Polar subcodes for encoding and blind decoding of variable-sized data blocks

Kirill Ivanov*, Peter Trifonov†
*EPFL, Switzerland; e-mail: kirill.ivanov@epfl.ch
†Saint Petersburg Polytechnic University, Russia; e-mail: petert@dcn.icc.spbstu.ru

***Abstract*—A scheme for encoding variable-sized data blocks and their low-complexity blind decoding, i.e. decoding without the information about an actual size of the payload, is presented. This scheme relies on polar subcodes and includes the code design rule as well as the idea of the decoding algorithm. A code construction is suggested, which modifies the previously constructed family of polar subcodes. As an example of low-complexity blind decoding algorithm, a generalization of the sequential decoder is presented, which is able to jointly detect the payload size and correct errors in the received noisy sequence. Simulation results are provided and demonstrate the significant complexity reduction in comparison to the case of multiple decoding attempts and the performance close to the independent decoding of each code.**

I. Introduction

Long term evolution (LTE) networks are focused on achieving high data rates, low latency and high mobility. In order to achieve this, a control channel is used, which adapts to radio network conditions by changing the error-correcting code rate and modulation scheme being used. To avoid the redundant signaling, a blind decoding procedure is used on the receiver side. Namely, user equipment (UE) performs the decoding attempts in a pre-defined set of possible code parameters (*search space*) until CRC is satisfied.

Next-generation mobile networks (5G) also include blind decoding of the control information as an essential component. They have larger search space and allocate less time for the procedure, therefore better methods have to be developed. These methods should perform faster and with lower complexity.

Polar codes, introduced in 2008 by E. Arikan, achieve the capacity of an arbitrary binary memoryless channel [1]. Low-complexity procedures can be used for their encoding and decoding, but their performance is quite poor due to the low minimum distance. Polar subcodes [2] are a generalization of polar codes with improved minimum distance, which can be efficiently decoded by the same algorithms with only minor modifications.

Recently polar codes were adopted for 5G control signaling [3] and the topic of their blind decoding is actively investigated. The problem of distinguishing the polar-coded frames from random data and noise is addressed in [4], where a novel detection metric is presented. In [5], two-stage procedure is proposed. UE identifier is stored in information bits and used to reduce the search space during low-complexity SC decoding and then for early termination when more complex list decoding is performed. However, to the best of our knowledge, there are no attempts to use the tree representation of the polar SC decoding process in order to jointly decode several codes (and hence reduce the decoding complexity compared to the case of running the decoder for each code). Consequently, the problem of polar (sub)code design for the blind decoding setting was not studied. We target both problems and introduce a possible solution.

In this paper, we propose a polar subcode construction, which admits low-complexity blind decoding. Namely, given a set of possible data block sizes and the code length, we show how to construct a family of polar subcodes, so that the decoder can blindly identify the code dimension during the decoding process. These codes can be decoded with the generalizations of list/sequential decoding algorithms and we present the latter one as an example. Its performance and complexity are demonstrated in comparison to the independent decoding attempts. The proposed approach can be used in order to implement the control signaling in 5G wireless systems. It should be noted that this work investigates another aspect of blind decoding, that is different from the ones considered in [4] and [5]. These papers are focused on the replacement of CRC-based detection while using classical polar code design and decoding algorithms. On the contrary, we target polar (sub)code design and low-complexity decoding, which can be later combined with any detection scheme.

II. Background

A. Blind decoding in LTE

In LTE networks, the physical downlink control channel (PDCCH) is used to transmit the downlink control information (DCI). DCI carries the information how to find and decode the data blocks. Before the transmission, a CRC checksum is attached and then DCI is encoded with a tail-biting convolutional code. UE has no information about the parameters being used and should examine all possible combinations of PDCCH locations, PDCCH and DCI formats in the common and UE-specific search spaces. This process is called blind decoding, or blind detection. The search is performed until UE finds parameters which give a valid CRC.

According to the LTE standard [6], there are following specifications for the blind decoding setting:

- Data block size with 16-bit CRC attached: between 24 and 73 bits
- Target code lengths: from 72 to 576 bits

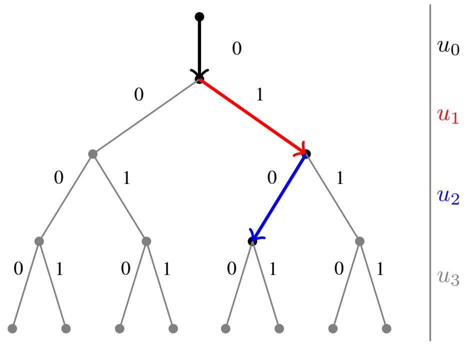

Fig. 1. Decoding tree

- Target frame error rate (FER): 10^{-2}
- Target false alarm rate (FAR): less than 10^{-4}. A false alarm event happens when a polar-encoded frame was not transmitted but is detected

We will further focus on the decoding of fixed-length codes with an unknown dimension. Assume that the code length n is known and the dimension k belongs to some set $\mathcal{K}$ known at the receiver side. The decoder needs to identify the code dimension as well as the payload data. In LTE networks, UE tries to decode (n, k_i) code for each possible $k_i \in \mathcal{K}$ and the process is stopped when the CRC check is passed. The worst-case complexity of this method is $\mathcal{D} = \sum_{i=0}^{|\mathcal{K}|-1} \mathcal{D}_i$, where $\mathcal{D}_i$ is the decoding complexity of (n, k_i) code. The complexity can be expressed in terms of number of floating-point operations needed for the decoding. UE power consumption depends on $\mathcal{D}$ and needs to be reduced.

B. Polar subcodes

A $(n = 2^m, k, d)$ polar subcode [2] is a set of vectors

$$c_0^{n-1} = u_0^{n-1} A_m,$$

where

$$u_{j_r} = \sum_{s=0}^{j_r-1} u_s V_{r,s}, 0 \le r < n-k, \tag{1}$$

for some $j_r \in \{0, \dots, n-1\}, V_{i,s} \in \mathbb{F}_2, A_m = F^{\otimes m}$, where $F = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. Matrix V is constructed so that each codeword of a polar subcode belongs to some $(n, k' > k, d)$ parent code and all positions j_r of last nonzero elements in rows $V_{r,*}$ are distinct and increasing. It can be noticed that this matrix defines the set of frozen symbols $\mathcal{F} = \{j_r | 0 \le r < n-k\}$ and the set of information symbols $\mathcal{I} = \{0, \dots, n-1\} \setminus \mathcal{F}$.

C. Sequential decoding

The sequential decoding algorithm [7], [8], [9], similarly to successive cancellation [1] and list decoding [10] algorithms, is based on the representation of polar decoding as a path search in a binary tree (Figure 1 illustrates this process in case of SC decoding). It uses a priority queue (PQ) to store node identifiers together with their scores and operates as follows:

1) Push into the PQ the root of the tree with score 0.
2) Extract from the PQ the node $u_0^{\phi-1}$ with the highest score.
3) If $\phi = n$, return codeword $u_0^{n-1} A$ and terminate the algorithm.
4) If the number of valid children of node u_0^{i-1} exceeds the amount of free space in the PQ, remove from it the element with the smallest score.
5) Compute the scores $M(u_0^{\phi}, y_0^{n-1})$ of valid children u_0^{ϕ} of the extracted node, and push them into the PQ.
6) If nodes of depth ϕ were extracted L or more times from the PQ, remove from it all nodes $\tilde{u}_0^{j-1}, j \le \phi$.
7) Go to step 2.

The parameter L has the same impact on the performance of the decoding algorithm as the list size in the Tal-Vardy list decoding algorithm. Step 6 ensures that the algorithm terminates in at most Ln iterations. This also gives an upper bound on the number of entries stored in the PQ. However, the algorithm can work with PQ of much smaller size Θ. Step 4 ensures that this size is never exceeded.

The performance and complexity of sequential decoding critically depends on the definition of score function $M(u_0^{\phi-1}, y_0^{n-1})$. It should allow meaningful comparison of paths $u_0^{\phi-1}$ of different lengths ϕ. In [11], the following score function is suggested. Let

$$\tilde{u}_0^{n-1} = \arg\max_{u_\phi^{n-1}} W_m^{(n-1)} \left\{ u_0^{n-1} | y_0^{n-1} \right\} \tag{2}$$

be the most probable continuation of path $u_0^{\phi-1}$ (which does not necessarily correspond to a valid codeword), and

$$R_m^{(\phi-1)}(u_0^{\phi-1}, y_0^{n-1}) = \log W_m^{(n-1)} \left\{ \tilde{u}_0^{n-1} | y_0^{n-1} \right\} \tag{3}$$

be its log-likelihood. One can notice that it is equal to the min-sum list decoding path metric from [12]. Then the score function can be calculated as

$$M(u_0^{\phi-1}, y_0^{n-1}) = R_m^{(\phi-1)}(u_0^{\phi-1}, y_0^{n-1}) - \Psi(\phi), \tag{4}$$

where

$$\Psi(\phi) = \mathbf{E}_{\mathbf{Y}} \left[\log W_m^{\phi-1} \{ \tilde{v}_0^{n-1} | \mathbf{Y} \} \right].$$

The heuristic function $\Psi(\phi)$ is in fact the expected value of log-likelihood $R_m^{(\phi-1)}$ for the correct path $u_0^{\phi-1} = v_0^{\phi-1}$.

III. Construction of codes for blind decoding

A. Decoding in a set of pre-defined codes

Assume that a family of polar subcodes $C^{(i)}$ with parameters $(n, k_i \in \mathcal{K})$, defined by their constraint matrices $V^{(i)}$, is used for transmission of variable-sized data blocks, so that each data block has some auxiliary information (e.g. CRC) used to verify its integrity and, in particular, identify its size.

The straightforward approach to decode a noisy sequence is to apply a list/sequential successive cancellation (SC) decoding algorithm for each of the codes $C^{(i)}$, obtain their most probable codewords, extract the corresponding data blocks, and select the one with valid CRC.

In the case of polar subcodes this approach results in many repetitive calculations. If first f rows of the constraint matrices

$V^{(i)}, V^{(i')}$ are equal, first f frozen symbols and constraints on them are the same in codes $C^{(i)}$ and $C^{(i')}$. Therefore, list decoding algorithm will perform identically steps for both codes until the symbol $u_{j_{f+1}}$ is processed. We want to avoid these calculations and hence reduce the decoding complexity by performing joint list/sequential decoding of codes $C^{(i)}$. The auxiliary information, contained in data blocks, defines some constraints on symbols u_j, similar to (1), which can be used by the decoder in order to select the most likely code before decoding is completed.

The auxiliary information should be constructed in such way, so that the most likely code corresponding to the received noisy sequence can be identified at early decoding steps.

B. *Code design for blind decoding*

Assume we want to transmit variable-length data blocks of size $k_i \in \mathcal{K}, |\mathcal{K}| = \kappa$, encoded into codewords of length n. Consider a family of codes $C^{(i)}(n, k_i + b)$ with constraint matrices $V^{(i)}$ and assume that the code identifier of length $b = \lceil \log_2 \kappa \rceil$ is attached to the beginning of the payload. Recall that the set of frozen symbols consists of the positions of last nonzero entries in rows of the constraint matrix.

In order to jointly decode codes $C^{(i)}$, the decoding steps should be identical for all codes until the last bit of the code identifier is processed. This allows to traverse first part of the decoding tree only once and hence to reduce the complexity.

Let $\mathcal{F}^{(i)}$ and $\mathcal{I}^{(i)}$ be the sets of frozen and information symbols defined by $V^{(i)}$ and $t_s^{(i)}$ be the s-th least element in $\mathcal{I}^{(i)}$. The property described above can be formally written as follows. For any pair of codes $C^{(i)}, C^{(i')}$ should hold

1) $t_s^{(i)} = t_s^{(i')} = t_s$ for $1 \leq s \leq b$;
2) $V_{r,*}^{(i)} = V_{r,*}^{(i')}$ for $0 \leq j_r < t_b$, where $V_{r,*}$ denotes the r-th row of V.

This property will be further addressed as *common tree prefix*.

Example Let us consider $(4, 2)$ and $(4, 3)$ polar subcodes $C^{(0)}, C^{(1)}$ with the corresponding constraint matrices

$$V^{(0)} = \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{smallmatrix}\right), V^{(1)} = \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \end{smallmatrix}\right).$$

In this case $b = 1, t_1 = 1$ and it is easy to notice that in both codes first non-frozen symbol is u_1 and the constraints on the preceding symbol u_0 are identical.

To encode a data block of size k_i, vector $u^{(i)} \in \mathbb{F}_2^n$ is constructed. It consists of symbols of three types:

1) Code identifier;
2) Payload data occupying k_i symbols;
3) $n - (k_i + b)$ frozen symbols defined by (1).

The bits corresponding to the code identifier are placed into symbols $\{u_{t_1}^{(i)}, ..., u_{t_b}^{(i)}\}$ and payload data occupies k_i remaining unfrozen positions. A codeword is finally obtained by multiplying vector $u^{(i)}$ by $F^{\otimes m}, F = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$.

C. *Code modification*

We propose the following method to construct codes with the common tree prefix. One can take a family of arbitrary polar subcodes $C^{(i)}$ and modify their constraint matrices $V^{(i)}$. This modification does not guarantee that the parameters of $C^{(i)}$ (e.g., minimum distance) remain unchanged, but our experiments show that impact on the performance is negligible.

First, one should construct set $\mathcal{I} = \cup_i \mathcal{I}^{(i)}$ and select b its least elements $t_s, 1 \leq s \leq b$. It may happen that symbol u_{t_s} is frozen in code $C^{(i)}$. This means that there exist some row r in $V^{(i)}$ with last nonzero entry in column t_s. In such occasion, one should find a non-frozen position $t' > t_b$ and swap values $V_{r,t_s}^{(i)}$ and $V_{r,t'}^{(i)}$ (i.e., freeze symbol t' and unfreeze symbol t_s).

Example Consider $(4, 2)$ and $(4, 3)$ polar subcodes $C^{(0)}, C^{(1)}$ with the constraint matrices

$$V^{(0)} = \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{smallmatrix}\right), V^{(1)} = \left(\begin{smallmatrix} 1 & 1 & 0 & 0 \end{smallmatrix}\right).$$

Code $C^{(1)}$ has the non-frozen position u_0 which is frozen in code $C^{(0)}$. Hence, modification procedure is applied to matrix $V^{(0)}$ and transforms it into

$$\tilde{V}^{(0)} = \left(\begin{smallmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{smallmatrix}\right).$$

One can easily verify that both matrices $\tilde{V}^{(0)}$ and $V^{(1)}$ define symbol u_0 as non-frozen and therefore have the target property.

Observe that in a classical polar code with a set of frozen symbols $\mathcal{F}$ any non-zero low-weight codeword is obtained as a linear combination of some subset of rows of the Arikan matrix A_m, which includes rows with low-weight indices $j_k \notin \mathcal{F}$. To ensure that non-frozen symbols u_{j_k} with low $\mathrm{wt}(j_k)$ are involved into linear constraints (1), we suggest to define an ordering

$$j_k \prec j_{k'} \Leftrightarrow \mathrm{wt}(j_k) < \mathrm{wt}(j_{k'}) \vee \\ \mathrm{wt}(j_k) = \mathrm{wt}(j_{k'}) \wedge j_k > j_{k'},$$

where $\mathrm{wt}(j)$ denotes the Hamming weight of an integer j, and take t' as the smallest (in terms of $\prec$) $j_{k'} > t_{b-1}$. This reduces the number of low-weight codewords in the constructed code.

IV. Low-complexity blind decoding

In this section, we discuss the modifications of list and sequential algorithms for low-complexity blind decoding of codes, designed in accordance with section III-B.

A. *Joint decoding*

The common tree prefix property of a polar subcodes family $C^{(i)}$ allows one to simply use matrix $V^{(0)}$ for list/sequential decoding until the bit u_{t_b} is processed. Bits $u_{t_s}, s \leq b$ are treated as non-frozen and stored during the decoding process in order to construct the code identifier v_l. It is further used to select the corresponding constraint matrix $V^{(v_l)}$ when a path is extended. When the decoding is finished, a codeword is returned along with its identifier which defined the code being used for the transmission.

One can use LLR-based SCL decoding algorithm [12] for joint decoding of codes $C^{(i)}$ with the modifications described above. However, one needs to reasonably compare paths, corresponding to different codes.

B. Path comparison

Path metric in SCL and the score function (4) in sequential decoding algorithms allow to accurately select the most probable path continuation for each code $C^{(i)}$ in the case of independent decoding attempts. However, they are inefficient in case of joint decoding since correct codewords of a low-rate code may have much bigger absolute metric values than of a high-rate one. Assume that $\hat{c}^{(0)}$ and $\hat{c}^{(1)}$ are the best paths corresponding to (n, k_0) and $(n, k_1), k_0 \ll k_1$ codes $C^{(0)}$ and $C^{(1)}$ with metrics $M^{(0)}$ and $M^{(1)}$, respectively, and $M^{(1)}$ is slightly (let's say 0.001%) higher than $M^{(0)}$. The decoding algorithm which relies on metric M would prefer $\hat{c}^{(1)}$, even though the codeword of code $C^{(1)}$ with score $M^{(1)}$ is less likely to be correct compared to the codeword with score $M^{(0)}$ of the lower-rate code $C^{(0)}$.

In order to overcome this problem, we propose the following approach. Assume the codeword c was transmitted over the AWGN channel and LLR vector $y_i = (-1)^{c_i} + \eta_i, \eta_i \backsim \mathcal{N}(0, \sigma^2)$ was received. Ellipsoidal weight of codeword c is defined as

$$EW(c, y) = \sum_{i:(-1)^{c_i} \neq \text{sgn}(y_i)} -|y_i|.$$

For the same code, higher value of $EW(c, y)$ means the higher probability of c being the transmitted codeword. For a path $u_0^{\phi-1}$, one can obtain an estimate of its ellipsoidal weight as

$$\tilde{w}(u_0^{\phi-1}, y_0^{n-1}) = (M(u_0^{\phi-1}, y_0^{n-1}) + \Psi(n)) \cdot \frac{\sigma^2}{2} \qquad (5)$$

Let $F^{(i)}(w)$ be the probability that a codeword c of code $C^{(i)}$ with an ellipsoidal weight w is incorrect (i.e., $P(EW(c, y) = w, c \neq \hat{c})$ where $\hat{c}$ is the transmitted codeword). It is very difficult to obtain an exact expression for $F^{(i)}(w)$, therefore we propose to use the approximation $\log F^{(i)}(w) \approx f^{(i)}(w) = aw^2 + bw + c$. In order to construct it, one can simply simulate independent decoding of codes $C^{(i)}$ and store the ellipsoidal weights of the obtained codewords along with the mark of successful decoding. Our experiments have shown that $f^{(i)}(w)$ is almost independent of the channel SNR. Figure 2 illustrates $F^{(i)}$ obtained by simulations for $(128, 32)$ and $(128, 64)$ polar subcodes along with its approximation $e^{f^{(i)}}$. One can notice that if $k_0 \ll k_1$ than $F^{(0)}(w) \ll F^{(1)}(w)$, so we can use it to balance the code preference in the decoder.

For each $C^{(i)}$ one can pre-compute the corresponding $f^{(i)}(w)$ and use them to compare paths with different code identifiers. We introduce the heuristic two-step comparison procedure, which uses the ratio of functions $f(w)$ as a test statistic and chooses the code according to some threshold T. In other words, we suggest the following procedure to compare two paths l and l' with $v_l \neq v_{l'}$ $(k_{v_l} \ll k_{v_{l'}})$:

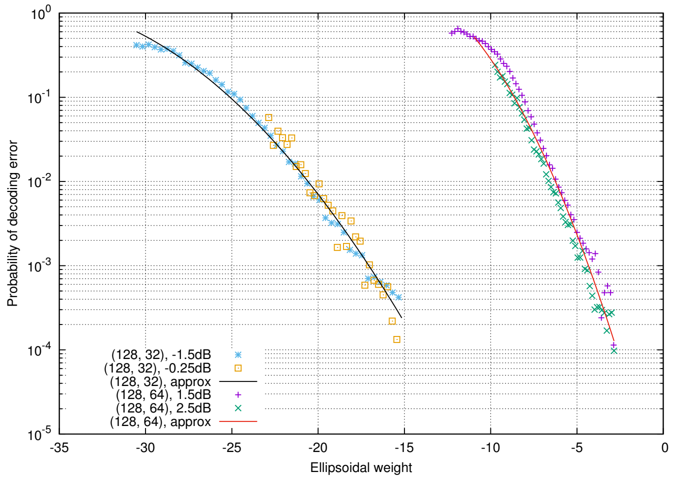

Fig. 2. Decoding error probability

1) Calculate EW estimates $\tilde{w}_l, \tilde{w}_{l'}$ using (5).
2) If $\tilde{w}_l > \tilde{w}_{l'}$ then path l is preferred to l'.
3) Else if $\frac{f^{(v_l)}(\tilde{w}_l)}{f^{(v_{l'})}(\tilde{w}_{l'})} > T$ then path l is preferred to l'.
4) Otherwise, path l' is preferred to l.

This procedure allows one to select a code such that its tree contains a path with sufficiently low $F^{(i)}$. Higher values of T make a higher-rate code more preferable (since $T \to \infty$ in fact corresponds to the decoding relying only on metric M) and setting the value of T close to 0 (or less) makes the decoder prefer only lower-rate code paths. Thus, T sets a preference tradeoff when comparing codes of different rates. The theoretic analysis of this procedure is hard to perform, but one can use Monte-Carlo simulations in order to select T.

C. Blind sequential decoding

In this section we propose some modifications of the sequential decoding algorithm for joint blind decoding of the codes which meet the requirements from section III-B.

The algorithm is presented at Figure 3. It employs data structures similar to [9] and operates in the same way with slight modifications. Paths, corresponding to codes $C^{(i)}$, are stored in distinct priority queues q_i.

Initially, a zero-length path with identifier $v_l = 0$ is pushed into q_0. At each iteration, the best path among all q_i is selected. The selection procedure is described in the previous section and its pseudocode is presented at figure 4. It makes use of the subroutine $GetMax$, which returns the best path from the queue without its removal. In order to process the next symbol, decoder selects the proper set of frozen symbols $\mathcal{F}^{(v_l)}$ (and the constraint matrix $V^{(v_l)}$) using the code identifier. All paths shorter than t_b are treated as belonging to the code $C^{(0)}$.

If currently estimated bit u_j is frozen ($j \in \{j_r | 0 \leq r < n-k\}$), the subroutine $EvaluateDynFrozen$ is used to evaluate its value using the constraint defined by $V_{r,*}$ and the decoded prefix u_0^{j-1}. If u_j carries the value of s-th code identifier bit ($j = t_s$), than the subroutine $StoreIDBit$ is used to set the corresponding bit of a partial code identifier $\tilde{v}_l$. When all b identifier bits are obtained, the subroutine *GetCodeID* is called, which simply returns the value of $\tilde{v}_l$. If the number of codes is less than 2^b, it is also possible to perform the procedure, similar to the described in [5] and not continue the paths with invalid identifiers. Path cloning operation also copies the values of $\tilde{v}_l$ and v_l. When a length-n path l is extracted, the corresponding codeword is returned with its identifier v_l.

V. Numeric results

The code lengths needed in LTE control channel are not the powers of 2. In [13], a method for the construction of polar codes with $n = 2^{m_1} + 2^{m_2}$ was presented. The generalization of the approach defined in section III-B and the decoding algorithm to the case of chained polar codes is straightforward.

The performance and the complexity of the proposed codes and decoding algorithm was studied in the case of AWGN channel and BPSK modulation. We consider encoding of data blocks of size $k \in \{28, 67\}$ into codewords of length $n \in \{288, 576\}$. List size $L = 8$ is used in the decoding algorithms and the threshold T is set to 3.5.

Figures 5-6 illustrate how the proposed approach performs in comparison to the independent sequential decoding of each code. We present simulation results for the case of LTE tailbiting and randomized chained polar subcodes. It can be seen that at the target LTE FER level (10^{-2}) our algorithm has almost negligible performance degradation. Figures 7-8 show the decoding complexity, expressed in average number of algorithm iterations. One can notice the significant complexity reduction in case of joint blind decoding, especially in high-SNR region. When the independent decoding attempts are performed, even though the correct code decoding needs less iterations for higher SNR, for the incorrect code decoder performs significantly more iterations. This leads to the increasing number of iterations with SNR.

In order to obtain the target FAR level, one can either use the scheme similar to [5] and [4], or simply use CRC both both decoding and detection purposes. For example, one can employ CRC-19 in order to achieve the FAR level 10^{-5} with list size $L = 8$. The proposed construction performs in the similar way for the CRC-aided decoding.

VI. Conclusion

A method for encoding variable-sized data blocks into fixed-length codewords is proposed, which employs a family of polar subcodes, as well as the joint blind sequential decoding algorithm. This enables one to significantly reduce the decoding complexity compared to the independent decoding of each code with only marginal performance loss.

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, 7 2009.

[2] P. Trifonov and V. Miloslavskaya, "Polar subcodes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 254–266, Feb 2016.

[3] 3GPP, Draft report of TSG RAN WG1 #87, 11 2016, version 0.2.0.

```
BlindSequentialDecoding($y_0^{n-1}, m, \mathcal{F}^{(i)}, V^{(i)}, L, \Theta, \sigma^2$)
Initialize($m, \Theta$)
$l \leftarrow$ AssignInitialPath()
$S \leftarrow$ GetArrayPointerS($l, 0$)
$v_l \leftarrow 0$
$S[\beta] \leftarrow y_\beta, 0 \le \beta < 2^m$
$\phi_l \leftarrow 0$
$B[0, \dots, 2^m - 1] \leftarrow 0$
$\widehat{R}[l] \leftarrow 0$
$Q_0$.Push($l, 0$)
while At least one $Q_i$ is not empty
do $l \leftarrow$ ExtractMax($\sigma^2$)
    if ($\phi_l > 0$) or ($\phi_l$ is even)
        then RecursivelyUpdateC($l, m, \phi_l - 1$)
    if $\phi_l = 2^m$
        then return [$v_l$, GetArrayPointerC($l, 0, 0$)]
    $B[\phi_l]++$
    RecursivelyCalcS($l, m, \phi_l$)
    $S \leftarrow$ GetArrayPointerS($l, m$)
    $C' \leftarrow$ GetArrayPointerC($l, m, \phi_l \bmod 2$)
    if $\phi_l \in \mathcal{F}^{(v_l)}$
        then $C'[0] \leftarrow$ EvaluateDynFrozen($\phi_l, v_l$)
             if $\mathrm{sgn}(S[0]) \ne (-1)^{C'[0]}$
                then $\widehat{R}[l] -= |S[0]|$
             $Q_{v_l}$.Push($l, \widehat{R}[l] - \Psi(\phi_l)$)
             $\phi_l++$
    else while $Q_{v_l}$.Size() $\ge \Theta - 2$
             do $l'' = Q_{v_l}$.PopMin()
                KillPath($l''$)
             $C'[0] \leftarrow (1 - \mathrm{sgn}(S[0]))/2$
             $Q_{v_l}$.Push($l, \widehat{R}[l] - \Psi(\phi_l)$)
             $l' \leftarrow$ ClonePath($l$)
             $C'' \leftarrow$ GetArrayPointerC($l', m, \phi_l \bmod 2$)
             $C''[0] \leftarrow C'[0] \oplus 1$
             $\widehat{R}[l'] \leftarrow \widehat{R}[l] - |S[0]|$
             $Q_{v_{l'}}$.Push($l', \widehat{R}[l'] - \Psi(\phi_l)$)
             if $\phi_l = t_s, s \le b$
                then StoreIDBit($C'[0], s, l$)
                     StoreIDBit($C''[0], s, l'$)
                     if $\phi_l = t_b$
                        then $v_l \leftarrow$ GetCodeID($l$)
                             $v_{l'} \leftarrow$ GetCodeID($l'$)
             $\phi_{l'} \leftarrow ++\phi_l$
    if $B[\phi_l - 1] \ge L$
       then for all paths $l''_i$ stored in all $Q_i$
            do if $\phi_{l''_i} < \phi_l$
                  then $Q_i$.Delete($l''_i$)
                       KillPath($l''_i$)
```

Fig. 3. Blind sequential decoder

```
ExtractMax($\sigma^2$)
 1  $w \leftarrow -\infty$
 2  $v_l \leftarrow -1$
 3  for $i \in \{0..\kappa - 1\}$
 4  do if $Q_i$.Empty()
 5        then continue
 6      $(S', l') \leftarrow Q_i$.GetMax()
 7      $w' \leftarrow (R[l'] - \Psi(\phi_{l'}) + \Psi(n)) \cdot \frac{\sigma^2}{2}$
 8      if $w' > w$
 9        then if $v_l = -1$ or $\frac{f^{(v_l)}(w)}{f^{(i)}(w')} < T$
10               then $w \leftarrow w'$
11                    $v_l \leftarrow i$
12  return $Q_{v_l}$.ExtractMax()
```

Fig. 4. Extract the best path among κ queues

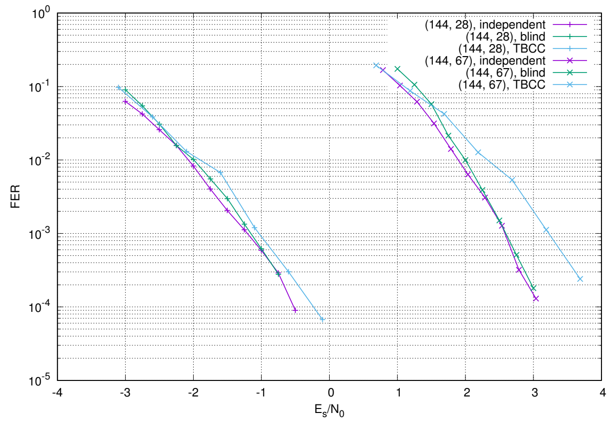

Fig. 5. Error probability, $n = 144$

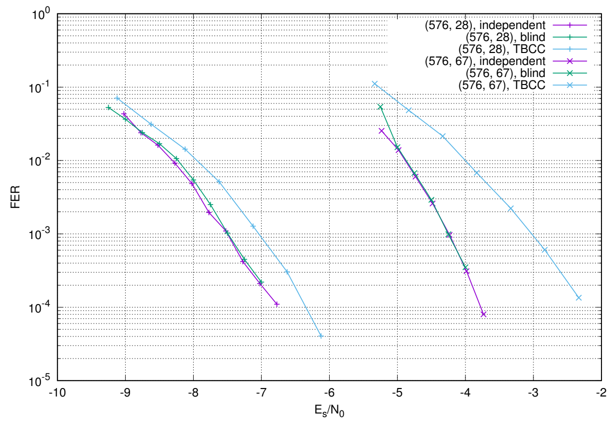

Fig. 6. Error probability, $n = 576$

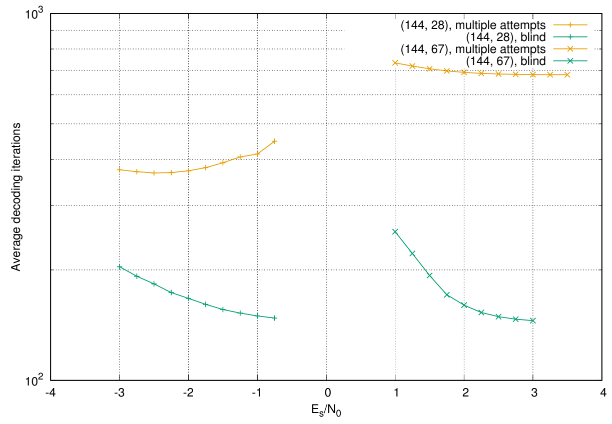

Fig. 7. Average complexity of blind decoding, $n = 144$

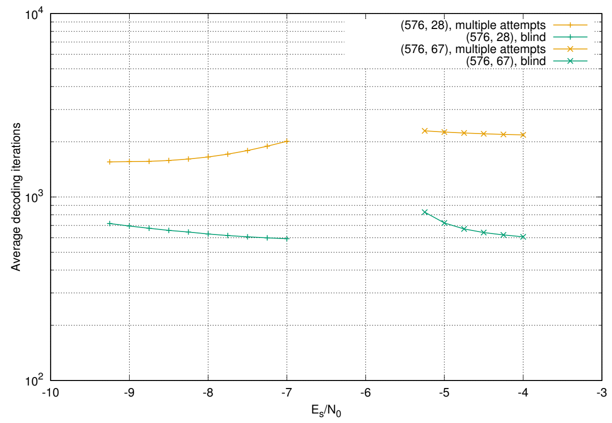

Fig. 8. Average complexity of blind decoding, $n = 576$

[4] P. Giard, A. Balatsoukas-Stimming, and A. Burg, "Blind detection of polar codes," *IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, Oct 2017.

[5] C. Condo, S. A. Hashemi, and W. J. Gross, "Blind detection with polar codes," *IEEE Communications Letters*, vol. 21, no. 12, pp. 2550–2553, Dec 2017.

[6] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding," Technical Specification (TS) 36.212, 08 2016, version 13.2.0.

[7] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. Wiley-IEEE Press, 1999.

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

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

[10] I. Tal and A. Vardy, "List decoding of polar codes," in *IEEE International Symposium on Information Theory*, July 2011, pp. 1–5.

[11] P. Trifonov, "A score function for sequential decoding of polar codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1470–1474.

[12] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "Llr-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, Oct 2015.

[13] P. Trifonov, "Randomized chained polar subcodes," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2018, pp. 25–30.