Fifty-first Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 2 - 3, 2013

Folded Tree Maximum-Likelihood Decoder for Kronecker Product-Based Codes

Sinan Kahraman, Emanuele Viterbo, and Mehmet E. Çelebi

***Abstract*— In this paper, we propose efficient maximum-likelihood (ML) decoding for binary Kronecker product-based (KPB) codes. This class of codes, have a matrix defined by the n-fold iterated Kronecker product $\mathbf{G}_n = \mathbf{F}^{\otimes n}$ of a binary upper-triangular kernel matrix F, where some columns are suppressed given a specific puncturing pattern. Polar and Reed-Muller codes are well known examples of such KPB codes.**

The triangular structure of $\mathbf{G}_n$ enables to perform ML decoding as a binary tree search for the closest codeword to the received point. We take advantage of the highly regular fractal structure of $\mathbf{G}_n$ and the "tree folding" technique to design an efficient ML decoder, enabling to decode relatively longer block lengths than with a standard binary tree search. The tree κ-folding operation transforms the binary tree with N levels into a non-binary tree with $N/2^\kappa$ levels, where the search can be significantly accelerated by a suitable ordering of the branch metrics. For a given κ we can find $\binom{n}{\kappa}$ different folding which lead to decoders with different complexity, for a given code.

Using the proposed folded tree decoder, we provide exact ML performances of some Reed-Muller and polar codes over a binary AWGN channel for the block length up to 256.

***Keywords*: Polar codes, Reed-Muller codes, tree search, ML decoding, Sierpinski triangle, Kronecker product-based (KPB) codes, Folded tree ML decoder.**

I. Introduction

Shannon's noisy channel coding theorem proves the existence of capacity-achieving codes [1]. However, there had been no explicit code construction on a binary-discrete memoryless channel (B-DMC) until the channel polarization was recently introduced by Arıkan in his celebrated paper [2]. Polar coding is known as the first provable class of linear block codes that are achieving capacity by the use of low complexity encoding and decoding methods. These codes have been intensively studied within the coding theory community and leveraged by the simplicity simple successive cancelation (SC) decoding.

Exact maximum-likelihood (ML) decoding has been ruled out as inefficient for long polar codes. A few exact ML decoding methods have been proposed for short block lengths. In [6], ML performance is investigated for short block lengths using Viterbi algorithm. In [7], the code based binary tree search algorithm is introduced as an ML decoder for short polar codes and shown to be effective for a code length up to 64. As a result of this, it was also shown that Reed-Muller (RM) codes outperform polar codes under ML decoding for short block lengths thanks to their larger Hamming distance [5]. List decoding is proposed as an advanced SC decoder for polar codes in [8].

Polar codes achieve capacity for very long block lengths under successive cancelation decoding. It is an open question whether they are able to approach capacity under ML decoding for shorter block lengths.

In this study, we focus on developing an efficient decoding technique that is capable of performing ML decoding for longer codes of length up to 256. We propose a tree folding technique to enable efficient ML decoding for a general class of Kronecker product-based (KPB) codes, which include polar and RM codes.

The general definition of Kronecker product-based (KPB) codes of length N, and dimension K is given as follows.

Definition 1: Let $n = \log_2 N$ and consider the matrix $\mathbf{G}_n = \mathbf{F}^{\otimes n}$ obtained from the n-fold Kronecker product of the 2×2 binary *kernel matrix* $\mathbf{F} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. An $(N, K, \mathcal{F})$ *KPB code* of rate $R = K/N$ is uniquely defined by a set of indices $\mathcal{F}$ of size $N - K$ and its $N \times K$ generator matrix $\mathbf{G}_{\mathcal{F},n}$ is obtained by suppressing the corresponding columns of $\mathbf{G}_n$.

Polar codes are KPB codes where the set $\mathcal{F}$ corresponds to the set of *frozen bits*, selected according to the channel polarization properties and n is the number of polarization steps [2]. For RM codes, the set $\mathcal{F}$ may be chosen to remove all the columns $\mathbf{F}^{\otimes n}$ with Hamming weight below a certain threshold [3],[4]. In the following we will refer to $\mathcal{F}$ as the set of frozen bits of the KPB code. The set $\mathcal{F}$ uniquely identifies the KPB code and is shared between encoder and decoder rather than the generator matrix. The triangular structure of $\mathbf{G}_n$ and the frozen bits set $\mathcal{F}$ enable to perform ML decoding as a binary tree search for the closest codeword to a received point [7]. In particular, we consider *folded tree decoding* based on a non-binary tree search of a tree with fewer levels.

Briefly, the paper contributions can be summarized as follows:

- We note that the matrix $\mathbf{G}_n$ of a KPB code has a fractal structure, that is self similar patterns repeating in successively smaller scales.
- We take advantage of such fractal structure to perform

S. Kahraman is supported by The Scientific and Technological Research Council of Turkey (TUBITAK). He is with Istanbul Technical University, Istanbul, 34469 Turkey, e-mail: kahraman@ieee.org. This work was performed during his visit to the Software Defined Telecommunications (SDT) Lab. at Monash University, Australia.

E. Viterbo is with Monash University, Melbourne, VIC3800 Australia, e-mail: emanuele.viterbo@monash.edu. His work was supported by the Monash Professional Fellowship and the Australian Research Council under Discovery grants ARC DP 130100103.

M. E. Çelebi is with Istanbul Technical University, Istanbul, 34469 Turkey, e-mail: mecelebi@itu.edu.tr.

the *basic folding* operation of the initial binary search tree. The resulting non-binary search tree has half depth but more branches at each node.
- We show that there are $n = \log_2 N$ alternative tree folding operations and n different non-binary search trees can be realized for a given code.
- We then iterate the folding operation κ times and show that there are $\binom{n}{\kappa}$ alternative non-binary folded search trees of height $L = N/2^{\kappa}$.
- The core of the decoding algorithm with the folded tree search is based on a suitable ordering of the branch metrics at each node.
- We provide ML performances of RM and polar codes for $N = 256$ using our folded tree decoder.
- We discuss complexity and memory requirements of our folded tree decoder.

The rest of this paper is organized as follows. In Section II, we discuss the KPB codes and the system model for binary AWGN channel. In Section III, we propose the folding operation. In Section IV, we describe multiple folding operations. In Section V, high level description of the folded tree decoder is given. In Section VI, simulation results are discussed for polar and RM codes under exact ML decoding. Therefore, the complexity of the proposed algorithm is presented. In Section VII, conclusions and future directions are discussed.

II. System Model

From Definition 1 we can generate the codewords $\mathbf{x} = (x_0, \ldots, x_{N-1})^T$ of an $(N, K, \mathcal{F})$ KBP code as

$$\mathbf{x} = \mathbf{G}_{\mathcal{F},n}\widetilde{\mathbf{d}} = \mathbf{G}_n\mathbf{d} \tag{1}$$

using either the $N \times K$ generator matrix $\mathbf{G}_{\mathcal{F},n}$ and the vector $\widetilde{\mathbf{d}}$ of K information bits or the matrix $\mathbf{G}_n$ and the information bit vector $\mathbf{d} = (d_0, \ldots, d_{N-1})^T$, where the $N - K$ bits in positions $\mathcal{F}$ are frozen to '0'.

Analysis of polar code constructions and the channel polarization rule are beyond the scope of this paper. The reader may find a detailed description in [9], while a heuristic method is given in [5] and [6].

Example 1: Let us consider a polar code $P(8, 6)$ with $\mathcal{F} = \{0, 2\}$. The matrix $\mathbf{G}_3$ is given by

$$\mathbf{G}_3 = \mathbf{F}^{\otimes 3} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

Then, $K = 6$ bits are selected in $\mathbf{d}$ to be used for information transmission, according to the construction rule of channel polarization [2] and the bits d_0 and d_2 are frozen. The encoder scheme of the polar code $P(8, 6)$ is shown in Fig. 1 and $\mathbf{x} = \mathbf{G}_3\mathbf{d}$ is given by

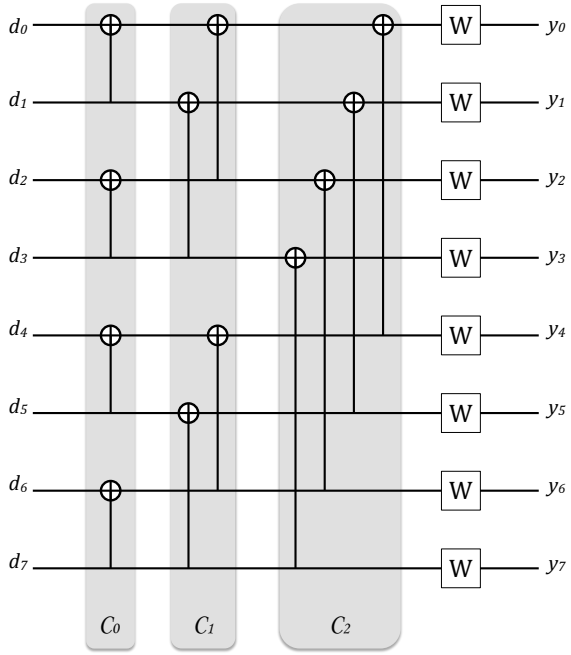

Fig. 1. Encoder scheme for polar code with the block length $N = 8$.

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ d_1 \\ 0 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix}. \tag{3}$$

We now consider the maximum-likelihood detection problem over the AWGN channel, assuming a BPSK modulation (i.e., '1'$\rightarrow +1$, '0' $\rightarrow -1$):

$$\widetilde{\mathbf{y}} = \widetilde{\mathbf{x}} + \mathbf{z}, \tag{4}$$

where $\mathbf{z}$ is the additive white Gaussian noise with zero mean and variance σ^2, and $\widetilde{\mathbf{x}}$ is the vector containing the BPSK modulated signals corresponding to the coded bits in $\mathbf{x}$. By shifting and scaling the received vector $\widetilde{\mathbf{y}}$, we get $\mathbf{y} = \frac{\widetilde{\mathbf{y}}+1}{2}$ and ML decoding is given by

$$\widehat{\mathbf{d}}_{\mathrm{ML}} = \underset{\mathbf{d}|\mathbf{d}^{(\mathcal{F})}=\mathbf{0}}{\arg\min} \|\mathbf{y} - \mathbf{G}_n\mathbf{d}\|^2 \tag{5}$$

where $\mathbf{d}^{\mathcal{F}}$ is the sub-vector of $\mathbf{d}$ with only frozen bits and, by an abuse of notation, we assume binary components '0','1' in $\mathbf{G}_n\mathbf{d}$ are converted to real numbers 0, 1.

III. The folding operation

The definition of KPB codes, is based on the n-fold Kronecker product $\mathbf{G}_n = \mathbf{F}^{\otimes n}$ and results in a fractal structure. In fact, $\mathbf{G}_n$ has the form of Sierpinski triangle, which is a well known fractal in chaotic phenomena introduced in 1915 by W. Sierpinski [10]. An example of the fractal form of the Sierpinski triangle can be seen for $\mathbf{G}_7 = \mathbf{F}^{\otimes 7}$ in Fig.2.

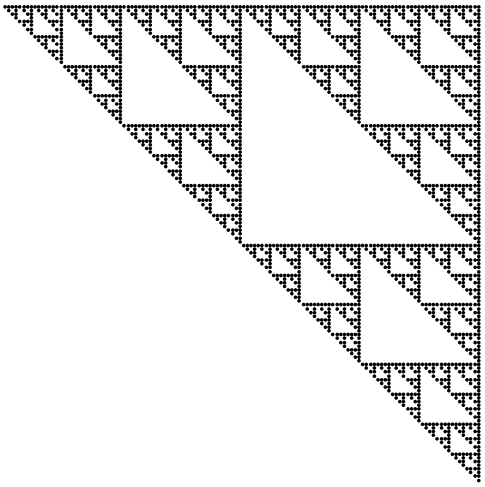

Fig. 2. The matrix $\mathbf{G} = \mathbf{F}^{\otimes 7}$ has the fractal form of a Sierpinski triangle.

The self similarities of the fractal structure repeating in different scales can be seen for any given n polarization steps as

$$\mathbf{F}^{\otimes n} = \begin{bmatrix} \mathbf{F}^{\otimes(n-1)} & \mathbf{F}^{\otimes(n-1)} \\ \mathbf{0} & \mathbf{F}^{\otimes(n-1)} \end{bmatrix}$$

and

$$\mathbf{F}^{\otimes(n+2)} = \begin{bmatrix} \mathbf{F}^{\otimes n} & \mathbf{F}^{\otimes n} & \mathbf{F}^{\otimes n} & \mathbf{F}^{\otimes n} \\ \mathbf{0} & \mathbf{F}^{\otimes n} & \mathbf{0} & \mathbf{F}^{\otimes n} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}^{\otimes n} & \mathbf{F}^{\otimes n} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{F}^{\otimes n} \end{bmatrix}.$$

This attractive property is used for the folding operation. In this way, KPB codes can be ML decoded by the use of non-binary search tree. Let us consider a folded search tree of height $L = N/2$ to show how complexity can be reduced. Writing $\mathbf{x} = \mathbf{F}^{\otimes n}\mathbf{d}$ as

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{\otimes(n-1)} & \mathbf{F}^{\otimes(n-1)} \\ \mathbf{0} & \mathbf{F}^{\otimes(n-1)} \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{N-1} \end{bmatrix}, \tag{6}$$

we can simply split it into two parts

$$\begin{cases} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N/2-1} \end{bmatrix} = \mathbf{F}^{\otimes(n-1)} \begin{bmatrix} d_0 \oplus d_{N/2} \\ d_1 \oplus d_{N/2+1} \\ \vdots \\ d_{N/2-1} \oplus d_{N-1} \end{bmatrix} \triangleq \mathbf{F}^{\otimes(n-1)}\,\mathbf{u_0} \\ \begin{bmatrix} x_{N/2} \\ x_{N/2+1} \\ \vdots \\ x_{N-1} \end{bmatrix} = \mathbf{F}^{\otimes(n-1)} \begin{bmatrix} d_{N/2} \\ d_{N/2+1} \\ \vdots \\ d_{N-1} \end{bmatrix} \triangleq \mathbf{F}^{\otimes(n-1)}\,\mathbf{u_1} \end{cases} \tag{7}$$

In general, we can write the set of pairs of bit indices which are added in $\mathbf{u}_0$ as

$$\mathcal{I} = \{\mathcal{I}_\ell\} = \left\{ \left(\frac{N}{2} - \ell, N - \ell \right), \ \ell = 1, \ldots, L \right\} \tag{8}$$

This particular pairing results in the *basic folding* operation.

With the folding operation, the Euclidean distance term in (5) can be written as a sum of squared norms of two vectors of half the dimension, i.e.,

$$\widehat{\mathbf{d}}_{\mathrm{ML}} = \underset{\mathbf{d}|\mathbf{d}^{(\mathcal{F})}=\mathbf{0}}{\arg\min} \left\{ \left\| \mathbf{y}' - \mathbf{F}^{\otimes(n-1)}\mathbf{u_0} \right\|^2 + \left\| \mathbf{y}'' - \mathbf{F}^{\otimes(n-1)}\mathbf{u_1} \right\|^2 \right\} \tag{9}$$

where

$$\begin{aligned} \mathbf{y}' &= \left(y_0, y_1, \cdots, y_{N/2-1}\right)^T \\ \mathbf{y}'' &= \left(y_{N/2}, y_{N/2+1}, \cdots, y_{N-1}\right)^T. \end{aligned}$$

The new decoding problem can now be solved by a search in a non-binary tree with $N/2$ levels. It should be noticed that $\mathbf{u_0}$ and $\mathbf{u_1}$ are simple linear functions of the pairs $\mathcal{I}_\ell$ of information bits in $\mathbf{d}$, which are used to label the branches of the corresponding non-binary tree. In order to perform exact ML decoding the constraints on the frozen bits pass onto the auxiliary vectors $\mathbf{u_0}$ and $\mathbf{u_1}$. These constraints result in some tree nodes with less than four outgoing branches.

Example 2: Consider the a KPB code in Example 1 of block length $N = 8$. The basic folding is given by

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 \oplus d_4 \\ d_1 \oplus d_5 \\ d_2 \oplus d_6 \\ d_3 \oplus d_7 \end{bmatrix}$$

and

$$\begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix}.$$

The four pairs of input bits which are added in $\mathbf{u_0}$ are $\mathcal{I} = \{(d_7, d_3), (d_6, d_2), (d_5, d_1), (d_4, d_0)\}$. The corresponding non-binary tree is shown in Fig. 6a. At each node on level ℓ a pair of bits $\mathcal{I}_\ell$ is used to select the branch through the corresponding bits $u_{0,\ell}$ and $u_{1,\ell}$.

We can use the block diagram as an alternative description of the folding operation (see Fig.1 for the example with $n = 3$). In the general case, there are n sets of $N/2$ XOR-connections, which reflect the Kronecker product structure of the $\mathbf{G}_n$. These sets are denoted by C_i for $i = 0, \ldots, n-1$ and the XORs connect input bits with a difference in position index of 2^i. The basic folding can be interpreted as moving the XORs in the set C_{n-1} to precode the information bits. The folded equivalent encoder scheme for $N = 8$ is shown in Fig.3.

We can have n possible different foldings for any selected C_i for $i = 0, \ldots, n-1$ sets, which yield different non-binary trees, with $N/2$ levels. In this way, different pairs of bits are XORed depending on the selected C_i. The frozen bits will appear in different levels of the non-binary tree and the actual decoder will behave differently.

Example 3: Let us consider the KPB code of Example 1, $P(8, 6)$ with frozen bits d_0 and d_2. We can provide two additional different foldings based on the structures in Figs. 4 and 5 by moving the XOR connections in C_0 and C_1, respectively. Each folding operation is defined by the different pairs

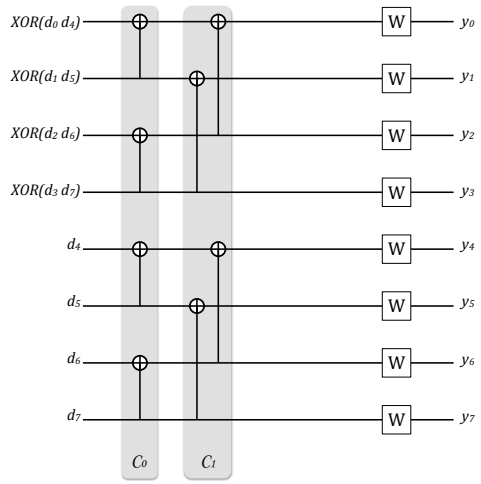

Fig. 3. Equivalent encoder scheme of *basic folding* is given for $N = 8$ by moving the C_2 set of XOR-connections.

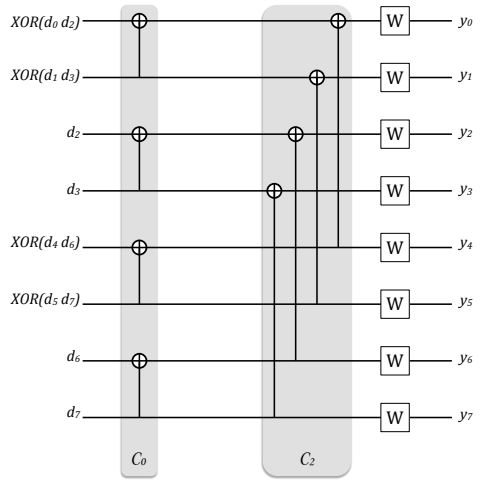

Fig. 4. Equivalent encoder scheme of the folding is given for $N = 8$ by the deletion of C_1. C_1 is the set of xor-connections with the length of 2.

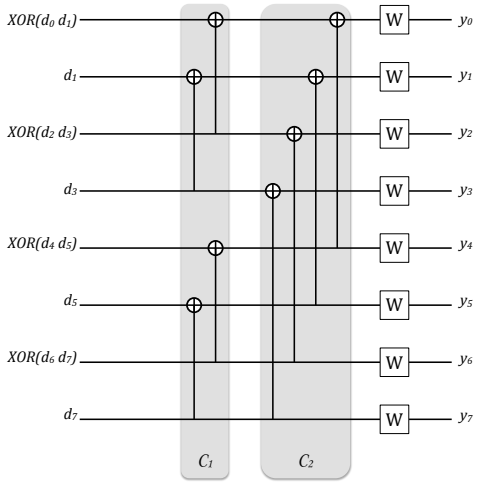

Fig. 5. Equivalent encoder scheme of the folding is given for $N = 8$ by the deletion of C_0. C_0 is the set of xor-connections with the length of 1.

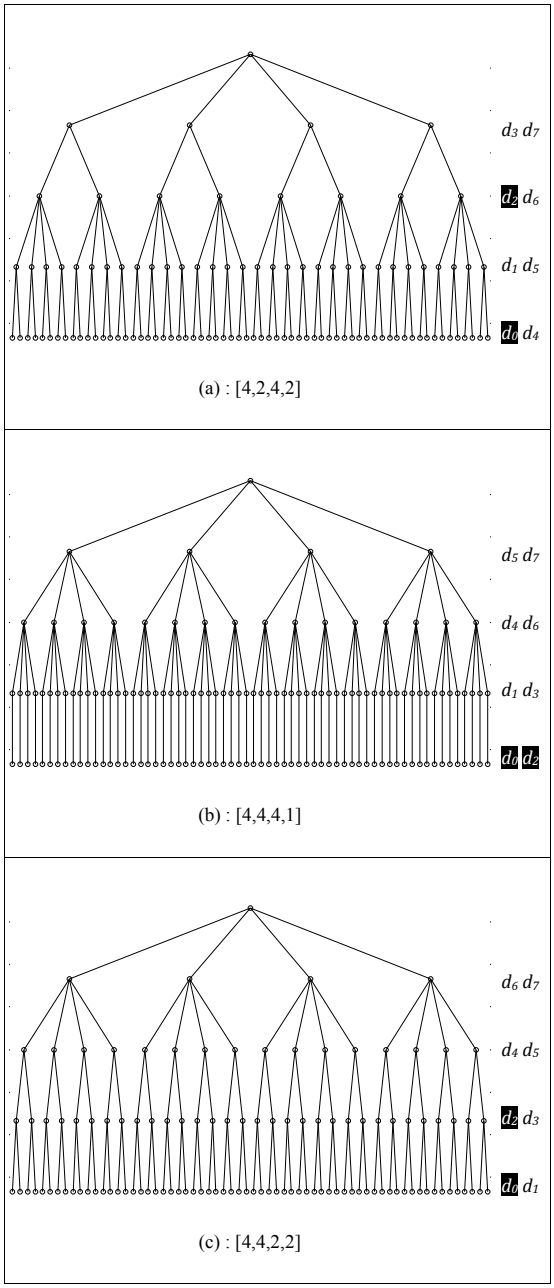

Fig. 6. Non-binary trees for $P(8, 6)$. The frozen bits are d_0 and d_2. (a) folding by deletion of C_2, (b) folding by deletion of C_1, (c) folding by deletion of C_0.

of input bits: $\mathcal{I} = \{(d_7, d_6), (d_5, d_4), (d_3, d_2), (d_1, d_0)\}$ for C_0 and $\mathcal{I} = \{(d_7, d_5), (d_6, d_4), (d_3, d_1), (d_2, d_0)\}$ for C_1. All three non-binary trees with 4 levels, that can be constructed for this code, are shown in Fig.6.

We will later show an example of how the complexities of different non-binary trees for a given KPB code can vary due to the different tree structures.

IV. Multiple Folding

Folding operations can be repeated more than once to further reduce the number of tree levels. In particular, we can consider the multiple folding of order $1 < \kappa \leq n$, which is

TABLE I

NUMBER OF DIFFERENT FOLDED TREES FOR A GIVEN N AND κ

κ	N=8	16	32	64	128	256	512	1024	2048
1	3	4	5	6	7	8	9	10	11
2	3	6	10	15	21	28	36	45	55
3	1	4	10	20	35	56	84	120	165
4	0	1	5	15	35	70	126	210	330
5	0	0	1	6	21	56	126	252	462
6	0	0	0	1	7	28	84	210	462
7	0	0	0	0	1	8	36	120	330
8	0	0	0	0	0	1	9	45	165

defined by choosing κ sets among the n XOR connection sets C_i. There are a total of $\binom{n}{\kappa}$ different κ-foldings, resulting in different decoding trees of hight $L = N/2^\kappa$. We can take advantage in selecting the different folding that yields the smallest decoding complexity for a given KPB code. For convenience, Table 1 gives the number of different folded trees for a given N and κ.

In the general case, we can rewrite the equations in (7) in terms of $\mathbf{F}^{\otimes(n-2)}$. Let us now consider the $\kappa = 2$ folding obtained by further applying the basic folding to each of the two equations in (7). We then have four equations with vectors of length $N/4$:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N/4-2} \\ x_{N/4-1} \end{bmatrix} = \mathbf{F}^{\otimes(n-2)} \begin{bmatrix} d_0 \oplus d_{N/4} \oplus d_{N/2} \oplus d_{3N/4} \\ d_1 \oplus d_{N/4+1} \oplus d_{N/2+1} \oplus d_{3N/4+1} \\ \vdots \\ d_{N/4-2} \oplus d_{N/2-2} \oplus d_{3N/4-2} \oplus d_{N-2} \\ d_{N/4-1} \oplus d_{N/2-1} \oplus d_{3N/4-1} \oplus d_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} x_{N/4} \\ x_{N/4+1} \\ \vdots \\ x_{N/2-2} \\ x_{N/2-1} \end{bmatrix} = \mathbf{F}^{\otimes(n-2)} \begin{bmatrix} d_{N/4} \oplus d_{3N/4} \\ d_{N/4+1} \oplus d_{3N/4+1} \\ \vdots \\ d_{N/2-2} \oplus d_{N-2} \\ d_{N/2-1} \oplus d_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} x_{N/2} \\ x_{N/2+1} \\ \vdots \\ x_{3N/4-2} \\ x_{3N/4-1} \end{bmatrix} = \mathbf{F}^{\otimes(n-2)} \begin{bmatrix} d_{N/2} \oplus d_{3N/4} \\ d_{N/2+1} \oplus d_{3N/4+1} \\ \vdots \\ d_{3N/4-2} \oplus d_{N-2} \\ d_{3N/4-1} \oplus d_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} x_{3N/4} \\ x_{3N/4+1} \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \mathbf{F}^{\otimes(n-2)} \begin{bmatrix} d_{3N/4} \\ d_{3N/4+1} \\ \vdots \\ d_{N-2} \\ d_{N-1} \end{bmatrix}.$$

This is equivalent to moving both of the XOR connection sets C_{n-1} and C_{n-2} to the input. We can derive a formula for the indices of the grouped bits after κ basic foldings. The set of indices of the 2^κ bits placed in the ℓth level of the tree is given by

$$\mathcal{I}_\ell = \left(\frac{1 \cdot N}{2^\kappa} - \ell, \frac{2 \cdot N}{2^\kappa} - \ell, \cdots, \frac{2^\kappa \cdot N}{2^\kappa} - \ell \right), \ell = 1, \ldots, L \tag{10}$$

TABLE II

INITIALIZATION

$r = \infty$

$L = N/2^\kappa$

$\ell = 1$

$\rho = (\rho_1, \rho_2, \ldots, \rho_L)_{1\times L}$

$\mathbf{c} = (0, 0, \ldots, 0)_{1\times L}$

$\mathbf{ind} = (0, 0, \ldots, 0)_{1\times L}$

$\mathbf{m} = (0, 0, \ldots, 0)_{1\times L}$

$\mathbf{t} = (0, 0, \ldots, 0)_{1\times N}$

$\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_L\}$

$\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_L\}$

$\mathbf{table}_\ell = \text{sort}_{\mathbf{u}} \left(\left\| \mathbf{u} - \mathbf{y}^{(\mathcal{I}_\ell)} \right\|^2 \right), \forall \ell = 1, \ldots, L$

We will refer to this particular choice of $\mathcal{I}_\ell$s as the *basic* κ-*folding* operation. The other choices of κ XOR connections sets yield different expressions for the groupings $\mathcal{I}_\ell$.

V. THE FOLDED TREE ML DECODER

In this section, we describe the folded tree ML decoding algorithm for general KPB codes and provide the full pseudo-code of a non recursive implementation. We denote vector variables by the boldface letters and consider a κ folding operation for a given $(N, K, \mathcal{F})$ KPB code.

As illustrated in the previous section, a non-binary tree with height $L = N/2^\kappa$ is constructed and the set $\mathcal{I} = \{\mathcal{I}_\ell, \ell = 1, \ldots, L\}$ defines the labeling for κ folded tree branches.

The decoder takes as inputs the received noisy vector $\mathbf{y} = (y_0, y_1, \ldots, y_{N-1})^T$, the indices of the $N - K$ frozen bits are denoted by $\mathcal{F}$ and set $\mathcal{I} = \{\mathcal{I}_\ell, \ell = 1, \ldots, L\}$. Let $\mathcal{F}_\ell$ denote the set of frozen indices appearing at level ℓ of the tree. The exact structure of the non-binary tree depends on the labeling $\mathcal{I}$ and the indices of frozen bits $\mathcal{F}$.

Let ρ_ℓ denote the number of non-frozen bits in $\mathcal{I}_\ell$, for $\ell = 1, \ldots, L$. We assume the root of the tree is level 0 and, the leafs are level L. For a given code, different folding choices yield different $\mathcal{I}$ and different ρ_ℓs. In this case, each node at level ℓ has 2^{ρ_ℓ} valid candidates. This means a valid candidate has no conflict between frozen bits.

The initialization steps for the decoder are given in Table II, where: r is the radius of the search sphere, $\mathbf{c}$ contains the current branch counter for each level, $\mathbf{ind}$ contains the current index in $\mathbf{table}_\ell$, $\mathbf{m}$ contains the partial metrics for each level, and $\mathbf{t}$ is the current candidate vector.

For all levels, $\ell = 1, \ldots, L$, the initialization generates a $\mathbf{table}_\ell$ by sorting in increasing order the Euclidean distances between all possible binary vectors $\mathbf{u}$ of 2^κ bits and the received sub-vector $\mathbf{y}^{(\mathcal{I}_\ell)}$. Each record of the sorted tables stores two entries: the decimal representation of the binary vector $\mathbf{u}$ and the corresponding squared distance. These guarantee that the shortest metric branches are visited first during the tree search.

TABLE III
MAIN ALGORITHM

while $\ell \neq 0$,
  if $c_\ell < 2^{\rho_\ell}$
    $c_\ell = c_\ell + 1$
    $(\widetilde{m}_\ell, \mathbf{t}^{(\mathcal{I}_\ell)}, ind_\ell) =$
     $PickNextCandidate\left(\mathbf{t}, \ell, ind_\ell, \mathbf{table}_\ell, \mathcal{I}_\ell, \mathcal{F}_\ell, \bar{\mathcal{F}}_\ell\right)$
    $m_\ell = \widetilde{m}_\ell$
    if $sum(\mathbf{m}) < r$
      if $\ell = L$
        $store\ \widehat{\mathbf{d}} = \mathbf{t}$
        $update\ r = sum(\mathbf{m})$
        $\mathbf{t}^{(\mathcal{I}_\ell)} = \mathbf{0}$
        $m_\ell = 0$
        $\ell = \ell - 1$
      else
        $\ell = \ell + 1$
        $c_\ell = 0$
        $ind_\ell = 0$
      end
    else
      $\mathbf{t}^{(\mathcal{I}_\ell)} = \mathbf{0}$
      $m_\ell = 0$
      $\ell = \ell - 1$
    end
  else
    $\ell = \ell - 1$
  end
end
ML_Estimation : $return\ \widehat{\mathbf{d}}$
Minimum_metric : $return\ r$.

TABLE IV
PICKNEXTCANDIDATE FUNCTION

function $PickNextCandidate\left(\mathbf{t}, \ell, ind_\ell, \mathbf{table}_\ell, \mathcal{I}_\ell, \mathcal{F}_\ell, \bar{\mathcal{F}}_\ell\right)$
repeat
  $ind_\ell = ind_\ell + 1$
  $\mathbf{u} = dec2bin(\mathbf{table}_\ell(1, ind_\ell))$
  $solve\ \hat{\mathbf{d}}^{(\mathcal{I}_\ell)} = \mathbf{F}^{\otimes\kappa} \cdot [(\mathbf{F}^{\otimes n}\mathbf{t})^{(\mathcal{I}_\ell)} \oplus \mathbf{u}]$
until $\hat{\mathbf{d}}^{(\mathcal{F}_\ell)} = \mathbf{0}$
$fill\ non\ frozen\ bits\ \mathbf{t}^{(\bar{\mathcal{F}}_\ell)} = \hat{\mathbf{d}}^{(\bar{\mathcal{F}}_\ell)}$
$\widetilde{m}_\ell = \mathbf{table}_\ell(2, ind_\ell)$
$return\ (\widetilde{m}_\ell, \mathbf{t}^{\mathcal{I}_\ell}, ind_\ell)$

Such table is pre-computed to speed up the tree search phase for the exact ML solution. The pseudo code of the main algorithm is given in Table III and the auxiliary function in Table IV.

The algorithm essentially operates as a sphere decoder [11] and successively reduces the search radius r as closer points to the received vector are found along the tree search. The faster the radius is reduced the more the tree branches are pruned and less nodes need to be visited. The tree search starts from the root node at level $\ell = 1$ and, the search radius r is set to infinity. The algorithm moves to lower levels under the radius constraint, i.e., only if the current accumulated metric is smaller than the radius $sum(\mathbf{m}) < r$. If the radius constraint is not valid in the visited level, algorithm moves to the upper level. The current metric element m_ℓ and the currently estimated vector $\mathbf{t}$ must be updated in both cases.

If the radius constraint is valid at the bottom level L, then the ML estimate $\hat{\mathbf{d}}$ is updated by the current $\mathbf{t}$. The radius is updated by $sum(\mathbf{m})$ and the algorithm moves to the upper level.

Let us denote by $\mathcal{F}_\ell$ the set of indices of the frozen bits at level ℓ and $\bar{\mathcal{F}}_\ell$ the set of non-frozen bits, such that $\mathcal{I}_\ell = \mathcal{F}_\ell \cup \bar{\mathcal{F}}_\ell$. At each level ℓ, the next candidate branch is chosen by the function $PickNextCandidate$ (see Table IV) to fill the bits in positions $\mathcal{I}_\ell$ of the vector $\mathbf{t}$. Note that $|\mathcal{F}_\ell|$ of the 2^κ bits in $\mathbf{t}^{(\mathcal{I}_\ell)}$ are frozen and only $\mathbf{t}^{(\overline{\mathcal{F}_\ell})}$ need to be updated. In this process, $PickNextCandidate$ uses $\mathbf{table}_\ell$ to pick up the next candidate in the table, which does not conflict with the frozen bits constraints and the already selected bits in $\mathbf{t}$. It also returns the updated index in the table and the branch metric.

As observed at the beginning of the section, this algorithm behaves as a sphere decoder and searches the non binary tree in such a way to minimize the distance to the received point, by only pruning the the non-competing branches that have a larger accumulated metric. This yields the exact ML decision.

VI. SIMULATION RESULTS

In this section, we discuss the exact ML performances of RM and polar codes for block lengths 128 and 256. The folded tree decoder with $\kappa = 4$ is used for KPB codes $(128, 120)$ and $(256, 247)$. Bit error rates versus E_b/N_0 are given in Figs. 7-8. Also the Shannon bounds for the rates 120/128 and 247/256 are shown. Frame error rates vs. E_b/N_0 are given in Figs. 9-10. By design, polar codes outperform RM codes with suboptimal SC decoding in terms of BER. It is interesting to observe that the RM codes outperform the polar codes under ML decoding. It remains an open question if this is true for larger code lengths.

We discuss expected complexity of the proposed algorithm in terms of the average number of visited nodes. Fig.11 shows that the expected complexity is affected by the positions of the frozen bits in the non-binary tree. It can be seen that the binary search tree in [7] required more than 10^5 node visits for $RM(64, 57)$. Here, two non-binary trees with $\kappa = 3$ and $\kappa = 4$ are considered for the $RM(64, 57)$ in Fig.12. The folded tree decoder with ML performance requires significantly lower decoding complexity for $RM(64, 57)$. Additionally, the expected complexity for $RM(128, 120)$ is shown in Fig. 13 for $\kappa = 3$ and 4. Selecting the most appropriate folding can significantly reduce the complexity at low SNR.

We now discuss the memory requirement for the L stored tables labeled as $\mathbf{table}_\ell$ in the setup of the proposed algorithm. Each table contains the indices of all possible vectors with 2^κ bits and requires 2^{2^κ} entries. Moreover, $L = N/2^\kappa$ tables are required for κ foldings. Thus, the total number of sorted indices can be given as

$$\Delta = \frac{N}{2^\kappa} 2^{2^\kappa}.$$

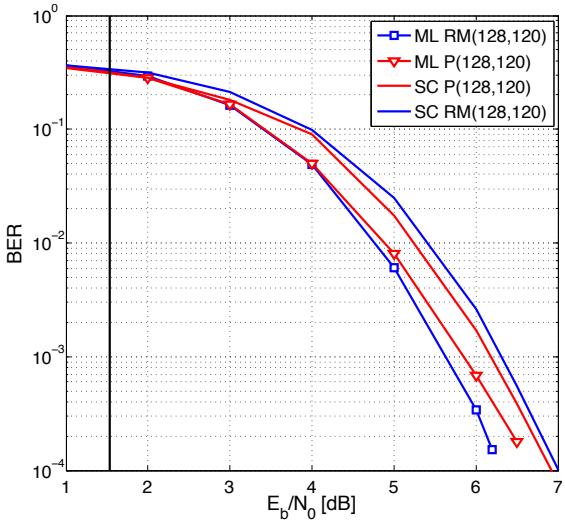

Fig. 7. Performance comparison BER vs. E_b/N_0 for P(128,120) and RM(128,120). Exact ML decoding by the folded tree decoder with $\kappa = 4$ and SC decoding.

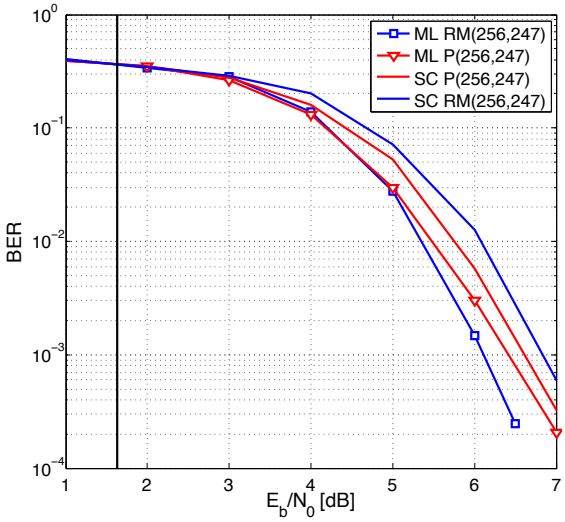

Fig. 8. Performance comparison BER vs. E_b/N_0 for P(256,247) and RM(256,247). Exact ML decoding by the folded tree decoder with $\kappa = 4$ and SC decoding.

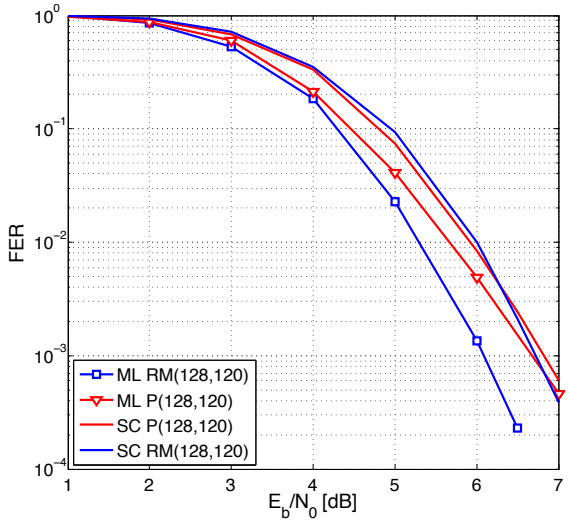

Fig. 9. Performance comparison FER vs. E_b/N_0 for P(128,120) and RM(128,120). Exact ML decoding by the folded tree decoder with $\kappa = 4$ and SC decoding.

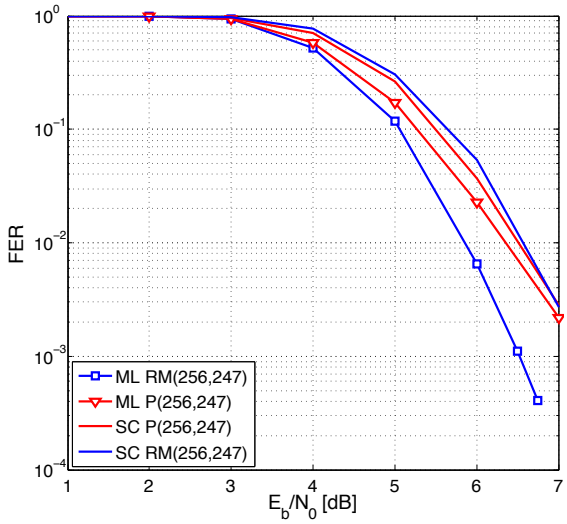

Fig. 10. Performance comparison FER vs. E_b/N_0 for P(256,247) and RM(256,247). Exact ML decoding by the folded tree decoder with $\kappa = 4$ and SC decoding.

Since we need 2^κ bits to define an index in the tables, the total number of required bits is given by

$$2^\kappa \Delta = N 2^{2^\kappa} \tag{11}$$

and shown in the table below

TABLE V
NUMBER OF REQUIRED BITS FOR A GIVEN N AND κ

κ	N=128	256	512	1024	2048
1	1KB	2KB	4KB	8KB	16KB
2	4KB	8KB	16KB	32KB	64KB
3	32KB	64KB	128KB	256KB	512KB
4	1MB	2MB	4MB	8MB	16MB

VII. CONCLUSION

In this work, we observed the fractal structure of the Kronecker product-based codes such as polar codes and, Reed-Muller codes. We proposed the κ folding method as a tool for efficient decoding of such family of codes. Moreover, we showed that different foldings are available to construct different non-binary tree structures for a given code. Using this, we proposed an efficient exact ML decoding algorithm. By the use of this algorithm, we provide the exact ML decoding of the Reed-Muller and polar codes with the block length 256. We are currently investigating methods for reducing complexity and memory requirements of the proposed algorithm and as future work we will explore a suboptimal variant of the algorithm to enable the decoding of even longer block lengths.

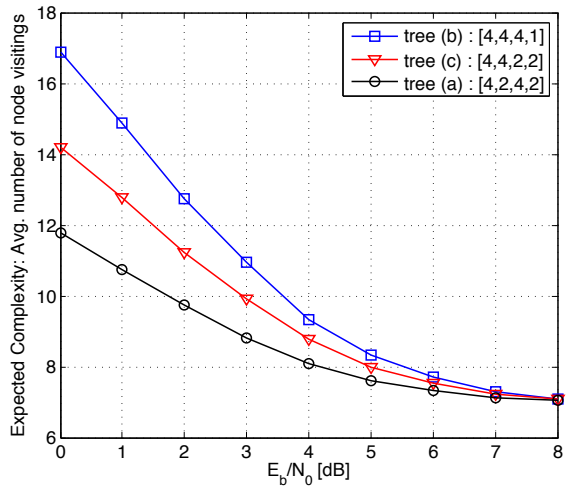

Fig. 11. Expected decoding complexities of non-binary trees (a) (b) (c) for $P(8,6)$. The frozen bits are d_0 and d_2.

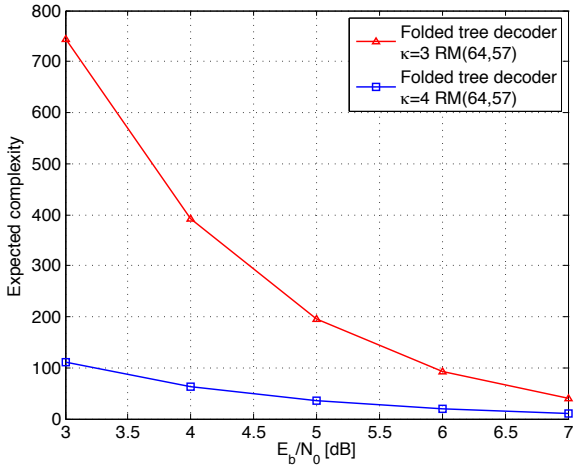

Fig. 12. Expected decoding complexity for $RM(64,57)$.

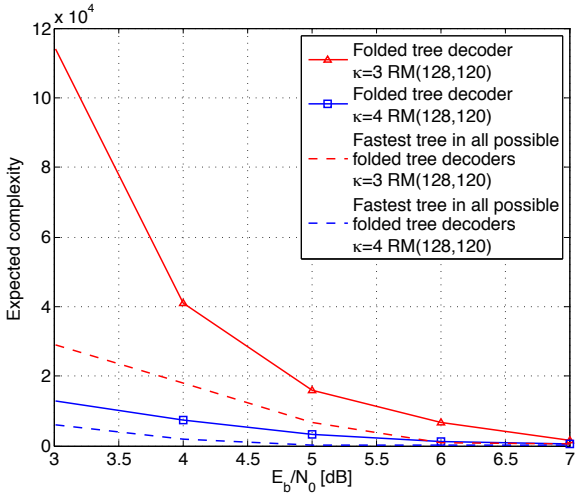

Fig. 13. Expected decoding complexity for $RM(128,120)$.

REFERENCES

[1] C.E. Shannon, "A mathematical theory of communication," *Bell System Tech. J.*, 27(2), pp. 379-423, 623–656, 1948.

[2] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inform. Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[3] D. E. Muller, "Application of boolean algebra to switching circuit design and to error correction," *IRE Trans. Electron. Computers*, vol. EC-3, pp. 6–12, Sept. 1954.

[4] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme,," *IRE Trans. Inform. Theory*, vol. 4, pp. 39–44, Sept. 1954.

[5] E. Arıkan, "A survey of reed-muller codes from polar coding perspective," in *Proc. IEEE Inform. Theory Workshop,* 6-8 Jan. 2010.

[6] E. Arıkan, K. Haesik, M. Garik, Ö. Üstün, and E. Efecan, "Performance of short polar codes under ML decoding," in *Proc. ICT Mobile Summit 2009*, (Santander, Spain), 10-12 June 2009.

[7] S. Kahraman, and M. E. Çelebi, "Code based efficient maximum-likelihood decoding of short polar codes," in *Proc. 2012 IEEE Int. Symp. Inform. Theory,* (Cambridge, USA), pp. 1967–1971, 2012.

[8] I. Tal, and A. Vardy, "List Decoding of Polar Codes," in *Proc. 2011 IEEE Int. Symp. Inform. Theory,* (St.Petersburg, Russia), pp. 1–5, 2011.

[9] I. Tal, and A. Vardy, "How to Construct Polar Codes," in [online available] *http://arxiv.org/abs/115.6164*, 2011.

[10] W. Sierpinski, "Sur une courbe dont tout point est un point de ramification," *C.R. Acad. Paris,* 160, pp. 302–305, 1915.

[11] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.