Low-complexity algorithm for the minimum distance properties of PAC codes

Malek Ellouze, Romain Tajan, Camille Leroux, Christophe Jégo, Charly Poulliat

Low-complexity algorithm for the minimum distance properties of PAC codes

Malek Ellouze*, Romain Tajan*, Camille Leroux*, Christophe Jégo* and Charly Poulliat †
*University of Bordeaux, Bordeaux INP, IMS Lab, UMR CNRS 5218, France
†University of Toulouse, INPT-ENSEEIHT, IRIT Lab, UMR CNRS 5505, France

***Abstract*—In this paper, a low-complexity algorithm that enables to compute the minimum distance as well as its associated number of occurrences for PAC codes is described. The method relies on the computation of the minimum weight properties of polar cosets while taking into consideration the initial precoding. Due to the exponential rise of the number of cosets, we propose a method that discards the cosets that have no impact on the computation of the minimum distance. This method does not impose any specific rules on the choice of the frozen bit set and the initial precoding.**

***Index Terms*—Polar codes, PAC codes, Minimum distance computation, Minimum weight codewords.**

I. Introduction

Polar codes [1] have poor performance at finite length under Successive Cancellation (SC) decoding. As an approximation of Maximum Likelihood (ML) decoding, the Successive Cancellation List (SCL) decoder [2] allows to partially close this gap if the list size is large enough. However, the mediocre distance properties of polar codes prevent them from better performance for medium / low length. In order to improve the distance properties, the polar coding structure has to be modified.

The first option, used in the 5G standard, is to use a CRC as an outer code [3]. This increases the distance of the code and allows the SCL decoder to discard candidate codewords at the end of the decoding. A more recent alternative is to apply a rate-1 precoding before the polar transformation. As such, the *dynamic frozen bits polar codes* use an upper triangular transformation on the frozen bits [4]. In the case of *Polarization-Adjusted Convolutional* (PAC) codes, a rate-1 convolutional encoder [5] is used. Precoding a polar code can be very efficient : near optimal performance is for instance reported in [6] for a (128,64) PAC codes.

Computing the distance properties of polar codes and precoded polar codes allows to rapidly estimate the decoding performance of the ML decoder at high SNR.

The computation of the minimum distance of polar codes was first proposed in [7]. In [8], the number of minimum weight codewords is computed under the assumption that the considered polar code is a decreasing monomial code. A more general approach is used in [9] where Monte Carlo simulation of an SCL decoder with very large list size allows to estimate the partial distance of polar codes. It is however very complex as the code length grows.

In the case of PAC codes, it is also possible to use the complex Monte Carlo approach [6]. In [10], the low-weight codewords of polar and precoded polar codes are enumerated based on a recursive decomposition. Although very general, this approach becomes complex for precoded polar codes because the decomposition is less favorable than for regular polar codes.

In [11], a low complexity method allows to enumerate the minimum weights codewords of PAC codes. However, it is only applicable under frozen set of decreasing monomial codes.

In this paper, a very low-complexity algorithm capable of computing both the minimum distance and its associated number of occurrences for PAC codes is proposed. Unlike [11], the method is "universal" in the sense that it does not assume any particular structure for the frozen bit set. Besides, stating that like polar codes, PAC codes can be described as a union of well-defined code cosets, a drastic complexity reduction is achieved by considering the relevant cosets. Considering that this method can be applied regardless of the frozen bit set, it can help design frozen bit sets that can be more adapted to PAC codes and consequently significantly improve their performance.

II. Preliminaries

A. Polar codes and polar cosets

The polar transformation T_N is defined as

$$T_N = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n}$$

where $\bigotimes n$ denotes the n fold Kronecker product and $N = 2^n$.

The codewords of a (N, K) polar code are obtained such that $\mathbf{c} = \mathbf{u}T_N$, where $\mathbf{u} \in \mathbb{F}_2^N$. $\mathbf{u}$ is a binary vector for which K positions are assigned to the information bits, whereas the remaining ones are *frozen* to some known values. We define $\mathcal{F}$ as the indice set of the components of $\mathbf{u}$ corresponding to the frozen bits.

As in [12], given $\mathbf{u}_0^{i-1} = (u_0, u_1, ..., u_{i-1}) \in \mathbb{F}_2^{i-1}$ and $u_i \in \mathbb{F}_2$, a polar coset $\mathcal{C}_N^{(i)}$ can be defined as:

$$\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i) = \{[\mathbf{u}_0^{i-1}, u_i, \mathbf{u}_{i+1}^{N-1}]T_N | \mathbf{u}_{i+1}^{N-1} \in \mathbb{F}_2^{N-i-1}\} \quad (1)$$

A polar coset $\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i)$ thus describes the codewords' space generated by the prefix $\mathbf{u}_0^i$ without considering the

frozen bits contained in $\mathbf{u}_{i+1}^{N-1}$. A polar code $\mathcal{C}$ can be defined as the disjoint union of the following cosets:

$$\mathcal{C} = \bigcup_{\substack{u_i \in \{0,1\}, \forall i \notin \mathcal{F} \\ u_i = 0, \forall i \in \mathcal{F}}} \mathcal{C}_N^{(s)}(\mathbf{u}_0^{s-1}, u_s = 0) \tag{2}$$

where s denotes the index of the last frozen bit.

B. PAC codes

PAC codes denoted by PAC$(N, K, \mathbf{g})$ consist in polar codes where the input vector $\mathbf{u}_0^{N-1}$ is obtained by a convolutional transformation using the generator polynomial $\mathbf{g}$ of degree $m-1$ with coefficients $[g_0, g_1, ..., g_{m-1}]$, i.e. given a vector $\mathbf{v}_0^i$, the associated u_i is obtained as follows:

$$u_i = \mathbf{g}(\mathbf{v}_0^i) = \sum_{j=0}^{m-1} g_j v_{i-j} \tag{3}$$

In other terms, the input element u_i of the polar encoder depend on the $i-1$ previous elements as well as the current element v_i. Conventionally, we have $g_0 = g_{m-1} = 1$. Unless specified otherwise, in this paper the following polynomial is used : $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$. Similar to regular polar codes, PAC codes can be defined as follows:

$$\mathcal{C}_{\text{PAC}} = \bigcup_{\mathcal{B}} \mathcal{C}_N^{(s)}(\mathbf{u}_0^{s-1}, u_s) \tag{4}$$

where $\mathcal{B}$ defines the set:

$$\mathcal{B} = \begin{cases} \mathbf{u}_0^s \in \{0,1\}^{s+1} | u_i = \mathbf{g}(\mathbf{v}_0^i), \forall i \in [\![0; s]\!] \\ v_i \in \{0,1\}, \forall i \notin \mathcal{F}, v_i = 0, \forall i \in \mathcal{F} \end{cases} \tag{5}$$

C. Minimum distance properties of PAC codes

In [12], it is shown that the distance properties of a polar code can be deduced from the analysis of the polar cosets. This is also true for PAC codes. When taking advantage of the equation (4), C_{PAC} can be expressed as follows:

$$\mathcal{C}_{\text{PAC}} = \bigcup_{\substack{\mathcal{B} \\ \mathbf{v}_0^s \neq \mathbf{0}_0^s}} \mathcal{C}_N^{(s)}(\mathbf{u}_0^{s-1}, u_s) \bigcup \mathcal{C}_N^{(s)}(g(\mathbf{0_0^{s-1}}), g(0))$$

It is important to note that as $g_0 = g_{m-1} = 1$ then $\forall i \in [\![0; N-1]\!]$ and $b \in \{0, 1\}$:

$$\mathbf{g}([\mathbf{0}_0^{i-1}, b]) = [\mathbf{0}_0^{i-1}, b]$$

this results in:

$$\mathcal{C}_{\text{PAC}} = \bigcup_{\substack{\mathcal{B} \\ \mathbf{v}_0^s \neq \mathbf{0}_0^s}} \mathcal{C}_N^{(s)}(\mathbf{u}_0^{s-1}, u_s) \bigcup \mathcal{C}_N^{(s)}(\mathbf{0_0^{s-1}}, 0) \tag{6}$$

$\mathcal{C}_N^{(s)}(\mathbf{0_0^{s-1}}, 0)$ can also be expressed as:

$$\mathcal{C}_N^{(s)}(\mathbf{0_0^{s-1}}, 0) = \bigcup_{i \in [\![s+1, N-1]\!]} \mathcal{C}_N^{(i)}(\mathbf{0}_0^{i-1}, u_i = 1) \bigcup \mathcal{C}_N^{(s)}(\mathbf{0_0^{N-2}}, 0) \tag{7}$$

When taking advantage of the equations (6) and (7), the overall minimum distance of a PAC code d^* can be expressed as follows:

$$\begin{aligned} d^* &\triangleq w^*(\mathcal{C}_{PAC} \setminus \mathcal{C}_N^{(N-1)}(\mathbf{0}_0^{N-2}, u_{N-1} = 0)) \\ &= \min(d_1, d_2) \end{aligned} \tag{8}$$

where

$$d_1 = \min_{\mathbf{u}_0^{s-1} \neq \mathbf{0}_0^{s-1}} w^*(\mathcal{C}_N^{(s)}(\mathbf{u}_0^{s-1}, u_s = 0)) \tag{9}$$

$$d_2 = \min_{i \in [\![s+1, N-1]\!]} w^*(\mathcal{C}_N^{(i)}(\mathbf{0}_0^{i-1}, u_i = 1)). \tag{10}$$

Where $w^*(.)$ defines the minimum weight of a set of codewords. In order to achieve a deterministic computation of d^* and A^*, the total number of cosets to explore can be expressed as:

$$n_d = 2^\gamma + (N - s) \tag{11}$$

where γ, refer to the mixing factor in [12], denotes the total number of information bits before the last frozen bit. Knowing that γ can reach 98 for a (256,128,$\mathbf{g}$) PAC code, the resulting computational complexity of exploring all possible cosets may become prohibitive even for moderate code sizes. In order to reply to this limitation, we propose here an algorithm that reduces the number of explored cosets.

III. Minimum distance properties evaluation of PAC codes

A. Low complexity algorithm

Knowing that we are able to compute w^* and A^* for any polar coset $\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i)$, it is possible to propose an enumeration structure similar to a list decoder that has the advantage of pruning cosets with a constraint of their minimal weight and only exploring relevant cosets. This enhances the efficiency of the algorithm by avoiding unnecessary computations on cosets having a large minimum weight.

Knowing that any coset $\mathcal{C}_N^{(i)}(\mathbf{0}_0^{i-1}, u_i = 1) \forall i > s$ only describes codewords, its is possible to use the minimum weight of those cosets as an upper bound on the minimum distance. Therefore, d_2 is computed first with the associated occurrences in order to have a first bound on the minimum distance.

Algorithm 1 gives the details of the proposed algorithm. Its main steps can be summarized as follows :

- For a considered polar code and the associated frozen bits strategy, $d_2 = w^*(\mathcal{C}_N^{(i)}(\mathbf{0}_0^{i-1}, u_i = 1 | i \in [\![s+1; N-1]\!]))$ is computed together with the associated occurrences.
- All the possible information vector $\mathbf{v}_0^i$ at an exploration stage i such that $i \leq s$ are listed and the associated prefixes $\mathbf{u}_0^i$ are computed using the convolutional transformation.
- For each of the aforementioned paths, $w^*(\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i))$ is computed. See section II.B for details
- The cosets with $w^*(\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i)) > d_2$ are discarded.

Algorithm 1: Computing d^* and A^* of a PAC code

Input: PAC code $\mathcal{C}_{PAC}(N, K, \mathcal{F}, \mathbf{g}), C_{max}$

$s \leftarrow$ index of the last frozen bit
$L \leftarrow 1$
for $i \in [\![s+1; N-1]\!]$ **do**
  Compute w^* and A^* of $\mathcal{C}_N^{(i)}(\mathbf{0}_0^{i-1}, u_i = 1))$
end
$d_2 \leftarrow \min_{i \in [\![s+1, N-1]\!]} w^*(\mathcal{C}_N^{(i)}(\mathbf{0}_0^{i-1}, u_i = 1))$
$A_2 \leftarrow$ Occurrences of d_2
for $i \in [\![1; s]\!]$ **do**
  if $i \in \mathcal{F}$ **then**
    for $l \in [\![1; L]\!]$ **do**
      $v_i[l] \leftarrow 0$
      $[u_i[l], s_{i+1}[l]] \leftarrow \text{conv}(v_i[l], s_i[l])$
      Compute w^* of $\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i[l]))$
      Discard the paths for which $w^* > d_2$
      $L \leftarrow |\mathcal{L}|$
    end
  else
    $\mathcal{L} \leftarrow \mathcal{L} \bigcup \mathcal{L}'$ /* $\mathcal{L}'$ is a copy of $\mathcal{L}$ */
    for $l \in [\![1; L]\!]$ **do**
      $[v_i[l], v_i[l']] \leftarrow [0, 1]$
      $[u_i[l], s_{i+1}[l]] \leftarrow \text{conv}(v_i[l], s_i[l])$
      $[u_i[l'], s_{i+1}[l']] \leftarrow \text{conv}(v_i[l'], s_i[l'])$
      Compute w^* of $\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i[l]))$ and $\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i[l']))$
      Discard the cosets for which $w^* > d_2$
      $L \leftarrow |\mathcal{L}|$
      if $\mathcal{L} > C_{max}$ **then**
        $\mathcal{L} \leftarrow$ Keep the L_{max} cosets with the least minimum weights
      end
    end
  end
  if $i = s$ **then**
    $d_1 \leftarrow$ minimum weight of the cosets remaining in the list except the all zero coset
    $A_1 \leftarrow$ Occurrences of d_1
  end
end
$d^* \leftarrow \min(d_1, d_2)$
$A^* \leftarrow$ Occurrences of d^*
Return (d^*, A^*)

- If the number of considered cosets reaches a maximum value $C_{\max}$, only $L_{\max}$ cosets with the least $w^*(\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i))$ are kept at a exploration stage i.
- When $i = s$ the minimum weights of the cosets remaining in the list as well as their numbers of occurrences are computed.

It is important to note that if $C_i < C_{\max}\ \forall i \in [\![1; s]\!]$ then the proposed algorithm provides the exact minimum distance as well as its associated number of occurrences. In fact, at each computation stage i, only the cosets having a minimum weight greater than the upper bound are discarded. If the number of relevant cosets does not exceed the maximum size C_{max}, we are sure that all the cosets with minimum weight have been explored.

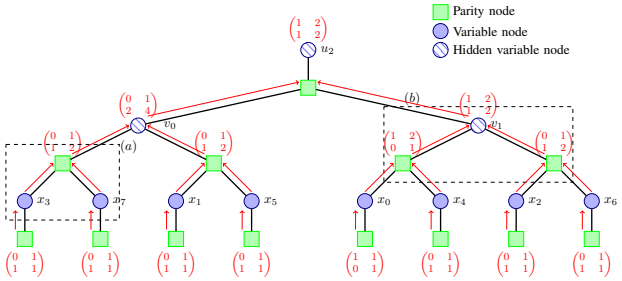

Fig. 1: Graph decoding of u_2 for a polar code with $N = 8$

B. Computation of the minimum distance properties of a coset on a factor graph

We now provide a sketch of proof of a method that enables to determine both the minimum weight and its number of occurrences for any polar coset.

Proposition III.1. For any polar coset $\mathcal{C}_N^{(i)}(\mathbf{u}_0^{i-1}, u_i)$, the minimum weight w^* and related occurrences A^* can be computed using the factor graph of each u_i described in [13].

Sketch of proof 1. In the following, we will denote by T_0^i and T_{i+1}^{N-1} the upper $(i+1)$ first rows and the $(N-i-1)$ last rows of T_N. Given $\mathbf{p} = \mathbf{u}_0^{i-1} T_0^{i-1}$ and $\mathbf{s} = \mathbf{u}_i^{N-1} T_i^{N-1}$, we prove that w^* and A^* can be expressed as:

$$w^*(\mathbf{p}, u_i) = \min_{\substack{\mathbf{s} \in \mathbb{F}_2^N \\ \mathbf{v} \in \mathbb{F}_2^d}} w(\sum_{j=0}^{N-1} p_j \oplus s_j) - \log(\mathbb{1}_{([\mathbf{s}, \mathbf{v}, u_i] H^{(i)T} = 0)}) \quad (12)$$

$$A^*(\mathbf{p}, u_i) = |\operatorname*{argmin}_{\substack{\mathbf{s} \in \mathbb{F}_2^N \\ \mathbf{v} \in \mathbb{F}_2^d}} w(\sum_{j=0}^{N-1} p_j \oplus s_j) - \log(\mathbb{1}_{([\mathbf{s}, \mathbf{v}, u_i] H^{(i)T} = 0)})| \quad (13)$$

where $H^{(i)^T}$ denotes the extended parity check matrix associated to T_{i+1}^{N-1} so that the decoding graph of u_i is a tree,$\mathbf{v}$ denotes all the hidden variables resulting from the Kronecker product and the $|.|$ operator defines the cardinality of a set.

Permuting the sum and the minimum operator in (12) is possible according to [14]. It enables to build a graphical representation of the factorization. In the specific case of polar codes, the decoding factor graphs of each u_i is a tree as it was shown in [13]. This tree representation allows a simple way to compute efficiently the minimum weight and associated occurrences. Figure 1 gives an illustration of a decoding factor graph of bit u_2 for a polar code of length $N = 8$ for $\mathbf{p} = [10000000]$.

During the computation, two configurations can be encountered. The dashed sub-graph (a) of Figure 1 describes the case where two variable nodes x_1 and x_2 are connected to a variable node x_p through a check node f. As in message passing formalism, messages are functions of the different variables. Each one of those messages can be represented by a matrix 2×2 $\boldsymbol{\theta}_x$ given by:

$$\boldsymbol{\theta}_x = \begin{pmatrix} \mu_x^{(0)} & A_x^{(0)} \\ \mu_x^{(1)} & A_x^{(1)} \end{pmatrix} \tag{14}$$

where $\mu_x^{(0)}$ and $\mu_x^{(1)}$ denote the minimum weight of the configurations with $x = 0$ and $x = 1$, while $A_x^{(0)}$ and $A_x^{(1)}$ are defined as the associated number of occurrences.

As depicted in Figure 1 in the dashed sub-graph (a), $\boldsymbol{\theta}_{x_p}$ can be computed as follows:

$$\begin{aligned} \mu_{x_p}^{(0)} =& \min(\underbrace{\mu_{x_1}^{(0)} + \mu_{x_2}^{(0)}}_{w_1}, \underbrace{\mu_{x_1}^{(1)} + \mu_{x_2}^{(1)}}_{w_2}) \\ A_{x_p}^{(0)} =& A_{x_1}^{(0)} A_{x_2}^{(0)} \mathbb{1}(\mu_{x_p}^{(0)} = w_1) + A_{x_1}^{(1)} A_{x_2}^{(1)} \mathbb{1}(\mu_{x_p}^{(0)} = w_2) \end{aligned} \tag{15}$$

Similarly, we have:

$$\begin{aligned} \mu_{x_p}^{(1)} =& \min(\underbrace{\mu_{x_1}^{(0)} + \mu_{x_2}^{(1)}}_{w_1}, \underbrace{\mu_{x_1}^{(1)} + \mu_{x_2}^{(0)}}_{w_2}) \\ A_{x_p}^{(1)} =& A_{x_1}^{(0)} A_{x_2}^{(1)} \mathbb{1}(\mu_{x_p}^{(1)} = w_1) + A_{x_1}^{(1)} A_{x_2}^{(0)} \mathbb{1}(\mu_{x_p}^{(1)} = w_2) \end{aligned} \tag{16}$$

For instance, in the example given in Figure 1, considering the dashed sub-graph (a), we have:

$$\boldsymbol{\theta}_{v_0} = \begin{pmatrix} \min(1+1, 0+0) & A_{v_0}^{(0)} \\ \min(1+0, 0+1) & A_{v_0}^{(1)} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \tag{17}$$

Figure 1 in the dashed sub-graph (b) describes the case where two check nodes f_1 and f_2 are connected to a variable node x. Given the two incoming messages from the check nodes to the variable node x, the messages $\mu_x^{(b)}$ and $A_x^{(b)}$, $b = \{0, 1\}$, can be expressed as follows:

$$\begin{cases} \mu_x^{(b)} &= \mu_{x_1}^{(b)} + \mu_{x_2}^{(b)} \\ A_x^{(b)} &= A_{x_1}^{(b)} A_{x_2}^{(b)} \end{cases} \tag{18}$$

Proposition III.2. *The overall time complexity defined as the number of arithmetic operations of computing d^* and A^* for a coset is at most* $12(N-1)$.

Proof. In the described algorithm, depending on the configuration (parity node case or variable node case), a certain number of arithmetic operations has to be performed.
In the parity node case, equations (13) and (14) are computed. Computation of $\mu_x^{(0)}$ and $\mu_x^{(1)}$ need 3 arithmetic operations each resulting in 6 operations. The computation of $A_x^{(0)}$ and $A_x^{(1)}$ need at most 3 arithmetic operations each. We will consider this worst case.
In the variable node case, Computation of $\mu_x^{(0)}$, $\mu_x^{(1)}$, $A_x^{(0)}$ and $A_x^{(1)}$ need 1 arithmetic operations each resulting in 4 operations. The parity node case has the highest complexity so we will only consider this case. Given that on each graph factor of u_i, $N-1$ nodes have to be evaluated, the time complexity is at most equal to $12(N-1)$ □

TABLE I: Minimum distance and number of occurrences

(N, K)	$\mathcal{F}$	d^*	A^*_{PAC}	C^*	n_c	n_e [11]
$(64, 22)$	RM	16	500	57	1061	65192
$(128, 64)$	RM	16	3120	2825	58329	7265295
$(256, 192)$	GA 4dB	8	53456	10326	227624	977664
$(512, 256)$	GA 2dB	16	36256	6074	139822	133968

IV. Experimental results

A. Complexity comparison

Algorithm 1 consists in $N-s$ loop iterations where a single coset is evaluated and s loop iterations where C_i cosets are evaluated, with $C_i < C_{max}$. The complexity of the proposed method is driven by the total number of evaluated cosets n_c:

$$n_c = \sum_{i=0}^{s} C_i + (N - s) \tag{19}$$

In all our experimentations, C_{max} was set to 2000000 and we observed that $C^* < C_{max}$, with $C^* = \max_i(C_i)$. This means that the number of explored cosets never reaches C_{max} and the reported (d^*, A^*) values are exact.

Algorithm 1 was applied on a range of PAC codes, for different code rates and different frozen set constructions. We will refer to the Gaussian Approximation (GA) construction with design-SNR of XdB [15] as GA XdB and to the Reed Muller construction as RM.
Table I summarizes the minimum distance d^*, the associated occurrences A^* for both polar and PAC configurations. The (d^*, A^*) values highlighted in green were corroborated with results in [11]. It is possible to estimate the complexity of [11] as the total number of explorations : $n_e = \sum_{i \in B} 2^{|\mathcal{K}_i^f|}(f - i)$.

The complexity comparison also shows that the proposed algorithm outperforms the algorithm proposed in [11]. In particular, for RM constructions, the number of explorations reduction is very significant. In terms of execution time, our MATLAB code time execution was compared to the MATLAB code time execution of [11] that can be found in [16]. For a (128,64) PAC code with a RM construction, our simulation time is around 11 seconds whereas [16]'s running time is around 25 minutes. Note that unlike Algorithm 1, the algorithm proposed in [11] is only valid for polar and RM constructions, no comparisons can be undertaken with other kind of frozen bit set constructions.

The complexity of the proposed algorithm was also compared to the method introduced in [10] in which the results are shown not only for the codewords with minimum weights but also for the codewords with a weight up to a certain value w_{max}. Algorithm 1 was slightly modified to accommodate the computation of the number of codewords up to a fixed weight. The changes consist the following changes in Algorithm 1:

TABLE II: Low weight codewords enumeration

(N, K)	$(128, 64)$	$(128, 80)$
(w, A_w)	$(8, 288)$	$(8, 4312)$
	$(12, 832)$	$(10, 2368)$
	$(16, 75864)$	$(12, 374240)$
	$(18, 6272)$	$(14, 1267648)$
$\Sigma\mathcal{N}^{(\mathcal{X})}_{Subcode}$ [10]	3831×10^9	4585×10^9
$6n_c(N-1)$	1532×10^5	2396×10^7

- Deleting lines $3-7$
- Replacing $[\![1; s]\!]$ in line 8 by $[\![1; N]\!]$
- Replacing $w^* > w_{start}$ in lines 14 and 25 by $w^* > w_{max}$.

It is also important to note that results for precoded polar codes are shown for dynamic frozen bits in [10]. Polar codes with dynamic frozen bits can be seen as PAC codes with a variable generator polynomial as follows:

$$\begin{cases} \mathbf{g} = [1], \forall i \notin \mathcal{F} \\ \mathbf{g} = [g_0, ..., g_{i-1}], \forall i \in \mathcal{F} \end{cases} \tag{20}$$

In this study case, $g_0, ..., g_{i-1}$ are chosen randomly.
The algorithm's overall time complexity is at most equal to $6n_c(N-1)$ as only d^* needs to be computed whereas [10]'s algorithm time complexity is dominated by the term $\Sigma\mathcal{N}^{(\mathcal{X})}_{Subcode}$. Table II shows the computed partial weight distribution obtained when introducing the aforementioned modifications to Algorithm 1 (Since the precoding is done randomly, it is not possible to reproduce the exact same values but only values in a close range) and compares the overall complexity of Algorithm 1 to the complexity of the algorithm in [10]. We can see from the table that the computational complexity of the proposed algorithm is lower by several orders of magnitude. [10] shows the overall running time for a C++ implementation on a computer with 6 cores i7 and a 3.2GHz processor. For example, the running time for a randomly precoded (128,64) is around one hour and a half in [10], whereas our running time for a MATLAB implementation on a computer with 2 cores i5 and a 3.1GHz processor is less than 10 minutes.

B. d^* and A^* computation

The computation of d^* and A^* for $N = 512$ and $K \in [\![1; N]\!]$ for the 5G standard frozen bit set is summarized in Figure 2 for both polar and PAC codes for a generator polynomial $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$. The frozen bit sets are the ones specified in the 5G standard [3]. This proves that we are able to compute d^* and A^* for any desired rate for a specific frozen bit set. Moreover, it can be observed that d^* is the same for polar and PAC codes for any rate for this specific frozen set.

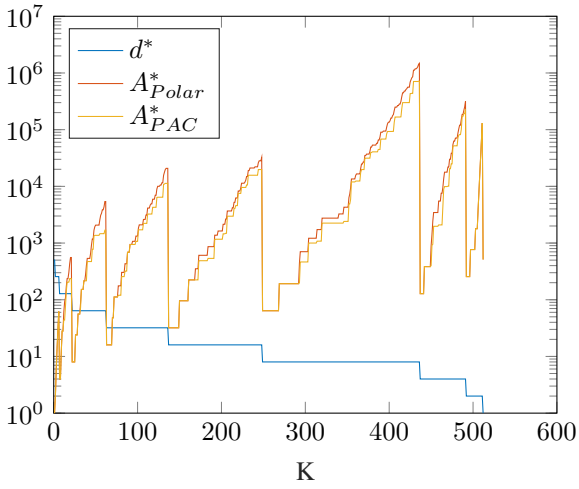

Fig. 2: d^* et A^* for $N = 512$ and $\mathbf{g} = [1, 0, 1, 1, 0, 1, 1]$ depending on K

V. Conclusion

A low-complexity algorithm that only explores cosets that follows a constraint on their minimum weight in order to compute the minimum distance and its associated number of occurrences for PAC codes has been introduced in this article. It has been proven to exhibit a low complexity in comparison to other algorithms. Besides, it enables computing the minimum distance as well as its associated number of occurrences for any frozen bit set.

References

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. on Inf. Theory*, 2009.

[2] I. Tal and A. Vardy, "List decoding of polar codes," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2011.

[3] 3GPP TS 38.212 V17.4.0, "5G; NR; multiplexing and channel coding," 2023.

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

[5] M. Moradi, A. Mozammel, K. Qin, and E. Arikan, "Performance and complexity of sequential decoding of PAC codes," *CoRR*, 2020.

[6] T. Tonnellier and W. J. Gross, "On systematic polarization-adjusted convolutional (pac) codes," *IEEE Communications Letters*, 2021.

[7] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *IEEE ISIT*, 2009.

[8] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *IEEE ISIT*, 2016.

[9] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Communications Letters*, 2012.

[10] V. Miloslavskaya, B. Vucetic, and Y. Li, "Computing the partial weight distribution of punctured, shortened, precoded polar codes," *IEEE Transactions on Communications*, 2022.

[11] M. Rowshan and J. Yuan, "Fast enumeration of minimum weight codewords of PAC codes," in *IEEE ITW*, 2022.

[12] H. Yao, A. Fazeli, and A. Vardy, "A deterministic algorithm for computing the weight distribution of polar code," *IEEE Transactions on Information Theory*, 2023.

[13] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *IEEE ISIT*, 2009.

[14] T. Richardson and R. Urbanke, *Modern Coding Theory*, Chapter 2, Cambridge University Press, 2008.

[15] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Transactions on Communications*, 2012.

[16] https://github.com/mohammad-rowshan/Fast-Enumeration-of-Minimum-Weight-Codewords-of-PAC-Codes.