Efficient SC Decoding of Convolutional Polar Codes

Ruslan Morozov, Peter Trifonov
St. Petersburg Polytechnic University
e-mail: {rmorozov, petert}@dcn.icc.spbstu.ru

***Abstract*—An efficient numerically stable successive cancellation decoding algorithm for convolutional polar codes is proposed. The proposed algorithm makes use of only summation and comparison operations, and admits immediate extension to the case of list SC decoding, which provides near-ML performance.**

I. Introduction

Convolutional polar codes (CPCs) are introduced by A. J, Ferris, C. Hirche and D. Poulin [1] as a special case of branching multi-scale entanglement renormalization ansatz (branching MERA) codes. Both open-boundary and periodic-boundary CPCs are presented in [1]. In this paper by CPCs we always mean open-boundary CPCs.

CPCs are shown to have a performance gain under the successive cancellation (SC) decoding algorithm compared to classical polar codes [2].

In this paper we provide an explicit description of the SC decoding algorithm for CPCs. Furthermore, we derive a min-sum approximation for it, which requires only addition and comparison operations. The proposed approach can be extended to the case of list SC decoding, using the same techniques as in [3].

II. Convolutional polar codes

An $(n=2^m,k)$ convolutional polar code (CPC) is defined as a set of vectors $c_0^{n-1}=u_0^{n-1}Q_n$, where $u_{\mathcal{F}}=0_0^{n-k-1}$, $\mathcal{F}\subset[n],|\mathcal{F}|=n-k$,$[n]$ denotes set $\{0,1,\ldots,n-1\}$, and bits $u_{\mathcal{I}}=a_0^{k-1}$, where $\mathcal{I}=[n]\setminus\mathcal{F}$ and a_0^{k-1} are uniformly distributed data bits. Here and throughout the paper, a_B denotes vector of elements of a with indices from B. The convolutional polarizing transformation (CPT) Q_n is an $n\times n$ matrix, which has recursive structure, depicted in Fig. 1, and is given by

$$Q_n=\mathrm{diag}(I_1,I_{n/2-1}\otimes Q_2,I_1)(I_{n/2}\otimes Q_2)R_n(I_2\otimes Q_{n/2})R_n^{-1}, \quad (1)$$

where R_n is the matrix of the permutation "even first odd last" $(0,2,\ldots,n-2,1,3,\ldots,n-1)$, $Q_2=\begin{pmatrix}1&0\\1&1\end{pmatrix}$, and I_s is the $s\times s$ identity matrix.

Algorithm 1 implements non-systematic encoding of CPC. Each layer λ of the CPT, defined in lines 1.2–1.9, consists of two sub-layers: the first sub-layer in lines 1.3–1.5, and the second sub-layer in lines 1.6–1.8, which is equivalent to the layer of the Arikan polarizing transformation.

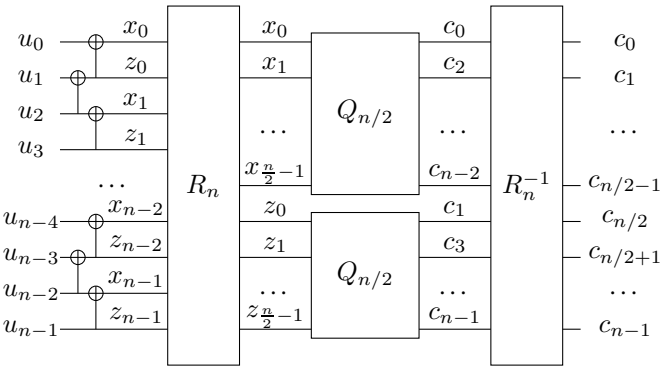

Fig. 1: Recursive construction of CPT Q_n of size n

Algorithm 1: $\mathtt{Encode}(m,a_0^{k-1},\mathcal{I})$

1.1 $n\leftarrow 2^m$, $c\leftarrow 0_0^{n-1}$, $c_{\mathcal{I}}\leftarrow a_0^{k-1}$, $N\leftarrow 2$
1.2 **for** $\lambda\leftarrow m\ldots 1$ **do**
1.3 **for** $g\leftarrow 0\ldots n/N-2$ **do**
1.4 **for** $j\leftarrow 0\ldots N/2-1$ **do**
1.5 $i\leftarrow gN+N/2+j$, $c_i\leftarrow c_i+c_{i+N/2}$
1.6 **for** $g\leftarrow 0\ldots n/N-1$ **do**
1.7 **for** $j\leftarrow 0\ldots N/2-1$ **do**
1.8 $i\leftarrow gN+j$, $c_i\leftarrow c_i+c_{i+N/2}$
1.9 $N\leftarrow 2N$
1.10 **return** c_0^{n-1}

III. Successive Cancellation Decoding

In this section we provide an explicit derivation of the SC decoding algorithm for CPC. The presented algorithm has time and space complexity $\Theta(n\log n)$.

A. Problem statement

Consider transmission of codeword $c_0^{n-1}=u_0^{n-1}Q_n$ through binary-input memoryless channel $\mathcal{W}:\mathbb{F}_2\rightarrow\mathcal{Y}$. Let y_0^{n-1} be the output of this channel. After demodulation, the probabilities $W(c_i|y_i)=\mathcal{W}(y_i|c_i)/\left(\mathcal{W}(y_i|0)+\mathcal{W}(y_i|1)\right)$ for $c_i\in\mathbb{F}_2$ are provided to the decoding algorithm.

Given the prior hard decisions $\hat{u}_0\ldots\hat{u}_{\phi-1}$, at the ϕ-th phase the SC decoding algorithm calculates probabilities $W_m^{(\phi)}(\hat{u}_0^{\phi-1},u_\phi|y_0^{n-1})$ defined as

$$W_m^{(\phi)}(u_0^{\phi}|y_0^{n-1})=\sum_{u_{\phi+1}^{n-1}\in\mathbb{F}_2^{n-\phi-1}}W^n(u_0^{n-1}Q_n|y_0^{n-1}), \quad (2)$$

where $W^n(c_0^{n-1}|y_0^{n-1}) = \prod_{i=0}^{n-1} W(c_i|y_i)$. Then, the hard decision on u_ϕ is made by

$$\hat{u}_\phi = \begin{cases} 0, & \phi \in \mathcal{F} \\ \arg\max\limits_{u_\phi \in \mathbb{F}_2} W_m^{(\phi)}(\hat{u}_0^{\phi-1}, u_\phi | y_0^{n-1}), & \phi \notin \mathcal{F}. \end{cases} \quad (3)$$

Below we obtain polar-like recursive formulae for computing probabilities (2).

B. Recursive formulae for probabilities calculation

Consider layer λ of CPT. Let $\Lambda = 2^\lambda$. Equation (1) implies

$$\begin{aligned} & W^\Lambda\left(u_0^{\Lambda-1} Q_\Lambda | y_0^{\Lambda-1}\right) \\ =& W^\Lambda\left((x_0, z_0, \ldots, x_{\frac{\Lambda}{2}-1}, z_{\frac{\Lambda}{2}-1}) R_\Lambda (I_2 \otimes Q_{\Lambda/2}) R_\Lambda^{-1} | y\right) \\ \overset{(1)}{=} & W^\Lambda\left((x_0^{\Lambda/2-1}, z_0^{\Lambda/2-1}) \begin{pmatrix} Q_{\Lambda/2} & 0 \\ 0 & Q_{\Lambda/2} \end{pmatrix} | y', y''\right) \\ =& W^{\Lambda/2}\left(x_0^{\Lambda/2-1} Q_{\Lambda/2} | y'\right) W^{\Lambda/2}\left(z_0^{\Lambda/2-1} Q_{\Lambda/2} | y''\right), \end{aligned} \quad (4)$$

where shortcuts $y = y_0^{\Lambda-1}, y' = y_{0,e}^{\Lambda-1}, y'' = y_{0,o}^{\Lambda-1}$,

$$a_{b,e}^c = (a_i | b \le i \le c, i \equiv 0 \bmod 2),$$
$$a_{b,o}^c = (a_i | b \le i \le c, i \equiv 1 \bmod 2),$$

and $x_0^{\Lambda-1} = \pi_0(u_0^{\Lambda-1}), z_0^{\Lambda-1} = \pi_1(u_0^{\Lambda-1})$ are vectors propagated on the next layer of CPT, where

$$\pi_0(u_0^i) = \begin{cases} u_{0,e}^i + u_{0,o}^i + (u_{2,e}^i, 0), & i \equiv 1 \bmod 2 \\ u_{0,e}^{i-1} + u_{0,o}^{i-1} + u_{2,e}^i, & i \equiv 0 \bmod 2 \end{cases} \quad (5)$$

$$\pi_1(u_0^i) = \begin{cases} u_{0,o}^i + (u_{2,e}^i, 0), & i \equiv 1 \bmod 2 \\ u_{0,o}^{i-1} + u_{2,e}^i, & i \equiv 0 \bmod 2 \end{cases} \quad (6)$$

Equality $\overset{(1)}{=}$ holds since $\mathcal{W}$ is memoryless and therefore conditional and event part can be permuted by R_Λ.

Using expansion (4), the recursive formula for $W_\lambda^{(2i+1)}(u_0^{2i+1}|y_0^{\Lambda-1})$ is derived in (7) at the top of the next page. Equality $\overset{(2)}{=}$ is obtained by change of variables $w = u_{2i+3} + u_{2i+4}$. Note that summation over $w = u_{2i+3} + u_{2i+4}$ reduces to summation over $w = u_{2i+3}$ for the case of $i = \Lambda/2 - 2$. Equality $\overset{(3)}{=}$ is obtained by replacing the sum over $u_{2i+4}^{\Lambda-1}$ with the product of sums over $x_{i+2}^{\Lambda/2-1}$ and $z_{i+2}^{\Lambda/2-1}$. This transformation is valid, since for each fixed pair of values $(x_{i+2}^{\Lambda/2-1}, z_{i+2}^{\Lambda/2-1}) = (\pi_0(u_{2i+4}^{\Lambda-1}), \pi_1(u_{2i+4}^{\Lambda-1}))$ there exists exactly one value of $u_{2i+4}^{\Lambda-1}$, which can be found by solving the system of linear equations with non-singular matrix. Thus, there is the one-to-one mapping between $(x_{i+2}^{\Lambda/2-1}, z_{i+2}^{\Lambda/2-1})$ and $u_{2i+4}^{\Lambda-1}$, and $\overset{(3)}{=}$ holds. In equality $\overset{(4)}{=}$, sums are replaced with (2).

One can proceed in a similar manner for $W_\lambda^{(2i)}$ and $W_\lambda^{(\Lambda-1)}$ and obtain

$$W_\lambda^{(2i)}(u_0^{2i}|y) = \sum_w W_{\lambda-1}^{(i)}(\pi_0(u_0^{2i}, w)|y') W_{\lambda-1}^{(i)}(\pi_1(u_0^{2i}, w)|y'') \quad (8)$$

$$\begin{aligned} W_\lambda^{(\Lambda-1)}(u_0^{\Lambda-1}|y) =& W_{\lambda-1}^{(\Lambda/2-1)}(\pi_0(u_0^{\Lambda-1})|y') \\ & \times W_{\lambda-1}^{(\Lambda/2-1)}(\pi_1(u_0^{\Lambda-1})|y''). \end{aligned} \quad (9)$$

The base of the recursion is $W_0^{(0)}(u_0|y_0) = W(u_0|y_0)$.

C. Probabilities calculation in the SC decoding algorithm

The SC decoding algorithm consists in successive calculation of $W_m^{(\phi)}(\hat{u}_0^{\phi-1}, u_\phi | y_0^{n-1})$ and making hard decisions according to (3) for $\phi = 0 \ldots n-1$. By re-using intermediate values in the recursive formulae (7)–(9), one can obtain an implementation of the SC algorithm with complexity $\Theta(n \log n)$, similarly to the case of Arıkan polar codes. However, since the CPT is substantially different from Arikan transformation, some changes in data structures are needed.

Observe that computing $W_\lambda^{(2i+1)}(u_0^{2i+1}|y_0^{\Lambda-1})$ for each $u_{2i+1} \in \mathbb{F}_2$ requires one to know values of $W_{\lambda-1}^{(i+1)}\left(x_0^{i+1}|y'\right)$ and $W_{\lambda-1}^{(i+1)}\left(z_0^{i+1}|y''\right)$. At the $(2i+1)$-th decoding phase on layer λ only values of u_0^{2i} are available, hence only x_0^{i-1} and z_0^{i-1} are known, and one should enumerate possible values of x_i^{i+1} and z_i^{i+1}. Since expressions (5)–(6) for $\pi_s(u_0^{2i+1}), s = 0, 1$, involve fixed values $u_{0,e}^{2i+1}$, only 8 combinations out of 16 need to be considered. Thus, one has to compute probabilities $W_{\lambda-1}^{(i+1)}\left(\pi_0(u_0^{2i}, a, b, c)|y'\right)$ and $W_{\lambda-1}^{(i+1)}\left(\pi_1(u_0^{2i}, a, b, c)|y''\right)$ for all 8 possible values of triple (a, b, c). For any $t \in \mathbb{N}$, a tuple of 2^t probabilities indexed by a vector of t bits is referred to as a t-cluster. So, in the case of CPC one needs to obtain 3-clusters, while for Arikan polar codes it is sufficient to obtain 1-clusters.

By examining expressions (7)–(9), one can see that computing 3-clusters $W_{\lambda-1}^{(i+1)}(\pi_s(u_0^{2i}, a, b, c)|y)$ does *not* require one to compute t-clusters for $t > 3$ on layers $\lambda' < \lambda - 1$. In some cases one can simplify calculations by computing 2-clusters instead of 3-clusters. For example, the output of $W_1^1(u_0^1|y_0^1)$ is a 2-cluster, containing 4 probabilities for each u_0^1.

Fig. 2 shows all patterns of cluster evolution. In general, in each pattern two input clusters P and Q on layer $\lambda - 1$ are processed and the output cluster $R^{(J)}, 1 \le J \le 6$ is calculated on layer λ, using already known hard decisions denoted by $\hat{u}_j$, and performing summation (marginalization) over not yet known symbols, denoted by Σ. Here P, Q and $R^{(J)}$ are 2 or 3-clusters that are depicted as "boxes" with 2 or 3 inputs corresponding to the value of each bit of the pair or the triple. The values $\hat{u}_i$ are either derived from those on layer $\lambda + 1$, or explicitly given by the decisions (3).

In Fig. 2a $P[x_0, x_1] = W_1^{(1)}(x_0^1|y')$, $Q[z_0, z_1] = W_1^{(1)}(z_0^1|y'')$, and $R^{(1)}[a, b, c] = W_2^{(2)}(a, b, c|y)$. From (8) one obtains

$$R^{(1)}[a, b, c] = \sum_{w \in \mathbb{F}_2} P[a + b + c, c + w] Q[b + c, w]. \quad (10)$$

In Fig. 2b, inputs P and Q are 3-clusters. Here $R^{(2)}[a, b, c] = W_\lambda^{(2)}(a, b, c|y)$, $\lambda > 2$, $P[x_0, x_1, x_2] = W_{\lambda-1}^{(2)}(x_0^2|y')$, $Q[z_0, z_1, z_2] = W_{\lambda-1}^{(2)}(z_0^2|y'')$, and

$$R^{(2)}[a, b, c] = \sum_{w \in \mathbb{F}_2} \sum_{z \in \mathbb{F}_2} P[a + b + c, c + w, z] \cdot \sum_{z \in \mathbb{F}_2} Q[b + c, w, z]. \quad (11)$$

$$W_\lambda^{(2i+1)}(u_0^{2i+1}|y_0^{\Lambda-1}) = \sum_{u_{2i+2}^{\Lambda-1}} W^{\Lambda/2}\left(\pi_0(u_0^{\Lambda-1})Q_{\Lambda/2}|y'\right) W^{\Lambda/2}\left(\pi_1(u_0^{\Lambda-1})Q_{\Lambda/2}|y''\right)$$

$$\stackrel{(2)}{=} \sum_{u_{2i+2},w} \sum_{u_{2i+4}^{\Lambda-1}} W^{\Lambda/2}\left(\left[\pi_0(u_0^{2i+2},w),\pi_0(u_{2i+4}^{\Lambda-1})\right] Q_{\Lambda/2}|y'\right) W^{\Lambda/2}\left(\left[\pi_1(u_0^{2i+2},w),\pi_1(u_{2i+4}^{\Lambda-1})\right] Q_{\Lambda/2}|y''\right)$$

$$\stackrel{(3)}{=} \sum_{u_{2i+2},w} \sum_{x_{i+2}^{\Lambda/2-1}} W^{\Lambda/2}\left(\left(\pi_0(u_0^{2i+2},w),x_{i+2}^{\Lambda/2-1}\right) Q_{\Lambda/2}|y'\right) \cdot \sum_{z_{i+2}^{\Lambda/2-1}} W^{\Lambda/2}\left(\left(\pi_1(u_0^{2i+2},w),z_{i+2}^{\Lambda/2-1}\right) Q_{\Lambda/2}|y''\right)$$

$$\stackrel{(4)}{=} \sum_{u_{2i+2},w} W_{\lambda-1}^{(i+1)}\left(\pi_0(u_0^{2i+2},w)|y'\right) W_{\lambda-1}^{(i+1)}\left(\pi_1(u_0^{2i+2},w)|y''\right), 0 \le i < \Lambda/2-1 \quad (7)$$

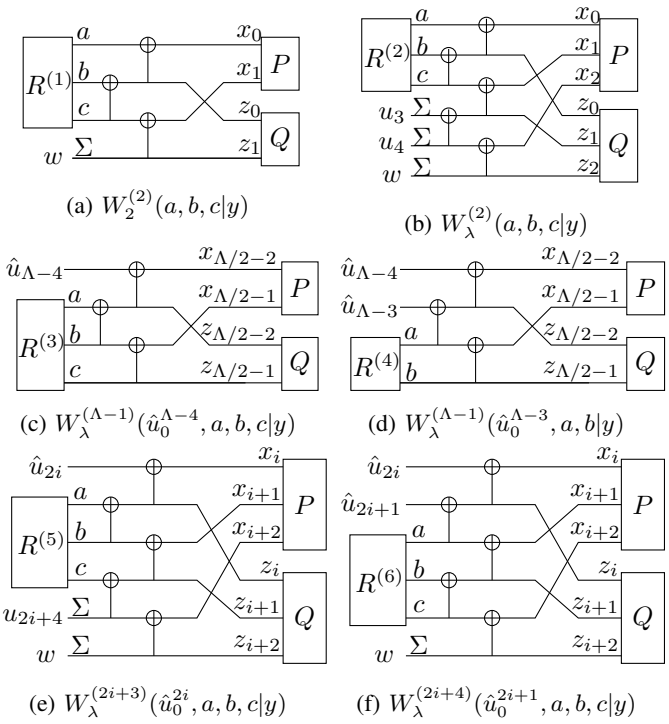

Fig. 2: Patterns which occur in the expansion of CPT

In Fig. 2c, one has $R^{(3)}[a,b,c] = W_\lambda^{(\Lambda-1)}(\hat{u}_0^{\Lambda-4},a,b,c|y)$, $P[x_{\Lambda/2-2},x_{\Lambda/2-1}] = W_{\lambda-1}^{(\Lambda/2-1)}(\hat{x}_0^{\Lambda/2-3},x_{\Lambda/2-2}^{\Lambda/2-1}|y')$, $Q[z_{\Lambda/2-2},z_{\Lambda/2-1}] = W_{\lambda-1}^{(\Lambda/2-1)}(\hat{z}_0^{\Lambda/2-3},z_{\Lambda/2-2}^{\Lambda/2-1}|y'')$, and (9) implies that the transformation is given by

$$R^{(3)}[a,b,c] = P[\hat{u}_{\Lambda-4}+a+b,b+c]Q[a+b,c]. \quad (12)$$

Fig. 2d provides another implementation of the pattern shown in Fig. 2c for the case of one more known value $\hat{u}_{\Lambda-3}$. Here one has

$$\begin{aligned} R^{(4)}[a,b] &= R^{(3)}[\hat{u}_{\Lambda-3},a,b] \\ &= P[\hat{u}_{\Lambda-4}+\hat{u}_{\Lambda-3}+a,a+b]Q[\hat{u}_{\Lambda-3}+a,b]. \end{aligned} \quad (13)$$

In Fig. 2e one needs to marginalize over u_{2i+4} and $w = u_{2i+5} + u_{2i+6}$. Hence, one obtains $R^{(5)}[a,b,c] = W_\lambda^{(2i+3)}(\hat{u}_0^{2i},a,b,c|y)$, $P[x_i,x_{i+1},x_{i+2}] = W_{\lambda-1}^{(i+2)}(\pi_0(\hat{u}_0^{2i}),x_i^{i+2}|y')$, $Q[z_i,z_{i+1},z_{i+2}] = W_{\lambda-1}^{(i+2)}(\pi_1(\hat{u}_0^{2i}),z_i^{i+2}|y'')$, and the transformation is given by

$$\begin{aligned} R^{(5)}[a,b,c] = \sum_{u_{2i+4},w} & P[\hat{u}_{2i}+a+b,b+c+u_{2i+4},u_{2i+4}+w] \\ & \times Q[a+b,c+u_{2i+4},w]. \end{aligned} \quad (14)$$

In Fig. 2f one has $R^{(6)}[a,b,c] = W_\lambda^{(2i+4)}(\hat{u}_0^{2i+1},a,b,c|y)$, P and Q are defined similarly to the previous case, and the transformation is given by

$$\begin{aligned} R^{(6)}[a,b,c] = \sum_{w} & P[\hat{u}_{2i}+\hat{u}_{2i+1}+a,a+b+c,c+w] \\ & \times Q[\hat{u}_{2i+1}+a,b+c,w]. \end{aligned} \quad (15)$$

An implementation of the SC algorithm based on the above expressions for $R^{(6)}[a,b,c]$ may suffer from severe numeric problems, which can be avoided by normalization of the probability clusters so that the sum of probabilities in a cluster is equal to 1. However, an LLR-domain implementation is presented below, which is much more numerically stable and does not require any multiplications.

IV. Efficient SC Decoding

Here we present a numerically stable log-domain implementation of the SC algorithm, which requires memory of size $\Theta(n)$.

A. Moving to log-domain

We propose to use min-sum approximations of formulae (10)–(15) as follows. Define log-likelihood of bit u_ϕ as

$$\widetilde{L}_m^{(\phi)}(u_0^\phi|y_0^{n-1}) = \ln \max_{u_{\phi+1}^{n-1}\in\mathbb{F}_2^{n-\phi-1}} W^n(u_0^{n-1}Q_n|y_0^{n-1}). \quad (16)$$

A 2-cluster of log-likelihood ratios (LLRs) is defined as $\widetilde{R}[a,b] = \widetilde{L}_\lambda^{(\phi)}(\hat{u}_0^{\phi-2},0,0|y) - \widetilde{L}_\lambda^{(\phi)}(\hat{u}_0^{\phi-2},a,b|y)$. A 3-cluster of LLRs is defined as $\widetilde{R}[a,b,c] = \widetilde{L}_\lambda^{(\phi)}(\hat{u}_0^{\phi-3},0,0,0|y) - \widetilde{L}_\lambda^{(\phi)}(\hat{u}_0^{\phi-3},a,b,c|y)$. The expressions for $\widetilde{R}^{(J)}$ can be obtained by replacing all sums with minima and all products with sums in (10)–(15):

$$\widetilde{R}^{(1)}[a,b,c] = -\min_w \left(\widetilde{P}[0,w] + \widetilde{Q}[0,w]\right)$$

$$+\min_{w}\Big(\widetilde{P}[a+b+c,c+w]+\widetilde{Q}[b+c,w]\Big) \quad (17)$$

$$\widetilde{R}^{(2)}[a,b,c]=-\min_{w}\Big(\min_{z}\widetilde{P}[0,w,z]+\min_{z}\widetilde{Q}[0,w,z]\Big)$$
$$+\min_{w}\Big(\min_{z}\widetilde{P}[a+b+c,c+w,z]+\min_{z}\widetilde{Q}[b+c,w,z]\Big) \quad (18)$$

$$\widetilde{R}^{(3)}[a,b,c]=-\widetilde{P}[\tilde{u}_0,0]+\widetilde{P}[\tilde{u}_0+a+b,b+c]+\widetilde{Q}[a+b,c] \quad (19)$$

$$\widetilde{R}^{(4)}[a,b]=-\widetilde{P}[\tilde{u}_0+\tilde{u}_1,0]-\widetilde{Q}[\tilde{u}_1,0]$$
$$+\widetilde{P}[\tilde{u}_0+\tilde{u}_1+a,a+b]+\widetilde{Q}[\tilde{u}_1+a,b] \quad (20)$$

$$\widetilde{R}^{(5)}[a,b,c]=-\min_{w,z}\Big(\widetilde{P}[\tilde{u}_0,w,w+z]+\widetilde{Q}[0,w,z]\Big)$$
$$+\min_{w,z}\Big(\widetilde{P}[\tilde{u}_0+a+b,b+c+w,w+z]+\widetilde{Q}[a+b,c+w,z]\Big) \quad (21)$$

$$\widetilde{R}^{(6)}[a,b,c]=-\min_{w}\Big(\widetilde{P}[\tilde{u}_0+\tilde{u}_1,0,w]+\widetilde{Q}[\tilde{u}_1,0,w]\Big)$$
$$+\min_{w}\Big(\widetilde{P}[\tilde{u}_0+\tilde{u}_1+a,a+b+c,c+w]+\widetilde{Q}[\tilde{u}_1+a,b+c,w]\Big) \quad (22)$$

The LLR-domain SC decoder computes an LLR for u_i as

$$S^{(\phi)}(\hat{u}_0^{\phi-1}|y)=\widetilde{L}_m^{(\phi)}(\hat{u}_0^{\phi-1},0|y)-\widetilde{L}_m^{(\phi)}(\hat{u}_0^{\phi-1},1|y). \quad (23)$$

After one obtains a 3-cluster $\widetilde{R}[u_\phi^{\phi+2}]$ on layer m using the recursive formulae (17)–(22), it can be converted to LLRs $\mathcal{L}_i(\widetilde{R},\hat{u}_\phi^{\phi+i-1})=S^{(\phi+i)}(\hat{u}_0^{\phi+i-1}|y)$, $0\le i\le 2$, defined in (23), as

$$\mathcal{L}_0(\widetilde{R})=\min_{b,c\in\mathbb{F}_2}\widetilde{R}[1,b,c]-\min_{b,c\in\mathbb{F}_2}\widetilde{R}[0,b,c] \quad (24)$$

$$\mathcal{L}_1(\widetilde{R},\hat{u}_\phi)=\min_{c\in\mathbb{F}_2}\widetilde{R}[\hat{u}_\phi,1,c]-\min_{c\in\mathbb{F}_2}\widetilde{R}[\hat{u}_\phi,0,c] \quad (25)$$

$$\mathcal{L}_2(\widetilde{R},\hat{u}_\phi,\hat{u}_{\phi+1})=\widetilde{R}[\hat{u}_\phi,\hat{u}_{\phi+1},1]-\widetilde{R}[\hat{u}_\phi,\hat{u}_{\phi+1},0] \quad (26)$$

Hence, hard decisions on u_ϕ are successively obtained as

$$\hat{u}_\phi=\begin{cases}0, & i\in\mathcal{F}\ \vee\ S^{(\phi)}(\hat{u}_0^{\phi-1}|y)>0\\ 1, & \text{otherwise.}\end{cases} \quad (27)$$

B. Memory-Efficient SC Decoding

In Alg. 2 the head function of log-domain SC decoding algorithm is presented. The inputs are: logarithm of code length m, noisy output $y_0^{2^m-1}$ of the channel, the set of information subchannels $\mathcal{I}=\{\mathcal{I}_i\}_{i=0}^{k-1}$, and function $W(x|y)$.

Algorithm 2: `DecodeSC`$(m, y_0^{n-1}, \mathcal{I}, W)$

2.1 $n\leftarrow 2^m$
2.2 **for** $i\leftarrow 0\dots n/2-1$ **do**
2.3 **for** $(a,b)\in\mathbb{F}_2^2$ **do**
2.4 $T_i^{(1)}[a,b]\leftarrow \ln\frac{W(0|y_i)W(0|y_{i+n/2})}{W(a+b|y_i)W(b|y_{i+n/2})}$
2.5 **for** $\xi\leftarrow 0\dots n-1$ **do**
2.6 $S^{(\xi)}(\hat{u}_0^{\xi-1}|y_0^{n-1})\leftarrow$ `CalcLLR`(m,ξ,T,C)
2.7 $C_{\xi \bmod 2}^{(m)}\leftarrow \mathbf{1}[\xi\in\mathcal{I}\ \wedge\ S^{(\xi)}(\hat{u}_0^{\xi-1}|y_0^{n-1})<0]$
2.8 `UpdateC`(m,ξ,C)
2.9 $B\leftarrow C_{0\dots n-1}^{(0)}Q_n^{-1}$
2.10 **return** $B_{\mathcal{I}}$

Variable T consists of m layers $T^{(\lambda)}, \lambda=1\dots m$, where each layer is an array of $2^{m-\lambda}$ LLR clusters. In lines 2.2–2.4 the first layer $T^{(1)}$ is initialized with LLRs of $W^2(a+b,b|y_i,y_{i+n/2})$ for each a,b. In line 2.6 LLR of u_ϕ is computed, given by (23). In line 2.7 the hard decision is computed, given by (27). Here, $\mathbf{1}$[statement] equals 1 if the statement is true, and 0 otherwise. Array C contains layers $C^{(\lambda)}$ for $\lambda=0\dots m$, each layer λ contains $2^{m-\lambda+1}$ hard decisions on corresponding layer of CPT. In line 2.8 the decision $\hat{u}_\phi$ made on layer m is propagated to previous layers via procedure `UpdateC`, presented in Alg. 3.

Algorithm 3: `UpdateC`(m,ξ,C)

3.1 $\lambda\leftarrow m,\ n\leftarrow 2^m,\ B\leftarrow 1,\ \Lambda\leftarrow n,\ \phi\leftarrow\xi$
3.2 **while** $\phi\neq 0\wedge\ \lambda\neq 0$ **do**
3.3 $\psi\leftarrow\lfloor\frac{\phi-1}{2}\rfloor,\ b\leftarrow\psi \bmod 2$
3.4 **if** $\phi\equiv 1 \bmod 2$ **then**
3.5 **for** $i\leftarrow 0\dots B-1$ **do**
3.6 **if** $\lambda=1$ **then**
3.7 $C_i^{(\lambda-1)}\leftarrow C_{2i}^{(\lambda)}+C_{2i+1}^{(\lambda)},\ C_{i+B}^{(\lambda-1)}\leftarrow C_{2i+1}^{(\lambda)}$
3.8 **else**
3.9 $C_{2i+b}^{(\lambda-1)}\leftarrow C_{2i}^{(\lambda)}+C_{2i+1}^{(\lambda)}, C_{2i+b+2B}^{(\lambda-1)}\leftarrow C_{2i+1}^{(\lambda)}$
3.10 **if** $\phi\neq\Lambda-1$ **then break**
3.11 **else**
3.12 **for** $i\leftarrow 0\dots B-1$ **do**
3.13 $C_{2i+b}^{(\lambda-1)}\leftarrow C_{2i+b}^{(\lambda-1)}+C_{2i}^{(\lambda)}$
3.14 $C_{2i+b+2B}^{(\lambda-1)}\leftarrow C_{2i+b+2B}^{(\lambda-1)}+C_{2i}^{(\lambda)}$
3.15 $\lambda\leftarrow\lambda-1,\ \phi\leftarrow\psi,\ B\leftarrow 2B,\ \Lambda\leftarrow\Lambda/2$

Algorithm 4: `CalcLLR`(m,ξ,T,C)

Output: LLR $S^{(\xi)}(\hat{u}_0^{\xi-1}|y)$
4.1 $n\leftarrow 2^m,\ \phi\leftarrow\xi$
4.2 **if** $\phi=n-2$ **then return** $\mathcal{L}_1(T_0^{(m)},C_1^{(m)})$
4.3 **if** $\phi=n-1$ **then**
4.4 **return** $\mathcal{L}_2(T_0^{(m)},C_2^{(m-1)}+C_0^{(m)},C_0^{(m)})$
4.5 **if** $\phi=0$ **then** $\mu\leftarrow 2$ **else** $\mu\leftarrow m$
4.6 **while** $(\phi\equiv 1 \bmod 2)\wedge(\mu>2)\wedge(\phi\neq 1)$ **do**
4.7 $\phi\leftarrow\lfloor\phi/2\rfloor,\ \mu\leftarrow\mu-1$
4.8 **for** $\lambda\leftarrow\mu\dots m$ **do**
4.9 $J\leftarrow$ `ChooseTransform`(ϕ,λ)
4.10 **for** $i\leftarrow 0\dots 2^{m-\lambda}-1$ **do**
4.11 $T_i^{(\lambda)}\leftarrow\widetilde{R}^{(J)}\left(T_i^{(\lambda-1)},T_{i+2^{m-\lambda}}^{(\lambda-1)},C_{2i}^{(\lambda)},C_{2i+1}^{(\lambda)}\right)$
4.12 **if** $\phi\neq 0$ **then** $\phi\leftarrow 2\phi+1$
4.13 **return** $\mathcal{L}_0(T_0^{(m)})$

Here, the inputs are logarithm of code length, index ξ of newly obtained u_ξ, and a pointer to the C array. B is equal to the number of CPTs on current layer λ, Λ is the size of

Algorithm 5: `ChooseTransform`(ϕ, λ)

5.1 **if** $\lambda = 2$ *and* $\phi < 2$ **then return** $1 + 2\phi$
5.2 **if** $\phi = 2^\lambda - 2$ **then return** 4
5.3 **if** $\phi = 0$ **then return** 2
5.4 **return** $6 - (\phi \bmod 2)$

TABLE I: Complexity of cluster operations

Op.	$\mathcal{L}_0$	$\widetilde{R}^{(1)}$	$\widetilde{R}^{(2)}$	$\widetilde{R}^{(3)}$	$\widetilde{R}^{(4)}$	$\widetilde{R}^{(5)}$	$\widetilde{R}^{(6)}$
$+$	1	16	23	7	7	38	23
$<>$	5	8	40	0	0	24	8
All	$\mathcal{C}_0 = 6$	$\mathcal{C}_1 = 24$	$\mathcal{C}_2 = 63$	$\mathcal{C}_3 = 7$	$\mathcal{C}_4 = 7$	$\mathcal{C}_5 = 62$	$\mathcal{C}_6 = 31$

these CPTs. Variable ϕ equals the local phase on layer λ, i.e., the index of newly obtained input bit on this layer. Variable ψ is the local phase on layer $\lambda - 1$. In line 3.15 these variables are updated for the layer $\lambda - 1$ which is processed at the next iteration of the loop.

Memory-efficient decision propagation works as follows. Recall (see (4)–(6)) that propagation to layer $\lambda - 1$ from layer λ is given by $x_\psi = u_{2\psi} + u_{2\psi+1} + u_{2\psi+2}$, $z_\psi = u_{2\psi+1} + u_{2\psi+2}$, where $u_{2\psi}$, $u_{2\psi+1}$, x_ψ and z_ψ are stored in $C_{2i}^{(\lambda)}$, $C_{2i+1}^{(\lambda)}$, $C_{2i+b}^{(\lambda-1)}$, $C_{2i+b+2B}^{(\lambda-1)}$, respectively. After computing $u_{2\psi+2}$, one obtains values of x_ψ and z_ψ. Thus, symbols on layer λ are propagated to layer $\lambda - 1$ if ϕ is non-zero and even, which is reflected in exit conditions of the main loop in line 3.2. For odd ϕ, the pre-summation operation is performed in lines 3.5–3.9 as $x'_\psi = u_{2\psi} + u_{2\psi+1}, z'_\psi = u_{2\psi+1}$. This enables one to overwrite $u_{2\psi}$ with $u_{2\psi+2}$ at the next phase. Finally, in lines 3.12–3.14 one obtains $x_\psi = x'_\psi + u_{2\psi+2}$ and $z_\psi = z'_\psi + u_{2\psi+2}$. Thus, for layer λ only $2^{m-\lambda+1}$ hard decisions are stored, similarly to [3]. The required amount of memory is $n + \sum_{\lambda=1}^{m} 2^{m-\lambda+1} = 3n - 2 = \Theta(n)$.

LLR clusters evolution is implemented in function `CalcLLR`, presented in Alg. 4. It returns LLR of u_ξ, defined in (23). In lines 4.5–4.12 3-cluster $T_0^{(m)} = \widetilde{L}_m^{(\xi+2)}(\hat{u}_0^{\xi-1}, a, b, c|y)$ is computed by recursion (17)–(22). For $\xi \geq n - 2$ the needed LLR is obtained by marginalization (25) or (26) (lines 4.2–4.4). In lines 4.5–4.7 the bottom layer μ to be updated at current phase is identified. Expressions (7)–(8) imply that for computing $\widetilde{L}_\lambda^{(2i+2)}$ the layer $\lambda - 1$ can be reused, since it is the same as for computing $\widetilde{L}_\lambda^{(2i+1)}$. Thus, the layer $\lambda - 1$ is updated only if local phase ϕ is odd. Also, for $\phi = 1$ the layer $\lambda - 1$ is not updated, because after computing $\hat{u}_0$ on layer λ no symbol is yet known on layer $\lambda - 1$. These conditions are reflected in line 4.6. In line 4.9 the appropriate transformation $\widetilde{R}^{(J)}$ is chosen via function `ChooseTransform`, shown in Alg. 5. In lines 4.10–4.11 this transformation is applied to LLR clusters on layer $\lambda - 1$ to obtain LLR clusters on layer λ.

In Table I the complexity of cluster operations is presented. The overall complexity of SC decoding is $\mathcal{C}(n) \approx \frac{n}{2} + n\mathcal{C}_0 + \sum_{\lambda=3}^{m} 2^{m-\lambda} \cdot \big(\mathcal{C}_2 + \mathcal{C}_3 + \mathcal{C}_4 + (\mathcal{C}_5 + \mathcal{C}_6) \cdot (2^{\lambda-1} - 2)\big) + \frac{n}{4}(\mathcal{C}_1 + \mathcal{C}_3 + \mathcal{C}_4) \approx 46.5mn - 104.25n + 109$. The equality is approximate because cluster computation at layer $\lambda = m$ can be simplified, and in lines 4.2–4.4 of Alg. 4 functions $\mathcal{L}_1$ and $\mathcal{L}_2$ are used instead of $\mathcal{L}_0$. The complexity of SC decoding of classical polar codes is mn.

Following the ideas presented in [3], the proposed approach can be easily extended to the case of list decoding.

V. Numerical Results

In Fig. 3 the performance of CPC and CPC with cyclic redundancy check (CRC) under SC and list SC decoding for AWGN channel is provided. For comparison, we report also the results for polar codes [3] and randomized polar subcodes [4]. Monte-Carlo method was used to find the set $\mathcal{F}$ for CPCs. Note that both the probability-domain (prob.) and the min-sum SC decoding algorithms have the same performance. Th CPC under SC decoding provides a huge performance gain compared to SC decoding of the polar code. The CPC under CRC-aided list decoding with $L = 4$ performs better then both polar codes under CRC-aided list decoding and randomized polar subcodes under list decoding with the same list size.

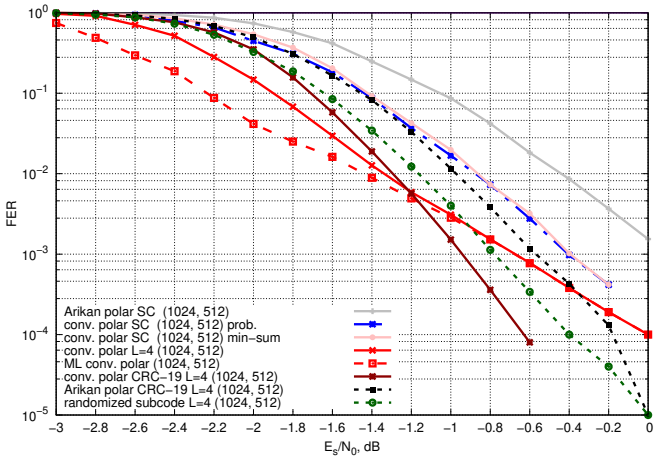

Fig. 3: FER of SC and list decoding of $(1024, 512)$ CPC

VI. Conclusions

In this paper numerically stable SC decoding algorithm for convolutional polar codes was presented, which employs only addition and comparison operations, and admits extension to list SC decoding. CPCs were shown to outperform both polar codes and subcodes.

References

[1] A. J. Ferris, C. Hirche, and D. Poulin, "Convolutional polar codes," *CoRR*, vol. abs/1704.00715, 2017. [Online]. Available: http://arxiv.org/abs/1704.00715

[2] 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.

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

[4] P. Trifonov and G. Trofimiuk, "A randomized construction of polar subcodes," in *Proceedings of IEEE International Symposium on Information Theory*. Aachen, Germany: IEEE, 2017, pp. 1863–1867.