A Deterministic Algorithm for Computing the Weight Distribution of Polar Codes

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

***Abstract*—We present a deterministic algorithm for computing the entire weight distribution of polar codes. As the first step, we derive an efficient recursive procedure to compute the weight distributions that arise in successive cancellation decoding of polar codes *along any decoding path*. This solves the open problem recently posed by Polyanskaya, Davletshin, and Polyanskii. Using this recursive procedure, we can compute the entire weight distribution of certain *polar cosets* in time $O(n^2)$. Any polar code can be represented as a disjoint union of such cosets; moreover, this representation extends to polar codes with dynamically frozen bits. This implies that our methods can be also used to compute the weight distribution of polar codes with CRC precoding, of polarization-adjusted convolutional (PAC) codes and, in fact, general linear codes. However, the number of polar cosets in such representation *scales exponentially* with a parameter introduced herein, which we call the *mixing factor*. To reduce the exponential complexity of our algorithm, we make use of the fact that polar codes have a large automorphism group, which includes the lower-triangular affine group $\mathrm{LTA}(m,2)$. We prove that $\mathrm{LTA}(m,2)$ acts transitively on certain subsets of polar codes, thereby drastically reducing the number of polar cosets we need to evaluate. This complexity reduction makes it possible to compute the weight distribution of *any polar code* of length up to $n = 128$.**

***Index Terms*—coding theory, polar codes, weight distribution, monomial codes, permutation group**

I. Introduction

The weight distribution of an error correction code counts the number of codewords in this code of any given weights. The weight distribution is one of the main characteristic of a code, and it plays a significant role in determining the capabilities of error detection and correction of a given code.

Polar coding, pioneered by Arıkan [1], gives rise to the first explicit family of codes that provably achieve capacity with efficient encoding and decoding for a wide range of channels. Since it's invention, the interest and research effort on polar codes has been constantly rising in academia and industry in the past decade. Now polar codes have been adopted as part of the fifth generation (5G) wireless communications standard [2]. Understanding the weight distribution of polar codes thus has great importance in both theoretical and practical aspects.

There are many prior attempts towards the weight distribution of polar codes. In [3], the authors provide an explicit formula for the number of codewords of minimal weight in polar codes. In [4], the authors propose a way to search for low weight codewords of polar codes by transmitting an all-zero codeword through a high SNR AWGN channel in simulation, and decode the received word using successive cancellation list (SCL) decoding with a huge list size. The authors in [5] improve this approach in terms of its memory usage. In [6], a probabilistic computation method is proposed to estimate the weight distribution of polar code. This method is later improved in [7] in both accuracy and complexity. We remark that besides the results in [3], all the aforementioned approaches in the literature are non-deterministic, and they only provide an estimate on the weight distribution of polar codes. Also, in [4] and [5], only part of the weight distribution can be derived.

In this paper, we present a deterministic algorithm for computing the entire weight distribution of polar codes. Our algorithm is based on an efficient recursive procedure to compute the weight distribution of certain *polar cosets* to be defined later, that arise in successive cancellation decoding. In a prior work by Polyanskaya, Davletshin, and Polyanskii [8], they introduce an algorithm that computes the weight distribution of these polar cosets along the all-zero decoding path. And how to compute the weight distribution of polar cosets along *any* decoding path remains open. In this work, we solve this problem by establishing a recursive relation followed by the weight enumerating functions (WEF) of those cosets. Our recursive computation procedure has two applications: computing the entire weight distribution of polar codes; analysing the successive cancellation (SC) decoding performance as shown in [8].

To compute the entire weight distribution of a polar code, we first represent the code as a disjoint union of some polar cosets, and then obtain the WEF of the entire code as the sum of the WEF of those cosets. This representation extends to polar codes with dynamically frozen bits. This implies our method can be used to compute the weight distribution of polar codes with CRC precoding [9], of polar subcodes [10], of polarization-adjusted convolutional (PAC) codes [11], etc. Since any binary linear codes can be represented as polar codes with dynamically frozen bits [12], our algorithm applies to general linear codes as well. However, the number of polar cosets in this representation scales exponentially with a code parameter that we refer as the *mixing factor*. To analyse the complexity of our algorithm, we provide upper bounds on the mixing factors of polar codes for block lengths up to 1024.

Our algorithm works for polar codes in a general setting, where we are allowed to select any subsets of rows in the polar transformation matrix as generators for the code. In a more restricted definition of polar codes, where we only select the bit-channels with the smallest Bhattacharyya parameters, we can reduce the exponential complexity of our algorithm using automorphism group of polar codes. Polar codes as decreasing monomial codes [3] have a large automorphism group, which includes the lower triangular affine group $\mathrm{LTA}(m,2)$ [3]. We prove that $\mathrm{LTA}(m,2)$ acts transitively on certain subsets of polar codes, which allows us to drastically reduce the number of cosets we need to evaluate. This complexity reduction

makes it possible to compute the weight distribution of any polar codes up to length 128. In particular, since Reed-Muller codes can also be viewed as decreasing monomial codes, our complexity reduction applies to Reed-Muller codes as well. This enables our algorithm to compute the entire weight distribution of Reed-Muller codes for all rates and length up to 128 with reasonable complexity.

II. Polar Codes and Polar Cosets

First, we give the definition for *polar cosets* and their weight enumerating functions (WEF). For the details of polar codes, we refer the readers to Arıkan's seminal paper [1]. In our 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 of the symbols in the vectors start with 0. We use $\boldsymbol{u}_i$ to represent the length $(i+1)$ vector $(u_0, u_1, \cdots, u_i)$. Also, we use $\boldsymbol{u}_{i,\text{even}}$ and $\boldsymbol{u}_{i,\text{odd}}$ to denote the subvectors of $\boldsymbol{u}_i$ with only even indices and only odd indices respectively.

Assuming $n = 2^m$, an (n,k) polar code is a binary linear block code generated by k rows of the polar transformation matrix $G_n = B_n K_2^{\otimes m}$, where

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

$K_2^{\otimes m}$ is the m-th Kronecker power of K_2, and B_n is an $n \times n$ bit-reversal permutation matrix. We denote by $\mathcal{A}$ the set of information indices, which are the row indices for the k selected rows in G_n. And we denote by $\mathcal{F}$ the set of frozen indices with $\mathcal{F} = \{0,1,\cdots,n-1\}\backslash\mathcal{A}$. We denote by $\boldsymbol{u}$ the information vector of length n, and by $\boldsymbol{c} = \boldsymbol{u}G_n$ the codeword.

Denote by $g_0, g_1, \cdots, g_{n-1}$ the rows in matrix G_n. For $0 \leqslant i \leqslant n-1$, let $\boldsymbol{u}_{i-1} \in \{0,1\}^{i-1}$ and $u_i \in \{0,1\}$, we define the ***polar coset*** $C_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)$ given $\boldsymbol{u}_{i-1}$ and u_i as

$$C_n^{(i)}(\boldsymbol{u}_{i-1}, u_i) = \sum_{j=0}^{i} u_j g_j + \langle g_{i+1}, \cdots, g_{n-1} \rangle, \qquad (1)$$

where $\langle g_{i+1}, \cdots, g_{n-1} \rangle$ is the linear space spanned by rows $g_{i+1}, \cdots, g_{n-1}$. In this definition and the rest of this paper, we assume u_0^{i-1} to be the empty vector when $i = 0$. We denote the empty vector by ψ. We also define $A_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)(X)$ as the WEF for $C_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)$ to be the polynomial:

$$A_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)(X) = \sum_{w=0}^{n} A_w X^w, \qquad (2)$$

where A_w is the number of words in $C_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)$ with Hamming weight w.

III. Computing the WEF of Polar Cosets

In this section, we present one of the key results of this paper: a recursive procedure that computes the WEFs for polar cosets $C_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)$ with arbitrary $\boldsymbol{u}_{i-1}$. Recently in [8], the authors introduce an algorithm that computes the weight distribution of polar coset $C_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)$ with $\boldsymbol{u}_{i-1} = \boldsymbol{0}$. And how to efficiently compute the weight distribution for $C_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)$ with any $\boldsymbol{u}_{i-1}$ remains open. In this section, we present a recursive computation procedure that solves this problem. This procedure is based on a recursive relation shown in Proposition 1. We leave the proof of Proposition 1 to the extended version of this paper due to lack of space [13].

Algorithm 1: CalcA$(n, \boldsymbol{u}_{i-1})$

Input: block length n and vector $\boldsymbol{u}_{i-1}$
Output: a pair of polynomials $\left(A_n^{(i)}(\boldsymbol{u}_{i-1}, 0)(X), A_n^{(i)}(\boldsymbol{u}_{i-1}, 1)(X)\right)$

```
if n = 1 then                                   // Stopping condition
    return (1, X)
else
    if i mod 2 = 0 then
        (f_0, f_1) ← CalcA(n/2, u_{i-1,even} ⊕ u_{i-1,odd})
        (g_0, g_1) ← CalcA(n/2, u_{i-1,odd})
        return (f_0 g_0 + f_1 g_1, f_0 g_1 + f_1 g_0)
    else
        (f_0, f_1) ← CalcA(n/2, u_{i-2,even} ⊕ u_{i-2,odd}^{i-2})
        (g_0, g_1) ← CalcA(n/2, u_{i-2,odd})
        if u_{i-1} = 0 then
            return (f_0 g_0, f_1 g_1)
        else
            return (f_1 g_0, f_0 g_1)
```

Proposition 1. *For any $m \geqslant 0$, $n = 2^m$, and $0 \leqslant i \leqslant n-1$,*

$$A_{2n}^{(2i)}(\boldsymbol{u}_{2i-1}, u_{2i})(X) = \sum_{u_{2i+1} \in \{0,1\}} A_n^{(i)}(\boldsymbol{u}_{2i-1,even} \oplus \boldsymbol{u}_{2i-1,odd},\, u_{2i} \oplus u_{2i+1})(X) \cdot A_n^{(i)}(\boldsymbol{u}_{2i-1,odd}, u_{2i+1})(X), \qquad (3)$$

and

$$A_{2n}^{(2i+1)}(\boldsymbol{u}_{2i}, u_{2i+1})(X) = A_n^{(i)}(\boldsymbol{u}_{2i-1,even} \oplus \boldsymbol{u}_{2i-1,odd},\, u_{2i} \oplus u_{2i+1})(X) \cdot A_n^{(i)}(\boldsymbol{u}_{2i-1,odd}, u_{2i+1})(X). \qquad (4)$$

Using (3) and (4) in Proposition 1, we can compute the WEF $A_n^{(i)}(\boldsymbol{u}_{i-1}, u_i)(X)$ of any polar coset recursively with the stopping conditions

$$A_1^{(0)}(\psi, 0) = 1, \quad A_1^{(0)}(\psi, 1) = X. \qquad (5)$$

The steps of this recursive procedure are shown in Algorithm 1. Next we analyze its complexity. Denote by $T(n)$ the run time for Algorithm 1, where n is the block length. Notice n is also the maximal degree of the polynomials f_0, f_1, g_0, g_1 in the algorithm. For every recurrence, the computation is divided into two recursive calls on the same algorithm, each with half of the parameter n. And there are up to three extra polynomial operations including addition and multiplication. Assume multiplication of two degree-n polynomials takes time $O(n^2)$, the recurrence relation shows $T(n) = 2T(n/2) + O(n^2)$, which by the Master theorem [14] gives us $T(n) = O(n^2)$. So Algorithm 1 has complexity $O(n^2)$. This complexity may be improved assuming multiplication of two degree-n polynomials takes time $O(n \log n)$ with the Fast-Fourier Transform.

IV. Computing the Entire Weight Distribution

In this section, we introduce our main deterministic algorithm that computes the entire weight distribution of polar codes, and polar codes with dynamically frozen bits.

A. Representing Polar Codes with Polar Cosets

First, We define the ***last frozen index*** and the ***mixing factor*** for polar codes as follows:

Definition 1. *Consider an (n,k) polar code $\mathbb{C}$ specified in terms of its set of information indices $\mathcal{A}$. With $\mathcal{F}$ being the set of frozen indices, we let $\tau(\mathbb{C}) = \max\{\mathcal{F}\}$ denote its **last frozen index** and define its **mixing factor** as:*

$$\mathrm{MF}(\mathbb{C}) = \big|\{i \in \mathcal{A} \ : \ i < \tau(\mathbb{C})\}\big| \tag{6}$$

Loosely speaking, the mixing factor $\mathrm{MF}(\mathbb{C})$ counts the number of information bits in $\mathbb{C}$ that are *mixed-in* among the frozen bits. To show that any polar code can be represented as a disjoint union of polar cosets, let's start with an example:

Example 1. The (16,11,4) extended Hamming code $\mathbb{C}$ can be generated by rows in the polar transformation matrix G_{16}. So we can view $\mathbb{C}$ as a polar code of length 16. The polar transformation matrix G_{16} is given by

$$\begin{array}{c} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \end{array} \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0 \\ 1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0 \\ 1&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0 \\ 1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0 \\ 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0 \\ 1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{bmatrix} \tag{7}$$

In (7), the information bits are highlighted in red and blue, and the frozen bits are black. We color the information bits that are mixed-in among the frozen bits in red, and color the rest of the information bits in blue. The last frozen bit of $\mathbb{C}$ is u_8, so $\tau(\mathbb{C}) = 8$. The mixing factor of $\mathbb{C}$ counts the number of red bits, so $\mathrm{MF}(\mathbb{C}) = 4$.

For any vector $\boldsymbol{u}_7$ with $u_0 = u_1 = u_2 = u_4 = 0$, and u_3, u_5, u_6, u_7 taking some values in $\{0,1\}$, the polar coset $C_{16}^{(8)}(\boldsymbol{u}_7, u_8 = 0)$ is a subset of $\mathbb{C}$. There are $2^4 = 16$ such disjoint polar cosets, and code $\mathbb{C}$ is their union:

$$\mathbb{C} = \bigcup_{\boldsymbol{u}_7 \in \{0,1\}^8:\ u_0=u_1=u_2=u_4=0} C_{16}^{(8)}(\boldsymbol{u}_7, u_8 = 0) \tag{8}$$

Therefore, we can compute the WEF $A_{\mathbb{C}}(X)$ for the entire code $\mathbb{C}$ as the sum of the WEFs for those cosets:

$$A_{\mathbb{C}}(X) = \sum_{\boldsymbol{u}_7 \in \{0,1\}^8:\ u_0=u_1=u_2=u_4=0} A_{16}^{(8)}(\boldsymbol{u}_7, 0)(X) \tag{9}$$

As shown in Example 1, in general, any polar code can be represented as a disjoint union of polar cosets:

Proposition 2. *Consider a polar code $\mathbb{C}$ with the set of frozen indices $\mathcal{F}$, and its last frozen index $\tau(\mathbb{C}) = \tau$. We can represent $\mathbb{C}$ as a disjoint union of polar cosets as follows:*

$$\mathbb{C} = \bigcup_{\boldsymbol{u}_{\tau-1} \in \{0,1\}^\tau:\ u_i = 0 \text{ for all } i \in \mathcal{F}} C_n^{(\tau)}(\boldsymbol{u}_{\tau-1}, u_\tau = 0) \tag{10}$$

The number of polar cosets in this representation equals $2^{\mathrm{MF}(\mathbb{C})}$. Denote the WEF for the entire code $\mathbb{C}$ as $A_{\mathbb{C}}(X)$, then $A_{\mathbb{C}}(X)$ is the sum of the WEFs for those polar cosets:

$$A_{\mathbb{C}}(X) = \sum_{\boldsymbol{u}_{\tau-1} \in \{0,1\}^\tau:\ u_i = 0 \text{ for all } i \in \mathcal{F}} A_n^{(\tau)}(\boldsymbol{u}_{\tau-1}, 0)(X) \tag{11}$$

B. Representing Polar Codes with Dynamically Frozen Bits

Our representation for polar codes extends to polar codes with *dynamically frozen bits*. Polar codes with dynamically frozen bits, first introduced in [15], are polar codes where each of the frozen bit u_i is not fixed to be zero, but set to be a linear function of its previous bits as $u_i = f_i(u_0, u_1, \cdots, u_{i-1})$. We refer the collection of these functions $\{f_i : i \in \mathcal{F}\}$ as the *dynamic constraints* for this code. Polar codes with dynamically frozen bits include polar codes with CRC precoding [9], polar subcodes [10], polarization-adjusted convolutional (PAC) codes [11], etc. Since any binary linear codes can be represented as polar codes with dynamically frozen bits [12], our algorithm extends to all binary linear codes as well.

We define the last frozen index and the mixing factor for polar codes with dynamically frozen bits the same way as in Definition 1. Then Proposition 2 extends to polar codes with dynamically frozen bits as follows:

Proposition 3. *Consider a polar code $\mathbb{C}$ with dynamically frozen bits, with the set of frozen indices $\mathcal{F}$ and the dynamic constraints $\{f_i : i \in \mathcal{F}\}$. Let its last frozen index be $\tau(\mathbb{C}) = \tau$. We can represent $\mathbb{C}$ as a disjoint union of polar cosets as follows:*

$$\mathbb{C} = \bigcup_{\substack{\boldsymbol{u}_{\tau-1} \in \{0,1\}^\tau:\ u_i = f_i(\boldsymbol{u}_{i-1}) \text{ for all } i \in \mathcal{F}, \\ u_\tau = f_\tau(\boldsymbol{u}_{\tau-1})}} C_n^{(\tau)}(\boldsymbol{u}_{\tau-1}, u_\tau) \tag{12}$$

The number of polar cosets in this representation equals $2^{\mathrm{MF}(\mathbb{C})}$. Denote by $A_{\mathbb{C}}(X)$ the WEF of the entire code $\mathbb{C}$, then $A_{\mathbb{C}}(X)$ is the sum of the WEFs for those polar cosets:

$$A_{\mathbb{C}}(X) = \sum_{\substack{\boldsymbol{u}_{\tau-1} \in \{0,1\}^\tau:\ u_i = f_i(\boldsymbol{u}_{i-1}) \text{ for all } i \in \mathcal{F}, \\ u_\tau = f_\tau(\boldsymbol{u}_{\tau-1})}} A_n^{(\tau)}(\boldsymbol{u}_{\tau-1}, u_\tau)(X) \tag{13}$$

C. Computing the Entire Weight Distribution

As shown in Proposition 2 and Proposition 3, we can represent any polar code, or any polar code with dynamically frozen bits as a disjoint union of polar cosets. With this representation, we can first use Algorithm 1 to compute the WEFs for those polar cosets, and then sum them up to obtain the WEF for the entire code. This procedure is shown in Algorithm 2.

In Algorithm 2, the number of polar cosets we need to evaluate for code $\mathbb{C}$ equals $2^{\mathrm{MF}(\mathbb{C})}$. So Algorithm 2 has complexity $O(2^{\mathrm{MF}(\mathbb{C})}\, n^2)$. This complexity is largely governed by the mixing factor of the code. In the next section, we identify

Algorithm 2: Compute the WEF of polar codes and polar codes with dynamically frozen bits

Input: block length n, set of frozen indices $\mathcal{F}$, and the dynamic constraints $\{f_i : i \in \mathcal{F}\}$

Output: WEF $A_{\mathbb{C}}(X)$

$\tau \leftarrow \max\{\mathcal{F}\}$
$A_{\mathbb{C}}(X) \leftarrow 0$
for $\boldsymbol{u}_{\tau-1} \in \{0,1\}^{\tau} : u_i = f_i(\boldsymbol{u}_{i-1})$ *for all* $i \in \mathcal{F}$ **do**
  $(f_0, f_1) \leftarrow \text{CalcWEF}(n, \boldsymbol{u}_{\tau-1})$
  $u_\tau \leftarrow f_\tau(\boldsymbol{u}_{\tau-1})$
  if $u_\tau = 0$ **then**
    $A_{\mathbb{C}}(X) \leftarrow A_{\mathbb{C}}(X) + f_0$
  else
    $A_{\mathbb{C}}(X) \leftarrow A_{\mathbb{C}}(X) + f_1$
return $A_{\mathbb{C}}(X)$

polar codes as decreasing monomial codes, and upper bound the mixing factors for polar codes at different block lengths.

We remark that although any binary linear codes can be represented as polar codes with dynamically frozen bits, many such representations have a large mixing factor, making Algorithm 2 less practical for a relative large block length.

V. Mixing Factors of Polar Codes as Decreasing Monomial Codes

In Section II, we introduce (n,k) polar codes in a broad sense, where we can pick any k rows in the polar transformation matrix G_n as generators for the code. If we follow a more restricted definition, where we pick the k bit-channels having the smallest Bhattacharyya parameters the same as Arıkan's definition in [1], the constructed polar code becomes a decreasing monomial code as introduced in [3]. In this section, We present results on the largest mixing factor of polar codes at each block length as decreasing monomial codes.

A. Decreasing Monomial Codes

We first recast the definition for decreasing monomial codes. For details of decreasing monomial codes and their algebraic properties, we refer the readers to the paper by Bardet, Dragoi, Otmani, and Tillich [3].

For $n = 2^m$, define the polynomial ring $\mathcal{R}_m$ as

$$\mathcal{R}_m = \mathbb{F}[x_0, \cdots, x_{m-1}]/(x_0^2 - x_0, \cdots, x_{m-1}^2 - x_{m-1}).$$

We associate each polynomial $p \in \mathcal{R}_m$ by a binary vector in $\mathbb{F}_2^n$ as the evaluation of p in all the binary m-tuples $\boldsymbol{x} = (x_0, \cdots, x_{m-1}) \in \mathbb{F}_2^m$. In other words, we associate polynomial p with $\text{ev}(p) = (p(\boldsymbol{x}))_{\boldsymbol{x}\in\mathbb{F}_2^m}$ where $\text{ev} : \mathcal{R}_m \to \mathbb{F}_2^n$ is a homomorphism. Denote the monomials in $\mathcal{R}_m$ as

$$\mathcal{M}_m = \left\{ x_0^{b_0} \cdots x_{m-1}^{b_{m-1}} \mid (b_0, \cdots, b_{m-1}) \in \mathbb{F}_2^m \right\}.$$

The monomial codes are defined as follows:

Definition 2. *Let $n = 2^m$ and $\mathcal{I} \subseteq \mathcal{M}_m$, the monomial code $\mathbb{C}(\mathcal{I})$ generated by $\mathcal{I}$ is the linear space spanned by $\{ev(f) : f \in \mathcal{I}\}$.*

Since all rows in the polar transformation matrix G_n can be obtained as $\text{ev}(f)$ with $f \in \mathcal{M}_m$, polar codes can be viewed as monomial codes. For a monomial $f \in \mathcal{M}_m$ given by $f = x_{i_1} x_{i_2} \cdots x_{i_d}$, we write $\deg f = d$ as the degree of f, and $\text{ind}(f) = \{i_1, i_2, \ldots, i_d\}$ as the set of indices for the variables in f. If the evaluation of f is the i-th row in the polar transformation matrix G_n, we write $[\![f]\!] = i$ to be its row index. Henceforth, whenever we write a monomial as $f = x_{i_1} x_{i_2} \cdots x_{i_d}$ we assume that $i_1 < i_2 < \cdots < i_d$, unless stated otherwise.

A partial order for the monomials in $\mathcal{M}_m$, and the decreasing monomial codes following this order can be defined as:

Definition 3. *Two monomials in $\mathcal{M}_m$ of the same degree are ordered as $x_{i_1} \cdots x_{i_d} \preccurlyeq x_{j_1} \cdots x_{j_d}$ if and only if $i_\ell \leqslant j_\ell$ holds for any $\ell \in \{1, \cdots, d\}$. Two monomials $f, g \in \mathcal{M}_m$ of different degrees are ordered as $f \preccurlyeq g$ if there is a divisor g^* of g having the same degree as f, and $f \preccurlyeq g^*$.*

Definition 4. *A set $\mathcal{I} \subseteq \mathcal{M}_m$ is decreasing, if ($g \in \mathcal{I}$ and $f \preccurlyeq g$) implies $f \in \mathcal{I}$. We call the monomial code generated by a decreasing set a decreasing monomial code.*

B. Largest Mixing Factors for Decreasing Monomial Codes

To compute the largest mixing factor for decreasing monomial codes at a given length, we first consider all decreasing monoimal codes with last frozen index τ:

Proposition 4. *Let $n = 2^m$, and let $\mathfrak{C}_\tau$ be the set of all length-n decreasing monomial codes with last frozen index τ, then*

$$\max_{\mathbb{C}\in\mathfrak{C}_\tau} \text{MF}(\mathbb{C}) = |\{g \in \mathcal{M}_m : [\![g]\!] < \tau, g \not\preccurlyeq f_\tau \text{ and } g \not\succcurlyeq f_\tau\}| \quad (14)$$

We leave the proof for Proposition 5 to the our extended paper [13]. Let $\mathfrak{C}$ be the set of all lengh-n decreasing monomial codes, we can then compute their largest mixing factor as

$$\max_{\mathbb{C}\in\mathfrak{C}} \text{MF}(\mathbb{C}) = \max_{0\leqslant\tau\leqslant n-1} \left(\max_{\mathbb{C}\in\mathfrak{C}_\tau} \text{MF}(\mathbb{C}) \right) \quad (15)$$

The results for block length up to 1024 are shown in the first row of Table 1.

Since by the MacWilliams identity [16], one can easily obtain the weight distribution of a code from the weight distribution of its dual, to get a better complexity cap, we can further restrict our searching space to codes with rates at most 1/2.

Proposition 5. *Let $n = 2^m$, and let $\mathfrak{C}_{\tau,R\leqslant 1/2}$ be the set of all length-n decreasing monomial codes with last frozen index τ, and rates at most 1/2, then*

$$\max_{\mathbb{C}\in\mathfrak{C}_{\tau,R\leqslant 1/2}} \text{MF}(\mathbb{C}) \leqslant \min\left\{ \max_{\mathbb{C}\in\mathfrak{C}_\tau} \text{MF}(\mathbb{C}),\ (\tau + 1 - n/2) \right\} \quad (16)$$

So let $\mathfrak{C}_{R\leqslant 1/2}$ be the set of all lengh-n decreasing monomial codes with rates at most 1/2, an upper bound on their mixing factors can be computed with

$$\max_{\mathbb{C}\in\mathfrak{C}_{R\leqslant 1/2}} \text{MF}(\mathbb{C}) = \max_{0\leqslant\tau\leqslant n-1} \left(\max_{\mathbb{C}\in\mathfrak{C}_{\tau,R\leqslant 1/2}} \text{MF}(\mathbb{C}) \right) \quad (17)$$

The results for block length up to 1024 are shown in the second row of Table 1.

$\log(n)$	3	4	5	6	7	8	9	10
$\max_{\mathbb{C}\in\mathfrak{C}} \mathrm{MF}(\mathbb{C})$	1	4	11	27	68	156	339	721
$\max_{\mathbb{C}\in\mathfrak{C}_{R\leqslant 1/2}} \mathrm{MF}(\mathbb{C}) \leqslant$	1	2	9	18	49	98	225	450

Table 1. Largest mixing factor for decreasing monomial codes at each block length.

We observe that when $\log(n) = 3,5,7,9$, those upper-bounds at the second row of Table 1 are met by the rate 1/2 Reed-Muller codes. We thus conjecture that, with the options of applying Algorithm 2 to either the code or its dual, the rate 1/2 Reed-Muller code has the highest complexity among decreasing monomial codes with the same length in general.

VI. Reducing Complexity using the Lower Triangular Affine Permutation Group

Polar codes as decreasing monomial codes have a large automorphism group that includes the *lower triangular affine* group $\mathrm{LTA}(m,2)$ [3, Theorem 2]. In this section, we show that $\mathrm{LTA}(m,2)$ acts transitively on certain subsets of decreasing monomial codes. This allow us to reduce the coset WEFs we need to compute in Algorithm 2. Since Reed-Muller codes are decreasing monomial codes as well, our complexity reduction also applies to Reed-Muller codes. Although the complexity remains exponential, this reduction makes it possible to compute the weight distribution of any polar code and Reed-Muller code for all rates of length up to $n = 128$.

A. Lower Triangular Affine Group Acts Transitively on Subsets of Decreasing Monomial Codes

First, we recast the definition for $\mathrm{LTA}(m,2)$.

Definition 5. *The lower triangular affine group over $\mathbb{F}_2^m$, denoted as LTA$(m,2)$, consists of all affine transformations over $\mathbb{F}_2^m$ with the form $\boldsymbol{x} \mapsto A\boldsymbol{x} + \boldsymbol{b}$, where $A \in \mathbb{F}_2^{m\times m}$ is a non-singular $m \times m$ lower triangular binary matrix, and $\boldsymbol{b} \in \mathbb{F}_2^m$.*

Since any affine transformations in $\mathrm{LTA}(m,2)$ is a permutation of all the m-tuples in $\mathbb{F}_2^m$, we can also think of the transformation as a permutation on the coordinates of the monomial codes. If we can prove that $\mathrm{LTA}(m,2)$ acts transtively on certain subsets of decreasing monomial codes, since permutation on the coordinates doesn't change the Hamming weights of the codewords, this will imply that all those subsets share the same weight distribution.

To describe the subsets on which $\mathrm{LTA}(m,2)$ acts transitively, we define a new relation called ***one-variable descendance*** for the monomials in $\mathcal{M}_m$.

Definition 6. *Let $f,g \in \mathcal{M}_m$, we say g is a **one-variable descendant** of f if $ind(f)\backslash ind(g) = \{i\}$ is a singleton, and either one of the following holds:*

1) *$ind(g)\backslash ind(f) = \{j\}$ is also a singleton, and $j < i$.*
2) *$ind(g)\backslash ind(f) = \emptyset$, or g divides f.*

If g is a one-variable descendant of f, We write $g \prec_{one} f$.

We also make the following definitions.

Definition 7. *Let $\mathbb{C}$ be a decreasing monomial code generated by $\mathcal{I} \subseteq \mathcal{M}_m$, and let $f \in \mathcal{I}$. Define $\mathcal{S}(\mathbb{C};f)$ as the set of monomials:*

$$\mathcal{S}(\mathbb{C};f) = \{h \in \mathcal{M}_m \,:\, h \prec_{one} f \text{ and } [\![h]\!] < \tau(\mathbb{C})\} \quad (18)$$

Define $\mathbb{C}_f$ as the subcode of $\mathbb{C}$ generated by the set of monomials that are not in $(\{f\} \cup \mathcal{S}(\mathbb{C};f))$

$$\mathbb{C}_f = \mathbb{C}(\mathcal{I}\backslash\ (\{f\} \cup \mathcal{S}(\mathbb{C};f))\ ) \quad (19)$$

Define $\mathcal{X}(\mathbb{C};f)$ as the set of cosets of $\mathbb{C}_f$ as

$$\mathcal{X}(\mathbb{C};f) = \left\{ ev(f) + \sum_{h\in\mathcal{S}(\mathbb{C};f)} u_h \cdot ev(h) + \mathbb{C}_f \ : \ u_h \in \{0,1\} \text{ for all } h \in \mathcal{S}(\mathbb{C};f) \right\} \quad (20)$$

Now we state our main theorem of this section, and its corollary that helps with the complexity reduction. Due to lack of space, we leave its proof for our extended paper [13].

Theorem 1. *Let $\mathbb{C}$ be a decreasing monomial code generated by $\mathcal{I} \subseteq \mathcal{M}_m$, and let $f \in \mathcal{I}$. The group action of LTA$(m,2)$ on $\mathcal{X}(\mathbb{C};f)$ is transitive.*

Corollary 1. *All the cosets of $\mathbb{C}_f$ in $\mathcal{X}(\mathbb{C};f)$ have the same weight distribution.*

B. Complexity Reduction for Decreasing Monomial Codes

Now we use Corollary 1 to reduce the complexity of Algorithm 2. First, observe that there are $2^{|\mathcal{S}(\mathbb{C};f)|}$ disjoint cosets of $\mathbb{C}_f$ in the set $\mathcal{X}(\mathbb{C};f)$, and the union of those cosets with the subcode $\mathbb{C}(\mathcal{I}\backslash\{f\})$ is the entire code $\mathbb{C}$:

$$\mathbb{C} = \mathbb{C}(\mathcal{I}\backslash\{f\}) \cup \left(\bigcup_{X\in\mathcal{X}(\mathbb{C};f)} X \right)$$

By Corollary 1, all cosets of $\mathbb{C}_f$ in $\mathcal{X}(\mathbb{C};f)$ share the same WEF. So with Algorithm 2, if we can compute the WEF for $\mathbb{C}(\mathcal{I}\backslash\{f\})$ as $B(X)$, and compute the WEF for a single coset in $\mathcal{X}(\mathbb{C};f)$ as $C(X)$, we can obtain the WEF $A_{\mathbb{C}}(X)$ for the entire code $\mathbb{C}$ as

$$A_{\mathbb{C}}(X) = B(X) + 2^{|\mathcal{S}(\mathbb{C};f)|} \cdot C(X) \quad (21)$$

If we apply Algorithm 2 directly to code $\mathbb{C}$, equivalently we will compute the WEFs for all the cosets in $\mathcal{X}(\mathbb{C};f)$. With equation (21), we only need to compute the WEF for a single coset in $\mathcal{X}(\mathbb{C};f)$.

If we pick f to be the monomial in $\mathcal{I}$ with the smallest $[\![f]\!]$, then the subcode $\mathbb{C}(\mathcal{I}\backslash\{f\})$ is also a decreasing monomial code, as shown by the following proposition. We leave the proof for Proposition 6 to [13].

Proposition 6. *Let $\mathbb{C}(\mathcal{I})$ be a decreasing monomial code generated by $\mathcal{I} \subseteq \mathcal{M}_m$. If we pick f to be the monomial in $\mathcal{I}$ with the smallest $[\![f]\!]$, then the subcode $\mathbb{C}(\mathcal{I}\backslash\{f\})$ is also a decreasing monomial code.*

So in (21), when we use Algorithm 2 to compute the WEF for the subcode $\mathbb{C}(\mathcal{I}\backslash\{f\})$, we can reapply the same complexity reduction, and unfold the WEF for $\mathbb{C}(\mathcal{I}\backslash\{f\})$ again. So on and so forth.

To demonstrate the overall reduction of the complexity exponent, consider the self-dual Reed-Muller code RM(3,7) and a (128,64) polar code. We show in the extended paper on arxiv [13] that, for the Reed-Muller code, the complexity of our algorithm reduces from 2^{49} polar coset evaluations to $2^{35.53}$, while for the polar code, the complexity reduces from 2^{37} polar coset evaluations to $2^{25.86}$.

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] 3GPP TSG RAN WG1 Meeting #87, Final report, Reno, NV, November 2016.

[3] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 230–234.

[4] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Communications Letters*, vol. 16, no. 12, pp. 2044–2047, 2012.

[5] Z. Liu, K. Chen, K. Niu, and Z. He, "Distance spectrum analysis of polar codes," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2014, pp. 490–495.

[6] M. Valipour and S. Yousefi, "On probabilistic weight distribution of polar codes," *IEEE communications letters*, vol. 17, no. 11, pp. 2120–2123, 2013.

[7] Q. Zhang, A. Liu, and X. Pan, "An enhanced probabilistic computation method for the weight distribution of polar codes," *IEEE Communications Letters*, vol. 21, no. 12, pp. 2562–2565, 2017.

[8] R. Polyanskaya, M. Davletshin, and N. Polyanskii, "Weight distributions for successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 68, no. 12, pp. 7328–7336, 2020.

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

[10] P. Trifonov and V. Miloslavskaya, "Polar subcodes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 254–266, 2015.

[11] E. Arıkan, "From sequential decoding to channel polarization and back again," *arXiv preprint arXiv:1908.09594*, 2019.

[12] 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.

[13] H. Yao, A. Fazeli, and A. Vardy, "A deterministic algorithm for computing the weight distribution of polar codes," *arXiv preprint arXiv:xxxx.yyyyy*, 2021.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[15] 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.

[16] J. A. MacWilliams, "A theorem on the distribution of weights in a systematic code," *Bell System Technical Journal*, vol. 42, no. 1, pp. 79–94, 1963.