Sequential Decoding of Polar Codes with Arbitrary Binary Kernel

Vera Miloslavskaya, Peter Trifonov
Saint-Petersburg State Polytechnic University
Email: {veram,petert}@dcn.icc.spbstu.ru

***Abstract*—The problem of efficient soft-decision decoding of polar codes with any binary kernel is considered. The proposed approach represents a generalization of the sequential decoding algorithm introduced recently for the case of polar codes with Arikan kernel. Numeric results show that the proposed algorithm enables near-ML decoding of polar codes with BCH kernel.**

I. Introduction

Polar codes were shown to be able to achieve the capacity of a wide class of communication channels [1]. Namely, it was shown that m-fold application of a 2×2 linear transformation given by matrix $A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ (Arikan kernel) transforms the original memoryless binary input output-symmetric channel into a number of bit channels, and their capacities converge with m to 0 and 1. However, the rate of convergence (rate of polarization) appears to be quite low. It was shown in [2] that high-dimensional kernels (e.g. based on BCH codes) provide higher polarization rate than the Arikan kernel. That is, the decoding error probability of such polar codes decreases much faster with code length compared to similar Arikan codes. However, there are still no efficient methods for their decoding.

The successive cancellation (SC) algorithm is the classical decoding method for polar codes. However, it is not able to recover from errors which may occur at early phases of the decoding, and fails therefore to achieve the ML performance. This problem was addressed in [3], where a list decoding algorithm for Arikan polar codes was introduced. It was shown in [4], [5] that the same performance can be achieved with much smaller complexity by employing stack decoding algorithm. It was shown in [6] that further complexity reduction can be obtained at the expense of negligible performance degradation.

In this paper we present a generalization of the sequential decoding algorithm suggested in [6] to the case of polar codes with an arbitrary binary kernel. The paper is organized as follows. Background on polar codes and their sequential decoding is provided in Section II. The proposed decoding method is described in Section III. Some improvements for it are derived in Section IV, and implementation issues are discussed in Section V. Numeric results are provided in Section VI.

II. Background

A. Polar codes

An $(n = l^m, k)$ polar code based on $l \times l$ kernel B is a linear block code generated by k rows of matrix $G_n = M^{(n)} B^{\otimes m}$, where $B^{\otimes m}$ denotes m-times Kronecker product of matrix with itself and $M^{(n)}$ is permutation matrix, such that $M^{(n)}_{t,t'} = 1$ for any $t = \sum_{i=0}^{m-1} t_i l^i$ and $t' = \sum_{i=0}^{m-1} t_{m-1-i} l^i$, $t_i \in \{0, \ldots, l-1\}$.

Let $c_0^{n-1} = (c_0, \ldots, c_{n-1})$. Any codeword of a polar code can be represented as $c_0^{n-1} = u_0^{n-1} G_n$, where u_0^{n-1} is the input sequence consisting of k information symbols and $n-k$ frozen symbols, which are equal to zero. Let $\mathcal{F}$ be the set of $n-k$ indices of frozen symbols.

B. Successive cancellation decoding

The decoding problem for polar codes consists in finding

$$\widehat{u}_0^{n-1} = \arg \max_{u_0^{n-1} : u_{0,\mathcal{F}}^{n-1} = \mathbf{0}} P(u_0^{n-1} | y_0^{n-1}),$$

where $u_{0,\mathcal{F}}^{n-1}$ denotes subvector of u_0^{n-1} consisting of elements with index $j \in \mathcal{F}$. At the i-th phase the SC decoder computes estimates

$$\widehat{u}_i = \begin{cases} \arg\max_{u_i} P(\widehat{u}_0^{i-1}, u_i | y_0^{n-1}), & t \notin \mathcal{F} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Given some path u_0^i, its probability can be recursively computed as

$$P(u_0^{ls+t} | y_0^{n-1}) = \sum_{u_{ls+t+1}^{l(s+1)-1}} \prod_{j=0}^{l-1} \pi_j, \quad (2)$$

where $0 \le t < l$, $\pi_j = P\left(\theta_B\left[u_0^{(s+1)l-1}, j\right] | y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}\right)$ $\theta_B\left[u_0^{l(s+1)-1}, j\right]_r = \left(u_{lr}^{l(r+1)-1} G_n\right)_j$, $0 \le r \le s$, $P(u_r|y_r) = \frac{P(y_r|u_r)P(u_r)}{P(y_r)}$, and $P(y_r|u_r)$ is the channel transition probability function.

The SC algorithm for binary polar codes can be implemented with complexity $O(n \log_l(n))$ unit calculations, where a unit calculation corresponds to single evaluation of (2).

C. Sequential decoding of polar codes with Arikan kernel

A major drawback of the SC algorithm is that it cannot correct errors which may occur at early phases of the decoding process. This problem is solved in stack/list algorithms by keeping a list of the most probable paths within code tree [3], [4], [5]. A path of length i is identified by values $u_0^{i-1} \in \{0,1\}^i$. Each path is associated with its score, which depends on its probability. Stack algorithms [4], [5] keep the paths in a stack (priority queue). At each iteration the decoder selects

for extension path u_0^{i-1} with the largest score, and performs the i-th phase of SC decoding. That is, if $i \in \mathcal{F}$ the path is extended to obtain u_0^i, where $u_i = 0$, and the extended path is stored in the stack together with its score. Otherwise, the path is cloned to obtain new paths $(u_0, \ldots, u_{i-1}, 0)$ and $(u_0, \ldots, u_{i-1}, 1)$, which are stored in the stack together with their scores. In order to keep the size of the stack limited, paths with low scores can be purged from the stack, so that the total number of paths considered simultaneously by the decoder does not exceed some parameter Θ. Furthermore, if the decoder returns to phase i more than L times, all paths shorter than $i+1$ are eliminated. Decoding terminates as soon as path of length n appears at the top of the stack, or the stack becomes empty. Hence, the worst case complexity of stack decoding is given by $O(Ln\log_l(n))$ unit calculations. Average decoding complexity depends on how path scores are defined.

In [6] a low-complexity version of the stack algorithm for binary polar codes with 2×2 Arikan kernel was introduced. Let $v[j]_0^{n-1}$, $0 \leq j < 2^{n-i-1}$, be different paths such that $v[j]_0^i = u_0^i$ and $v[j]_{i+1}^{n-1} \in \{0,1\}^{n-i-1}$. Let J be a random variable, which is equal to $j \in \left\{0, \ldots, 2^{n-i-1}-1\right\}$ if the most probable codeword of the polar code corresponds to path $v[j]_0^{n-1}$. The solution of the optimization problem $Z = \max\limits_{u_0^{n-1}:u_{0,\mathcal{F}}^{n-1}=\mathbf{0}} P_{U_0^{n-1}|Y_0^{n-1}}(u_0^{n-1}|y_0^{n-1})$ is equal to $P_{U_0^{n-1}|Y_0^{n-1}}(v[j]_0^{n-1}|y_0^{n-1})$ with probability $P\{J=j\}$, i.e. it is a function of random variable J. Here U_h is a random variable corresponding to the h-th input symbol of polarizing transformation and Y_h is a random variable corresponding to the h-th received symbol. Observe that $J = j$ implies that $v[j]_h = 0, h \in \mathcal{F}$. Hence, one can estimate Z as $E_J[P_{U_0^{n-1}|Y_0^{n-1}}(v[J]_0^{n-1}|y_0^{n-1})]$. It can be seen that increasing i reduces the number of possible values of random variable J, improving thus the accuracy of such estimate.

Let us assume without loss of generality that $v[0]_0^{n-1}$ is the most probable path, i.e.

$$\arg \max_{0 \leq j < 2^{n-i-1}} P_{U_0^{n-1}|Y_0^{n-1}}(v[j]_0^{n-1}|y_0^{n-1}) = 0.$$

Event $J = 0$ is equivalent to $v[0]_h = 0, h \in \mathcal{F}$. Therefore, one obtains

$$\begin{aligned} &E_J[P_{U_0^{n-1}|Y_0^{n-1}}(v[J]_0^{n-1}|y_0^{n-1})] = \\ &\sum_{j=0}^{2^{n-i-1}-1} P_{U_0^{n-1}|Y_0^{n-1}}(v[j]_0^{n-1}|y_0^{n-1}) P\{J=j\} \geq \\ &\underbrace{P_{U_0^{n-1}|Y_0^{n-1}}(v[0]_0^{n-1}|y_0^{n-1})}_{R(u_0^i, y_0^{n-1})} P\{J=0\}. \end{aligned} \tag{3}$$

It is possible to show that

$$R(u_0^i, y_0^{n-1}) = \max_{u_{i+1}^{n-1}} P_{U_0^{n-1}|Y_0^{n-1}}(u_0^{n-1}|y_0^{n-1})$$

can be computed exactly as [6]

$$\begin{aligned} &R(u_0^{2s+t}, y_0^{n-1}) = \\ &\max_{u_{2s+t+1}^{2(s+1)-1}} \prod_{j=0}^{1} R\left(\theta_A\left[u_0^{2(s+1)-1}, j\right], y_{j\frac{n}{2}}^{(j+1)\frac{n}{2}-1}\right), \end{aligned} \tag{4}$$

where $t \in \{0,1\}$, and initial values for these recursive expressions are given by $R(b, y_j) = P(b|y_j)$, $b \in \{0,1\}$.

It is difficult to compute the second term in (3) exactly for any given y_0^{n-1} at phase i of the SC decoder. However, it can be averaged over all possible received sequences. Average probability of the most probable path u_0^{n-1} having prefix u_0^i, having zeroes in positions $j \in \mathcal{F}$, is lower bounded by probability of the SC decoder, which starts from u_0^i and does not take into account any freezing constraints, makes decisions $u_j = 0, j > i, j \in \mathcal{F}$, i.e. does not make errors in these positions. Probability of this event is given by

$$\hat{\Omega}(i) = \prod_{j \in \mathcal{F}, j > i} (1 - P_j), \tag{5}$$

where P_j is the j-th subchannel error probability, provided that exact values of all previous bits $u_{j'}$, $j' < j$, are available. It depends only on n, $\mathcal{F}$ (i.e. the code being considered), channel properties and phase i. For any given channel, probabilities P_j can be pre-computed using density evolution.

Thus, the score for path u_0^i can be defined as

$$\hat{T}(u_0^i, y_0^{n-1}) = R(u_0^i, y_0^{n-1})\hat{\Omega}(i). \tag{6}$$

This approach enables one to compare paths u_0^i with different lengths, and prevent the decoder from switching frequently between different paths.

III. Proposed approach

A. Decoding algorithm

We propose to generalize decoding algorithm [6] to the case of polar codes based on arbitrary binary kernel. The idea of this algorithm does not use any kernel properties. Similarly to the case of Arikan kernel, in the case of $l \times l$ kernel B one obtains

$$\begin{aligned} &R(u_0^{ls+t}, y_0^{n-1}) = \\ &\max_{u_{ls+t+1}^{l(s+1)-1}} \prod_{j=0}^{l-1} R\left(\theta_B\left[u_0^{l(s+1)-1}, j\right], y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}\right), \end{aligned} \tag{7}$$

where $0 \leq t < l$.

This enables one to perform decoding of a polar code with arbitrary binary kernel in exactly the same way as in the case of Arikan kernel, i.e. keep paths u_0^i in a stack, and select at each iteration for extension a path with the highest value of $\hat{T}(u_0^i, y_0^{n-1})$, until a path of length n is obtained.

However, the task of computing (7) is not as simple as in (4). Moreover, efficient techniques for computing $\hat{\Omega}(i)$ should be provided.

B. Computing path score

1) Function $R(u_0^i, y_0^{n-1})$: Let $i = ls+t, 0 \le t < l$. Finding $u_{ls+t+1}^{l(s+1)-1}$ maximizing (7) is equivalent to decoding in a coset of $(l, \kappa = l-t-1)$ code C_t, which is generated by last κ rows of matrix B. The coset is given by $u_{ls}^{ls+t}B_{0..t}+C_t$, where $B_{0..t}$ is the submatrix of B consisting of first $t+1$ rows. Indeed, task (7) can be considered as finding

$$p[u_{ls+t}]_t = \max_{u_{ls+t+1}^{l(s+1)-1}} P(u_{ls}^{l(s+1)-1}|z_0^{l-1}), \tag{8}$$

where transition probabilities $P(b|z_j) = \Delta_j R\left(\left(\theta_B\left[u_0^{ls-1}, j\right], b\right), y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}\right)$, $b \in \{0,1\}$, $0 \le j < l$, and Δ_j are some scaling factors. This enables one to employ any soft decision decoding algorithm for finding $u_{ls+t+1}^{l(s+1)-1}$ maximizing (8).

Hence, the total complexity of the sequential decoding algorithm is $O(Ln\log_l(n))$ unit operations, where unit operations correspond to searching for the most probable codeword in codes generated by rows of $l \times l$ kernel B.

2) Function $\hat{\Omega}(i)$: Finding error probability $P_j, j \in \mathcal{F}$, for a bit subchannel induced by non-Arikan polarizing transformation, provided that exact values of u_0^{j-1} are available to the decoder, still remains an open problem. Therefore, we have to use simulations to compute these values. Possible alternatives include employing union bound together with multilevel code weight enumerator computation techniques introduced in [7], and Gaussian approximation method based on Gram-Charlier series expansion of the expression for symbol LLR [8]. The first approach appears to work poorly on bad bit subchannels, which are included into the set of frozen symbols. The second approach may result in divergent series.

IV. Improved decoding method

It can be seen that estimation of input symbols u_i, $sl \le i < (s+1)l$, is performed using probabilities $R\left(\left(\theta_B\left[u_0^{ls-1}, j\right], b\right), y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}\right), b \in \{0,1\}$, and the corresponding subset of frozen symbols is $\mathcal{F} \cap \{sl, \ldots, sl+l-1\}$. The former probabilities are independent from $t = i \bmod l$. Therefore, symbols $u_{sl}^{(s+1)l-1}$ can be estimated jointly. This enables one to significantly reduce the decoding complexity.

Assume that at some iteration β of the decoding algorithm described in Section III-A path u_0^{sl-1} is selected for extension as the most probable one. Let us define the set of paths of length $(s+1)l$ which can be obtained from path u_0^{sl-1}

$$V = \left\{u_0^{(s+1)l-1} | u_{sl}^{(s+1)l-1} \in \{0,1\}^l, u_{sl,\mathcal{F}}^{(s+1)l-1} = \mathbf{0}\right\}.$$

Recall, that the sequential decoding algorithm involves stack purging operation, which eliminates paths with lowest scores, so that the total number of paths does not exceed Θ, and list pruning operation, which ensures that at most L paths can be extended till phase i for any i. Let us assume for the sake of simplicity that all paths given by V are inserted into the stack, and then stack purging and list pruning operations are applied. Let $\Phi \subset V$ be the set of paths which survive these operations.

It can be seen from (5)–(6) that scores $\hat{T}(u_0^{(s+1)l-1}, y_0^{n-1})$ have common factor $\hat{\Omega}((s+1)l-1)$, i.e. Φ consists of $\phi = |\Phi|$ paths $u_0^{(s+1)l-1}$ with largest probabilities $R(u_0^{(s+1)l-1}, y_0^{n-1})$. This implies that these paths correspond to ϕ most probable codewords $u_{sl}^{(s+1)l-1}B$ of (l, λ_s) code Υ_s generated by rows of B with indices from Ξ_s, where $\Xi_s = \{sl, sl+1, \ldots, (s+1)l-1\} \backslash \mathcal{F}$ is the set of non-frozen symbols in the s-th block of input symbols, and $\lambda_s = |\Xi_s|$. These codewords can be identified using any list soft decision decoding algorithm for binary linear code Υ_s.

Let $\hat{t}_j$ be the j-th largest path score stored in the stack at iteration β, $0 \le j < \Theta$. If there are $\tilde{\Theta} < \Theta$ active paths in the stack, assume that $\hat{t}_j = 0, \tilde{\Theta} \le j < \Theta$. Let Q_i be the set of paths of length i stored in the stack. Let $\tilde{t}_j$ be the scores of paths in $Q_{(s+1)l}$ arranged in the descending order. Let $\tilde{t}_j = 0, \tilde{L} \le j < L$, where $\tilde{L} = |Q_{(s+1)l}|$. Φ includes all paths from V which can survive stack purging and list pruning operations, i.e.

$$u_0^{(s+1)l-1} \in V : \ \hat{T}(u_0^{(s+1)l-1}, y_0^{n-1}) \ge \max(\hat{t}_{\Theta-\phi}, \tilde{t}_{L-\phi}). \tag{9}$$

Observe that if $L - \widetilde{L} \ge 2^\lambda$ and $\Theta - \widetilde{\Theta} \ge 2^{\lambda_s}$, then exhaustive search over codewords $w_0^{l-1} \in \Upsilon_s$ is performed and for each w_0^{l-1} corresponding path is pushed into the stack together with its score. It appears that set Φ given by (9) is highly redundant. More efficient algorithm can be obtained by replacing Φ with

$$\hat{\Phi} = \left\{u_0^{(s+1)l-1} \in \Phi | \hat{T}(u_0^{(s+1)l-1}, y_0^{n-1}) \ge e^{-\alpha}\rho\right\}, \tag{10}$$

where $\rho = \widehat{T}\left((u_0^{sl}, w_0^{l-1}), y_0^{n-1}\right)$, and $w_0^{l-1}B$ is the most probable codeword of Υ_s. Here α is a parameter which affects the size of the obtained list. It must be recognized that for sufficiently high α this approach may require decoding of Υ_s beyond its Johnson bound [9]. This may result in list size, which is exponential in l. In order to obtain a practical algorithm, it may be necessary to further restrict $\hat{\Phi}$ to contain L_0 vectors with the largest score for some L_0.

The improved decoding algorithm operates as follows. At each iteration it selects for extension path u_0^{sl-1} with the largest score. Then list decoding of code Υ_s is performed, and paths given by (10) are pushed into the stack together with their scores. The input data for the list decoder are still given by $R\left(\left(\theta_B\left[u_0^{ls-1}, j\right], b\right), y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}\right)$, $b \in \{0,1\}$. These values are given by (7), and are computed recursively by employing a soft-input hard-output decoder for codes generated by rows of B. Path score is defined as

$$\hat{T}(u_0^{(s+1)l-1}, y_0^{n-1}) = R(u_0^{(s+1)l-1}, y_0^{n-1})\tilde{\Omega}(s+1),$$

where

$$\hat{\Omega}(s) = \prod_{\sigma=s+1}^{n/l-1}(1-\Gamma_\sigma),$$

and Γ_σ is the probability of non-zero values appearing in positions $j \in \{\sigma l, \ldots, (\sigma+1)l-1\} \cap \mathcal{F}$ of the most probable path with prefix $u_0^{\sigma l-1}$. Stack purging and list pruning

operations are performed in the same way as described above. Probabilities Γ_σ can be obtained from simulations. Decoding terminates as soon as path of length n appears at the top of the stack.

V. Implementation

Both encoding and decoding operations for a polar code of length $n = l^m$ can be decomposed into m layers, where each layer corresponds to application of the linear transformation given by B to l^{m-1} data units obtained at the previous layer. In the case of decoding, layer $m-1$ (final layer) corresponds to recovery of input symbols u_i, where list decoding is needed, as described in Section IV. At layers $0, \ldots, m-2$ (intermediate layers) one needs to identify just single most probable codeword given by (8) for each data unit.

A. Intermediate layers

Any soft decision decoding algorithm for coset $u_{ls}^{ls+t} B_{0..t} + C_t$ can be used to find a solution of (8). Let us consider for the case of concreteness the case of Box and Match algorithm [10]. For a code of dimension κ, $(\kappa+\psi, \gamma)$ Box and Match algorithm performs search for up to 2γ errors in positions from set $\Lambda^{(t)} \cup \Psi^{(t)}$, where $\Lambda^{(t)}$ is the most reliable information set and $\Psi^{(t)}$ is the set of ψ most reliable positions except positions in $\Lambda^{(t)}$. Parameter ψ specifies a trade-off between time and space complexity.

Let $\omega[b]_j = P(b + u_{ls}^{ls+t} B_{0..t,j} | z_j)$ be the input probabilities. The input LLR vector χ_0^{l-1} is defined as

$$\chi_j = (-1)^{u_{ls}^{ls+t} B_{0..t,j}} \log \frac{\omega[1]_j}{\omega[0]_j},$$

where $B_{0..t,j}$ is the vector consisting of first $t+1$ elements of the j-th column of matrix B. In order to identify the most probable codeword $v_{t+1}^{l-1} B_{t+1..l-1} \in C_t$ one needs to construct sets $\Lambda^{(t)}$ and $\Psi^{(t)}$. The first step of the Box and Match algorithm consists in sorting of the reliability vector $(|\chi_0|, \ldots, |\chi_{l-1}|)$. It can be seen that the permutation obtained as a result of this step does not depend on t, i.e. it can be performed only once for any given s. Observe that $\Lambda^{(0)} \supset \Lambda^{(1)} \supset \cdots \supset \Lambda^{(l-2)}$. In order to reduce the complexity, we propose to construct information sets $\Lambda^{(t)}$ jointly. Assuming that C_{l-2} is a repetition code, one obtains that $\Lambda^{(l-2)} = \{j_{l-2}\} : j_{l-2} = \arg\max_j |\chi_j|$. Given $\Lambda^{(t)}$, one can construct $\Lambda^{(t-1)} = \Lambda^{(t)} \cup \{j_{t-1}\}$, where j_{t-1} is such that $B_{t..l-1,j_{t-1}}$ is linearly independent of $B_{t..l-1,i}, i \in \Lambda^{(t)}$, and $|\chi_{j_{t-1}}|$ is as high as possible. $\Psi^{(t)}$ can be obtained similarly. Given the output $v_{t+1}^{l-1} B_{t+1..l-1}$of the decoding algorithm, one should compute $p[v_t]_t = \prod_{j=0}^{l-1} \omega[(v_0^{l-1} B)_j]_j$. Even if $v_{t+1}^{l-1} B_{t+1..l-1}$ fails to be the ML solution of the decoding problem due to suboptimality of the decoder being used, this does not necessarily result in a failure of the proposed polar code decoding algorithm, since $p[v_t]_t$ is typically close to its true value.

Recall that for any t one needs to compute both $p[0]_t$ and $p[1]_t$. This requires finding most probable codewords $c[0]_0^{l-1}$ and $c[1]_0^{l-1}$ in cosets $(v_0^{t-1}, 0) B_{0..t} + C_t$ and $(v_0^{t-1}, 1) B_{0..t} + C_t$, respectively. Observe that the most probable codeword c_0^{l-1} in coset $v_0^{t-1} B_{0..t-1} + C_{t-1}$ has already been computed at step $t-1$. Obviously, $c[0]_0^{l-1}$ or $c[1]_0^{l-1}$ is equal to c_0^{l-1}, so at the t-th step it remains to invoke decoder for C_t only once.

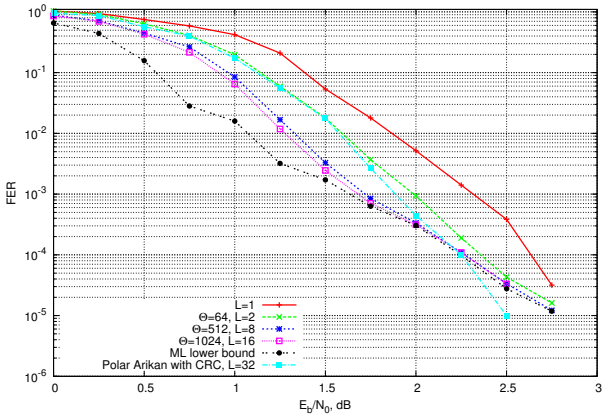

Figure 1. Performance of $(1024, 512)$ polar codes

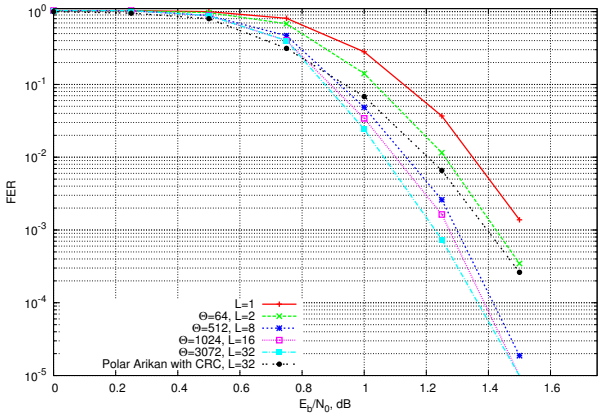

Figure 2. Performance of $(4096, 2048)$ polar codes

It can be seen that the complexity of computing $R(u_0^{sl+t}, y_0^{n-1}), u_{ls+t} \in \{0,1\}, 0 \le t < l$, for intermediate layers of the decoder is given by $\sum_{t=0}^{l-2} N(l, l-t-1, \psi, \gamma) + O(l \log l) + O(l^3)$, where $N(l, \kappa, \psi, \gamma)$ is the complexity of the $(\kappa+\psi, \gamma)$ Box and Match algorithm for (l, κ) code, excluding sorting and information set construction steps.

B. Final layer

At the last layer one needs to construct a list of codewords given by (10). This can be implemented using either ordered statistics [11] or a list extension of the Box and Match algorithm [12].

Observe that one does not need to perform any calculations, including any processing at intermediate layers, while decoding fully frozen blocks of input symbols.

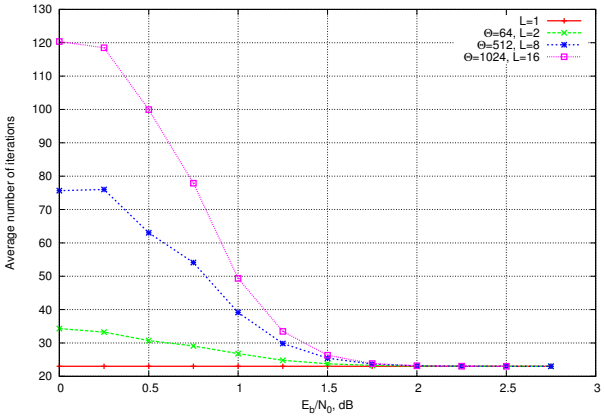

Figure 3. Average decoding complexity for $(1024, 512)$ polar code

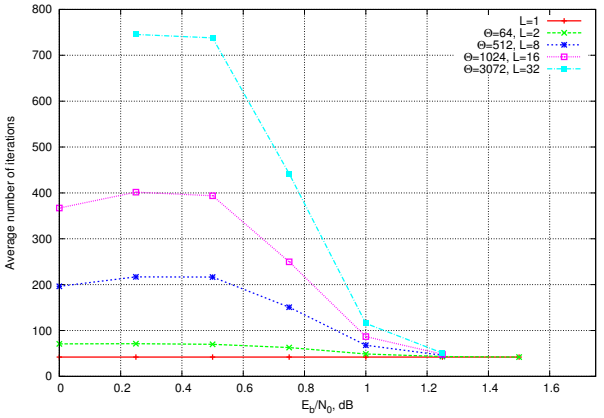

Figure 4. Average decoding complexity for $(4096, 2048)$ polar code

VI. Numeric results

Figure 1 illustrates the performance of $(1024, 512, 16)$ polar code[1] with BCH kernel [2] and $l = 32$ for the case of BPSK transmission over AWGN channel. For comparison, the results for Arikan polar code concatenated with CRC and list decoding (see [3]) is also shown. It can be seen that for sufficiently large L the proposed decoding algorithm enables one to achieve near-ML performance. At low SNR polar codes with BCH kernel outperform Arikan polar code concatenated with CRC. However, at high SNR the performance of the former one becomes limited by poor minimum distance. Figure 2 provides similar results for the case of $(4096, 2048, 32)$ polar code with BCH kernel and $l = 64$. Although minimum distance is still quite low, error floor does not appear down to error probability 10^{-5}. It is possible to improve the performance of polar code with BCH kernel by employing dynamic subchannel freezing techniques introduced in [5].

Figures 3–4 illustrate the average number of iterations performed by the proposed decoding algorithm for the considered codes. Only iterations corresponding to non-empty blocks of non-frozen symbols are counted. It can be seen that at sufficiently high SNR, where codeword error probability drops below 10^{-3}, the proposed algorithm requires in average t iterations, where $t \leq n/l$ is the number of non-empty blocks Ξ_s.

[1]This polar code was constructed using the method given in [13], and has codes $\Upsilon_s, 0 \leq s < n/l$, with dimensions 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 5, 7, 11, 13, 16, 21, 21, 24, 26, 26, 28, 29, 31, 31, 31, 31, 31, 32, 32, 32, 32.

VII. Conclusions

In this paper a novel decoding algorithm for binary polar codes with arbitrary kernel was proposed. The algorithm is a generalization of the sequential decoding algorithm introduced earlier for the case of polar codes with Arikan kernel. It involves near-ML soft-decision decoding of codes generated by rows of submatrices of the kernel. Simulation results show that the proposed approach enables one to perform near-ML decoding of polar codes with BCH kernel. These codes were shown to outperform Arikan polar codes concatenated in CRC, at least in the low-SNR region. The proposed algorithm can be used for decoding of polar codes with dynamic frozen symbols and non-Arikan kernels, which provide higher minimum distance, and can avoid therefore an error floor.

Acknowledgements

This work was supported by Russian Foundation for Basic Research under the grant 12-01-00365-a.

References

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

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

[4] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Communications Letters*, vol. 16, no. 10, October 2012.

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

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

[7] U. Wachsmann, R. F. H. Fischer, and J. B. Huber, "Multilevel codes: Theoretical concepts and practical design rules," *IEEE Transactions On Information Theory*, vol. 45, no. 5, pp. 1361–1391, July 1999.

[8] A. Abedi and A. Khandani, "An analytical method for approximate performance evaluation of binary linear block codes," *IEEE Transactions On Communications*, vol. 52, no. 2, February 2004.

[9] I. Dumer, G. Kabatiansky, and C. Tavernier, "Soft-decision list decoding of Reed-Muller codes with linear complexity," in *Proceedings of IEEE International Symposium on Information Theory*, 2011.

[10] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 5, May 2004.

[11] M. P. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, September 1995.

[12] P. A. Martin, D. Taylor, and M. P. Fossorier, "Soft-input soft-output list-based decoding algorithm," *IEEE Transactions On Communications*, vol. 52, no. 2, February 2004.

[13] V. Miloslavskaya and P. Trifonov, "Design of polar codes with arbitrary kernels," in *Proceedings of IEEE Information Theory Workshop*, 2012, pp. 119–123.