Binary Successive Cancellation Decoding of Polar Codes with Reed-Solomon Kernel

Peter Trifonov
Distributed Computing and Networking Department
Saint-Petersburg State Polytechnical University
Email: petert@dcn.icc.spbstu.ru

***Abstract*—Reduced complexity implementation of the successive cancellation decoding algorithm for polar codes with Reed-Solomon kernel is presented. The proposed approach is based on the representation of Reed-Solomon codes as Arikan polar codes over $\mathbb{F}_{2^m}$ with dynamic frozen symbols, and application of list successive cancellation decoding algorithm.**

I. Introduction

Polar codes were recently shown to be able to achieve the capacity of a wide class of communication channels [1]. However, the original Arikan kernel polarizes the channel quite slowly. This causes short polar codes to perform quite poorly. It was shown in [2], [3] that much higher rate of polarization can be achieved by employing Reed-Solomon kernel instead of Arikan one. However, implementation of a successive cancellation (SC) decoding algorithm for such codes involves soft-input soft-output decoding of Reed-Solomon codes, which is prohibitively complex even for shortest codes.

In this paper we present a reduced-complexity implementation of SC decoding algorithm for polar codes with Reed-Solomon kernel. The proposed approach is based on a representation of Reed-Solomon component codes as Arikan polar codes with dynamic frozen symbols, and their list successive cancellation decoding.

The paper is organized as follows. Section II introduces the necessary background. The proposed algorithm is derived in Section III. Numeric results are given in Section IV.

II. Background

A. Arikan polar codes

Consider a linear transformation $\mathbb{F}_2^{2^m} \rightarrow \mathbb{F}_2^{2^m}$ given by matrix $A = B_m^{(2)} G_2^{\otimes m}$, where $B_m^{(2)}$ is a bit-reversal permutation matrix, $G_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ and $G_2^{\otimes m}$ denotes m-times Kronecker product of matrix G_2 with itself. It was shown in [1] that such transformation allows one to split a memoryless binary-input output symmetric channel $W(y|x)$ into $n = 2^m$ subchannels $W(y_0^{n-1}, u_0^{i-1}|u_i)$ with capacities converging to 0 and 1. This enables one to construct polar code as a set of vectors $c = u_0^{n-1} A$, where $u_i = 0, i \in \mathcal{F}$, $\mathcal{F}$ is the set of low-capacity subchannels, which are "frozen", and the remaining u_i are given by the payload data symbols being encoded.

The successive cancellation decoding algorithm computes $W(y_0^{n-1}, u_0^{i-1}|u_i) = W_n^{(i)}(u_0^i|y_0^{n-1}) \frac{W(y_0^{n-1})}{P\{u_i\}}$, and makes decisions on non-frozen symbols $u_i, i \notin \mathcal{F}$. These decisions are used instead of true values of u_i at the subsequent steps of the algorithm for computing $W(y_0^{n-1}, u_0^{i-1}|u_i)$. This approach does not provide any way to correct erroneous estimates of u_i, causing thus the successive cancellation decoder to be highly suboptimal for moderate-length codes.

This problem was addressed in [4], where it was suggested to track at most L paths u_0^{i-1}. For each $i \notin \mathcal{F}$ each path splits into two paths $u_0^i, u_i \in \{0, 1\}$. If the number of obtained paths exceeds L, the paths with lowest values of $W_n^{(i)}(u_0^i|y_0^{n-1})$ are killed, where

$$W_n^{(2i)}(u_0^{2i}|y_0^{n-1}) = \sum_{u_{2i+1}=0}^{1} W_{\frac{n}{2}}^{(i)}(u_{0,e}^{2i+1} \oplus u_{0,o}^{2i+1}|y_0^{\frac{n}{2}-1}) \cdot W_{\frac{n}{2}}^{(i)}(u_{0,o}^{2i+1}|y_{\frac{n}{2}}^{n-1}) \quad (1)$$

$$W_n^{(2i+1)}(u_0^{2i+1}|y_0^{n-1}) = W_{\frac{n}{2}}^{(i)}(u_{0,e}^{2i+1} \oplus u_{0,o}^{2i+1}|y_0^{\frac{n}{2}-1}) \cdot W_{\frac{n}{2}}^{(i)}(u_{0,o}^{2i+1}|y_{\frac{n}{2}}^{n-1}), \quad (2)$$

$W_1^{(0)}(x|y) = W(x|y)$, $u_{0,e}^{2i+1}$ and $u_{0,o}^{2i+1}$ denote subvectors of u_0^{2i+1} with even and odd indices, respectively. List SC decoding with sufficiently large L enables one to achieve near-ML decoding performance.

It was suggested in [5] to represent an arbitrary binary linear block code of length n with parity check matrix H as a polar code with dynamic frozen symbols, so that $uAH^T = 0$, and

$$u_{j_i} = \sum_{s<j_i} V_{i,s} u_s, 0 \leq i < n - k, \quad (3)$$

where $V = QHA^T$ is a matrix, such that at most one row ends[1] in each column, Q is an invertible matrix, j_i is the column in which the i-th row ends, and $\mathcal{F} = \{j_0, \ldots, j_{n-k-1}\}$ is a set of dynamic frozen symbols, which depend on the code being considered. This enables one to employ (list) successive cancellation decoding algorithm for an arbitrary code. The set of dynamic frozen symbols, however, may include many good subchannels, while bad subchannels may remain unfrozen. Therefore quite large list size may be needed in order to achieve good performance.

Observe that this representation corresponds to the encoding scheme given by $c = fPB_m^{(2)} G_2^{\otimes m}$, where f is the message vector, and P is a $k \times n$ matrix, such that $PV^T = 0$.

[1]Row i ends in column j_i iff $V_{i,j_i} \neq 0$, and for all $t > j_i$ $V_{i,t} = 0$.

B. Reed-Solomon kernel

It was shown in [2] that if the input alphabet of a channel is $\mathbb{F}_q$, matrix G_q is not diagonal, and for each i there exists $j < i$, such that $(G_q)_{i,j}$ is a primitive element[2], then the capacities of the symbol subchannels induced by transformation $c = fB_m^{(q)}G_q^{\otimes m}$ converge to 0 or 1 q-ary symbols per channel use, where $B_m^{(q)}$ is a permutation matrix corresponding to reversal of the digits of q-ary integers. Matrix G_q can be constructed so that its submatrices are generator matrices of nested extended Reed-Solomon codes, e.g.

$$G_q = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 & \alpha_0^{q-1} \\ \alpha_{q-1}^{q-2} & \alpha_{q-2}^{q-2} & \dots & \alpha_2^{q-2} & \alpha_1^{q-2} & \alpha_0^{q-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \alpha_{q-1}^{1} & \alpha_{q-2}^{1} & \dots & \alpha_2^{1} & \alpha_1^{1} & \alpha_0^{1} \\ \alpha_{q-1}^{0} & \alpha_{q-2}^{0} & \dots & \alpha_2^{0} & \alpha_1^{0} & \alpha_0^{0} \end{pmatrix}, \quad (4)$$

where α_i are distinct elements of $\mathbb{F}_q$ with $\alpha_0 = 0$.

Similarly to the case of binary Arikan codes, one can obtain a polar code of length $n = q^m$ over $\mathbb{F}_q$ by freezing low-capacity subchannels. The successive cancellation decoding algorithm requires one in this case to compute

$$\tilde{W}_n^{(qj+i)}(f_0^{qj+i}|y_0^{n-1}) = \sum_{f_{qj+i+1}^{qj+q-1} \in \mathbb{F}_q^{q-i-1}} \prod_{s=0}^{q-1} \tilde{W}_{\frac{n}{q}}^{(j)}((f_{qt}^{qt+q-1}G_q)_s, 0 \le t \le j | y_{\frac{n}{q}s}^{\frac{n}{q}s+\frac{n}{q}-1})$$

for $f_i \in \mathbb{F}_q$, where $\tilde{W}_1^{(0)}(x|y) = W(x|y)$. This can be recognized as SISO decoding of a coset of the Reed-Solomon code generated by rows $i, \dots, q-1$ of kernel G_q. The complexity of straightforward evaluation of this expression, provided that $\tilde{W}_{\frac{n}{q}}^{(j)}(\cdot)$ values are already available, is given by $O(q^{q-i})$. The objective of this paper is to provide more efficient techniques for this problem in the case of $q = 2^m$.

This problem was considered in [6], where it was suggested to construct an extended trellis of the code, so that one can employ standard APP decoding algorithms for computing the required probabilities. However, this approach does not allow one to reuse intermediate results for different i.

III. Efficient decoding of polar codes with Reed-Solomon kernel

A. Successive cancellation decoding of Reed-Solomon codes

$(2^m, k, q-k+1)$ Reed-Solomon code over $\mathbb{F}_{2^m}$, is a set of vectors c obtained by evaluating $f(x) = \sum_{i=0}^{k-1} f_{2^m-1-i}x^i$, $f_i \in \mathbb{F}_{2^m}$, at distinct points $x \in \mathbb{F}_{2^m}$. Let $a_0, \dots, a_{m-1}$ be a basis of $\mathbb{F}_{2^m}$. Then $x = \sum_{s=0}^{m-1} x_s a_s, x_i \in \mathbb{F}_2$. Hence, one obtains

$$f(x) = f(x_0, \dots, x_{m-1}) = \sum_{i=0}^{k-1} f_{2^m-1-i} \sum_{t \in C_i} p_{i,t} \prod_{j=0}^{m-1} x_j^{t_j}, \quad (5)$$

[2]This requirement is in fact not necessary.

where $p_{i,t} \in \mathbb{F}_{2^m}$ are the coefficients of the expansion of $(\sum_{s=0}^{m-1} x_s a_s)^i$, and C_i is the set of multi-degrees $t \in \{0,1\}^m$ of non-zero terms in this expansion. This corresponds to the encoding scheme $c = f_0^{2^m-1}PA$, where P is a $q \times q$ matrix consisting of $p_{i,t}$ values, and $f_i = 0, i < 2^m - k$. Observe that matrix A can be seen as a table of values of various monomials $\prod_{j=0}^{m-1} x_j^{t_j}$, and matrix P is invertible. Expression (5) can be considered as a generalized Zhegalkin polynomial $f : \mathbb{F}_2^m \to \mathbb{F}_{2^m}$.

On the other hand, one can take check matrix of the Reed-Solomon code, and obtain its representation as an Arikan polar code with dynamic frozen symbols, where the dynamic freezing constraints are given by (3), and $PV^T = 0$. This enables one to employ the successive cancellation algorithm for decoding of Reed-Solomon codes. However, (1) should be replaced with

$$W_n^{(2i)}(u_0^{2i}|y_0^{n-1}) = \sum_{u_{2i+1} \in \mathbb{F}_{2^m}} W_{\frac{n}{2}}^{(i)}(u_{0,e}^{2i+1} \oplus u_{0,o}^{2i+1}|y_0^{\frac{n}{2}-1}) \cdot W_{\frac{n}{2}}^{(i)}(u_{0,o}^{2i+1}|y_{\frac{n}{2}}^{n-1}) \quad (6)$$

This expression can be recognized as m-dimensional 2-point cyclic convolution. Fast Hadamard transform can be used to evaluate it with complexity $O(m2^m)$ [7]. However, the performance of the classical successive cancellation decoding algorithm appears to be unsatisfactory in this case. It can be improved by employing list successive cancellation decoding algorithm presented in [4], subject to the following changes:

- At phases ϕ corresponding to non-frozen symbols u_ϕ, where $u_0^{2^m-1} = f_0^{2^m-1}P$, each path expands into q branches corresponding to different values of f_ϕ.
- Expression (3) is used for continuation of paths at phases corresponding to frozen symbols u_ϕ
- Path probabilities are computed according to (6) and (2).

Example 1. *Consider* $(4, 2, 3)$ *Reed-Solomon code over* $\mathbb{F}_{2^2}$. *Its codewords correspond to evaluation of polynomials* $f(x) = f_3 + f_2 x = f_3 + f_2(a_0x_0 + a_1x_1)$, *where* (a_0, a_1) *is a basis of* $\mathbb{F}_{2^2}$. *One obtains* $u_3 = f_3, u_1 = f_2a_0$ *(non-frozen symbols), and* $u_2 = \frac{a_1}{a_0}u_1, u_0 = 0$ *(frozen symbols).*

B. Soft-output successive cancellation decoding

Let us consider for the sake of simplicity the case of successive cancellation decoding of a polar code over $\mathbb{F}_q$ with $q \times q$ Reed-Solomon kernel $G_q = PA$ and $n = q = 2^m$, i.e. just a single layer of polarizing transformation. The successive cancellation algorithm computes probabilities $\tilde{W}_q^{(i)}(f_0^i|y_0^{q-1})$, and makes decision on f_i by setting it to the value corresponding to the highest probability. Observe that this essentially reduces to SISO decoding of a Reed-Solomon code. The successive cancellation decoding algorithm for the case of Reed-Solomon codes computes the probabilities of various paths u_0^j, such that $u_0^{q-1}A = f_0^{q-1}PA$, i.e. $f_0^{q-1} = u_0^{q-1}P^{-1}$. Observe that it is possible to reconstruct f_0^i from $u_0^{\tau_i}$, where τ_i is the position of the last non-zero symbol in the i-th row of P^{-1}. However, successive cancellation decoding of polar codes with RS kernel

requires one to compute $\tilde{W}_q^{(i)}(f_0^i|y_0^{q-1}), f_i \in \mathbb{F}_q$. Fixing the values f_0^{i-1} may impose constraints on $u_j, j > \tau_i$, which must be taken into account while computing these probabilities.

Indeed, vectors f_0^{q-1} and u_0^{q-1} satisfy the system of equations

$$\underbrace{\begin{pmatrix} S & I \end{pmatrix}}_{\Theta'} \begin{pmatrix} f_{q-1} & \dots & f_1 & f_0 & u_0 & u_1 & \dots & u_{q-1} \end{pmatrix}^T = 0,$$

where $q \times q$ matrix S is obtained by transposing P and reversing the order of columns in the obtained matrix. By applying elementary row operations, matrix Θ' can be transformed into minimum-span form Θ, such that the i-th row starts in the i-th column, and ends in column z_i, where all z_i are distinct, and $\Theta_{i,z_i} = 1$. This enables one to obtain dynamic freezing constraints in the form of

$$u_{j_i} = \sum_{s=0}^{i} f_s \Theta_{n-1-i,n-1-s} + \sum_{t=0}^{j_i-1} u_t \Theta_{n-1-i,n+t}, \quad (7)$$

where $j_i = z_{n-1-i} - n$.

Hence, one obtains

$$\tilde{W}_q(f_0^i|y_0^{q-1}) = \sum_{u_j \in \mathbb{F}_q, j \in T_i} W_q(u_0^{h_i}|y_0^{q-1}), \quad (8)$$

where summation is performed over vectors $u_0^{h_i}$ satisfying (7), i.e.

$$h_i = \max_{0 \le i' \le i} j_{i'},$$

and $T_i = \{0, \dots, h_i\} \setminus \{j_0, \dots, j_i\}$ is the set of non-frozen symbols of Arikan polarizing transformation. If this set is empty, (8) is assumed to contain a single summand. The values $W_q(u_0^{h_i}|y_0^{q-1})$ can be computed according to (6) and (2).

Observe that the proposed approach reduces to computing the probabilities of a number of paths $u_0^{h_i}$. For each i fixing a value of f_i activates one more constraint (7), causing thus some of these paths to be eliminated. Let $\sigma_\phi = i : j_i = \phi$ be the smallest i, such that a constraint on u_ϕ becomes active.

Example 2. *Consider 4×4 Reed-Solomon kernel over $\mathbb{F}_4$ given by*

$$G_4 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ \alpha & \alpha^2 & 1 & 0 \\ \alpha^2 & \alpha & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

It can be seen that

$$P = G_4 A^{-1} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & \alpha^2 & 0 \\ 0 & 1 & a & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Therefore,

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \alpha^2 & 0 & \alpha & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} f_3 \\ \vdots \\ f_0 \\ u_0 \\ \vdots \\ u_3 \end{pmatrix} = 0,$$

i.e. $h_0^3 = (0, 2, 2, 3)$, $\sigma_0^3 = (0, 2, 1, 3)$, and

$$\tilde{W}_4^{(0)}(f_0|y_0^3) = W_4^{(0)}(f_0|y_0^3),$$

$$\tilde{W}_4^{(1)}(f_0^1|y_0^3) = \sum_{u_1 \in \mathbb{F}_4} W_4^{(2)}(f_0, u_1, f_1 + \alpha^2 f_0 + \alpha u_1|y_0^3),$$

$$\tilde{W}_4^{(2)}(f_0^2|y_0^3) = W_4^{(2)}(f_0, f_2 + f_1 + f_0, f_0 + \alpha^2 f_1 + \alpha f_2|y_0^3),$$

$$\tilde{W}_4^{(3)}(f_0^3|y_0^3) = W_4^{(3)}(f_0, f_2 + f_1 + f_0, f_0 + \alpha^2 f_1 + \alpha f_2, f_3|y_0^3)$$

Computing $\tilde{W}_4^{(1)}(f_0^1|y_0^3)$ requires one to consider four paths corresponding to different values of u_1. As soon as f_1 becomes fixed, three of them (those with u_2 not satisfying (7)*) are eliminated.*

Observe that computing $\tilde{W}_4^{(0)}(f_0|y_0^4)$ using code trellis requires $(4-1)4^2 = 48$ multiply-add operations. The proposed approach involves computing Hadamard transforms of $W(c|y_i), 0 \le i < 4, c \in \mathbb{F}_4$, componentwise multiplication of the obtained 4-dimensional vectors, and computing inverse Hadamard transform. Hence, the total cost is $5 \cdot 8 = 40$ additions and 12 multiplications. Complexity savings for the remaining steps are even more significant.

In order to estimate the number L of paths $u_0^{h_i}$ needed to obtain exact values of $\tilde{W}_q^{(i)}(f_0^i|y_0^{q-1}), f_i \in \mathbb{F}_q$, observe that computing it requires one to consider all paths $u_0^{h_i}$ satisfying i constraints (7). Each path is assumed to be associated with q distinct values of u_{h_i}. Hence, the maximal number of such paths is given by

$$L_0 = \max_{0 \le i < q} q^{h_i - i} \quad (9)$$

C. Reducing list size

The number of paths L_0 needed to obtain exact values of $\tilde{W}_q^{(i)}(f_0^i|y_0^{q-1})$, in general, increases exponentially with kernel dimension q. In order to reduce the complexity, one may consider $L < L_0$ most probable paths. However, it may happen that paths $u_0^{h_i}$ corresponding to the required values of f_0^i are killed at some early stages of list successive cancellation decoding algorithm. This may cause significant inaccuracies in the obtained estimates of $\tilde{W}_q(f_0^i|y_0^{q-1})$. In order to avoid this, it is necessary to restart list successive cancellation decoding algorithm as soon as one obtains $j_i < h_i$, provided that some paths are killed earlier. This may increase the complexity by a factor of $\sigma < q$, where σ is the number of times the list successive cancellation decoder is restarted. Hence, the complexity of the proposed algorithm for the case of $q \times q$ Reed-Solomon kernel over $\mathbb{F}_q, q = 2^m$, is given by $O(\sigma L q^2 \log^2 q)$.

The details of the proposed approach are provided in algorithms 1 – 3. These algorithms keep global variables $\phi, activePath[l], P_{l,i}[\beta, x], C_{l,i}[\beta, j], 0 \le l < L, 0 \le i \le m, 0 \le \beta < 2^{m-i}, x \in \mathbb{F}_{2^m}, 0 \le j < 2, 0 \le \phi < n$, which have similar meaning to those used in [4]. K is a global variable which is equal to the last phase ϕ, where a path was killed due to memory overflow. One should set $\phi = 0, K = -1$ for every new instance of received noisy vector $(y_0, \dots, y_{n-1})$. For the sake of simplicity, global variables

Algorithm 1: GetSymbolDistribution

Data: $i, (y_0, \ldots, y_{n-1})$
Result: $\tilde{W}_n^{(i)}(f_0, \ldots, f_i | y_0, \ldots, y_{n-1}), f_i \in \mathbb{F}_{2^\mu}$

begin
  if $\phi = 0$ **then**
    $l \leftarrow$ `AssignInitialPath()`
    $P_{l,0}[\beta, u] \leftarrow W(u|y_\beta), u \in \mathbb{F}_{2^m}, 0 \leq b < n$
    $K = -1$
  while $\phi \leq h_i$ **do**
    `RecursivelyCalcP`(m, ϕ)
    if $\sigma_\phi < i$ **then**
      for $l \leftarrow 0$ **to** $L - 1$ **do**
        if $activePath[l]$ **then**
          $u_{l,\phi} \leftarrow \sum_{s=0}^{\sigma_\phi} \Theta_{n-1-\sigma_\phi, n-1-s} f_s + \sum_{t=0}^{\phi-1} \Theta_{n-1-\sigma_\phi, n+t} u_{l,t}$
          $P_{l,m}[0, x] \leftarrow 0, x \in \mathbb{F}_{2^m} \setminus \{u_{l,\phi}\}$
          $C_{l,m}[0, \phi \bmod 2] \leftarrow u_{l,\phi}$
    else
      if $\phi < h_i$ **then**
        $Q \leftarrow$ `Sort`$([(P_{l,m}[0, x], l, x), l < L, x \in \mathbb{F}_{2^m}])$
        $c_l \leftarrow |\{(p, l', u) \in Q_0^{L-1} | l' = l\}|, l < L$
        for $l \leftarrow 0$ **to** $L - 1$ **do**
          if $c_l = 0 \wedge activePath[l]$ **then**
            `KillPath`(l)
            $K \leftarrow \phi$
        for $j \leftarrow 0$ **to** $|Q_L| - 1$ **do**
          $l \leftarrow Q[j][1]$
          if $c_l > 1$ **then**
            $l' \leftarrow$ `ClonePath`(l)
            $C_{l',m}[0, \phi \bmod 2] \leftarrow u_{l',\phi}$
            $u_{l',\phi} \leftarrow Q[j][2]$
          $C_{l,m}[0, \phi \bmod 2] \leftarrow u_{l,\phi} \leftarrow Q[j][2]$
          $c_l \leftarrow c_l - 1$
      if $(\phi \equiv 1 \bmod 2) \wedge (\phi < h_i)$ **then**
        `RecursivelyUpdateC` (m, ϕ)
    $\phi \leftarrow \phi + 1$
  $\Pi \leftarrow (0, \ldots, 0)$
  for $l \leftarrow 0$ **to** $L - 1$ **do**
    if $activePath[l]$ **then**
      for $u_{l,\phi-1} \in \mathbb{F}_{2^m}$ **do**
        $f_i \leftarrow \frac{1}{\Theta_{n-1-i,n-1-i}} \left(\sum_{t=0}^{j_i} u_{l,t} \Theta_{n-1-i,n+t} + \sum_{s=0}^{i-1} f_s \Theta_{n-1-i,n-1-s} \right)$
        $\Pi[f_i] \leftarrow \Pi[f_i] + P_{l,m}[0, u_{\phi-1}]$
  return Π

$u_{l,\phi}$ are used in the description of the algorithms. However, their values can be recovered from $C_{l,m}[\beta, j]$. Algorithm 2 makes use of fast Hadamard transform. The description of the remaining auxiliary functions is presented in [4].

For each i one obtains $\tilde{W}_n^{(i)}(f_0, \ldots, f_i | y_0, \ldots, y_{n-1}), f_i \in \mathbb{F}_{2^\mu}$ by calling $GetSymbolDistribution(i, (y_0, \ldots, y_{n-1}).$

Algorithm 2: SetValue

Data: i, x
begin
  $f_i \leftarrow x$
  if $j_i \leq K$ **then**
    $\phi \leftarrow 0$
    return
  for $l \leftarrow 0$ **to** $L - 1$ **do**
    if *activePath[l]* **then**
      $\tilde{u}_{l,j_i} \leftarrow \sum_{s=0}^{i} \Theta_{n-1-i,n-1-s} f_s + \sum_{t=0}^{j_i-1} \Theta_{n-1-i,n+t} u_{l,t}$
      if $j_i < \phi - 1$ **then**
        if $\tilde{u}_{l,j_i} \neq u_{l,j_i}$ **then**
          `KillPath` (l)
      if $j_i = \phi - 1$ **then**
        $P_{l,m}[0, x] \leftarrow 0, x \in \mathbb{F}_{2^m} \setminus \{\tilde{u}_{l,j_i}\}$
        $C_{l,m}[0, i \bmod 2] \leftarrow u_{l,j_i} \leftarrow \tilde{u}_{l,j_i}$
        if $\phi - 1 \equiv 1 \bmod 2$ **then**
          `RecursivelyUpdateC` $(m, \phi - 1)$

Algorithm 3: RecursivelyCalcP

Data: l, λ, ϕ
begin
  if $\lambda = 0$ **then**
    return
  $\psi \leftarrow \lfloor \phi / 2 \rfloor$
  if $\phi \equiv 0 \bmod 2$ **then**
    `RecursivelyCalcP` $(\lambda - 1, \psi)$
  for $l \leftarrow 0$ **to** $L - 1$ **do**
    if $activePath[l]$ **then**
      for $\beta = 0$ **to** $2^{m-\lambda} - 1$ **do**
        if $\phi \equiv 0 \bmod 2$ **then**
          $p' \leftarrow \mathtt{FHT}(P_{l,\lambda-1}[2\beta, u], 0 \leq u < 2^m)$
          $p'' \leftarrow \mathtt{FHT}(P_{l,\lambda-1}[2\beta + 1, u], 0 \leq u < 2^m)$
          for $i \leftarrow 0$ **to** $2^m - 1$ **do**
            $p_i \leftarrow p'_i p''_i$
          $P_{l,\lambda}[\beta, \cdot] \leftarrow \frac{1}{2^\mu} \mathtt{FHT}(p)$
        else
          $u' \leftarrow C_{l,\lambda}[\beta, 0]$
          for $u'' \in \mathbb{F}_{2^m}$ **do**
            $P_{l,\lambda}[\beta, u''] \leftarrow P_{l,\lambda-1}[2\beta, u' \oplus u''] P_{l,\lambda-1}[2\beta + 1, u'']$

This call must be followed by a call to $SetValue(i, x)$, where x is the (presumably) correct value of f_i.

IV. Numeric results

Figure 1 presents the performance of polar codes with Reed-Solomon kernel over $\mathbb{F}_q$ based on $q \times q$ kernel for $q \in \{4, 8, 16\}$, as well as Arikan polar code, and Arikan polar code concatenated with CRC [4]. For the case of polar codes with Reed-Solomon kernel list decoding was employed only

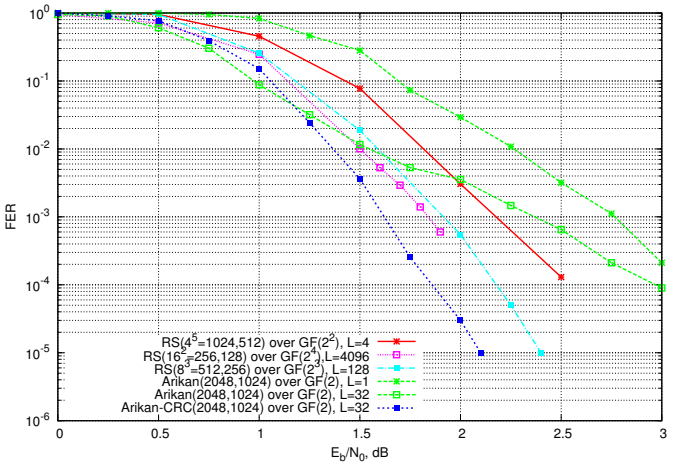

Fig. 1. Performance of polar codes

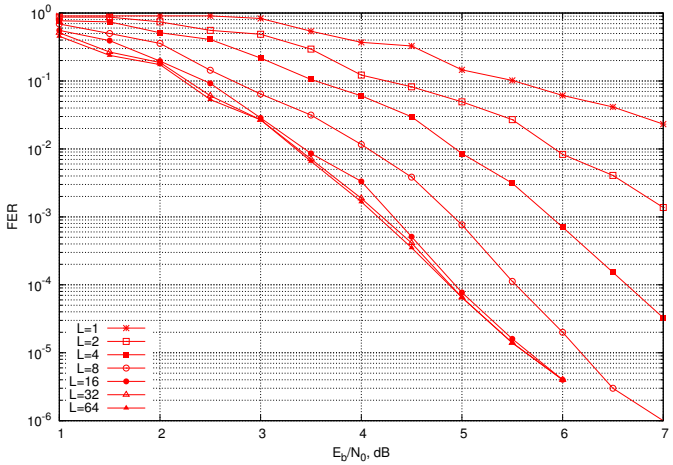

Fig. 2. Performance of list successive cancellation decoding of $(16, 11, 6)$ Reed-Solomon code

for computing probabilities $\tilde{W}_{s^s}^{(i)}(\cdot)$, as described in Section III-C, while classical successive cancellation decoding scheme was used for making the decisions on information symbols $f_i, 0 \leq i < q^m$. For the case of codes based on Arikan kernel list decoding was used to identify the most probable sequence $u_0^{2^m-1}$. Simulations were performed for the case of BPSK transmission of the binary image of the codes over AWGN channel.

It can be seen that codes with Reed-Solomon kernel provide significant performance gain compared to codes with Arikan kernel. However, this requires employing list decoding with very large list size in order to calculate the probabilities needed by the successive cancellation decoding algorithm, except for very small values of q. Furthermore, due to suboptimality of the successive cancallation decoding method, Arikan polar codes with outer CRC still outperform polar codes with Reed-Solomon kernel. This problem can be avoided by employing list decoding techniques not only for computing $\tilde{W}_{q^s}^{(i)}(\cdot)$, but also for finding the most probable values $f_0^{q^m-1}$.

Figure 2 presents the performance of list successive cancellation decoding algorithm for the case of $(16, 11, 6)$ Reed-Solomon code over $\mathbb{F}_{2^4}$, represented as an Arikan polar code with dynamic frozen symbols. It can be seen that near-optimal performance is achieved already with list size $L = 16$. This implies that one can substantially reduce the decoding complexity for polar codes with Reed-Solomon kernel by employing "hard-output" list successive cancellation decoding of codes at the outer layer with small L. Soft-output successive cancellation decoding, which requires large list size, needs to be employed only at the internal layers of the Reed-Solomon polarizing transformation.

V. Conclusions

In this paper an efficient algorithm for computing the probabilities needed by the successive cancellation decoding method for polar codes with Reed-Solomon kernel was proposed. Although the complexity of the proposed algorithm is still exponential in kernel dimension, it remains practical for $q = 4$, and the rate of polarization provided by such kernel $E(G_4) = 0.573$ already exceeds that of rather large binary kernels [8].

The approach proposed in this paper, which is based on the representation of codes generated by kernel submatrices as Arikan polar codes with dynamic frozen symbols, is a general one, and can be extended to the case of other kernel types. However, it becomes particularly efficient in the case of Reed-Solomon kernel, since codes generated by submatrices of such kernel are Reed-Solomon ones, and their codewords correspond to generalized Zhegalkin polynomials of sufficiently low degree.

Acknowledgement

This work was supported by Russian Foundation for Basic Research under 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] R. Mori and T. Tanaka, "Channel polarization on q-ary discrete memoryless channels by arbitrary kernels," in *Proceedings of IEEE International Symposium on Information Theory*, 2010.

[3] ——, "Non-binary polar codes using Reed-Solomon codes and algebraic geometry codes," in *Proceedings of IEEE Information Theory Workshop*, 2010.

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

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

[7] D. Declercq and M. P. Fossorier, "Decoding algorithms for nonbinary LDPC codes over $GF(q)$," *IEEE Transactions On Communications*, vol. 55, no. 4, April 2007.

[8] 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.