On Efficient Decoding of Polar Codes with Large Kernels

Sarit Buzaglo, **Arman Fazeli**, **Paul H. Siegel**, **Veeresh Taranalli**, and **Alexander Vardy**
University of California San Diego, La Jolla, CA 92093, USA
{*sbuzaglo, afazelic, psiegel, vtaranalli, avardy*}*@ucsd.edu*

***Abstract*—Defined through a certain 2×2 matrix called *Arikan's kernel*, polar codes are known to achieve the symmetric capacity of binary-input discrete memoryless channels under the *successive cancellation* (SC) decoder. Yet, for short block-lengths, polar codes fail to deliver a compelling performance under the low complexity SC decoding scheme. Recent studies provide evidence for improved performance when Arikan's kernel is replaced with larger kernels that have smaller *scaling exponents*. However, for $\ell \times \ell$ kernels the time complexity of the SC decoding increases by a factor of 2^ℓ.**

In this paper we study a special type of kernels called *permuted kernels*. The advantage of these kernels is that the SC decoder for the corresponding polar codes can be viewed as a permuted version of the SC decoder for the conventional polar codes that are defined through Arikan's kernel. This *permuted successive cancellation* (PSC) decoder outputs its decisions on the input bits according to a permuted order of their indices. We introduce an efficient PSC decoding algorithm and show simulations for two 16×16 permuted kernels that have better scaling exponents than Arikan's kernel.

I. Introduction

Introduced by Arikan [1], polar codes are the first codes that were proved to achieve the symmetric capacity of a binary-input discrete memoryless channel (B-DMC) $\mathcal{W}$. Polar codes can be viewed as part of a much larger family of codes that are generated according to the 2×2 matrix

$$F_2 \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

These length $n = 2^m$ codes are cosets of a linear subspace that is spanned by some k rows of $F_2^{\otimes m}$, the mth Kronecker power of F_2. One important example of such codes are Reed-Muller codes, for which the spanning k rows correspond to the k rows of $F_2^{\otimes m}$ that have maximum Hamming weight. The genius of Arikan's construction lies in the fact that the selection of the k spanning rows depends on the channel. Each row of $F_2^{\otimes m}$ is associated with a synthesized channel called a *bit channel*, which is defined through F_2 and the channel $\mathcal{W}$. The selected k rows correspond to the k least noisy bit channels, where the noisiness of the channel is measured by its Bhattacharyya parameter (see [1] for more information on the Bhattacharyya parameter). A remarkable property of F_2 is that as n grows, the bit channels exhibit a polarization phenomenon: some of the bit channels become very noisy, while the rest of the bit channels become almost noiseless. Moreover, the fraction of channels that are almost noiseless approaches the symmetric capacity of the channel, $I(\mathcal{W})$. As a result, for every $R < I(\mathcal{W})$ and for sufficiently large n there exists a rate R polar code for which the probability of error under a *successive cancellation* (SC) decoder is $O(n^{-\frac{1}{2+\epsilon}})$, where $\epsilon > 0$ is arbitrarily small.

Despite their impressive asymptotic behavior, empirical results indicate less impressive performance of the SC decoder for polar codes of short block-lengths, e.g. compared to LDPC codes. To improve performance, different decoders were suggested, e.g., belief propagation decoding [2], [5], [8], decoding with look-ahead [15], and list successive cancellation decoding [12] improved SC decoding [4]. Combined with cyclic redundancy check (CRC) pre-coding, the list successive cancellation decoder provides the best performance for polar codes up-to date. Meanwhile, a series of works pursued the same goal by replacing F_2 with larger matrices called *kernels* that may induce a faster polarization of the bit channels [6], [7], [9], [10], [11], [14].

To estimate how polarizing a kernel is, two figures of merit were proposed. The first is the *error exponent* [3], [10] which quantifies the exponential decay of the error probability with respect to the block-length n, for a fixed rate R. The second is the *scaling exponent* [6], [9] which reflects the dependence between the length of the code and its rate, for a fixed probability of error P_e. More precisely, a scaling exponent μ is a constant that depends only on the kernel K and the channel $\mathcal{W}$ for which $n = \Theta(I(\mathcal{W}) - R)^{-\mu}$, where the sum of the Bhattacharyya parameters of the nR least noisy bit channels is at most P_e. No method is known to compute the scaling exponent for general channels. In fact, it is not even clear whether or not the scaling exponent exists. Hassani *et. al.* [9] introduced a numerical method to compute the scaling exponent for the binary erasure channel (BEC) and found that $\mu = 3.627$, where the kernel is F_2. Fazeli and Vardy [6] presented an 8×8 kernel K_8 with $\mu = 3.577$ and showed that this is optimal for kernels of size $\ell \leq 8$. They also suggested a heuristic construction through which they found a 16×16 kernel K_{16} with $\mu = 3.356$. However, the time complexity of a SC decoding of polar codes with $\ell \times \ell$ kernels scales as $2^\ell n \log_\ell n$.

The goal of this paper is to achieve both good performance and efficient SC decoding. To this end we propose to use a special type of kernels called *permuted kernels*. These kernels are formed by permuting the rows of $F_2^{\otimes \ell}$. One example of a permuted kernel is the kernel K_8 from [6]. On the other hand, the kernel K_{16} defined in [6] is not a permuted kernel. While a successive cancellation decoder for a polar code with the kernel F_2 and of dimension k decides on the n input bits $u_0 u_1 \dots u_{n-1}$ (n uncoded bits, $n-k$ of them are known to the decoder) one after the other according to the sequential order

978-1-5090-5908-9/17/$31.00 ©2017 IEEE

from 0 to $n-1$, a SC decoder for polar codes with permuted kernels decides on the input bits according to a permuted order of their indices. Therefore, we call the SC decoder for a polar code with a permuted kernel a *permuted successive cancellation* (PSC) decoder. By exploiting the connection between permuted kernels and $F_2^{\otimes \ell}$, we introduce a more efficient implementation of the PSC decoder. Due to page limitations, we only describe our PSC decoding algorithm for length-ℓ polar codes. A more detailed presentation of the algorithm will be given in the full version of this paper. We also propose two new 16×16 permuted kernels and show simulation results for their performance.

The rest of this paper is organized as follows. In Section II, we present notation and definitions that are used throughout the paper and review the basic concepts of polar codes. In Section III we discuss the connection between the bit channels of a permuted kernel and the bit channels of $F_2^{\otimes \ell}$. Our PSC decoding algorithm for block-length ℓ decoder is presented in Section IV. Time complexity of the algorithm is discussed in Section V along with some simulation results.

II. Preliminaries

In this section we introduce notation and definitions used throughout the paper and review the basic concepts for polar codes.

For a positive integer n, denote by $[n]$ the set of n integers $\{0, 1, \ldots, n-1\}$. For two integers $a < b$, denote by $[a, b]$ the set of $b-a$ integers $\{a, a+1, \ldots, b-1\}$. For a positive integer ℓ and for all $r \in [\ell]$, denote by $[n]_r$ the set of all elements in $[n]$ that are equal to r modulo ℓ, where ℓ should be clear from the context. A binary vector of length n is denoted by $u_0^{n-1} = u_0 u_1 \ldots u_{n-1}$. For $\mathcal{A} \subseteq [n]$, denote by $u_{\mathcal{A}}$ the subvector of u_0^{n-1} that is specified according to indices from $\mathcal{A}$. In particular, if ℓ divides n and $r \in [\ell]$ then $u_{[n]_r} = u_r u_{\ell+r} \ldots u_{n-\ell+r}$. All operations on vectors and matrices in this paper are carried out over the field $GF(2)$. The componentwise addition modulo-2 of two binary vectors u_0^{n-1} and v_0^{n-1} is denoted by $u_0^{n-1} \oplus v_0^{n-1}$. For an $\ell \times \ell$ matrix K, denote by $K^{\otimes m}$ the mth Kronecker power of K.

Throughout this paper $\mathcal{W} : \mathcal{X} \to \mathcal{Y}$ is a generic B-DMC with input alphabet $\mathcal{X} = \{0, 1\}$, output alphabet $\mathcal{Y}$, and *transition probabilities* $\mathcal{W}(y|x)$, where for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, $\mathcal{W}(y|x)$ is the conditional probability that the channel output is y given that the transmitted input is x. For a positive integer n, denote by $\mathcal{W}^n : \mathcal{X}^n \to \mathcal{Y}^n$ the channel that corresponds to transmission over n independent copies of $\mathcal{W}$. Hence, for every $x_0^{n-1} \in \mathcal{X}^n$ and $y_0^{n-1} \in \mathcal{Y}^n$, the transition probability $\mathcal{W}^n(y_0^{n-1}|x_0^{n-1})$ is given by

$$\mathcal{W}^n(y_0^{n-1}|x_0^{n-1}) \stackrel{\text{def}}{=} \prod_{i=0}^{n-1} \mathcal{W}(y_i|x_i).$$

For an $n \times n$ matrix G, a set $\mathcal{A} \subset [n]$ of size k, and a vector z_0^{n-k-1}, let $\mathcal{C}$ be the code that encodes a length n input vector u_0^{n-1}, for which $u_{[n]\setminus\mathcal{A}} = z_0^{n-k-1}$, to the codeword $x_0^{n-1} = u_0^{n-1} G$. If $i \in [n] \setminus \mathcal{A}$, then u_i is called a *frozen bit*. For all $i \in [n]$, the *ith bit channel* with respect to G, $\mathcal{W}_G^{(i)} : \mathcal{X} \to \mathcal{Y}^n \times \mathcal{X}^i$, is defined by the transition probabilities

$$\mathcal{W}_G^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) \stackrel{\text{def}}{=} \sum_{u_{i+1}^{n-1} \in \mathcal{X}^{n-i-1}} \frac{1}{2^{n-1}} \mathcal{W}_G(y_0^{n-1}|u_0^{n-1} G). \tag{1}$$

A *successive cancellation* (SC) decoder for $\mathcal{C}$ outputs a *decision vector* $\hat{u}_0^{n-1}$ in n steps, where at the ith step the decoder decides on the value of $\hat{u}_i$ according to the following rule. If $i \in [n] \setminus \mathcal{A}$ then $\hat{u}_i$ is set to the value of the frozen bit u_i. Otherwise, the decoder calculates the pair of probabilities

$$\begin{aligned} P_i[0] &\stackrel{\text{def}}{=} \mathcal{W}_G^{(i)}(y_0^{n-1}, \hat{u}_0^{i-1}|u_i = 0), \\ P_i[1] &\stackrel{\text{def}}{=} \mathcal{W}_G^{(i)}(y_0^{n-1}, \hat{u}_0^{i-1}|u_i = 1). \end{aligned} \tag{2}$$

and sets $\hat{u}_i$ to the more likely value according to these probabilities, i.e.,

$$\hat{u}_i \stackrel{\text{def}}{=} \begin{cases} 0, & P_i[0] \geq P_i[1] \\ 1, & \text{otherwise.} \end{cases}$$

Notice that, in general, the complexity of the SC decoder may be exponential in n, since the calculation of $P_i[0], P_i[1]$ requires a summation of 2^{n-i-1} terms.

Let $n = 2^m$ and let $G_n \stackrel{\text{def}}{=} B_n F_2^{\otimes m}$, where B_n is the *bit-reversal* permutation matrix (see [1] for the definition of B_n). For $G = G_n$, a polar code with Arikan's kernel is defined by setting $\mathcal{A}$ to be the set of k indices corresponding to the bit-channels with the lowest Bhattacharyya parameters[1].

Due to the structure of G_n, the SC decoder for polar codes admits an efficient implementation that runs in $O(n \log n)$. This follows from the fact that for this choice of G, there is a recursive expression for the transition probabilities of the ith bit channel. That is, for all $i \in [n]$, if $i = 2\varphi$ then

$$\begin{aligned} \mathcal{W}_{G_n}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) = \\ \sum_{u_{2\varphi+1}} \frac{1}{2} \mathcal{W}_{G_{n/2}}^{(\varphi)}(y_0^{n/2-1}, u_{[2\varphi]_0} \oplus u_{[2\varphi]_1}|u_{2\varphi} \oplus u_{2\varphi+1}) \\ \cdot \mathcal{W}_{G_{n/2}}^{(i/2)}(y_{n/2}^{n-1}, u_{[2\varphi]_1}|u_{2\varphi+1}), \end{aligned} \tag{3}$$

and if $i = 2\varphi + 1$ then

$$\begin{aligned} \mathcal{W}_{G_n}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) = \\ \frac{1}{2} \mathcal{W}_{G_{n/2}}^{(\varphi)}(y_0^{n/2-1}, u_{[2\varphi]_0} \oplus u_{[2\varphi]_1}|u_{2\varphi} \oplus u_{2\varphi+1}) \\ \cdot \mathcal{W}_{G_{n/2}}^{(\lfloor i/2 \rfloor)}(y_{n/2}^{n-1}, u_{[2\varphi]_1}|u_{2\varphi+1}). \end{aligned} \tag{4}$$

$\mathcal{W}_{G_1}^{(0)} = \mathcal{W}$. Notice that, for $r \in \{0, 1\}$, $[2\varphi]_r$ is the set of all elements in $[2\varpi]$ that are equal to r modulo $\ell = 2$.

A *kernel* K is an invertible $\ell \times \ell$ matrix. As mentioned above, one can improve the performance of polar codes by replacing Arikan's kernel with larger kernels that admit better

[1] By the definition of polar codes, the values of the frozen bits are also required. If the channel is symmetric, then the frozen bits are all taken to be zero. For asymmetric channels, an assignment of the frozen bits that guarantees a vanishing probability of error is known to exist, however no practical method that finds such an assignment is known.

scaling exponents, and hence are considered to be more polarizing.

Next, we describe the matrix $G = G_{n,K}$ through which a polar code with kernel K is generated. Let $n = \ell^m$ and let R_n be the permutation matrix for which $u_0^{n-1} R_n = u_{[n]_0} u_{[n]_1} \dots u_{[n]_{\ell-1}}$, for all $u_0^{n-1} \in \mathcal{X}^n$. For an $\ell \times \ell$ kernel $K = (K_{r,s})$, define the matrix $G_{n,K}$ recursively by $G_{\ell,K} \stackrel{\text{def}}{=} K$ and $G_{n,K} \stackrel{\text{def}}{=} (I_{n/\ell} \otimes K) R_n (I_\ell \otimes G_{n/\ell,K})$. For ease of notation we denote the ith bit channel with respect to $G_{n,K}$ by $\mathcal{W}_{n,K}^{(i)}$. The recursive structure of the matrix $G_{n,K}$ induces recursive formulas for the bit channels as follows.

Lemma 1. *Let K be an $\ell \times \ell$ kernel that can be obtained by permuting the rows and columns of some lower triangular matrix. For all $i \in [n]$, if $i = \varphi\ell + \rho$ for some $\varphi \in [n/\ell]$ and $\rho \in [\ell]$ then*

$$\mathcal{W}_{n,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) = \frac{1}{2^{\ell-1}} \sum_{u_{\varphi\ell+\rho+1}^{(\varphi+1)\ell-1}} \prod_{s=0}^{\ell-1} \mathcal{W}_{n/\ell,K}^{(\varphi)}(y_{sn/\ell}^{(s+1)n/\ell-1}, T^{(s)}(u_{[\varphi\ell]})| \oplus_{r=0}^{\ell-1} K_{r,s} \cdot u_{\varphi\ell+r}),$$

where $T^{(s)}(u_{[\varphi\ell]}) \stackrel{\text{def}}{=} \oplus_{r=0}^{\ell-1} K_{r,s} \cdot u_{[\varphi\ell]_r}$.

Notice that $u_{[\varphi\ell]_r}$ is a vector of length φ and $K_{r,s} \in GF(2)$, hence $T^{(s)}(u_{[\varphi\ell]})$ is a vector of length φ as required by the definition of the φth bit channel. The term $\oplus_{r=0}^{\ell-1} K_{r,s} \cdot u_{\varphi\ell+r}$ is simply the inner product of $u_{\varphi\ell}^{(\varphi+1)\ell-1}$ with the sth column of K. Also notice that for $\ell = 2$, the expression in Lemma 1 reduces to the recursive formulas (3) and (4). From Lemma 1, it follows that a SC decoding algorithm at the kernel level, i.e., for a length-ℓ polar code with $G = K$, that has time complexity t can be extended to a SC decoding algorithm for a length-n polar code with kernel K that has time complexity $O(tn \log_\ell n)$. In particular, there exists an implementation for the SC decoder that runs in $O(2^\ell n \log_\ell n)$. In practice, this time complexity may be too large even for relatively small values of ℓ. For this reason we propose to use a special type of kernels that can simultaneously reduce the time complexity of the SC decoder and achieve better scaling exponents. These kernels are called *permuted kernels* since they are defined by a row permutation of the matrix $G_\ell = B_\ell F_2^{\otimes L}$, $\ell = 2^L$.

III. Permuted Kernels

In this section, we focus our discussion on polar codes of length $\ell = 2^L$, where the matrix G is an $\ell \times \ell$ permuted kernel. We will express the bit channels of these codes in terms of the bit channels of G_ℓ. The discussion in this section motivates the SC decoding algorithm that is formalized in Section IV.

A *permutation* π of length ℓ is a bijection $\pi : [\ell] \to [\ell]$. For a permutation π, the matrix permutation corresponding to π is denoted by M_π and defined by

$$(M_\pi)_{r,s} \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } s = \pi(r) \\ 0, & \text{otherwise,} \end{cases}$$

i.e., $u_0^{\ell-1} M_\pi = v_0^{\ell-1}$, where $v_{\pi(r)} = u_r$, for all $r \in [\ell]$. A *permuted kernel* with respect to π is defined by $K_\pi \stackrel{\text{def}}{=} M_\pi G_\ell = M_\pi B_\ell F_2^{\otimes L}$.

Let $\mathcal{C}_\pi$ be the polar code that is generated by $G = K_\pi$. For $i \in [\ell]$, let $D_i = \{\pi(0), \pi(1), \dots, \pi(i-1), \pi(i)\}$ and let $d_i = \max\{D_i\}$. For ease of notation, for all $i \in [n]$ we denote the ith bit channel with respect to G_{n,K_π} and G_n by $\mathcal{W}_{n,\pi}^{(i)}$ and $\mathcal{W}_n^{(i)}$, respectively.

Lemma 2. *For all $i \in [\ell]$, the ith bit channel with respect to K_π is equal to*

$$\mathcal{W}_{\ell,\pi}^{(i)}(y_0^{\ell-1}, u_0^{i-1}|u_i) = \sum_{v_{[d_i+1]\setminus D_i}} \mathcal{W}_\ell^{(d_i)}(y_0^{\ell-1}, v_0^{d_i-1}|v_{d_i}),$$

where $v_0^{\ell-1} = u_0^{\ell-1} M_\pi$, i.e., for all $r \in [\ell]$, $v_{\pi(r)} = u_r$.

Proof. By definition of the ith bit channel (1), we have that

$$\begin{aligned}
&\mathcal{W}_{\ell,K}^{(i)}(y_0^{\ell-1}, u_0^{i-1}|u_i) \\
&\quad = \frac{1}{2^{\ell-1}} \sum_{u_{i+1}^{\ell-1}} \mathcal{W}^\ell(y_0^{\ell-1}|u_0^{\ell-1} K_\pi) \\
&\quad = \frac{1}{2^{\ell-1}} \sum_{u_{i+1}^{\ell-1}} \mathcal{W}^\ell(y_0^{\ell-1}|(u_0^{\ell-1} M_\pi) G_\ell) \\
&\quad = \frac{1}{2^{\ell-1}} \sum_{v_{[\ell]\setminus D_i}} \mathcal{W}^\ell(y_0^{\ell-1}|v_0^{\ell-1} G_\ell) \\
&\quad = \sum_{v_{[d_i+1]\setminus D_i}} \frac{1}{2^{\ell-1}} \sum_{v_{d_i+1}^{\ell-1}} \mathcal{W}^{(\ell)}(y_0^{\ell-1}|v_0^{\ell-1} G_\ell) \\
&\quad = \sum_{v_{[d_i+1]\setminus D_i}} \mathcal{W}_\ell^{(d_i)}(y_0^{\ell-1}, v_0^{d_i-1}|v_{d_i}).
\end{aligned}$$

□

By Lemma 2, the ith bit channel with respect to K_π can be expressed in terms of the d_ith bit channel with respect to G_ℓ. In particular, using a SC decoder for the length-ℓ polar code with Arikan's kernel, one can define a SC decoder for the code $\mathcal{C}_\pi$ as follows. For all $i \in [\ell]$, to decode $\hat{u}_i$, given the previous decoded bits $\hat{u}_0^{i-1}$, the transition probabilities $\mathcal{W}_\ell^{(d_i)}(y_0^{\ell-1}, v_0^{d_i-1}|v_{d_i})$ are calculated for all possible choices of v_s, $s \in [d_i + 1] \setminus D_i$, where $v_{s=\pi(r)} = \hat{u}_r$ if $r \in [i]$. Thus, 2^{d_i+1-i} transition probabilities for the ith bit channel of G_ℓ are calculated in order to calculate the ith pair of probabilities defined in (2).

IV. Formalization of SC Decoder for Permuted Kernels

In this section we formalize our SC decoder for a length-ℓ code $\mathcal{C}_\pi$ defined by a permuted kernel K_π. As mentioned above, SC decoding for $\mathcal{C}_\pi$ is equivalent to SC decoding of a length-ℓ polar code with kernel F_2 that decides on the input bits in a permuted order according to π. For this reason, we call our SC decoding scheme *permuted successive cancellation* (PSC) decoding. We present an implementation of PSC decoding for any permutation π, which requires significantly less computational power compared to the conventional SC decoder at the kernel level. Analysis of the time complexity is also presented.

For $i \in [\ell]$, let $\hat{u}_0^{i-1}$ be the bits decoded so far. Recall that in the ith step, the SC decoder calculates the pair of probabilities defined in (2). Denote these probabilities by

$$Q_i[0] \stackrel{\text{def}}{=} \mathcal{W}_{\ell,\pi}^{(i)}(y_0^{\ell-1}, \hat{u}_0^{i-1} | u_i = 0),$$
$$Q_i[1] \stackrel{\text{def}}{=} \mathcal{W}_{\ell,\pi}^{(i)}(y_0^{\ell-1}, \hat{u}_0^{i-1} | u_i = 1).$$

For v_0^i, let

$$P_i[v_0^i] \stackrel{\text{def}}{=} \mathcal{W}_{\ell}^{(i)}(y_0^{\ell-1}, v_0^{i-1} | v_i).$$

Recall that $D_i = \{\pi(0), \pi(1), \ldots, \pi(i)\}$ and d_i is the maximum over D_i. The *decoding window* in the ith step is denoted by $DW_i = [d_i + 1] \setminus D_i$. By Lemma 2,

$$Q_i[b] = \sum_{v_s:\ s \in DW_i} P_{d_i}[v_0^{d_i}], \tag{5}$$

where $v_{\pi(i)} = u_i = b$ and for all $r \in [i]$, $v_{\pi(r)} = \hat{u}_r$.

The PSC decoding utilizes the conventional recursive SC decoder for the calculation of $P_{d_i}[v_0^{d_i}]$ with different settings (corresponding to all choices for v_s, $s \in DW_i$) so that it can generate the desired pair of probabilities $Q_i[b]$, $b \in \{0, 1\}$. The decoding window $DW_\pi(i)$, refers to the indices of the input bits in $v_0^{d_i}$ that are not decoded yet, and hence have to be set temporarily before execution of an instance of the traditional SC decoder. However, the original (and efficient) recursive algorithm requires that $\{\pi(0), \pi(1), \cdots, \pi(i)\} = \{0, 1, \cdots, i\}$, which may not be the case for many $i \in [\ell]$. To overcome this problem, the SC decoder must also calculate $P_j[v_0^j]$ for all $j \in [\tau_i + 1, d_i + 1]$, where

$$\tau_i \stackrel{\text{def}}{=} \begin{cases} \min(DW_i) - 1, & \text{if } DW_i \neq \varnothing \\ i - 1, & \text{otherwise.} \end{cases}$$

In other words, τ_i denotes the largest index j such that $v_0, v_1, \cdots, v_j$ have all been decoded before calculating the pair of probabilities associated with $u_i = v_{\pi(i)}$. To better understand the importance of this parameter, we recall that the conventional SC decoding algorithm requires memory, in which it stores both calculated probabilities and decoded values associated with the previous bit channels and bit channels in inner-layer nodes. Utilizing this memory, the algorithm avoids duplicate calculations and achieves $\mathcal{O}(n \log(n))$ overall complexity in decoding all of the bit channels. This old information, i.e., probabilities and decoded values of the previous bit channels, is only helpful if it can be propagated back through the polarization layers and reveal unknown inner nodes. However, that is only possible when the indices of the already-decoded bits form a consecutive set. The parameter τ_i determines the largest subset of consecutive indices of the already-decoded bits and hence acts as the starting point for the recursive SC decoding. For simplicity, we call the sets $[\tau_i + 1]$ and $[\tau_i + 1, d_i + 1]$ the *decoding tail* and *decoding lookahead* of u_i, respectively. Note that, depending on π, for some $i \in [\ell]$ we may have $DW_i = \varnothing$, which is equivalent to $\{\pi(0), \cdots, \pi(i)\} = \{0, 1, \cdots, i\}$ and hence only a single instance of the traditional SC decoder is required.

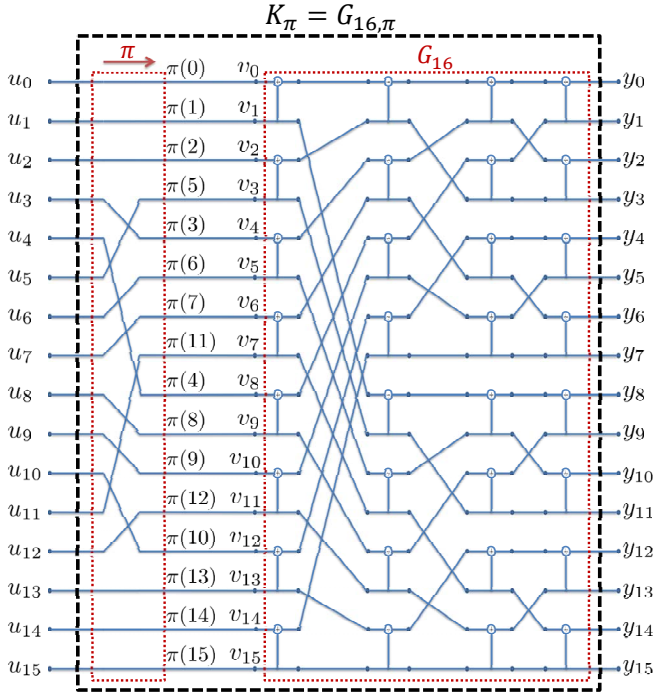

Fig. 1: Structure of the permuted 16×16 kernel K_π from Example 1. The recursive structure of G_{16} is exploited by the PSC decoder for this kernel.

Example 1. *Consider the permuted binary* 16×16 *kernel* $K_\pi = G_{16,\pi}$*, shown in Figure 1, where*

$$\pi = (0, 1, 2, 4, 8, 3, 5, 6, 9, 12, 7, 11, 13, 14, 15).$$

The scaling exponent of this kernel for BEC is $\mu(K_\pi) = 3.479$.

To decode u_0, u_1*, and* u_2 *we can simply call the traditional recursive SC decoder. However, to decode* u_3*, one needs to calculate the pair of probabilities*

$$Q_3[b] = P_4[\hat{v}_0 \hat{v}_1 \hat{v}_2 0 b] + P_4[\hat{v}_0 \hat{v}_1 \hat{v}_2 1 b], \tag{6}$$

where $\hat{v}_{\pi(r)}$ *denotes the decoded bit* $\hat{u}_r$*. By definition, we have that* $D_3 = \{0, 1, 2, 4\}$*,* $DW_3 = \{3\}$*, and* $\tau_3 = 2$*. Each term in the right-hand side can be calculated using the recursive SC decoder with input parameters* $y_0^{15}, \hat{v}_0^2, v_3 = 0$ *and* $y_0^{\ell-1}, \hat{v}_0^2, v_3 = 1$*. However, in order to calculate* $P_4[\hat{v}_0 \hat{v}_1 \hat{v}_2 v_3 v_4]$*, the recursive SC must first calculate* $P_3[\hat{v}_0 \hat{v}_1 \hat{v}_2 v_3]$.

Moving forward, to decode the next unknown bit u_4*, it is desired to output the probability pair*

$$Q_4[b] = \sum_{v_3 v_5 v_6 v_7} P_8[\hat{v}_0 \hat{v}_1 \hat{v}_2 v_3 \hat{v}_4 v_5^7 b] \tag{7}$$

which consists of $2^4 = 16$ *terms. In this case,* $D_4 = \{0, 1, 2, 4, 8\}$*,* $DW_4 = \{3, 5, 6, 7\}$*, and* $\tau_3 = 2$*. Again, in order to calculate* $P_8[\hat{v}_0 \hat{v}_1 \hat{v}_2 v_3 \hat{v}_4 v_5^7 b]$*, the recursive SC must first calculate* $P_j[\hat{v}_0 \hat{v}_1 \hat{v}_2 v_3^j]$*, for all* $j \in [\tau_3 + 1, 7]$.

Surprisingly, the decoding of u_5 *follows a very similar formulation to (7), as*

$$Q_5[b] = \sum_{v_5 v_6 v_7} P_8[\hat{v}_0 \hat{v}_1 \hat{v}_2 b \hat{v}_4 v_5^7 \hat{v}_8]. \tag{8}$$

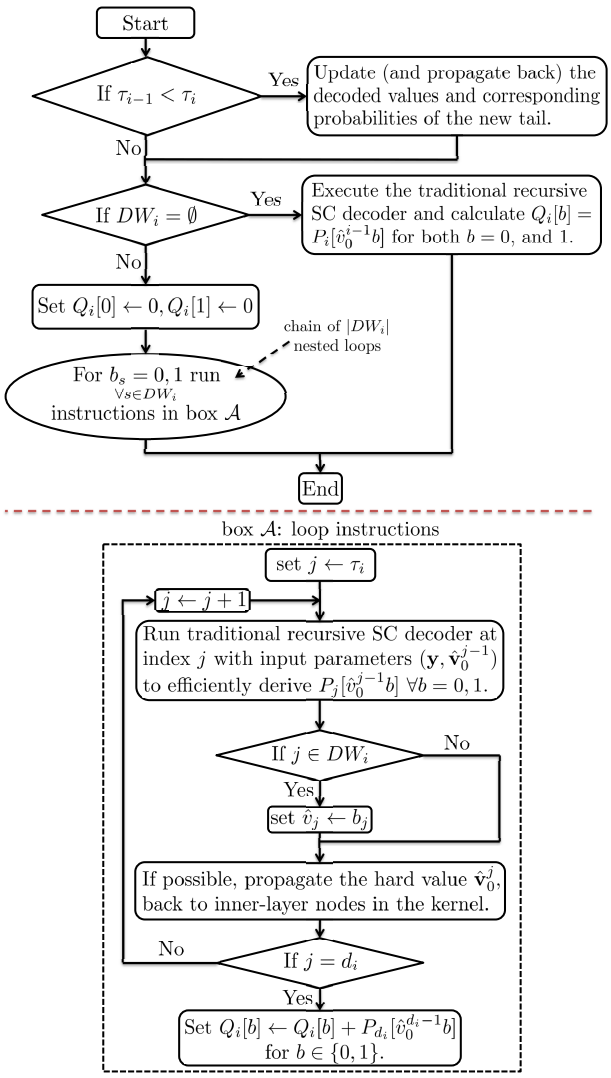

Fig. 2: Structure of the algorithm that calculates the transition probabilities of the ith bit channel $\mathcal{W}_{\ell,\pi}^{(i)}(y_0^{\ell-1}, \hat{u}_0^{i-1}|u_i)$ for PSC decoding.

Accordingly, u_5 can be decoded by simulating the conventional SC at the same bit channel[2].

An overview of the general PSC decoder is given in Figure 2. The diagram is self-explanatory. We add a few comments here to clarify some of the instructions. The reader is encouraged to read the full version of the paper for the detailed decoding algorithm (similar to source code) at both the kernel and the full-length level, where more details about efficient memory addressing, and duplication-free calculations are presented.

The algorithm starts by checking updates on the decoding tail. The decoding tail (and its propagated values) correspond to the fixed inner-layer nodes in the decoder structure, and hence by storing their corresponding probabilities/hard-values in memory we can avoid duplicate calculations. Next, the algorithm moves to the first unknown node, i.e., τ_i, and enters a nested chain of for-loops which alternate between all $2^{|DW_i|}$ initializations for the unknown bits in the decoding window of u_i. At each such initialization, the algorithm runs the traditional successive cancellation for all $j \in [\tau_i + 1, d_i + 1]$ with input parameters $y_0^{\ell-1}, \hat{v}_0^{j-1}$, where $\forall s \in [j]$, $\hat{v}_s$ is given either from the previously decoded values $\hat{u}_0^{i-1}$, or the temporary initialization of the elements in the decoding window. It is observed that both the size of the decoding window and the size of the decoding lookahead have a significant role in determining the time complexity of the decoding algorithm. In the next section, we present the simulations results for multiple permutations, along with a more detailed comparison on the complexity increase for each permutation.

[2]It is possible to exploit this fact by storing all of the 2×2^4 terms in (7) in an auxiliary memory to avoid duplicate calculations in the future stages (decoding u_5, u_6, and u_7). We refer the reader to the full version of the paper, where we explain this extension in detail, its improvement in throughput, and the extra required memory.

V. Simulation Results

In this section, we discuss the complexity increase in more detail and compare the performance of the polar codes based upon different permuted kernels with respect to their overall decoding complexity.

As explained earlier, the computational complexity of the permuted SC decoder for the ith bit channel in the permuted kernel may be more than the corresponding ith bit channel in the traditional $\ell \times \ell$ kernel. This is mainly because computing $Q_i[b]$ requires a summation over $2^{|DW_i|}$ terms, where each term can be computed by running $d_i - \tau_i$ instances of the conventional SC decoder; see (8).

However, not all such instances consume the same amount of computational power. Hence, to arrive at an explicit formulation of the computational complexity for the proposed algorithm, we need to analyze each instance separately. It is to be noted that time complexity and space complexity are the two main notions of computational complexity. This paper focuses on improving the decoder speed (throughput) and ignores the efficiency of the memory management. In the full version, we propose algorithms that sacrifice space efficiency for ultimate throughput, or vice versa.

The speed of a successive cancellation decoder is usually limited to the number of mathematical operations that have to be executed sequentially. This sequence in the conventional SC decoder consists of a double visit for each of $n \log(n)$ nodes in the decoder structure; the first visit is to output a pair of probabilities for the nodes on their left (closer to u_i's), and the second visit is to calculate and update the probabilities for the nodes on the right side (closer to y_i's). Let us define the *normalized complexity fraction* at bit channel i, ρ_i, to be the fraction of the total $2n \log(n)$ computational power that is spent on decoding W_{i,G_n}. It is easy to track the number of nodes contributing to revealing each v_i, and hence a simple program can derive the explicit values of ρ_i's. Table I shows the evaluation of ρ_i's for K_π from Example 1.

i	odd	$\{2, 6, 10, 14\}$	$\{4, 12\}$	$\{8\}$	$\{0\}$
ρ_i	0.008%	0.039%	0.101%	0.226%	0.351%

TABLE I: Table of normalized complexity fractions for 4^{th} power or Arikan's kernel. $\rho_i \times 2 \times 16 \log(16) =$ the number of visits (out of $2 \times 16 \log(16)$) the regular SC decoder makes on the inner nodes while decoding the ith bit channel.

As explained in the previous section, the PSC decoder executes the instances of traditional SC at each index i multiple times. Let us define the *frequencies*, f_i, by the total number of calls made by PSC on traditional SC at index i. The following theorem gives an explicit formula for the computational complexity of PSC decoder at the kernel level.

Theorem 1. *The* normalized computational complexity *of the PSC decoder at kernel level, $\mathcal{CC}(K_{\ell,\pi})$, is given by*

$$\mathcal{CC}(K_{\ell,\pi}) = \sum_{i=0}^{\ell-1} f_i \rho_i. \tag{9}$$

The proof is straightforward and thus it is omitted. Note that the overall decoding complexity at the full-length also scales with the same scalar as in (9). Table II tabulates the evaluation of f_i's for the permuted kernel in Example 1. Utilizing (9), we have $\mathcal{CC}(K_{16,\pi}) = 7.03$.

i	0	1	2	3	4	5	6	7
f_i	1	1	1	4	4	4	8	2
i	8	9	10	11	12	13	14	15
f_i	22	6	4	4	4	1	1	1

TABLE II: The frequency table for K_π from Example 1.

Example 2. *Consider a different permuted binary* 16×16 *kernel K_σ, defined by the permutation*

$$\sigma = (0, 1, 2, 3, 4, 6, 8, 10, 5, 9, 7, 11, 12, 13, 14, 15).$$

The scaling exponent of K_σ for the binary erasure channel is given by $\mu = 3.541$, which is again smaller than $\mu(F_2)$. Hence one would expect a performance improvement upon conventional polar codes. However, since $\sigma(i) = i$, for all $i \in [15] \setminus \{5, 6, 7, 8, 9, 10\}$, it is also expected to observe a smaller increase in computation complexity. We can verify this by following the formulation in (9), and show $\mathcal{CC}(K_\sigma) = 3.2$.

We leave the discussion on space complexity to the full version of the paper, where we also introduce even faster decoding algorithms which can increase the throughput even further at the cost of extra memory.

Figure 3 depicts the performance comparison of three polar codes at length $n = 2^8$ over binary erasure channels. Two of these codes are constructed via K_σ and K_π, respectively, and the other one is the conventional Arikan's polar code, constructed from the 2×2 kernel F_2. It is observed that the polar code constructed from K_π, and decoded with PSC, outperforms both the conventional polar code and the code constructed by K_σ. The latter outperforms the conventional polar code. The results are in agreement with the convention that the smaller the scaling exponent, the better the polar code performance. Notice that for some erasure channels the polar code constructed from K_π outperforms Arikan's polar code even with the ML decoder.

It is to be noted that the conventional construction algorithms such as [13] cannot be applied directly for the kernels with $\ell \geq 4$ due to the exponential increase in the number of bit channel outputs. In this paper, we utilized a Monte-Carlo construction algorithm, which may be improved by using a larger number of iterations. We leave the study of low-complexity construction algorithms for future work.

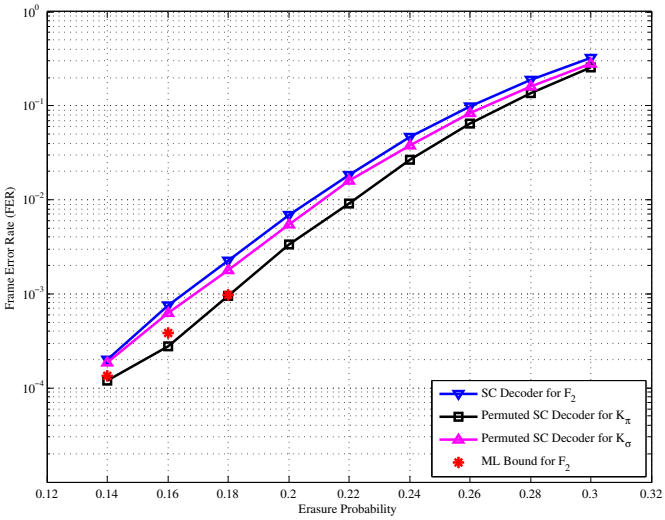

Fig. 3: Frame-error-rate comparison of three polar codes over binary erasure channels. The tree codes are constructed with F_2, K_σ, and K_π, respectively, at code length $n = 2^8$, and decoded with SC and PSC algorithms. All codes are optimized for the channel BEC(0.2) and rate $R = \frac{3}{5}$. The ML bound on performance of the F_2 polar code is also given, based upon methods in [12].

REFERENCES

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. on Inform. Theory,* vol. 55, pp. 3051–3073, July 2009.

[2] E. Arikan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Commun. Letters*, vol. 12, pp. 447–449, June 2008.

[3] E. Arikan and E. Telatar, "On the rate of channel polarization," *Proc. IEEE Int. Symp. Inform. Theory,* pp. 1493–1495, Seaoul, South Korea, July 2009.

[4] K. Chen, N. Kai, and L. Jiaru, "Improved successive cancellation decoding of polar codes," *IEEE Trans. on Commun.,* vol. 61, pp. 3100–3107, August 2013.

[5] U. U. Fayyaz and J. R. Bany, "Polar codes for partial response channels," *Proc. IEEE Int. Conf. Commun.,* Budapest, Hungary, pp. 4337–4341, June 2013.

[6] A. Fazeli and A. Vardy. "On the scaling exponent of binary polarization kernels, *Proc. Allerton Conf. on Commun., Control, and Computing,* October 2014.

[7] T. C. Gulcu, M. Ye, and A. Barg, "Construction of polar codes for arbitrary discrete memoryless channels," *Proc. IEEE Symp. Inform. Theory,* pp. 51–55, Barcelona, Spain, July 2016.

[8] J. Guo, M. Qin, A. G Fàbregas, and P. H. Siegel, "Enhanced belief propagation decoding of polar codes through concatenation," *Proc. IEEE Int. Symp. Inform. Theory,* pp. 2897–2991, Honolulu, Hawaii, July 2014.

[9] H. S. Hassani, K. Alishahi, and R. L. Urbanke, "Finite-length scaling of polar codes," *IEEE Trans. on Inform. Theory,* vol. 60, pp. 5875–5898, July 2014.

[10] S. B. Korada, E. Sasoglu, and R. L. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. on Inform. Theory,* vol. 56, pp. 6253–6264, December 2010.

[11] V. Miloslavskaya, and P. Trifonov, "Sequential decoding of polar codes with arbitrary binary kernel," *Proc. IEEE Information Theory Workshop,* pp. 376–380, San Diego, February 2014.

[12] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. on Inform. Theory,* vol. 61, pp. 2213–2226, May 2015.

[13] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. on Inform. Theory,* vol. 59, pp. 6562–6582, October 2013.

[14] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. on Commun.,* vol. 60, pp. 3221-3227, November 2012.

[15] P. Trifonov, "Binary successive cancellation decoding of polar codes with Reed-Solomon kernel," *Proc. IEEE Int. Symp. Inform. Theory,* pp. 2972–2976, Honolulu, Hawaii, July 2014.