Reduced complexity window processing of binary polarization kernels

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

***Abstract*—We propose a reduced complexity algorithm for computing log-likelihood ratios (LLRs) needed for successive cancellation (SC) decoding of polar codes with $2^t \times 2^t$ polarization kernels. This algorithm is applied to some polarization kernels of length 16 and 32 with high polarization rate. The complexity reduction is achieved by exploiting linear relationship of the considered kernels and Arikan matrix. Further complexity reduction is achieved by identification of common subexpressions. The proposed approach enables SC list decoding of polar codes with some large kernels with lower complexity compared to the codes based on the Arikan kernel with the same performance.**

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 application of the window processing algorithm [11] to some 16×16 polarization kernels was studied in [6], where some kernel specific optimizations of processing algorithm were also provided. In this paper we present the reduced complexity extension of window processing algorithm, which is suitable for processing of any $2^t \times 2^t$ polarization kernel. We applied the proposed method to decode polar codes with 32×32 kernel with polarization rate 0.529. 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 LLRs for the input symbols of the considered kernels are obtained from the LLRs computed via the Arikan recursive expressions. Further complexity reduction is achieved by identification and reusing of some common subexpressions, which arise in 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 $\{0, 1, \dots n-1\}$. A *polarization kernel* K is a binary invertible $l \times l$ matrix, which is not upper-triangular under any column permutation. The Arikan matrix is given by $F_m = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{\otimes m}$.

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 $I(W)$ [2]. Selecting $\mathcal{F}$ as the set of indices of low-capacity subchannels enables almost error-free communication. It is convenient to define probabilities

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

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

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

where $\theta_K[u_0^{(s+1)l-1}, j]_r = (u_{lr}^{l(r+1)-1} G_n)_j, r \in [s+1]$. Computing these probabilities reduces to soft-output decoding of non-systematically encoded codes, generated by rows $t+1, \dots, l-1$ of K. This problem was considered in [7].

At the receiver side, the SC decoding algorithm makes estimates

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

B. Computing kernel input symbols LLRs

1) General case: Let us consider the *kernel processing* problem, i.e. the task of computing (2). Let us assume for the sake of simplicity that $m = 1$.

We propose to introduce approximate probabilities

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

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 approximation was introduced in [8], [9], and shown to provide substantial reduction of the complexity of sequential decoding of polar codes.

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

$$\mathbf{S}_{1,i} = \ln \frac{\widetilde{\mathbf{W}}_1^{(i)}(u_0^{i-1}.0|y_0^{l-1})}{\widetilde{\mathbf{W}}_1^{(i)}(u_0^{i-1}.1|y_0^{l-1})} = R(0) - R(1), \qquad (5)$$

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 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 K, assuming that all $u_j, i < j < l$, are equiprobable.

2) Window processing: Straightforward evaluation of (5) for arbitrary kernel has complexity $O(2^l)$. However, we have a simple explicit recursive procedure for computing these values for the case of the Arikan matrix F_t.

Let $l = 2^t$. Consider encoding scheme

$$c_0^{l-1} = v_0^{l-1} F_t. \qquad (6)$$

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

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

It can be seen that

$$S_\lambda^{(2i)}(v_0^{2i-1}, y_0^{N-1}) = Q(a,b) = \operatorname{sgn}(a)\operatorname{sgn}(b)\min(|a|,|b|) \qquad (7)$$

$$S_\lambda^{(2i+1)}(v_0^{2i}, y_0^{N-1}) = P(a,b,v_{2i}) = (-1)^{v_{2i}}a + b, \qquad (8)$$

$$a = S_{\lambda-1}^{(i)}(v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}, y_{0,e}^{N-1}), \qquad (9)$$

$$b = S_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}, y_{0,o}^{N-1}). \qquad (10)$$

where $N = 2^\lambda$. Then the log-likelihood of a path v_0^i can be obtained as [10]

$$\begin{aligned} R(v_0^i|y_0^{l-1}) &= \log \widetilde{W}_t^{(i)}(v_0^i|y_0^{l-1}) \\ &= R(v_0^{i-1}|y_0^{l-1}) + \tau\left(S_t^{(i)}(v_0^{i-1}, y_0^{l-1}), v_i\right), \end{aligned} \qquad (11)$$

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 [11] and [6] to express values $\widetilde{\mathbf{W}}_1^{(i)}(u_0^i|y_0^{l-1})$ via $\widetilde{W}_t^{(j)}(v_0^j|y_0^{l-1})$ for some j. Indeed, $TK = F_t$, where T is an $l \times l$ matrix. Let $c_0^{l-1} = v_0^{l-1}F_t = u_0^{l-1}K$, so that $u_0^{l-1} = v_0^{l-1}T$.

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].

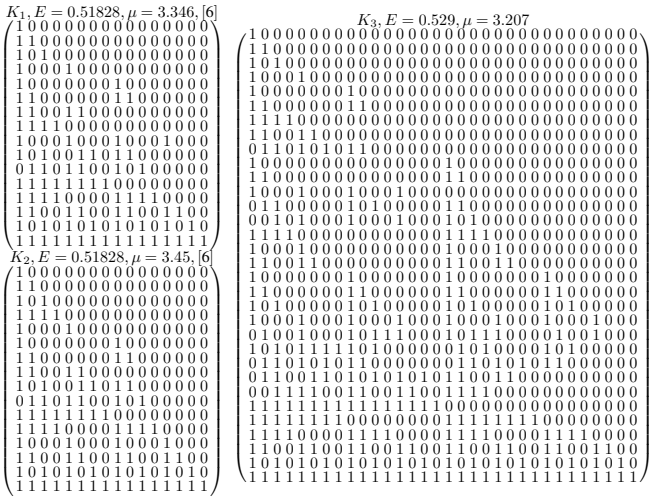


Fig. 1: Large polarization kernels with high polarization rate

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

$$u_i = \sum_{j=0}^{l-1} v_j T_{j,i} = \sum_{\substack{j=0 \\ T_{j,i}=1}}^{\tau_i} v_j. \qquad (12)$$

Let $h_i = \max_{i' \in [i+1]} \tau_{i'}$. It can be seen that[1]

$$\widetilde{\mathbf{W}}_1^{(j)}(u_0^j|y_0^{l-1}) = \max_{v_0^{h_j} \in \mathcal{Z}_j} \widetilde{W}_t^{(h_j)}(v_0^{h_j}|y_0^{l-1}), \qquad (13)$$

where $\mathcal{Z}_j$ is the set of vectors $v_0^{h_j}$, such that (12) 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

$$\mathbf{S}_{1,i} = \max_{v_0^{h_j} \in \mathcal{Z}_{i,0}} R(v_0^{h_i}|y_0^{l-1}) - \max_{v_0^{h_j} \in \mathcal{Z}_{i,1}} R(v_0^{h_i}|y_0^{l-1}). \qquad (14)$$

Observe that computing these values requires considering multiple vectors $v_0^{h_i}$ of input symbols of F_t.

Let $\mathcal{D}_i = [h_i + 1] \backslash \{\tau_0, \tau_1, \ldots, \tau_i\}$ be a *decoding window*, i.e. the set of indices of independent (from u_0^{i-1}) components of $v_0^{h_i}$. The number of such vectors, which determines the decoding complexity, is $2^{|\mathcal{D}_i|+1}$. In general, one has $|\mathcal{D}_i| = O(l)$ for an arbitrary kernel. The calculation of LLRs $\mathbf{S}_{1,i}$ via (14) will be referred to as the *window processing* algorithm.

III. Identification of common subexpressions

A. Kernels

Figure 1 presents some kernels, their polarization rate E and BEC scaling exponent μ [12], [3]. All kernels were derived by a generalization of the method presented in [3].

Table I presents the right-hand side of expression (12) for each $i \in [32]$ for kernel K_3 as well as the sizes of corresponding decoding windows.

[1]The method given in [5] is a special case of this approach.

TABLE I: Input symbols u_ϕ for kernel K_3 as functions of input symbols v for F_5

ϕ	u_ϕ	$\lvert\mathcal{D}_\phi\rvert$	comp-lexity	comp-lexity, with CSE	ϕ	u_ϕ	$\lvert\mathcal{D}_\phi\rvert$	comp-lexity	comp-lexity, with CSE
0	v_0	0	31	31	16	$v_{14} \oplus v_{21}$	5	97	97
1	v_1	0	1	1	17	v_{24}	7	2497	1057
2	v_2	0	3	3	18	$v_{11} \oplus v_{22} \oplus v_{25}$	7	385	385
3	v_4	1	17	17	19	$v_7 \oplus v_{26}$	7	641	641
4	v_8	4	313	137	20	$v_{11} \oplus v_{28}$	8	2815	2047
5	$v_6 \oplus v_9$	4	63	63	21	v_{11}	7	1	1
6	v_3	3	1	1	22	v_7	6	1	1
7	v_5	2	1	1	23	v_{22}	5	1	1
8	v_6	1	1	1	24	v_{14}	4	1	1
9	v_{16}	7	4673	961	25	v_{13}	3	1	1
10	$v_{10} \oplus v_{17}$	7	385	385	26	v_{15}	2	1	1
11	$v_{12} \oplus v_{18}$	7	767	575	27	v_{23}	1	1	1
12	v_{10}	6	1	1	28	v_{27}	0	1	1
13	v_{18}	5	1	1	29	v_{29}	0	31	31
14	$v_7 \oplus v_{11} \oplus v_{13} \oplus v_{19}$	5	129	129	30	v_{30}	0	3	3
15	v_{20}	5	289	193	31	v_{31}	0	3	1

B. Motivation

Our goal is to obtain ϕ-th input symbol LLR of the $2^t \times 2^t$ polarization kernel K. In the previous section we showed how to solve this task by window processing algorithm (14), which computes several log-likelihoods $R(v_0^{h_\phi}|y_0^{l-1})$ of paths $v_0^{h_\phi}$, considered in the context of Arikan SC decoding. The number of such paths is determined by the size of the ϕ-th decoding window $\mathcal{D}_\phi$. The values ϕ and h_ϕ will be referred to as *external phase* and *internal phase*, respectively.

According to (11), computing of a path score $R(v_0^{h_\phi}|y_0^{l-1})$ requires the values $S_t^{(h_\phi)} = S_t^{(h_\phi)}(v_0^{h_\phi-1}, y_0^{l-1})$ and $R(v_0^{h_\phi-1}|y_0^{l-1})$. Since $h_\phi - 1$ may be greater than $h_{\phi-1}$, one needs to compute several input symbol LLRs $S_t^{(h)}$, which correspond to the path $v_0^{h_\phi}$, where $h_{\phi-1} < h \le h_\phi$. Since the window processing algorithm considers several paths, it is convenient to define the vector $\mathbb{S}_\phi^{(h)}$ of $S_t^{(h)}$ indexed by $v_0^{h-1} \in \mathcal{Z}_\phi$. It can be observed, that the length of this vector is given by $2^{|\mathcal{D}_i|-(h_i-h)}$.

Example 1. *Consider computing the LLR $\mathbf{S}_{1,4}$ of K_3. The decoding window $\mathcal{D}_4$ is given by $\{3,5,6,7\}$, $h_4 = 8$ and $h_3 = 4$. To compute a path score $R(v_0^8|y_0^{31})$, one needs to sequentially obtain LLRs $S_5^{(h)}, h_3 < h \le h_4$, corresponding to path v_0^8. In other words,*

$$R(v_0^8|y_0^{31}) = R(v_0^4|y_0^{31}) + \sum_{h=5}^{8} \tau(S_5^{(h)}, v_h).$$

The number of path scores, which are needed to compute $\mathbf{S}_{1,4}$, is given by $2^{|\mathcal{D}_4|+1} = 32$.

The LLR $S_t^{(h)}(v_0^{h-1}|y_0^{l-1})$ of a single path v_0^{h-1} is obtained recursively as it shown in (7) and (8). More precisely, to obtain $S_t^{(h)}$ one needs to calculate $2^{t-\lambda}$ intermediate LLRs $S_\lambda^{(j)} = S_\lambda^{(j)}(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$, $N = 2^\lambda$, on each layer λ, $0 \le \lambda < t$, of the recursion. The value $\bar{v}_0^{j-1}$ is a partial sum of some elements from $v_0^{h_\phi-1}$. The structure of these sums is determined by the recursion (9) and (10) as well as a particular subvector $\bar{y}_0^{N-1}$ of y_0^{l-1}.

GETCSEPAIRS(h, ϕ)

$\widehat{Z}_\phi^{(h)} = \left\{v_0^{h-1} | v_i \in \{0,1\}, i \in \mathcal{D}_\phi, u_j = 0, j \in [\phi]\right\}$
$y_0^{l-1} = (y_0, y_1, \ldots, y_{l-1})$ in symbolic form
$X_t \leftarrow \left\{(v_0^{h-1}, y_0^{l-1}) | v_0^{h-1} \in \widehat{Z}_\phi^{(h)}\right\}$
$I \leftarrow h$
for $\lambda \leftarrow t$ **downto** 2
do $X_{\lambda-1} \leftarrow \emptyset, N \leftarrow 2^\lambda, I' \leftarrow I - (I \mod 2)$
  for each $(\bar{v}_0^{I-1}, \bar{y}_0^{N-1})$**in** X_λ
  do $X_{\lambda-1} \leftarrow X_{\lambda-1} \cup \left\{(v_{0,e}^{I'-1} \oplus v_{0,o}^{I'-1}, y_{0,e}^{N-1})\right\}$
    $X_{\lambda-1} \leftarrow X_{\lambda-1} \cup \left\{(v_{0,o}^{I'-1}, y_{0,o}^{N-1})\right\}$
  $I \leftarrow \lfloor I/2 \rfloor$
return $\{X_\lambda | \lambda \in [t]\}$

Fig. 2: Computing of CSE for given phases

In a straightforward implementation, all intermediate LLRs $S_\lambda^{(j)}(\bar{v}_0^j, \bar{y}_0^{N-1})$ should be separately computed for each $S_t^{(h)} \in \mathbb{S}_\phi^{(h)}$. It turns out, that $S_\lambda^{(j)}$ induced by recursive calculation of $S_t^{(h)}$ for different v_0^{h-1} may be the same. For instance, observe that the number of distinct vectors $\bar{v}_0^{j-1}$ and length-N subvectors $\bar{y}_0^{N-1}$ of y_0^{l-1} arising in (7)–(8) is at most 2^j and $2^{t-\lambda}$, respectively. Thus, the number of distinct pairs $(\bar{v}_0^j, \bar{y}_0^{N-1}) \le 2^{j+t-\lambda}$, which can be less than $|\mathbb{S}_\phi^{(h)}|$.

Therefore, we propose to compute $S_\lambda^{(j)}(\bar{v}_0^j, \bar{y}_0^{N-1})$ only for distinct pairs $(\bar{v}_0^j, y_0^{N-1})$ which arise at each layer of the recursion (7)–(10) for computing the vector $\mathbb{S}_\phi^{(h)}$ of LLRs $S_t^{(h)}$. We denote by X_λ the set of these distinct pairs $(\bar{v}_0^j, \bar{y}_0^{N-1})$ and call it as a set of *common subexpressions* (CSE).

C. The CSE identification algorithm

The sets X_λ of CSE can be identified offline for a given polarization kernel K of length $l = 2^t$, external phase ϕ and internal phase h, $h_{\phi-1} < h < h_{\phi+1}$. To do that, one should perform the algorithm shown in figure 2. This algorithm recursively enumerate pairs $(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$ according to recursion given in (7) and (8) under assumption that $u_i = 0, i \in [l]$.

Example 2. *Consider the identification of CSE for kernel K_3, $\phi = 4$ and $h = 8$. The LLR $S_5^{(8)}(v_0^7, y_0^{31}) = Q(a, b)$, where $a = S_4^{(4)}(v_{0,e}^7 \oplus v_{0,o}^7, y_{0,e}^{31})$, $b = S_4^{(4)}(v_{0,o}^7, y_{0,o}^{31})$ (7). For LLR a we have $\bar{v}_0^3 = v_{0,e}^7 \oplus v_{0,o}^7 = (v_0 \oplus v_1, v_2 \oplus v_3, v_4 \oplus v_5, v_6 \oplus v_7)$ and $\bar{y}_0^{15} = (y_0, y_2, \ldots, y_{30})$. On the other hand for LLR b we have $\bar{v}_0^3 = (v_1, v_3, v_5, v_7)$ and $\bar{y}_0^{15} = (y_1, y_3, \ldots, y_{31})$.*

In both cases the upper bound for the number of distinct vectors $\bar{v}_0^3$ is $2^4 = 16$. However, the symbols $u_i, i \in [4]$, are already estimated by the SC decoder of K_3, therefore, according to (12), the symbols v_0, v_1, v_2, v_4 are also known. Without loss of generality, we assume $u_i = 0, i \in [4]$. Then, $\bar{v}_0^3$ is given by $(0, v_3, v_5, v_6 \oplus v_7)$ or $(0, v_3, v_5, v_7)$ depending on the branch of the recursion. In both cases $\bar{v}_0^3$ takes only 8 different values.

Then, we can compose the set X_4 of CSE $(\bar{v}_0^3, \bar{y}_0^{15})$ in $S_4^{(4)}(\bar{v}_0^3, \bar{y}_0^{15})$ as $X_{4,0} \cup X_{4,1}$, where

COMPUTELLRS(h, ϕ)

1 $S_0^{(0)}(\epsilon, y_i) = y_i$
2 **for** $\lambda \leftarrow 1$ **to** $t-1$
3 **do** $I = \lfloor h/2^{t-\lambda} \rfloor, I' = I - I \mod 2$
4 **for each** $(\bar{v}_0^{I-1}, \bar{y}_0^{N-1})$ **in** X_λ
5 **do** add $\hat{u}_i, i \in [\phi]$ to $\bar{v}_0^{I-1}$
6 $a = S_{\lambda-1}^{(I'/2)}(v_{0,e}^{I'-1} \oplus v_{0,o}^{I'-1}, y_{0,e}^{N-1})$
7 $b = S_{\lambda-1}^{(I'/2)}(v_{0,o}^{2i-1}, y_{0,o}^{N-1})$
8 **if** I is even
9 **then** $S(\bar{v}_0^I, \bar{y}_0^{N-1}) = \mathrm{sgn}(a)\,\mathrm{sgn}(b)\min(|a|, |b|)$
10 **else** $S(\bar{v}_0^I, \bar{y}_0^{N-1}) = (-1)^{\bar{v}_I} a + b$
11 **return** $\mathbb{S}_\phi^{(h)}$

Fig. 3: Computing LLRs using CSE for given phases

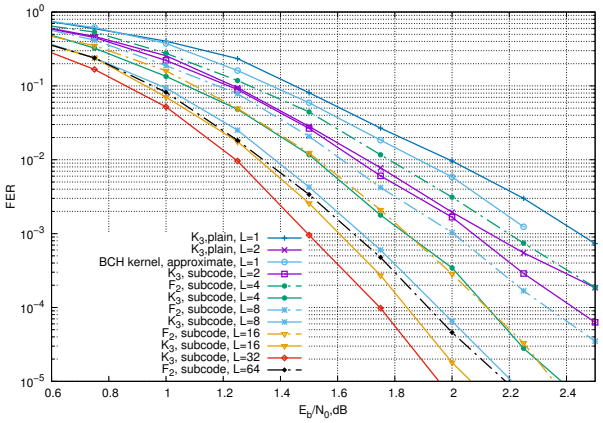


Fig. 4: Performance of (1024,512) polar codes

$X_{4,0} = \{((0, v_3, v_5, v_6 \oplus v_7), y_{0,e}^{31}) | v_i \in \{0,1\}, i \in \mathcal{D}_4\}$, $X_{4,1} = \{((0, v_3, v_5, v_7), y_{0,o}^{31}) | v_i \in \{0,1\}, i \in \mathcal{D}_4\}$.

Note that $|X_4| = 16$, which implies that only 16 distinct intermediate LLRs $S_4^{(4)}(\bar{v}_0^3, \bar{y}_0^{15})$ can appear during calculation of each LLR $S_5^{(8)}(v_0^7, y_0^{31})$ from $\mathbb{S}_4^{(8)}$. In a straightforward implementation 32 operations is required to calculate two $S_4^{(4)}(\bar{v}_0^3, \bar{y}_0^{15})$ for each $S_5^{(8)}(v_0^7, y_0^{31})$.

To obtain X_3, we can denote $\bar{v}_0^3$ as $(0, \bar{v}_1, \bar{v}_2, \bar{v}_3)$ and for each $(\bar{v}_0^3, \bar{y}_0^{15})$ compute pairs $((\bar{v}_1, \bar{v}_2 \oplus \bar{v}_3), \bar{y}_{0,e}^{15})$ and $((\bar{v}_1, \bar{v}_3), \bar{y}_{0,o}^{15})$, which corresponds to lines 7-9 of $GetCSEPairs$ algorithm. Since $(\bar{v}_1, \bar{v}_3)$ has 4 possible values, $|X_3| = 16$. In the same manner we can obtain X_2, $|X_2| = 16$. In CSE set X_1 we have pairs $(\epsilon, \bar{y}_0^1)$, where ϵ is an empty sequence, thus, $|X_1| = 16$.

The condition $u_j = 0, j \in [\phi]$ is chosen for simplicity, but in actual processing $u_j \in \{0,1\}$. This should be taken into account during the window processing. Recall that $TK = F_t$ and τ_i is the position of the last non-zero symbol in the i-th column of T. Observe that the expression (12) can be rewritten as $v_{\tau_i} = u_i \oplus \sum_{j=0}^{\tau_i - 1} v_j T_{j,i}$, thus, if some $u_i = 1, i \in [\phi]$ then it should be added to each entry of v_{τ_i} in partial sums $\bar{v}_0^j$.

Example 3. *The set X_2 from the Example 2 can be represented as $X_2 = \{(c_i, (y_i, y_{i+8}, y_{i+16}, y_{i+24})) | i \in [8]\}$, where $c_0^7 = v_0^7 F_3$. Recall that $v_i = u_i, i \in [3]$, $v_4 = u_3$. For offline identification of CSE via $GetCSEPairs$ $c = (0,0,0,v_3,0,v_5,v_6,v_7)F_3$, but in actual processing $c = (v_0,v_1,v_2,v_3,v_4,v_5,v_6,v_7)F_3$. Since the values $u_i, i \in [4]$ are constants, the size of sets $|X_\lambda|, 1 \le \lambda \le 5$ does not change.*

D. LLR computation

Finally, to obtain $\mathbb{S}_\phi^{(h)}$ on the external phase ϕ of the kernel processing, one should perform $ComputeLLRs$ algorithm, presented in Figure 3. This algorithm calculates intermediate LLRs for pairs, presented in CSE, where symbolic y_0^{l-1} is replaced by corresponding actual LLRs. The already estimated symbols $\hat{u}_i, i \in [\phi]$ are also used.

Observe, that sets X_λ for phases h and $h+1$ can be the same for some λ, therefore, one does not need to recompute corresponding intermediate LLRs $S_\lambda^{(i)}$. For instance, on external phase $\phi = 5$ of kernel K_3 we have $u_5 = v_6 \oplus v_9$ and $\mathcal{D}_5 = \mathcal{D}_4$. The value of $v_8 = u_4$ is used in P function (8) and does not appear on layers 1-4 of the calculations, which means that X_λ, $\lambda \in [5]/\{0\}$ have not changed and can be used to calculate $\mathbb{S}_5^{(9)}$.

As a result, the arithmetic complexity (number of summation and comparison operations) of computing $\mathbb{S}_\phi^{(h)}$ is given by $\sum_{\lambda=\lambda'}^{t} |X_\lambda|$, λ' is a layer, on which CSE are changed compared to the previous internal phase $h-1$, while in a straightforward implementation of the window processing algorithm one needs $|\mathcal{D}_\phi| \cdot \sum_{\lambda=\lambda'}^{t} 2^{t-\lambda}$ operations. It can be seen that $\sum_{\lambda=\lambda'}^{t} |X_\lambda| \le |\mathcal{D}_\phi| \cdot \sum_{\lambda=\lambda'}^{t} 2^{t-\lambda}$ by definition of the CSE.

The above described techniques can be also used to implement SCL decoder for polar codes with the considered kernels, using a straightforward generalization of the algorithm and data structures presented in [13].

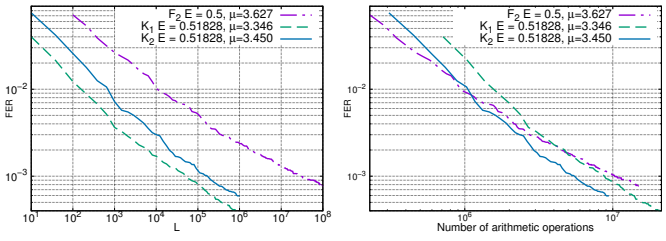
Other optimizations of kernel processing are described in [6]. Note that the algorithm of identification and reusing of the values of CSE can be applied for window processing of any binary polarization kernel of size 2^t to reduce overall arithmetic complexity.

IV. NUMERIC RESULTS

Table I presents kernel processing arithmetic complexity of the proposed algorithm for kernel K_3. We report the complexity of straightforward window processing algorithm and implementation with reusing of the values of CSE. It can be seen, that reusing of values of CSE allows one to reduce processing complexity almost by a factor of two: from 13154 to 6770 operations (summations and comparisons).

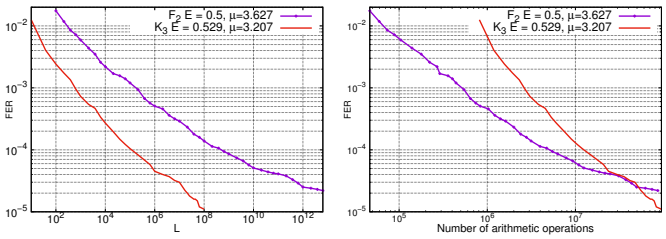
We constructed $(1024, 512)$ polar (sub)codes[14] with the kernels K_3 and F_1, and investigated their performance for the case of AWGN channel with BPSK modulation. The sets of frozen symbols were obtained by Monte-Carlo simulations.

Figure 4 illustrates the performance of polar (sub)codes with Arikan kernel F_1 and polar (sub)codes with kernel K_3 [15]. It can be seen that the codes based on kernel K_3 with



(a) Performance (b) Decoding complexity

Fig. 5: SCL decoding of (4096, 2048) polar subcodes



(a) Performance (b) Decoding complexity

Fig. 6: SCL decoding of (1024, 512) polar subcodes

improved polarization rate $E(K_3) = 0.529$ provide significant performance gain compared to polar codes with Arikan kernel. Moreover, polar subcode with kernels K_3 under SCL with $L = 8$ has almost the same performance as polar subcodes with Arikan kernel under SCL decoder with $L = 64$.

We also compared the performance of plain polar codes with kernel K_3 with performance of polar codes with 32×32 BCH kernel B_{32} and approximate processing method, presented in [8]. The simulation results are shown in Figure 4. We constructed plain $(1024, 512)$ polar codes with considered kernels. Due to higher polarization rate $E(B_{32}) = 0.537$, polar code with B_{32} kernel outperforms polar code with K_3 under SC algorithm. However, the decoding complexity of the approximate processing algorithm for B_{32} is $2.5 \cdot 10^6$ operations, while the proposed exact algorithm requires $4.3 \cdot 10^5$ operations for $L = 1$ and $8 \cdot 10^5$ operations for $L = 2$. In latter case it outperforms B_{32} with $L = 1$.

We implemented the proposed algorithm of identification and reusing of the values of CSE in the case of 16×16 kernels K_1 and K_2 proposed in [6]. Figure 5a presents simulation results for $(4096, 2048)$ polar subcodes with different kernels under SCL with different L at $E_b/N_0 = 1.25$ dB. 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 $O(L)$. Figure 5b shows the same results in terms of the number of arithmetical operations. 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 lower list size, required for achieving the same performance, which eventually enables one to compensate relatively high complexity of the LLR computation algorithm presented in Section III.

The same simulations were done for $(1024, 512)$ codes with 32×32 kernel K_3 and SNR = 1.75 dB. The corresponding results illustrated on Figure 6. Due to relatively high processing complexity of K_3, the overall SCL decoding complexity of the polar code with K_3 becomes lower compared to the polar code with F_1 kernel only for FER less than $2.5 \cdot 10^{-5}$.

That is, plots 5b and 6b show that polar codes with kernels with higher polarization rate requires substantially lower list size to achieve the same performance compared to F_1 kernel.

V. Conclusions

In this paper a reduced complexity decoding algorithm for polar codes with $2^t \times 2^t$ polarization kernels was proposed and applied to kernels of length 32 and 16. The algorithm computes kernel input symbols LLRs via the ones for the Arikan kernel, and exploit the structure of recursive calculations 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.

The complexity of decoding of polar codes with proposed 32×32 polarization kernel K_3 with high polarization rate is substantially lower compared to existing exact and approximate algorithms.

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 ITW*, 2018.

[2] 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.

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

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

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

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

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

[12] 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.

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

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

[15] 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.