "Polar codes with dynamic frozen symbols and their decoding by directed search," in *2013 IEEE Information Theory Workshop (ITW)*. IEEE, 2013, pp. 1–5.

[22] A. Fazeli, A. Vardy, and H. Yao, "Hardness of successive-cancellation decoding of linear codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 455–460.

[23] C.-Y. Lin, Y.-C. Huang, S.-L. Shieh, and P.-N. Chen, "Transformation of binary linear block codes to polar codes with dynamic frozen," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 333–341, 2020.

[24] S. B. Korada, E. Şaşoğlu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6253–6264, 2010.

[25] G. Trofimiuk and P. Trifonov, "Window processing of binary polarization kernels," *IEEE Transactions on Communications*, 2021.

[26] V. Bioglio and I. Land, "On the marginalization of polarizing kernels," in *2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*. IEEE, 2018, pp. 1–5.

[27] Z. Huang, S. Zhang, F. Zhang, C. Duanmu, F. Zhong, and M. Chen, "Simplified successive cancellation decoding of polar codes with medium-dimensional binary kernels," *IEEE Access*, vol. 6, pp. 26 707–26 717, 2018.

[28] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.