Construction of binary polarization kernels for low complexity window processing

Grigorii Trofimiuk, Peter Trifonov
ITMO University, Russia
Email: {gtrofimiuk,pvtrifonov}@itmo.ru

***Abstract*—An algorithm for construction of binary polarization kernels of size 16 and 32 with polarization rate greater than 0.5, which admit low complexity processing is proposed. Kernels are obtained by employing such linear transformations of the Arikan matrix, which minimize the complexity of the window processing algorithm, while preserving required rate of polarization. Simulation results show that polar subcodes with obtained kernels can outperform polar codes with Arikan kernel, while having lower decoding complexity.**

I. Introduction

Polar codes with large kernels were shown to provide asymptotically optimal scaling exponent [1]. Many kernels with various properties were proposed [2], [3], [4], [5]. Until recently, polar codes with large kernels were believed to be impractical due to very high decoding complexity.

The window processing algorithm for some polarization kernels of size 16 and 32 was introduced in [6] and [7] respectively. This 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.

In this paper we present a construction method for polarization kernels of size 16 and 32 with high rate of polarization, which admit efficient processing by window based approach. The proposed method constructs a set of polarization kernels by performing such linear transformations of Arikan matrix, which minimize the complexity of the window processing algorithm. At the same time, these transformations are aimed on achieving the required rate of polarization.

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. 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 $F_m = \begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}^{\otimes m}$.

For a positive integer n, denote by $[n]$ the set of integers $\{0,1,\dots n-1\}$. 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 $\sum_{i=0}^{m-1} t_i l^i \to \sum_{i=0}^{m-1} t_{m-1-i} l^i, t_i \in [l]$. 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 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 give rise to bit subchannels $W_{m,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)$, such that their capacities converge with n to 0 or 1, and fraction of almost noiseless subchannels converges to $I(W)$ [2]. Selecting $\mathcal{F}$ as the set of indices of low-capacity subchannels enables almost error-free communication. It is convenient to define probabilities

$$W_{m,K}^{(i)}(u_0^i|y_0^{n-1}) = \frac{W_{m,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)}{2W(y_0^{n-1})}. \tag{1}$$

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 $W_t^{(j)}(u_0^j|y_0^{2^t-1}) = W_{1,F_t}^{(j)}(u_0^j|y_0^{2^t-1})$ for Arikan matrix F_t. Due to the recursive structure of G_m, one has

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

where $\theta_K[u_0^{(s+1)l-1}, j]_r = (u_{lr}^{l(r+1)-1}G_m)_j, r\in[s+1]$. A trellis-based algorithm for computing these values was presented in [8].

At the receiver side, one can successively estimate

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

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

B. Properties of polarization kernels

1) Rate of polarization: The rate of polarization shows how fast bit subchannels $W_{m,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)$ of $K^{\otimes m}$ approach either almost noiseless or noisy channel with $n = l^m$ [2].

Let $\langle g_1, g_2, \dots, g_k\rangle$ be a linear code, generated by the vectors $g_1, g_2, \dots, g_k$. Let $d_H(a,b)$ be the Hamming distance between a and b. Let $d_H(b,\mathcal{C}) = \min_{c\in\mathcal{C}} d_H(b,c)$ be a minimal distance between vector b and linear block code $\mathcal{C}$. We denote the i-th row of a matrix M as $M[i]$.

The *partial distances* (PDs) $\mathbb{D}_i, i \in [l]$, of the $l \times l$ matrix K are defined as follows:

$$\mathbb{D}_i = d_H(K[i], \langle K[i+1], \ldots, K[l-1]\rangle), i \in [l-1]$$
$$\mathbb{D}_{l-1} = d_H(K[l-1], \mathbf{0}).$$

The vector $\mathbb{D}$ will be referred to as a *partial distances profile* (PDP). Here we assume that a kernel with considered PDP $\mathbb{D}$ exists. In [2] it was shown that for any B-DMC W and any $l \times l$ polarization kernels K with PDP $\mathbb{D}$, the rate of polarization $E(K)$ is given by $E(K) = \frac{1}{l}\sum_{i=0}^{l-1} \log_l \mathbb{D}_i$.

The Arikan kernel F_1 has rate of polarization $E(F_1) = 0.5$, whereas random kernels achieve $E = 1$. The kernels of size 16 and 32 with rate of polarization 0.51828 and 0.53656 respectively can be obtained [10].

2) Scaling exponent: Given W and P_e, suppose we wish to communicate at rate $I(W) - \Delta$ using a family of (n, k) polar codes with kernel K. It has been shown that this value of n scales as $O(\Delta^{-\mu(K)})$, where the constant $\mu(K)$ is known as the scaling exponent [3]. We will compute the scaling exponent for the case of binary erasure channel (BEC) [9],[3].

The Arikan kernel F_1 has $\mu(K) = 3.627$, whereas random codes achieve optimal $\mu = 2$. The best known scaling exponents for 16×16 and 32×32 polarization kernels are 3.346 [6] and 3.127 [10] respectively.

C. Computing kernel input symbols LLRs

In this section we consider computing of probabilities $\mathbf{W}_1^{(i)}(u_0^i|y_0^{n-1})$ for a polarization kernel K. The corresponding task will be referred to as *kernel processing*.

1) General case: In this work we use the approximate probabilities

$$\widetilde{\mathbf{W}}_1^{(j)}(u_0^j|y_0^{l-1}) = \max_{u_{j+1}^{l-1}} \mathbf{W}_1^{(l-1)}(u_0^{l-1}|y_0^{l-1}), \tag{4}$$

which were introduced in [11], [12].

Decoding can be implemented using the log-likelihood ratios of the approximated probabilities (4)

$$\mathbf{S}_{1,i} = \ln \frac{\widetilde{\mathbf{W}}_1^{(i)}(u_0^{i-1}.0|y_0^{l-1})}{\widetilde{\mathbf{W}}_1^{(i)}(u_0^{i-1}.1|y_0^{l-1})} = R(0) - R(1),$$

where $R(a) = \max_{u_{i+1}^{l-1}} \ln \mathbf{W}_1^{(l-1)}(u_0^{i-1}.a.u_{i+1}^{l-1}|y_0^{l-1})$. The value of i will be referred to as a processing *phase*. 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.

2) Window processing: Let $l = 2^t$. Consider encoding scheme $c_0^{l-1} = v_0^{l-1} F_t$. Similarly to (4), we define approximate probabilities $\widetilde{W}_t^{(i)}(v_0^i|y_0^{l-1})$ and modified log-likelihood ratios $S_t^{(i)}(v_0^{i-1}, y_0^{l-1}) = \log \frac{\widetilde{\mathbf{W}}_t^{(i)}(v_0^{i-1}.0|y_0^{l-1})}{\widetilde{\mathbf{W}}_t^{(i)}(v_0^{i-1}.1|y_0^{l-1})}$ of F_t.

It can be seen that

$$S_\lambda^{(2i)}(v_0^{2i-1}, y_0^{N-1}) = \operatorname{sgn}(a)\operatorname{sgn}(b)\min(|a|, |b|) \tag{5}$$
$$S_\lambda^{(2i+1)}(v_0^{2i}, y_0^{N-1}) = (-1)^{v_{2i}} a + b, \tag{6}$$

where $N = 2^\lambda$, $a = S_{\lambda-1}^{(i)}(v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}, y_{0,e}^{N-1})$, $b = S_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}, y_{0,o}^{N-1})$.

Then the log-likelihood of a path v_0^i can be obtained as [13]

$$R(v_0^i|y_0^{l-1}) = R(v_0^{i-1}|y_0^{l-1}) + \tau\left(S_t^{(i)}(v_0^{i-1}, y_0^{l-1}), v_i\right), \tag{7}$$

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 was suggested in [14] and [6] 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. Indeed, let $TK = F_t$, where the matrix T will be referred to as *transition matrix*. Let $c_0^{l-1} = v_0^{l-1} F_2^{\otimes t} = u_0^{l-1} K$, thus, $u_0^{l-1} = v_0^{l-1} T$.

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 column of T. For the sake of simplicity we assume that all $\tau_i, i \in [l]$ are distinct. The general case is considered in [6].

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

$$u_i = \sum_{j=0}^{l-1} v_j T[j, i], \tag{8}$$

where $T[i, j]$ is the j-th element of row $T[i]$.

Let $h_i = \max_{i' \in [i+1]} \tau_{i'}$ and $\mathcal{Z}_j$ be the set of vectors $v_0^{h_j}$, such that (8) holds for $i \in [j]$. 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 [6]

$$\mathbf{S}_{1,i} = \max_{v_0^{h_i} \in \mathcal{Z}_{i,0}} R(v_0^{h_i}|y_0^{l-1}) - \max_{v_0^{h_i} \in \mathcal{Z}_{i,1}} R(v_0^{h_i}|y_0^{l-1}). \tag{9}$$

Observe that computing these values requires considering multiple vectors $v_0^{h_i}$ of input symbols of the Arikan matrix F_t. Let $\mathcal{D}_i = [h_i + 1] \setminus \{\tau_0, \tau_1, \ldots, \tau_i\}$ be a *decoding window*, i.e. the set of indices of independent (from u_0^{i-1}) components of $v_0^{h_i}$. Note that $|\mathcal{D}_i| = |[h_i + 1]| - |\{\tau_0, \tau_1, \ldots, \tau_i\}| = h_i - i$ since all τ_i are distinct and $\{\tau_0, \tau_1, \ldots, \tau_i\} \subseteq [h_i + 1]$. The calculation of LLRs $\mathbf{S}_{1,i}$ via (9) will be referred to as the *window processing* algorithm.

The number of path scores to be computed in (9), which determines the processing complexity, is equal to $2^{|\mathcal{D}_i|+1}$. Let $\mathcal{M}(K)$ denote the $\max_{i \in [l]} |\mathcal{D}_i|$. In general, one has $\mathcal{M}(K) = O(l)$ for an arbitrary kernel K.

3) Complexity: In this section we derive an estimate of the complexity of the window processing algorithm in terms of the number of arithmetical (summation and comparison) operations.

The complexity of the window processing algorithm at the phase i depends on the number of considered paths $v_0^{h_i}$ in Arikan matrix F_t and complexity of calculation of the path score $R(v_0^{h_i}|y_0^{l-1})$ of the single path.

According to (7), calculation of each path score $R(v_0^{h_i}|y_0^{l-1})$ requires computing the LLR $S_t^{(h_i)} = S_t^{(h_i)}(v_0^{h_i-1}, y_0^{l-1})$. The complexity of computing of $S_t^{(h_i)}$ with reuse of intermediate LLRs is given by $2^{B(h_i)} - 1$ operations [15], where $B(h)$ is a position of the last nonzero bit in the binary representation of h, i.e. $h = 2^{b_0} + 2^{b_1} + \cdots + 2^{B(h)}$. If $h = 0$ then $B(h)$ is assumed to be t.

Then, to compute (7), one should add the value $R(v_0^{h_{i-1}}|y_0^{l-1})$ to the value $\tau\left(S_t^{(h_i)}, v_i\right)$. As far as $v_i \in [2]$, it can be done in at most 1 summation. Therefore, it gives $2^{|\mathcal{D}_i|}$ operations more. Moreover, if $h_i - h_{i-1} > 1$, then the above described computations should be done for LLRs $S_t^{(h)}$, $h_{i-1} < h \le h_i$. It can be seen, that the number of such LLRs is given by $2^{|\mathcal{D}_i|-(h_i-h)} = 2^{h-i}$.

In total, the complexity of calculation of path scores $R(v_0^{h_i}|y_0^{l-1})$, is given by $\Lambda(i) = \sum_{h=h_{i-1}+1}^{h_i} 2^{h+B(h)-i}$, where $v_0^{h_i} \in \mathcal{Z}_i$ and h_{-1} is assumed to be -1.

To compute the LLR $\mathbf{S}_{1,i}$ according to (9), one needs $2^{|\mathcal{D}_i|+1}$ comparisons. Note that in the case of $h_i = h_{i-1}$ path scores remain the same. We assume that the maximums of scores of paths from $\mathcal{Z}_{i,b}, b \in [2]$, are also known, since maximums from $\mathcal{Z}_{i-1,b}$ were calculated. Therefore, one substraction needed only to compute the input symbol LLR.

Finally, the complexity of the straightforward implementation of the window processing algorithm for kernel K can be estimated as

$$\Psi(K) = \sum_{i=0}^{l-1} \Phi(i), \tag{10}$$

where $\Phi(i) = \begin{cases} 2^{h_i-i+1} + \Lambda(i), & h_i > h_{i-1}, \\ 1, & \text{otherwise.} \end{cases}$

III. Construction of polarization kernels

Our goal is to construct polarization kernels with polarization rate greater that 0.5, which admit low complexity window processing. Such rate of polarization can be achieved for kernels of size $l = 16$ and $l \ge 23$ [2]. In this work we focus on polarization kernels of size 16 and 32.

The minimization of the processing complexity (10) by the exhaustive search over all polarization kernels K of desired size and rate of polarization is infeasible. Therefore, we propose to consider some restricted set of polarization kernels, which are expected to have moderate window processing complexity $\Psi(K)$ and required PDP.

Every $2^t \times 2^t$ polarization kernel K can be derived by application of elementary operations to rows of Arikan matrix F_t, since F_t is invertible. The window processing algorithm exploits this linear relationship to obtain LLRs $\mathbf{S}_{1,i}$ (9). The complexity $\Psi(K)$ of computing these LLRs is also determined by this linear relationship. Therefore, we will study, how elementary operations over F_t affect the window processing complexity $\Psi(K)$ of the obtained kernel K. Further, we construct the set of kernels by application of such elementary operations over rows of F_t, which are expected to result in kernels K with moderate processing complexity $\Psi(K)$, while having required PDs.

A. Elementary operations

1) Row permutation: Let K be an $l \times l, l = 2^t$, polarization kernel. We express the linear relationship between F_t and K by the transition matrix $T = F_t K^{-1}$, which is described in section II-C2. Recall that τ_i is the position of the last non-zero symbol in the i-th column of T, $h_i = \max_{i' \in [i+1]} \tau_{i'}$.

Let P_ρ be a matrix, which corresponds to the permutation $\rho = [\rho(0), \rho(1), \dots, \rho(l-1)]$. Consider the permuted Arikan kernel $F_{t,\rho} = P_\rho F_t$. The transition matrix T of $F_{t,\rho}$ is given by $F_t {F_{t,\rho}}^{-1} = P_\rho^T$. Thus, all $\tau_i, i \in [l]$ are given by $\rho(i)$.

It can be seen than in the case of permuted Arikan kernel, the value of $h_i - i$ becomes positive, once $\rho(i) = \tau_i > i, i \in [l]$, appears in T. Therefore the processing complexity of $F_{t,\rho}$ depends on the value of $\rho(i) - i$ in cases, where $\rho(i) > i$. It means, that we should keep the values $\rho(i) - i$ as small as possible to keep the overall processing complexity moderate.

We also introduce the permutation σ, where $\sigma(i) \in [l]$ are sorted according to the Hamming weight of its binary representation first and by ascending order second. We denote the kernel $F_{t,\sigma}$ as sorted Arikan matrix. It is noticeable, that $F_{t,\sigma}$ has the minimal window processing complexity among permuted Arikan matrices $F_{t,\beta}$.

For instance, consider the sorted Arikan kernel $F_{4,\sigma}$[5], where $\sigma = [0,1,2,4,8,3,5,6,9,10,12,7,11,13,14,15]$. The decoding windows $\mathcal{D}_i$ in this kernel becomes non-empty on the phases with $\rho(i) > i$. For example, $h_4 = \rho(4) = 8, \mathcal{D}_4 = \{3,5,6,7\}$. At the next phases $i \in \{5,6,7\}$, h_i still equals to 8, which preserves $h_i > i$ and leads to increased complexity.

2) Row addition: The i-th PD $\mathbb{D}[i]$ of the kernel K might be increased by performing addition operations over its rows. It is shown [3], that addition of the row $K[i]$ to row $K[j]$ with $i > j$ does not change the rate of polarization and scaling exponent of K. Thus, we consider row additions with $i < j$.

The addition of two rows can also increase the maximal size of the decoding windows. Indeed, let $X_{i,j}$ be an elementary matrix which corresponds to addition of row i to row $j, i \neq j$. In other words, $X_{i,j}$ is a matrix with 1's on the diagonal and $X_{i,j}[j,i] = 1$. Note that $X_{i,j}^{-1} = X_{i,j}$.

Let $K = T^{-1} F_t$. Suppose we added row $K[i]$ to $K[j], i < j$ and obtained the kernel $\bar{K} = X_{i,j} K$ with transition matrix $\bar{T} = F_t \bar{K}^{-1} = T X_{i,j}$. It means that the matrix $\bar{T}$ was obtained by addition of j-th column of T to the i-th one. Let $\bar{\tau}_i$ be the position of the last non-zero symbol in the i-th column of $\bar{T}$. After row addition of row i to row j in K, $\bar{\tau}_i = \max(\tau_i, \tau_j)$, which means that $|\bar{\tau}_i - i| \ge |\tau_i - i|$ and the size of the decoding window $|\mathcal{D}_i|$ may increase. Therefore, one should use addition matrices $X_{i,j}$ with as small as possible values $|j-i|$.

For example, if we add $F_3[3]$ to $F_3[6]$, then $u_3 = v_3 \oplus v_6$ (8), $h_3 = \tau_3 = \max(3,6) = 6$ and $\mathcal{D}_3 = \{3,4,5\}$.

B. Kernel construction

1) The general construction algorithm: In this section we describe the general principles of construction of the polarization kernels with low window processing complexity.

The construction starts from determining the desired PDP $\mathbb{D}_*$. For convenience, we consider first the case of monotonically increasing PDs (MI-PDs), i.e. $\mathbb{D}_*[i] \le \mathbb{D}_*[i+1], i \in [l-1]$. The upper-bound on the MI-PDs was proposed in [4].

Let $\beta \in B$ be a permutation, which results in a permuted Arikan matrix F_β with monotonically increasing PDP (MI-PDP) $\mathbb{D}_\beta = [1, 2_{*\binom{t}{1}}, 4_{*\binom{t}{2}}, \dots, 2^{t-1}_{*\binom{t}{t-1}}, 2^t]$, where a_{*b} denotes b-times repetition of an element a. In our construction, we also assume that $\mathbb{D}_*[i] \geq \mathbb{D}_\beta[i], i \in [l]$.

Our goal is to obtain an $l \times l, l = 2^t$, polarization kernel K with given MI-PDP $\mathbb{D}_*$ and as small as possible window processing complexity (10). We propose to minimize the complexity (10) over a set $\mathbb{K}$ of candidate matrices M, which are obtained by employing such elementary operations over rows of F_t, which minimize the complexity $\Psi(M)$, while preserving the required MI-PDP $\mathbb{D}_*$.

The naive construction of the set $\mathbb{K}$ is as follows. At first, for each permutation $\beta \in B$ we obtain the kernel $F_{t,\beta}$. Then, for each obtained $F_{t,\beta}$ we try to increase PDs. i.e. for each i, where $\mathbb{D}_*[i] > \mathbb{D}_\beta[i], i \in [l]$, we add rows $F_{t,\beta}[j], 0 \leq j < i$ to the row $F_{t,\beta}[i]$ to obtain row $M[i]$ with $w_H(M_i) = \mathbb{D}_*[i]$, where $w_H(c) = d_H(c, \mathbf{0})$. In the next sections we show that the set $\mathbb{K}$ can be significantly reduced.

2) The construction of 16×16 *kernels:* We illustrate the proposed general algorithm by construction of the 16×16 polarization kernels. We are going to minimize the processing complexity for kernels with $\mathbb{D}_* = [1, 2_{*4}, 4_{*4}, 6_{*2}, 8_{*4}, 16]$. Note that kernels with PDP $\mathbb{D}_*$ has maximal rate of polarization among 16×16 kernels, while the permuted Arikan matrix $F_{4,\beta}$ has MI-PDP $\mathbb{D}_\beta = [1, 2_{*4}, 4_{*6}, 8_{*4}, 16]$.

This implies that we need to transform rows $K_{4,\beta}[9]$ and $K_{4,\beta}[10]$ to rows $M[i], i \in \{9, 10\}$, with $\mathbb{D}_i = 6$. We also assume that $\mathbb{D}_i = w_H(M[i]), i \in [l]$. To reduce the size of the set $\mathbb{K}$ of the candidate matrices M, we propose to obtain these rows as a linear combination of rows $F_{4,\sigma}[i], i \in [5:11]$, which have $w_H(F_{4,\sigma}) = 4$, where $[a:b]$ denotes the set $\{a, a+1, \ldots, b-1\}$. Note that this construction allows one to keep unchanged rows $F_{4,\sigma}[i], i \in [32] \backslash [5:11]$, which have $w_H(F_{4,\sigma}) \neq 4$.

The constraints on the matrix $M \in \mathbb{K}$ are following:

- $M[i] = F_{4,\sigma}[i], i \in [16] \backslash \mathbb{M}_1$, where $\mathbb{M}_1 = [5:11]$,
- $M[9], M[10] \in \{c \in \langle\{F_{4,\sigma}[i], i \in \mathbb{M}_1\}\rangle | w_H(c) = 6\}$,
- $M[i] \in \{F_{4,\sigma}[j], j \in \mathbb{M}_1\}$, where $i \in [5:9]$.

It is easy to observe, that the proposed construction can produce kernels with partial distances distinct from $\mathbb{D}_*$ and even singular matrices. One should check the obtained matrices and eliminate those with invalid PDP.

C. Enhanced construction algorithm

In this section we provide the modifications of the algorithm, presented in III-B, and illustrate it by construction of 32×32 kernels. The maximal rate of polarization among kernels of size 32 is 0.53656, which is achieved by kernels with PDP $\mathbb{D}_* = [1, 2_{*5}, 4_{*5}, 6_{*5}, 8_{*5}, 12_{*5}, 16_{*5}, 32]$, while MI-PDP of permuted Arikan matrix $F_{5,\beta}$ is given by $\mathbb{D}_\beta = [1, 2_{*5}, 4_{*10}, 8_{*10}, 16_{*5}, 32]$. It means that one should transform $F_{5,\beta}$ to obtain matrices M with $w_H(M[i]) = \mathbb{D}[i] = 6, i \in [11:16]$ and $\mathbb{D}[i] = w_H(M[i]) = 12, i \in [21:26]$.

To reduce the search space, we propose to obtain rows of weight 6 and 12 as linear combination of rows with weight 4 and 8 respectively. We start minimization process considering the set $\mathbb{K}$ of matrices M with following constraints:

- $M[i] = F_{5,\sigma}[i], i \in [31] \backslash \mathbb{M}_2$, $\mathbb{M}_2 = [16:26]$,
- $M[i] \in \{c \in \mathcal{C} | d_H(c, \mathbf{0}) = 12\}$, where $\mathcal{C} = \langle\{F_{5,\sigma}[j], j \in \mathbb{M}_2\}\rangle$ and $i \in [21:26]$,
- $M[i] \in \{F_{5,\sigma}[j], j \in \mathbb{M}_2\}, i \in [16:21]$.

After that, one can consider similarly constructed set of matrices with rows of the Hamming weight 6.

One can further reduce the size of the set $\mathbb{K}$. For each codeword $c \in \mathcal{C}$, we can compute $v = cF_5$ and obtain the position $t(v) = j$ of the last non-zero element in v. We can construct the disjoint sets of codewords $\mathcal{C}_j = \{c \in \mathcal{C} | \tau(cF_5) = j, d_H(c, \mathbf{0}) = 12\}$ and consider matrices M with rows $M[i], i \in [21:26]$, from different $\mathcal{C}_j$. This allows one to exclude the matrices M, which correspond to addition of row $F_{5,\beta}[i]$ to row $F_{5,\beta}[j]$ with $i > j$, which does not change the rate of polarization and scaling exponent.

Moreover one can pick only unique combinations of $M[i], i \in \mathbb{M}_2$. For each combination, one should obtain the transition matrix T and sort two block of τ_i: $i \in [16:21]$ and $i \in [21:26]$ to minimize the processing complexity.

The same procedure can also be done for rows $[6:16]$ independently from rows $[16:26]$.

IV. Numeric results

A. Monotonic partial distances

1) Construction of 16×16 *kernels:* We constructed the set of 16×16 matrices by procedure, described in section III-B2 and picked the set $\mathbb{K}_*$ of polarization kernels K with $E(K) = 0.51828$. For each $K \in \mathbb{K}_*$ we compute complexity $\Psi(K)$ and BEC scaling exponent $\mu(K)$ (section II-B2).

As a result, we minimized the processing complexity among kernels with the lowest scaling exponent and found a kernel K_1, which has $\mu(K) = 3.346$, maximal size of the decoding window $\mathcal{M}(K_1) = 4$ and processing complexity $\Psi(K_1) = 740$. This kernel is presented on Figure 1.

It turns out, that the complexity of window processing can be significantly reduced by employing modified window processing algorithm [6], [7]. For instance, the kernel K_1 was reported in [6] to have processing complexity of 472 arithmetic operations instead of estimated 740 operations.

2) Construction of 32×32 *kernels:* We found a kernel K_{32}, illustrated on Figure 1, by enhanced construction algorithm, proposed in section III-C. The K_{32} kernel has $E(K_{32}) = 0.53656$ and $\mu(K_{32}) = 3.127$. This BEC scaling exponent is minimal for 32×32 polarization kernels at present [10]. The K_{32} has the maximal size of the decoding window $\mathcal{M}(K_{32}) = 14$ and processing complexity $\Psi(K_{32}) = 472247$ arithmetical operations, which is unfortunately too high for practical use.

B. Permuted partial distances

As we have seen in the previous section, the kernels with MI-PDs can have too high processing complexity. To cope with it, we propose to apply row permutations over rows of K, which reduce decoding windows, while preserving PDP.

$K_1, E = 0.51828, \mu = 3.346$

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

$K_2, E = 0.51828, \mu = 3.45$

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

$K_{32}, E = 0.53656, \mu = 3.127$

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

Fig. 1: Constructed polarization kernels

1) Row permuted 16×16 *kernel:* We performed a heuristic minimization of $\mathcal{M}(K_1)$ by permutation of rows of K_1, while preserving the polarization rate $E = 0.51828$. The search resulted in the kernel K_2, illustrated in Figure 1, with $E(K_2) = 0.51828$, $\mu(K) = 3.45$ and $\mathcal{M}(K) = 3$.

The kernel K_2 is given by $P_\rho K_1$, where P_ρ is a permutation matrix, where $\rho = [0, 1, 2, 7, 3, 4, 5, 6, 9, 10, 11, 12, 8, 13, 14, 15]$. It was shown in [6] that the kernel K_2 can be processed with 183 operations instead of 293 operations of a straightforward implementation.

2) Row permuted 32×32 *kernel:* In contrast to the kernels of the size 16, we were not able to reduce $\mathcal{M}(K_{32})$ and preserve its polarization rate by row permutations. Then we decided to decrease the required rate of polarization to reduce maximal size of the decoding window.

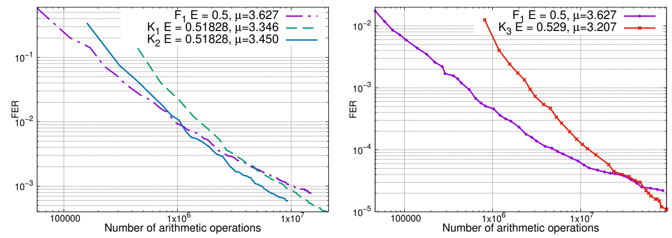
As a result, we obtained a kernel K_3 with $E(K_3) = 0.529248$ and $\mu(K_3) = 3.207$. The complexity of K_3 is given by $\Psi(K_3) = 13885$ operations and $\mathcal{M}(K_3) = 8$. Moreover, the kernel K_3 can be processed by reduced complexity window processing algorithm [7] with 6770 operations.

The kernel K_3 is given by $P_\rho K_{32}$, where P_ρ is a permutation matrix, $\rho(i) = \bar{\rho}(i), i \in [18]$, $\bar{\rho} = [0, 1, 2, 12, 3, 11, 4, 6, 15, 5, 7, 8, 14, 13, 16, 9, 17, 10]$ and $\rho(i) = i, i \in [18 : 32]$.

C. Performance of polar codes with the constructed kernels

We constructed $(4096, 2048)$ polar codes with kernels K_1 and K_2. We also constructed $(1024, 512)$ polar code with kernel K_3. The performance was investigated for the case of AWGN channel with BPSK modulation. The set of frozen symbols was obtained by the method, proposed in [16].

Figure 2a presents simulation results for $(4096, 2048)$ polar subcodes [17],[18] with different kernels under SCL with different list size at $E_b/N_0 = 1.25$ dB. The results are presented in terms of the actual decoding complexity. It can be seen that the polar subcode based on kernel K_2 can provide better performance with the same decoding complexity for FER $\leq 8 \cdot 10^{-3}$ compared to polar subcodes with F_1 kernel. 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 F_1 only at FER$= 2 \cdot 10^{-3}$.



(a) (4096, 2048) codes

(b) (1025, 512) codes

Fig. 2: SCL decoding of polar subcodes

The same simulations were done for $(1024, 512)$ codes with 32×32 kernel K_3 and SNR = 1.75 dB. The corresponding results are shown in figure 2b. Due to high processing complexity of K_3 only for FER less than $2.5 \cdot 10^{-5}$ the overall SCL decoding complexity of the polar code with K_3 has become lower compared to the polar code with Arikan kernel.

References

[1] A. Fazeli, S. H. Hassani, M. Mondelli, and A. Vardy, "Binary linear codes with optimal scaling: Polar codes with large kernels," in *Proceedings of IEEE Information Theory Workshop*, 2018.

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

[3] 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.

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

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

[6] G. Trofimiuk and P. Trifonov, "Efficient decoding of polar codes with some 16×16 kernels," in *Proceedings of IEEE ITW*, 2018.

[7] G. Trofimiuk and P. Trifonov, "Reduced complexity window processing of binary polarization kernels," in *Proceedings of IEEE ISIT*, 2019.

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

[9] S. H. Hassani, K. Alishahi, and R. Urbanke, "Finite-length scaling for polar codes," *IEEE Trans. On Inf. Theory*, vol. 60, no. 10, October 2014.

[10] H. Yao, A. Fazeli, and A. Vardy, "Explicit polar codes with small scaling exponent," in *Proceedings of IEEE ISIT*, Paris, France, July 2019.

[11] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes with arbitrary binary kernel," in *Proceedings of IEEE Information Theory Workshop*. Hobart, Australia: IEEE, 2014, pp. 377–381.

[12] ——, "Sequential decoding of polar codes," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1127–1130, 2014.

[13] P. Trifonov, "A score function for sequential decoding of polar codes," in *Proceedings of IEEE ISIT*, Vail, USA, 2018.

[14] ——, "Binary successive cancellation decoding of polar codes with Reed-Solomon kernel," in *Proceedings of IEEE ISIT*. Honolulu, USA: IEEE, 2014, pp. 2972 – 2976.

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

[16] P. Trifonov, "On construction of polar subcodes with large kernels," in *Proceedings of IEEE ISIT*, 2019.

[17] P. Trifonov and G. Trofimiuk, "A randomized construction of polar subcodes," in *Proceedings of IEEE ISIT*, 2017.

[18] P. Trifonov, "Design of randomized polar subcodes with non-arikan kernels," in *Proceedings of 16-th International Workshop on Algebraic and Combinatorial Coding Theory*, 2018.