Efficient decoding of polar codes with some 16×16 kernels

Grigorii Trofimiuk, Peter Trifonov
Saint Petersburg Polytechnic University
Email: {grigoriyt,petert}@dcn.icc.spbstu.ru

***Abstract*—A decoding algorithm for polar codes with binary 16×16 kernels with polarization rate 0.51828 and scaling exponents 3.346 and 3.450 is presented. The proposed approach exploits the relationship of the considered kernels and the Arikan matrix to significantly reduce the decoding complexity without any performance loss. Simulation results show that polar (sub)codes with 16×16 kernels can outperform polar codes with Arikan kernel, while having lower decoding complexity.**

I. Introduction

Polar codes are a novel class of error-correcting codes, which achieve the symmetric capacity of a binary-input discrete memoryless channel W, have low complexity construction, encoding and decoding algorithms [1]. However, the performance of polar codes of practical length is quite poor. The reasons for this are the presence of imperfectly polarized subchannels and the suboptimality of the successive cancellation (SC) decoding algorithm. To improve performance, successive cancellation list decoding (SCL) algorithm [2], as well as various code constructions were proposed [3], [4], [5].

Polarization is a general phenomenon, and is not restricted to the case of Arikan matrix [6]. One can replace it by a larger matrix, called *polarization kernel*, which can provide higher polarization rate. Polar codes with large kernels were shown to provide asymptotically optimal scaling exponent [7]. Many kernels with various properties were proposed [6], [9], [10], [8], but, to the best of our knowledge, no efficient decoding algorithms for kernels with polarization rate greater than 0.5 were presented, except [11], where an approximate algorithm was introduced. Therefore, polar codes with large kernels are believed to be impractical due to very high decoding complexity.

In this paper we present reduced complexity decoding algorithms for 16×16 polarization kernels with polarization rate 0.51828 and scaling exponents 3.346 and 3.45. We show that with these kernels increasing list size in the SCL decoder provides much more significant performance gain compared to the case of Arikan kernel, and ultimately the proposed approach results in lower decoding complexity compared to the case of polar codes with Arikan kernel with the same performance.

The proposed approach exploits the relationship between the considered kernels and the Arikan matrix. Essentially, the log-likelihood ratios (LLRs) for the input symbols of the considered kernels are obtained from the LLRs computed via the Arikan recursive expressions.

II. Background

A. Channel polarization

Consider a binary input memoryless channel with transition probabilities $W\{y|c\}, c \in \mathbb{F}_2, y \in \mathcal{Y}$, where $\mathcal{Y}$ is output alphabet. For a positive integer n, denote by $[n]$ the set of n integers $\{0, 1, \ldots n-1\}$. A *polarization kernel* K is a binary invertible $l \times l$ matrix, which is not upper-triangular under any column permutation. The Arikan kernel is given by $F_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. An $(n = l^m, k)$ polar code is a linear block code generated by k rows of matrix $G_m = M^{(m)} K^{\otimes m}$, where $M^{(m)}$ is a digit-reversal permutation matrix, corresponding to mapping $\sum_{i=0}^{m-1} t_i l^i \rightarrow \sum_{i=0}^{m-1} t_{m-1-i} l^i, t_i \in [l]$. The encoding scheme is given by $c_0^{n-1} = u_0^{n-1} G_m$, where $u_i, i \in \mathcal{F}$ are set to some pre-defined values, e.g. zero (frozen symbols), $|\mathcal{F}| = n - k$, and the remaining values u_i are set to the payload data.

It is possible to show that a binary input memoryless channel W together with matrix G_m gives rise to bit subchannels $W_{m,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)$ with capacities approaching 0 or 1, and fraction of noiseless subchannels approaching $I(W)$ [6]. Selecting $\mathcal{F}$ as the set of indices of low-capacity subchannels enables almost error-free communication. It is convenient to define probabilities

$$W_{m,K}^{(i)}(u_0^i|y_0^{n-1}) = \frac{W_{m,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)}{2W(y_0^{n-1})} = \sum_{u_{i+1}^{n-1}} \prod_{i=0}^{n-1} W((u_0^{n-1} G_m)_i | y_i). \quad (1)$$

Let us further define $\mathbf{W}_m^{(j)}(u_0^j|y_0^{n-1}) = W_{m,K}^{(j)}(u_0^j|y_0^{n-1})$, where kernel K will be clear from the context. We also need probabilities $W_t^{(j)}(u_0^j|y_0^{l-1}) = W_{1,F_2^{\otimes t}}^{(j)}(u_0^j|y_0^{l-1})$ for Arikan matrix $F_2^{\otimes t}$. Due to the recursive structure of G_n, one has

$$\mathbf{W}_m^{(sl+t)}(u_0^{sl+t}|y_0^{n-1}) = \sum_{u_{sl+t+1}^{l(s+1)-1}} \prod_{j=0}^{l-1} \mathbf{W}_{m-1}^{(s)}(\theta_K[u_0^{l(s+1)-1}, j] | y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}) \quad (2)$$

where $\theta_K[u_0^{(s+1)l-1}, j]_r = (u_{lr}^{l(r+1)-1} G_n)_j, r \in [s+1]$. A trellis-based algorithm for computing these values was presented in [12].

At the receiver side, one can successively estimate

$$\widehat{u}_i = \begin{cases} \arg\max_{u_i \in \mathbb{F}_2} \mathbf{W}_m^{(i)}(\widehat{u}_0^{i-1}.u_i|y_0^{n-1}), & i \notin \mathcal{F}, \\ \text{the frozen value of } u_i & i \in \mathcal{F}. \end{cases} \quad (3)$$

This is known as the successive cancellation (SC) decoding algorithm.

III. Computing kernel input symbols LLRs

A. General case

Our goal is to compute efficiently probabilities $\mathbf{W}_m^{(i)}(u_0^i|y_0^{n-1})$ for a given polarization transform $K^{\otimes m}$. Let us assume for the sake of simplicity that $m = 1$. The corresponding task will be referred to as *kernel processing*.

We propose to introduce approximate probabilities

$$\widetilde{\mathbf{W}}_1^{(j)}(u_0^j|y_0^{l-1}) = \max_{u_{j+1}^{l-1}} \mathbf{W}_1^{(l-1)}(u_0^{l-1}|y_0^{l-1}) = \max_{u_{j+1}^{l-1}} \prod_{i=0}^{l-1} W((u_0^{l-1}K)_i|y_i). \quad (4)$$

This is the probability of the most likely continuation of path u_0^j in the code tree, without taking into account possible freezing constraints on symbols $u_i, i > j$. Note that the same probabilities were introduced in [11], [13], and shown to provide substantial reduction of the complexity of sequential decoding of polar codes.

Decoding can be implemented using the log-likelihood ratios $\bar{\mathbf{S}}_{m,i} = \bar{\mathbf{S}}_m^{(i)}(u_0^{i-1}|y_0^{n-1}) = \ln \frac{\mathbf{W}_m^{(i)}(u_0^{i-1}.0|y_0^{n-1})}{\mathbf{W}_m^{(i)}(u_0^{i-1}.1|y_0^{n-1})}$. Hence, kernel output LLRs $\bar{\mathbf{S}}_{1,i}, i \in [l]$ can be approximated by

$$\bar{\mathbf{S}}_{1,i} \approx \mathbf{S}_{1,i} = \ln \frac{\widetilde{\mathbf{W}}_1^{(i)}(u_0^{i-1}.0|y_0^{l-1})}{\widetilde{\mathbf{W}}_1^{(i)}(u_0^{i-1}.1|y_0^{l-1})} = \max_{u_{i+1}^{l-1}} \ln \mathbf{W}_1^{(l-1)}(u(0)^i|y_0^{l-1}) - \max_{u_{i+1}^{l-1}} \ln \mathbf{W}_1^{(l-1)}(u(1)^i|y_0^{l-1}), \quad (5)$$

where $b(a)^i = (b_0^{i-1}.a.b_{i+1}^{l-1})$. The above expression means that $\mathbf{S}_{1,i}$ can be computed by performing ML decoding of the code, generated by last $l-i+1$ rows of the kernel K, assuming that all $u_j, i < j < l$, are equiprobable.

B. Binary algorithm

Straightforward evaluation of (5) for arbitrary kernel has complexity $O(2^l l)$. However, we have a simple explicit recursive procedure for computing these values for the case of the Arikan matrix $F_2^{\otimes t}$.

Let $l = 2^t$. Consider encoding scheme $c_0^{l-1} = v_0^{l-1}F_2^{\otimes t}$. Similarly to (4), define approximate probabilities

$$\widetilde{W}_t^{(i)}(v_0^i|y_0^{l-1}) = \max_{v_{i+1}^{l-1}} W_t^{(l-1)}(v_0^{l-1}|y_0^{l-1})$$

and modified log-likelihood ratios

$$S_\lambda^{(i)}(v_0^{i-1}, y_0^{l-1}) = \log \frac{\widetilde{W}_\lambda^{(i)}(v_0^{i-1}.0|y_0^{l-1})}{\widetilde{W}_\lambda^{(i)}(v_0^{i-1}.1|y_0^{l-1})}.$$

It can be seen that

$$S_\lambda^{(2i)}(v_0^{2i-1}, y_0^{N-1}) = Q(a,b) \quad (6)$$

$$S_\lambda^{(2i+1)}(v_0^{2i}, y_0^{N-1}) = P(a,b,v_{2i}), \quad (7)$$

where $N = 2^\lambda$, $a = S_{\lambda-1}^{(i)}(v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}, y_{0,e}^{N-1})$, $b = S_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}, y_{0,o}^{N-1})$, $Q(a,b) = \operatorname{sgn}(a)\operatorname{sgn}(b)\min(|a|,|b|)$, $P(a,b,c) = (-1)^c a + b$. Then the log-likelihood of a path v_0^i can be obtained as [14]

$$R(v_0^i|y_0^{l-1}) = \log \widetilde{W}_t^{(i)}(v_0^i|y_0^{l-1}) = R(v_0^{i-1}|y_0^{l-1}) + \tau\left(S_t^{(i)}(v_0^{i-1}, y_0^{l-1}), v_i\right), \quad (8)$$

where $R(\epsilon|y_0^{l-1})$ can be set to 0, ϵ is an empty sequence, and

$$\tau(S, v) = \begin{cases} 0, & \operatorname{sgn}(S) = (-1)^v \\ -|S|, & \text{otherwise.} \end{cases}$$

It can be verified that

$$\sum_{\beta=0}^{2^j-1} \tau(S_0^{(0)}(y_\beta), c_\beta) = \sum_{\beta=0}^{2^j-1} \tau(S_j^{(\beta)}(v_0^{\beta-1}, y_0^{2^j-1}), v_\beta), \quad (9)$$

where $c = v_0^{2^j-1}F_2^{\otimes j}$.

It was suggested in [15] to express values $\mathbf{W}_1^{(i)}(u_0^i|y_0^{l-1})$ via $W_t^{(j)}(v_0^j|y_0^{l-1})$ for some j. One can represent the kernel K as $K = TF_2^{\otimes t}$, where T is an $l \times l$ matrix. Let $v_0^{l-1} = u_0^{l-1}T$. Then, $c_0^{l-1} = v_0^{l-1}F_2^{\otimes t} = u_0^{l-1}K$, so that $u_0^{l-1} = v_0^{l-1}T^{-1}$.

Observe, that it is possible to reconstruct u_0^i from $v_0^{\tau_i}$, where τ_i is the position of the last non-zero symbol in the i-th row of T^{-1}. Recall that successive cancellation decoding of polar codes with arbitrary kernel requires one to compute values $\mathbf{W}_1^{(i)}(u_0^i|y_0^{l-1}), u_i \in \mathbb{F}_2$. However, fixing the values u_0^{i-1} may impose constraints on $v_j, j > \tau_i$, which must be taken into account while computing these probabilities.

Indeed, vectors u_0^{l-1} and v_0^{l-1} satisfy the equation

$$\Theta'(u_{l-1} \quad \dots \quad u_1 \quad u_0 \quad v_0 \quad v_1 \quad \dots \quad v_{l-1})^T = 0,$$

where $\Theta' = (\mathbb{S} \quad I)$, and $l \times l$ matrix $\mathbb{S}$ is obtained by transposing T and reversing the order of columns in the obtained matrix. By applying elementary row operations, matrix Θ' can be transformed into a minimum-span form Θ, such that the first and last non-zero elements of the i-th row are located in columns i and z_i, respectively, where all z_i are distinct. This enables one to obtain symbols of vector u as

$$u_i = \sum_{s=0}^{i-1} u_s \Theta_{l-1-i,l-1-s} + \sum_{t=0}^{j_i} v_t \Theta_{l-1-i,l+t}, \quad (10)$$

where $j_i = z_{l-1-i} - l$. Let $h_i = \max_{0 \le i' \le i} j_{i'}$. It can be seen that[1]

$$\mathbf{W}_1^{(j)}(u_0^j|y_0^{l-1}) = \sum_{v_0^{h_j} \in \mathcal{Z}_j} W_t^{(h_j)}(v_0^{h_j}|y_0^{l-1}) = \sum_{v_0^{h_j} \in \mathcal{Z}_j} \sum_{v_{h_j+1}^{l-1}} W_t^{(l-1)}(v_0^{l-1}|y_0^{l-1}), \quad (11)$$

[1]The method given in [8] is a special case of this approach.

where $\mathcal{Z}_j$ is the set of vectors $v_0^{h_j}$, such that (10) holds for $i \in [j]$. Similarly we can rewrite the above expression for the case of the approximate probabilities

$$\widetilde{\mathbf{W}}_1^{(j)}(u_0^j|y_0^{l-1}) = \max_{v_0^{h_j} \in \mathcal{Z}_j} \widetilde{W}_t^{(h_j)}(v_0^{h_j}|y_0^{l-1}) = \max_{v_0^{h_j} \in \mathcal{Z}_j} \max_{v_{h_j+1}^{l-1}} W_t^{(l-1)}(v_0^{l-1}|y_0^{l-1}). \quad (12)$$

Let $\mathcal{Z}_{i,b} = \left\{ v_0^{h_i} | v_0^{h_i} \in \mathcal{Z}_i, \text{where } u_i = b \right\}$. Hence, one obtains

$$\mathbf{S}_{1,i} = \max_{v_0^{h_j} \in \mathcal{Z}_{i,0}} R(v_0^{h_i}|y_0^{l-1}) - \max_{v_0^{h_j} \in \mathcal{Z}_{i,1}} R(v_0^{h_i}|y_0^{l-1}). \quad (13)$$

Observe that computing these values requires considering multiple vectors $v_0^{h_i}$ of input symbols of the Arikan transform $F_2^{\otimes t}$. Let $\mathcal{D}_i = \{0, \dots, h_i\} \backslash \{j_0, \dots, j_i\}$ be a *decoding window*, i.e. the set of indices of Arikan input symbols $v_0^{h_i}$, which are not determined by symbols u_0^{i-1}. The number of such vectors, which determines the decoding complexity, is $2^{|\mathcal{D}_i|}$. In general, one has $|\mathcal{D}_i| = O(l)$ for an arbitrary kernel.

IV. Efficient processing of 16×16 kernels

To minimize complexity of proposed approach (13) one needs to find kernels with small decoding windows while preserving required polarization rate (> 0.5 in our case) and scaling exponent. By computer search, based on heuristic algorithm presented in [9], we found a 16×16 kernel

$$K_1 = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0 \\ 1&0&1&0&0&1&1&0&1&1&0&0&0&0&0&0 \\ 0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0 \\ 1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0 \\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{pmatrix}$$

with BEC scaling exponent $\mu(K_1) = 3.346$ [9]. Furthermore, to minimize the size of decoding windows, we derived another kernel $K_2 = P_\sigma K_1$, were P_σ is a permutation matrix corresponding to permutation $\sigma = [0, 1, 2, 7, 3, 4, 5, 6, 9, 10, 11, 12, 8, 13, 14, 15]$, with scaling exponent $\mu(K_2) = 3.45$. Both kernels have polarization rate 0.51828.

Table I presents the right hand side of expression (10) for each $i \in [16]$, as well as the corresponding decoding windows $\mathcal{D}_i$, for both kernels. It can be seen that the maximal decoding windows size for K_1 and K_2 is 4 and 3, respectively. Note that by applying the row permutation to K_1, we have reduced decoding windows, but increased scaling exponent. Below we present efficient methods for computing some input symbol LLRs for these kernels.

TABLE I: Input symbols u_ϕ for kernels K_1, K_2 as functions of input symbols v for $F_2^{\otimes 4}$

ϕ	K_1			K_2		
	u_ϕ	D_ϕ	Cost	u_ϕ	D_ϕ	Cost
0	v_0	{}	15	v_0	{}	15
1	v_1	{}	1	v_1	{}	1
2	v_2	{}	3	v_2	{}	3
3	v_4	{3}	21	v_3	{}	1
4	v_8	{3, 5, 6, 7}	103	v_4	{}	7
5	$v_6 \oplus v_9$	{3, 5, 6, 7}	49	v_8	{5, 6, 7}	68
6	$v_5 \oplus v_6 \oplus v_{10}$	{3, 5, 6, 7}	95	$v_6 \oplus v_9$	{5, 6, 7}	25
7	v_3	{5, 6, 7}	1	$v_5 \oplus v_6 \oplus v_{10}$	{5, 6, 7}	47
8	v_{12}	{5, 6, 7, 11}	175	v_6	{6, 7}	1
9	v_6	{6, 7, 11}	1	v_{10}	{7}	1
10	v_{10}	{7, 11}	1	v_7	{}	1
11	v_7	{11}	1	v_{11}	{}	1
12	v_{11}	{}	1	v_{12}	{}	7
13	v_{13}	{}	1	v_{13}	{}	1
14	v_{14}	{}	3	v_{14}	{}	3
15	v_{15}	{}	1	v_{15}	{}	1

A. Processing of kernel K_1 with $\mu = 3.346$

It can be seen that for $\phi \in \{0, 1, 2, 13, 14, 15\}$ one has $\mathbf{S}_{1,\phi} = S_4^{(\phi)}(v_0^\phi, y_0^{15})$, i.e. LLR for $F_2^{\otimes 4}$. For $\phi = 3$, expressions (13) and (10) imply that LLR for u_3 is given by

$$\mathbf{S}_{1,3} = \max_{v_3} R(v_0 v_1 v_2 v_3 0|y_0^{15}) - \max_{v_3} R(v_0 v_1 v_2 v_3 1|y_0^{15}),$$

where $v_i = u_i, i \in [3]$, are already estimated symbols.

Instead of exhaustive enumeration of vectors $v_0^{h_j}$ in (13), we propose to exploit the structure of K_1 to identify some common subexpressions (CSE) in formulas for $R(v_0^{h_i}|y_0^{l-1})$, which can be computed once and used multiple times. In some cases computing these subexpressions reduces to decoding of well-known codes, which can be implemented with appropriate fast algorithms. Furthermore, we observe that the set of possible values of these subexpressions is less than the number of different $v_0^{h_j}$ to be considered. This results in further complexity reduction. To demonstrate this approach, we consider computing the LLR for u_4 of K_1.

This requires considering 16 vectors v_0^7 satisfying (10). According to (8), one obtains $R(v_0^8|y_0^{15}) = R(v_0^7|y_0^{15}) + \tau\left(S_4^{(8)}(v_0^7, y_0^{15}), v_8\right)$. Observe that $\left\{v_0^7 F_2^{\otimes 3} | v_0^7 \in \bar{\mathcal{Z}}_4\right\}$ is a coset of Reed-Muller code $RM(1, 3)$, where $\bar{\mathcal{Z}}_j$ is the set of vectors $v_0^{h_j - 1}$, so that (10) holds for $i \in [j-1]$. Furthermore,

$$R(v_0^7|y_0^{15}) = \frac{1}{2}\left(\sum_{i=0}^{7}(-1)^{c_i} s_i - \sum_{i=0}^{7}|s_i|\right),$$

where $s_i = S_1^{(0)}(\epsilon, (y_i, y_{i+8})), i \in [8]$, $c_0^7 = v_0^7 F_2^{\otimes 3}$. Assume for the sake of simplicity that $v_j = 0, j \in \{0, 1, 2, 4\}$. Then the first term in this expression can be obtained for each $v_0^7 \in \bar{\mathcal{Z}}_4$ via the fast Hadamard transform (FHT) [16], and the second one does not need to be computed, since it cancels in (13).

It remains to compute $S_4^{(8)}(v_0^7, y_0^{15}), v_0^7 \in \bar{\mathcal{Z}}_4$ and $|\bar{\mathcal{Z}}_4| = 16$. In a straightforward implementation, one would recursively apply formulas (6) and (7) to compute $S_4^{(8)}$ for 16 vectors v_0^7. It appears that there are some CSE arising in this computation.

At first, one needs to compute $S_1^{(1)}(c_i, y_i, y_{i+8}), i \in [8]$, $c_0^7 = v_0^7 F_2^{\otimes 3}$. Since $c_i \in \{0, 1\}$, the values

$S_1^{(1)}(j, y_i, y_{i+8}), j \in \{0,1\}, i \in [8]$ constitute the first set of CSE. We store them in the array L

$$L[4i+j] = S_1^{(1)}(j \bmod 2, y_{i+\bar{j}}, y_{i+8+\bar{j}}),$$

where $i, j \in [4]$, $\bar{j} = 4\lfloor j/2 \rfloor$. Computing these values requires 16 summations only, instead of $16 \cdot 8 = 128$ summations in a straightforward implementation.

The next step is to compute the values $S_2^{(2)}((c_i, c_{i+4}), (y_i, y_{i+4}, y_{i+8}, y_{i+12})), i \in [4]$ which are equal to $Q(S_1^{(1)}(c_i, y_i, y_{i+8}), S_1^{(1)}(c_{i+4}, y_{i+4}, y_{i+12}))$. Since $(c_i, c_{i+4}) \in \mathbb{F}_2^2$, $S_2^{(2)}$ gives us the second set of CSE. One can use values stored in L to compute $S_2^{(2)}$ as

$$X[i][j] = Q(L[4i + j/2], L[4i + (j \bmod 2) + 2])], i, j \in [4].$$

Observe that for any $c_0^7 \in RM(1,3)$ one has $c_i^{i+4} \in RM(1,2), i \in \{0,4\}$. That is, one needs to consider only vectors c_i^{i+4} of even weight while computing $S_3^{(4)}$. These values can be calculated as

$$Y[i][j+4k] = Q(X[i][j \oplus 3k], X[i+2][j]), i, k \in [2], j \in [4].$$

Finally, the values $S_4^{(8)}(v_0^7, y_0^{15})$ can be obtained as

$$Z[i+8j] = Q(Y[0][i \oplus 3j], Y[1][i]), i \in [8], j \in [2].$$

Each element of Z corresponds to some $c_0^7 \in RM(1,3)$. Finally, these values are used in (13) together with $R(v_0^7|y_0^{15})$ to calculate $\mathbf{S}_{1,4}$.

Similar CSE elimination techniques can be used to obtain efficient methods for computing $\mathbf{S}_{1,i}, 5 \le i \le 8$. For $8 < i < 13$ one needs just to find maximal elements in the sets of $R(v_0^{12}|y_0^{15})$ computed for $i = 8$, as shown in (13). Furthermore, since for these phases one has $\mathcal{Z}_{i+1} = \mathcal{Z}_{i,b}$, where b is 0 or 1, one can save and reuse the intermediate values of computing the maximum for $i = 8$.

Due to space constraints, we do not present the detailed description of the corresponding algorithm.

B. Processing of kernel K_2 with $\mu = 3.45$

Below we briefly present a complete processing algorithm for kernel K_2. It provides much better performance-complexity tradeoff compared to K_1. The algorithm uses the same CSE elimination techniques as described in section IV-A. After the pre-computation steps for $\phi \in \{5,6,7\}$, the LLR is obtained via (13).

- For $\phi \in \{0,1,2,3,4,11,12,13,14,15\}$, compute $\mathbf{S}_{1,\phi}$ as LLRs $S_4^{(\phi)}(v_0^{\phi-1}, y_0^{15})$ for the Arikan transform $F_2^{\otimes 4}$.
- For $\phi = 5$:
  1) Since for fixed $v_i, i \in \{0,1,2,3,4\}$, the set of vectors $c_0^7 = v_0^7 F_2^{\otimes 3}$ is a coset of $RM(1,2)$ concatenated with $(2,1,2)$ code, one can obtain 8 values of $R(v_0^7|y_0^{15})$ from the FHT of vector $(S_1^{(0)}(\epsilon, (y_i, y_{i+8})) + S_1^{(0)}(\epsilon, (y_{i+4}, y_{i+12})), i \in [4])$.
  2) Compute $L[i+8j] = S_1^{(1)}(j \bmod 2, y_i, y_{i+8}), i \in [8], j \in [2]$.
  3) Since $\{(c_i, c_{i+4})\} = \{(0,0),(1,1)\}$, compute all possible $S_2^{(2)}$ values as $X[i][j] = Q(L[i+8j], L[i+4+8j]), i \in [4], j \in [2]$.
  4) Since $\{(c_i, c_{i+4}, c_{i+2}, c_{i+6})\}$ is a code generated by $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$, compute all possible $S_3^{(4)}$ values as $Y[i][j] = Q(X[i][j/2], X[i+2][j \bmod 2]), i \in [2], j \in [4]$.
  5) For every $c_0^7 = v_0^7 F_2$ compute $S_4^{(8)}(v_0^7, y_0^{15})$ as $Z[i+4j] = Q(Y[0][i \oplus 3j], Y[1][i]), i \in [4], j \in [2]$.
- For $\phi = 6$, compute $S_4^{(9)}(v_0^8, y_0^{15})$ as $Z[i+4j] = P(Y[0][i \oplus 3j], Y[1][i], v_8), i \in [4], j \in [2]$.
- For $\phi = 7$:
  1) Compute $S_3^{(5)}$ as $\bar{Y}[0][j+4k] = P(X[0][j/2], X[2][j \bmod 2], \bar{c}_{0,k})$, $\bar{Y}[1][j+4k] = P(X[3][\bar{j}_k/2], X[1][\bar{j}_k \bmod 2], \bar{c}_{1,k})$ and $\bar{c}_{0,k} = v_8 \oplus u_6 \oplus k$, $\bar{c}_{1,k} = u_6 \oplus k$, $\bar{j}_k = j \oplus 3$, $i \in [2], j \in [4], k \in [2]$.
  2) Obtain $S_4^{(10)}$ as $Z[i] = Q(\bar{Y}[0][i], \bar{Y}[1][i]), i \in [8]$.
- For $\phi \in \{8,9,10\}$: Use 16 already computed values of $R(v_0^{10}|y_0^{15})$ to obtain $\mathbf{S}_{1,\phi}$.

Remark 1. *Let us comment the case of $\phi = 7$. In conventional Arikan SC for $F_2^{\otimes 4}$, after symbol v_9 is estimated, the LLRs $S_3^{(5)}$ for v_{10} are obtained by applying P function to $S_2^{(2)}$ and values $(v_8 \oplus v_9, v_8)$. In the case of K_2, we do not have a fixed value for v_9. Instead, we have a constraint $u_6 = v_6 + v_9$. Therefore, for each vector $v_0^9 \in \bar{\mathcal{Z}}_7$ the value of v_9 is changed according to v_6. This property is taken into account in the expressions for computing of $\bar{Y}[i][j]$.*

The cost, in terms of the total number of summations and comparisons, of computing $\mathbf{S}_{1,\phi}$ using the proposed algorithm is shown in Table I. The overall processing complexity is 472 and 183 operations for kernels K_1 and K_2 respectively, while the trellis-based algorithm [12] requires 7557 and 9693 operations, respectively.

The above described techniques can be also used to implement an SCL decoder for polar codes with the considered kernels, using a straightforward generalization of the algorithm and data structures presented in [2].

V. Numeric results

We constructed $(4096, 2048)$ polar codes with the considered kernels, and investigated their performance for the case of AWGN channel with BPSK modulation. The sets of frozen symbols were obtained by Monte-Karlo simulations.

Figure 1 illustrates the performance of plain polar codes, polar codes with CRC[2] and polar subcodes [4]. It can be seen that the codes based on kernels K_1 and K_2 with improved polarization rate $E(K_1) = E(K_2) = 0.51828$ provide significant performance gain compared to polar codes with Arikan kernel. Observe also that randomized polar subcodes provide better performance compared to polar codes with CRC. Moreover, polar subcodes with kernels K_1, K_2 under SCL with $L = 8$ have almost the same performance as polar subcodes with Arikan kernel under SCL with $L = 32$. Observe also that the

[2]CRC length was selected to minimize FER with $L = 8$.

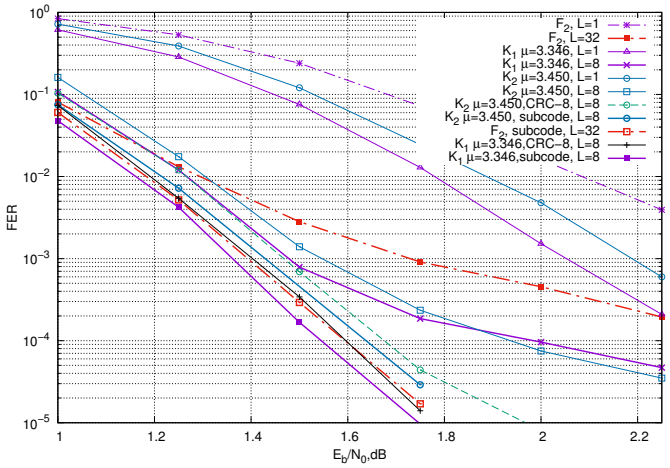
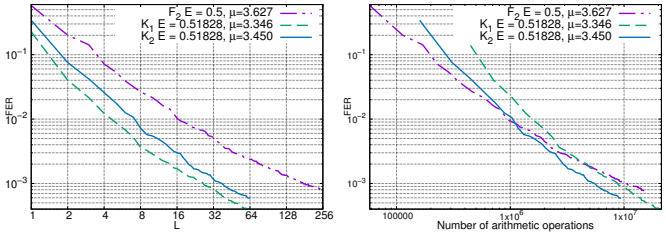


Fig. 1: Performance of $(4096, 2048)$ polar codes



(a) Performance (b) Decoding complexity

Fig. 2: SCL decoding of polar subcodes with different kernels

codes based on kernels with lower scaling exponent exhibit better performance.

Figure 2a presents simulation results for $(4096, 2048)$ polar subcodes with different kernels under SCL with different L at $E_b/N_0 = 1.25$ dB. It can be seen that the kernels with polarization rate 0.51828 require significantly lower list size L to achieve the same performance as the code with the Arikan kernel. Moreover, this gap grows with L. This is due to improved rate of polarization, which results in smaller number of unfrozen imperfectly polarized bit subchannels. The size of the list needed to correct possible errors in these subchannels grows exponentially with their number (at least for the genie-aided decoder considered in [17]). On the other hand, lower scaling exponent gives better performance with the same list L, but the slope of the curve remains the same for both kernels K_1, K_2.

Figure 2b presents the same results in terms of the actual decoding complexity. Recall that proposed kernel processing algorithm uses only summations and comparisons. The SCL algorithm was implemented using the randomized order statistic algorithm for selection of the paths to be killed at each phase, which has complexity $O(L)$. Observe that the polar subcode based on kernel K_2 can provide better performance with the same decoding complexity for FER $\leq 8 \cdot 10^{-3}$. This is due to higher slope of the corresponding curve in Figure 2a, which eventually enables one to compensate relatively high complexity of the LLR computation algorithm presented in Section IV.

Unfortunately, K_1 kernel, which provides lower scaling exponent, has greater processing complexity than K_2, so that its curve intersects the one for the Arikan kernel only at FER$= 2 \cdot 10^{-3}$.

VI. Conclusions

In this paper efficient decoding algorithms for some 16×16 polarization kernels with polarization rate 0.51828 were proposed. The algorithms compute kernel input symbols LLRs via the ones for the Arikan kernel, and exploit the structure of the codes induced by the kernel to identify and re-use the values of some common subexpressions. It was shown that in the case of SCL decoding with sufficiently large list size, the proposed approach results in lower decoding complexity compared to the case of polar (sub)codes with Arikan kernel with the same performance.

Extension of the proposed approach to the case of other kernels remains an open problem.

References

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. on Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

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

[3] P. Trifonov and V. Miloslavskaya, "Polar subcodes," *IEEE J. on Selected Areas in Comm.*, vol. 34, no. 2, pp. 254–266, February 2016.

[4] P. Trifonov and G. Trofimiuk, "A randomized construction of polar subcodes," in *Proc. of IEEE ISIT*, 2017, pp. 1863–1867.

[5] T. Wang, D. Qu, and T. Jiang, "Parity-check-concatenated polar codes," *IEEE Communications Letters*, vol. 20, no. 12, December 2016.

[6] S. B. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. on Inf. Theory*, vol. 56, no. 12, pp. 6253–6264, December 2010.

[7] A. Fazeli, S. H. Hassani, M. Mondelli, and A. Vardy, "Binary linear codes with optimal scaling and quasi-linear complexity," *arXiv:1711.01339*, 2017.

[8] S. Buzaglo, A. Fazeli, P. H. Siegel, V. Taranalli, and A. Vardy, "On efficient decoding of polar codes with large kernels," in *Proc. of 2017 IEEE WCNC Workshops*, March 2017, pp. 1–6.

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

[10] N. Presman, O. Shapira, S. Litsyn, T. Etzion, and A. Vardy, "Binary polarization kernels from code decompositions," *IEEE Trans. On Inf. Theory*, vol. 61, no. 5, May 2015.

[11] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes with arbitrary binary kernel," in *Proc of IEEE ITW*, 2014, pp. 377–381.

[12] H. Griesser and V. R. Sidorenko, "A posteriory probability decoding of nonsystematically encoded block codes," *Problems of Information Transmission*, vol. 38, no. 3, 2002.

[13] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1127–1130, 2014.

[14] P. Trifonov, "A score function for sequential decoding of polar codes," in *Proc. of IEEE ISIT*, 2018.

[15] ——, "Binary successive cancellation decoding of polar codes with Reed-Solomon kernel," in *Proc. of IEEE ISIT*, 2014, pp. 2972 – 2976.

[16] Y. Beery and J. Snyders, "Optimal soft decision block decoders based on fast Hadamard transform," *IEEE Trans. on Inf. Theory*, vol. 32, no. 3, May 1986.

[17] M. Mondelli, S. H. Hassani, and R. Urbanke, "Scaling exponent of list decoders with applications to polar codes," *IEEE Trans. On Inf. Theory*, vol. 61, no. 9, September 2015.