Neural Belief Propagation Decoding of CRC-Polar Concatenated Codes

Nghia Doan, Seyyed Ali Hashemi, Elie Ngomseu Mambou, Thibaud Tonnellier, Warren J. Gross

Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada

Email: {nghia.doan, seyyed.hashemi, elie.ngomseumambou}@mail.mcgill.ca, {thibaud.tonnellier, warren.gross}@mcgill.ca

Abstract—Polar codes are the first class of error correcting codes that provably achieve the channel capacity at infinite code length. They were selected for use in the fifth generation of cellular mobile communications (5G). In practical scenarios such as 5G, a cyclic redundancy check (CRC) is concatenated with polar codes to improve their finite length performance. This is mostly beneficial for sequential successive-cancellation list decoders. However, for parallel iterative belief propagation (BP) decoders, CRC is only used as an early stopping criterion with incremental error-correction performance improvement. In this paper, we first propose a CRC-polar BP (CPBP) decoder by exchanging the extrinsic information between the factor graph of the polar code and that of the CRC. We then propose a neural CPBP (NCPBP) algorithm which improves the CPBP decoder by introducing trainable normalizing weights on the concatenated factor graph. Our results on a 5G polar code of length 128 show that at the frame error rate of 10^{-5} and with a maximum of 30 iterations, the error-correction performance of CPBP and NCPBP are approximately 0.25 dB and 0.5 dB better than that of the conventional CRC-aided BP decoder, respectively, while introducing almost no latency overhead.

Index Terms—polar codes, 5G, neural belief propagation decoding, code concatenation, cyclic redundancy check.

I. INTRODUCTION

Polar codes are a breakthrough in the field of channel coding as they were proved to achieve channel capacity with efficient encoding and decoding algorithms [1]. Successive cancellation (SC) and belief propagation (BP) decoding algorithms are introduced in [1] to decode polar codes. Although SC decoding can provide a low-complexity implementation, its serial nature prevents the decoder to reach a high decoding throughput. In addition, the error-correction performance of SC decoding for short to moderate polar codes does not satisfy the requirements of the fifth generation of cellular mobile communications (5G). To improve the performance of SC decoding, SC list (SCL) decoding was introduced in [2] and it was shown that SCL can provide a vast error-correction performance improvement if it is aided by a cyclic redundancy check (CRC). Based on this observation, polar codes have been selected to be used in the enhanced mobile broadband (eMBB) control channel of 5G together with a CRC [3].

Unlike SC-based decoders, the iterative message passing process of BP decoding can be executed in parallel, hence enabling the decoder to reach high decoding throughput. However, with limited number of iterations BP decoding suffers from poor error-correction performance. Furthermore, 5G standard requires polar codes to be concatenated with an outer

CRC code. Thus, several attempts have been carried out to improve the performance of BP decoding for polar codes using a CRC. In [4], CRC is used as an early termination criterion to prevent the BP decoder from processing unnecessary iterations when the correct codeword is found. In [5], a post-processing algorithm is presented which uses a CRC to detect false-converged errors. In addition, it is observed in [6], [7] that BP decoding on a list of factor graph permutations of polar codes can benefit from CRC to achieve lower error probabilities than when no CRC is used. However, BP decoding in all of the aforementioned works is only applied on the factor graph of polar codes and the CRC is only used to verify the result of BP at each iteration, without exploiting the inherent factor graph of CRC. Note that BP decoding is used in [8]–[10] on the concatenated factor graphs of a low-density parity-check (LDPC) code and a polar code. However, this scheme is not selected for use in the eMBB control channel of 5G.

In this paper, we first show that by running BP decoding on the CRC-polar concatenated factor graph, significant error-correction performance improvement can be achieved in comparison with the conventional CRC-aided BP decoder. We call the proposed decoding method CRC-polar BP (CPBP). We then propose a neural CPBP (NCPBP) decoder to further improve the error-correction performance of CPBP with limited number of BP iterations. We devise an efficient weight-assignment scheme for NCPBP and show that the proposed scheme has fewer weights than the state-of-the-art neural BP decoder of [11] while providing a better error-correction performance. The proposed decoders are evaluated on a 5G polar code of length 128. At the frame error rate (FER) of 10^{-5} and with a maximum number of 30 iterations, we show that the error-correction performance of the CPBP and NCPBP decoders are approximately 0.25 dB and 0.5 dB better than that of the conventional CRC-aided BP decoder, respectively, with negligible latency overhead.

The rest of this paper is organized as follows. Section II briefly introduces polar codes and its BP-based decoders. Section III and IV describe the proposed CPBP and NCPBP decoders, respectively. Finally, concluding remarks are drawn in Section V.

II. PRELIMINARIES

A. Polar Codes

A polar code $\mathcal{P}(N, K)$ of length N with K information bits is constructed by applying a linear transformation to the
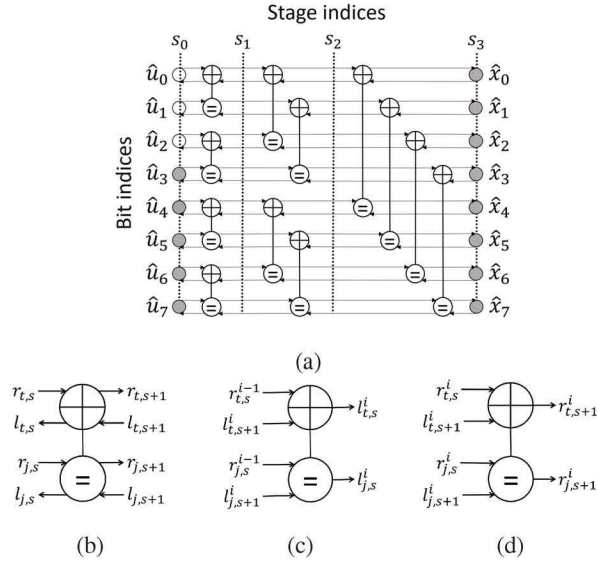
Fig. 1: (a) BP decoding on the factor graph of $\mathcal{P}(8,5)$ with $\{u_0, u_1, u_2\} \in \mathcal{A}^c$, (b) a PE, (c) a right-to-left message update of a PE on an unrolled factor graph, and (d) a left-to-right message update of a PE on an unrolled factor graph.

message word $\boldsymbol{u} = \{u_0, u_1, \ldots, u_{N-1}\}$ as $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}^{\otimes n}$ where $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ is the codeword, $\boldsymbol{G}^{\otimes n}$ is the n-th Kronecker power of the polarizing matrix $\boldsymbol{G} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$, and $n = \log_2 N$. The vector $\boldsymbol{u}$ contains a set $\mathcal{A}$ of K information bits and a set $\mathcal{A}^c$ of $N - K$ frozen bits. The positions of the frozen bits are known to the encoder and the decoder and their values are known to be 0. The codeword $\boldsymbol{x}$ is then modulated and sent through the channel. In this paper, binary phase-shift keying (BPSK) modulation and additive white Gaussian noise (AWGN) channel model are considered, therefore, the soft vector of the transmitted codeword received by the decoder is written as

$$\boldsymbol{y} = (\mathbf{1} - 2\boldsymbol{x}) + \boldsymbol{z}, \tag{1}$$

where $\mathbf{1}$ is an all-one vector of size N, and $\boldsymbol{z} \in \mathbb{R}^N$ is the AWGN noise vector with variance σ^2 and zero mean. In the log-likelihood ratio (LLR) domain, the LLR vector of the transmitted codeword is

$$\mathbf{LLR}_{\boldsymbol{x}} = \ln \frac{\Pr(\boldsymbol{x} = 0|\boldsymbol{y})}{\Pr(\boldsymbol{x} = 1|\boldsymbol{y})} = \frac{2\boldsymbol{y}}{\sigma^2}. \tag{2}$$

B. Belief Propagation Decoding of Polar Codes

Fig. 1a illustrates BP decoding on a factor graph representation of $\mathcal{P}(8,5)$. The messages are iteratively propagated through the processing elements (PEs) [12] located in each stage. An update iteration starts with a right-to-left message pass that propagates the LLR values from the channel (rightmost) stage, to the information bit (leftmost) stage, and ends with the left-to-right message pass which occurs in the reverse order. Fig. 1b shows a PE with its corresponding messages, where $r_{t,s}$ denotes a left-to-right message, and $l_{t,s}$ denotes a right-to-left message of the t-th bit index at stage s. Equivalently, BP decoding of polar codes can be

represented on an unrolled factor graph, in which BP iterations are performed sequentially. Fig. 1c and Fig. 1d illustrate the input and output messages of a PE for the right-to-left and left-to-right message updates on an unrolled factor graph, where the superscript i denotes the iteration number. The update rule [12] for the right-to-left messages of a PE is

$$\begin{cases} l_{t,s}^i &= f(l_{t,k}^i, r_{j,s}^{i-1} + l_{j,k}^i), \\ l_{j,s}^i &= f(l_{t,k}^i, r_{t,s}^{i-1}) + l_{j,k}^i, \end{cases} \tag{3}$$

and for the left-to-right messages is

$$\begin{cases} r_{t,k}^i &= f(r_{t,s}^i, l_{j,k}^i + r_{j,s}^i), \\ r_{j,k}^i &= f(r_{t,s}^i, l_{t,k}^i) + r_{j,s}^i, \end{cases} \tag{4}$$

where $j = t + 2^s$, $k = s + 1$, and

$$f(x,y) = 2\arctanh\left(\tanh\left(\frac{x}{2}\right)\tanh\left(\frac{y}{2}\right)\right), \tag{5}$$

for any $x, y \in \mathbb{R}$. Note that implementing (5) is costly in practice, instead the following approximation of (5) is used in this paper:

$$f(x,y) \approx \tilde{f}(x,y) = \text{sgn}(x)\,\text{sgn}(y)\min(|x|,|y|). \tag{6}$$

The BP decoding performs a predetermined $I_{\max}$ update iterations where the messages are propagated through all PEs in accordance with (3) and (4). Initially, for $0 \le t < N$ and $\forall i \le I_{\max}$, $l_{t,n}^i$ are set to the received channel LLR values $\mathbf{LLR}_{\boldsymbol{x}}$, $r_{t,0}^i$ are set to the LLR values of the information and frozen bits as

$$\mathbf{LLR}_{\mathcal{A} \cup \mathcal{A}^c} = \begin{cases} 0, & \text{if } t \in \mathcal{A}, \\ +\infty, & \text{if } t \in \mathcal{A}^c. \end{cases} \tag{7}$$

All the other left-to-right and right-to-left messages of the PEs at the first iteration are set to 0. After running $I_{\max}$ iterations, the decoder makes a hard decision on the LLR values of the t-th bit at the information bit stage to obtain the estimated message word as

$$\hat{u}_t = \begin{cases} 0, & \text{if } r_{t,0}^{I_{\max}} + l_{t,0}^{I_{\max}} \ge 0, \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

In the following sections the vector forms of the left-to-right and right-to-left messages at the s-th stage and the i-th iteration are denoted as $\boldsymbol{l}_s^i$ and $\boldsymbol{r}_s^i$, respectively, while that of the estimated message word is denoted as $\hat{\boldsymbol{u}}$.

A CRC is used for BP decoding to either early terminate the BP process [4], or to help select the correct codeword among a list of candidates [6], [7]. However, these CRC utilizations do not take into account the factor graph realization of CRC, on which the BP decoder can be applied.

C. Neural BP Decoding

Neural BP decoding was introduced in [11], [13] to improve the error-correction performance of BP decoding on Bose-Chaudhuri-Hocquenghem (BCH) codes by assigning trainable weights to the conventional BP decoding. Neural normalized min-sum recurrent neural network (NNMS-RNN) is a powerful variant of neural BP [11] with the following weight
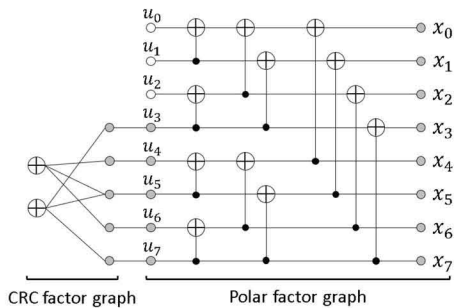
Fig. 2: Factor graph representation of a CRC-polar concatenated code. The polar code is $\mathcal{P}(8,3)$ and a 2-bit CRC is used.

assignment scheme for the message update rule of a PE in (3) and (4):

$$\begin{cases} l_{t,s}^i & = w_0 \tilde{f}(l_{t,k}^i, w_1 r_{j,s}^{i-1} + w_2 l_{j,k}^i), \\ l_{j,s}^i & = w_4(w_3 \tilde{f}(l_{t,k}^i, r_{t,s}^{i-1})) + w_5 l_{j,k}^i, \end{cases} \quad (9)$$

$$\begin{cases} r_{t,k}^i & = w_6 \tilde{f}(r_{t,s}^i, w_7 l_{j,k}^i + w_8 r_{j,s}^i), \\ r_{j,k}^i & = w_{10}(w_9 \tilde{f}(r_{t,s}^i, l_{t,k}^i)) + w_{11} r_{j,s}^i, \end{cases} \quad (10)$$

where $w_m \in \mathbb{R}$ ($0 \le m \le 11$) are the trainable weights.

The NNMS-RNN BP decoder suffers from a large number of weights which adversely affects its implementation cost. A neural normalized min-sum (NNMS) decoder was used to decode polar codes by only enabling the training for w_0, w_3, w_6 and w_9, while setting the other weights in (9) and (10) to 1 [14]. However, the error-correction performance improvement of [14] with respect to the conventional BP is incremental.

III. CRC-POLAR BP DECODING

In this section, we present the CPBP decoding algorithm which exploits the concatenated factor graph of a polar code and a CRC. Fig. 2 shows the concatenated factor graph of $\mathcal{P}(8,3)$ and a CRC of length 2. We run BP decoding algorithm on the concatenated factor graph to exploit the extrinsic information of the two constituent factor graphs. A similar approach was performed in [8] for a LDPC-polar concatenated code by passing the BP messages between the factor graphs of LDPC and polar code at each iteration. A direct application of the BP decoder in [8] to the CRC-polar concatenated code is not beneficial. This is due to the fact that the LDPC code is only connected to a few pre-selected information bits of polar codes which ensures the extrinsic information received by polar code is reliable enough, even in the initial iterations of BP decoding where the LLR values are not evolved yet. This is not the case for CRC-polar concatenated code since CRC is connected to all the information bits of polar code, some of which are highly unreliable during the early iterations of BP decoding.

In order to address the above issue, we first run BP decoding on the polar code factor graph for a maximum of I_{thr} iterations and if the BP decoding is not successful after

I_{thr} iterations, we then continue the BP decoder on the CRC-polar concatenated factor graph. In order to determine if the decoder has succeeded, we use the CRC at each iteration as an early stopping criterion. The proposed decoding algorithm is summarized in Algorithm 1. The LLR vectors l_s^i and r_s^i at all stages and iterations are initialized as explained in Section II-B. The BP_PolarLeft and BP_PolarRight functions compute (3) and (4) at all the bit indices to perform the polar right-to-left and left-to-right LLR updates, respectively. The estimated message word $\hat{u}$ is obtained at every iteration by making a hard decision based on l_0^i and r_0^i, which is done by executing (8) in the HardDecision function. A CRC is then applied on $\hat{u}$ and the decoding can be early terminated if the CRC is satisfied. After I_{thr} iterations, if the decoding is not terminated, BP decoding on the CRC-polar factor graph is carried out in the BP_CRC function. It is worth mentioning that BP decoding after I_{thr} iterations runs on the concatenated CRC-polar factor graph at every iteration.

Algorithm 1: CPBP Decoding Algorithm

Input : $I_{\max}, I_{\text{thr}}, n$
Output: $\hat{u}$

1 Initialize l_s^i, r_s^i ($1 \le i \le I_{max}, 0 \le s \le n$)
2 **for** $i \leftarrow 1$ **to** $I_{\max}$ **do**
3   **for** $s \leftarrow n-1$ **to** 0 **do**
4          $l_s^i \leftarrow$ BP_PolarLeft (l_{s+1}^i, r_s^{i-1})
5   **if** $i > I_{\text{thr}}$ **then**
6          $r_0^i \leftarrow$ BP_CRC (l_0^i)
7      $\hat{u} \leftarrow$ HardDecision $(r_0^i + l_0^i)$
8   **if** $\hat{u}$ *satisfies CRC* **then**
9   Terminate
10   **if** $i \le I_{\max} - 1$ **then**
11   **for** $s \leftarrow 1$ **to** $n-1$ **do**
12             $r_s^i \leftarrow$ BP_PolarRight (l_s^i, r_{s-1}^i)

13 **return** $\hat{u}$

Fig. 3 shows the FER performance of the proposed CPBP algorithm in comparison with the CRC-aided BP decoder of [4], for the $\mathcal{P}(128,80)$ concatenated with the 16-bit CRC, which is selected for 5G [3]. In this figure, we set $I_{\max} \in \{30, 200\}$ and we set $I_{\text{thr}} \in \{15, 30\}$ when $I_{\max} = 30$ and $I_{\text{thr}} \in \{0, 50, 100, 150, 200\}$ when $I_{\max} = 200$. We denote CPBP decoding with parameters $I_{\max}$ and I_{thr} as CPBP-$(I_{\max}, I_{\text{thr}})$. Note that CPBP-$(I_{\max}, I_{\max})$ is equivalent to the decoder in [4] and CPBP-$(I_{\max}, 0)$ is the direct application of the approach in [8]. It can be seen that, CPBP-(30,15) provides a gain of almost 0.25 dB in comparison with the CRC-aided BP decoder of [4] at the target FER of 10^{-5}. In addition, among the selected I_{thr} for $I_{\max} = 200$, CPBP-(200,50) provides the best error-correction performance at the target FER of 10^{-5}. Furthermore, CPBP-$(200,50)$ has an error-correction performance gain of about 0.75 dB at FER $= 10^{-5}$ in comparison with the CRC-aided BP decoder
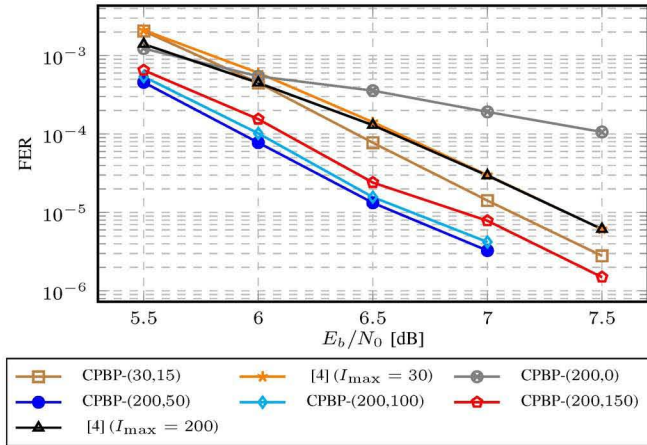
Fig. 3: FER performance of CPBP decoding for $\mathcal{P}(128, 80)$ and a 16-bit CRC used in 5G.

of [4]. It is worth mentioning that increasing $I_{\max}$ does not improve the error probabilities of the conventional BP decoder in [4] at high E_b/N_0 regime. On the contrary, the FER of the proposed CPBP decoder is greatly benefited from a high value of $I_{\max}$ as observed from Fig. 3.

We now evaluate the latency of the proposed CPBP decoding scheme and compare it with state-of-the-art. The latency of a BP-based decoder can be measured using the number of time steps required to finish the decoding process [12]. Let us consider the decoding process terminates at iteration I_{ET} $(1 \leq I_{\mathrm{ET}} \leq I_{\max})$. Then the decoding latency of a conventional BP decoder with early stopping criterion can be represented as

$$\mathcal{T}_{\mathrm{BP}} = (2n-1)(I_{\mathrm{ET}}-1) + n. \tag{11}$$

The latency of the proposed CPBP decoder depends on when the decoding process terminates and can be represented as

$$\mathcal{T}_{\mathrm{CPBP}} = \begin{cases} (2n-1)(I_{\mathrm{ET}}-1)+n, & \text{if } I_{\mathrm{ET}} \leq I_{\mathrm{thr}}, \\ (2n-1)(I_{\mathrm{ET}}-1)+n+2(I_{\mathrm{ET}}-I_{\mathrm{thr}}), & \text{otherwise.} \end{cases} \tag{12}$$

In fact, if $I_{\mathrm{ET}} \leq I_{\mathrm{thr}}$, (12) reverts to (11) since the CPBP decoder terminates without traversing the CRC factor graph. It should be noted that the worst case latency of the BP decoder and the proposed CPBP decoder can be calculated using (11) and (12) respectively, by setting $I_{\mathrm{ET}} = I_{\max}$.

Fig. 4 illustrates the average latency of the proposed CPBP decoding algorithm in comparison with a conventional CRC-aided BP decoder of [4] for the same code as in Fig. 3. For the proposed decoders, we set $I_{\mathrm{thr}} \in \{15, 30\}$ for $I_{\max} = 30$, and $I_{\mathrm{thr}} \in \{50, 100, 150, 200\}$ for $I_{\max} = 200$. It can be seen that the proposed CPBP algorithm incurs negligible latency overhead in comparison with [4], while providing significant performance gain. Moreover, the average latency of the CPBP decoder when $I_{\max} = 30$ is always smaller than that of the CPBP decoder when $I_{\max} = 200$. This average latency saving is more significant for lower E_b/N_0 values. Furthermore, the
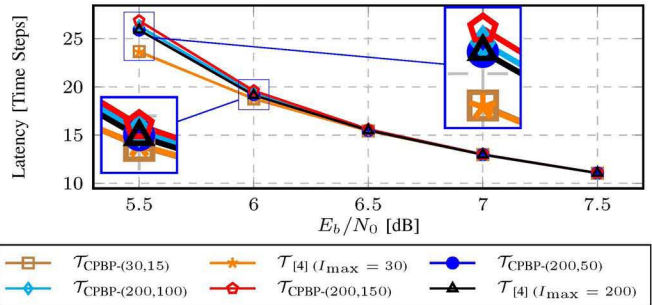

Fig. 4: Average decoding latency of CPBP decoding for $\mathcal{P}(128, 80)$ and a 16-bit CRC used in 5G.

worst case latency of CPBP(200,50) is 2887 time steps, and that of CPBP(30,15) is 407 time steps which is only 14% of the worst case latency of CPBP(200,50). For applications with stringent latency requirements, a small $I_{\max}$ is needed. However, the latency saving as a result of a small $I_{\max}$ comes at the cost of error-correction performance loss as shown in Fig. 3. In the next section, we propose a method to improve the error-correction performance of CPBP decoding for small values of $I_{\max}$, by using trainable weights.

IV. NEURAL CRC-POLAR BP DECODING

In this section, we propose the NCPBP decoder to improve the error-correction performance of CPBP decoding. The NCPBP decoder assigns trainable weights to the edges of the CRC-polar concatenated factor graph. Therefore, the NCPBP decoder resembles a neural network architecture by mapping the message updates of CPBP decoding to different computational layers in the neural network. In other words, each computational layer of the neural network is represented either as a set of PEs for BP decoding on the factor graph of polar codes, or as a set of operations required to perform BP decoding on the CRC factor graph. This network architecture greatly simplifies the training process since it can be adapted to recent deep learning frameworks, e.g. Tensorflow [15].

Fig. 5 depicts the architecture of the proposed NCPBP decoder for $\mathcal{P}(8, 3)$, with $I_{\max} = 2$ and $I_{\mathrm{thr}} = 0$. The architecture contains the unrolled CRC-polar concatenated factor graph. Therefore, the message updates of BP decoding on polar codes at a computational layer is represented as the ones in Fig. 1c and Fig. 1d. In order to assign the weights to the $\frac{N}{2}$ parallel PEs at polar code computational layers, we represent the product of the weights w_3 and w_4 in (9), and the product of the weights w_9 and w_{10} in (10), as single trainable weights $w_{3,4}$ and $w_{9,10}$, respectively. This is due to the fact that the product of two trainable weights can be merged into one as the new weight can also be optimized during training. In addition, we merge the weights w_1 and w_2 in (9) into $w_{1,2}$, and the weights w_7 and w_8 in (10) into $w_{7,8}$, to further reduce the number of trainable weights. As a result, we define the
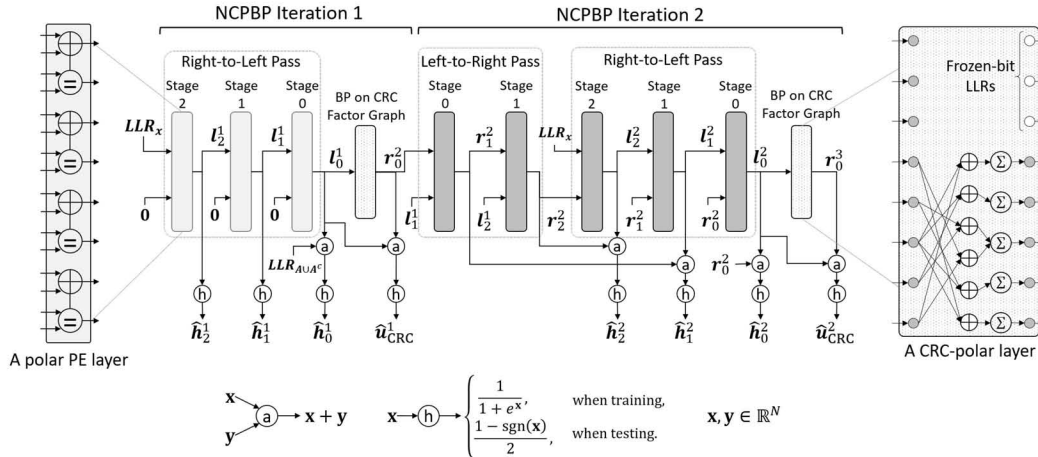
Fig. 5: NCPBP architecture with $I_{\max} = 2$ and $I_{\text{thr}} = 0$ for $\mathcal{P}(8,3)$ concatenated with a 2-bit CRC.

weight assignment scheme of a PE in NCPBP decoding as

$$
\begin{cases}
l_{t,s}^i &= w_0 \tilde{f}(l_{t,k}^i, w_{1,2}(r_{j,s}^i + l_{j,k}^i)), \\
l_{j,s}^i &= w_{3,4} \tilde{f}(l_{t,k}^i, r_{t,s}^i) + w_5 l_{j,k}^i,
\end{cases} \tag{13}
$$

$$
\begin{cases}
r_{t,k}^i &= w_6 \tilde{f}(r_{t,s}^i, w_{7,8}(l_{j,k}^{i-1} + r_{j,s}^i)), \\
r_{j,k}^i &= w_{9,10} \tilde{f}(r_{t,s}^i, l_{t,k}^{i-1}) + w_{11} r_{j,s}^i.
\end{cases} \tag{14}
$$

For the BP decoding on the CRC factor graph of the proposed NCPBP decoder, we adopt the weight assignment scheme of the NNMS-RNN decoder in [11]. It should be noted that the polar code computational layers share the same set of weights in each iteration of the proposed NCPBP decoding, while this set of weights is different for different iterations. This is illustrated in Fig. 5, in which the layers depicted in the same color indicate that they use the same set of weights. On the contrary, the weights used in all the CRC layers are shared among all the decoding iterations of NCPBP. This is particularly useful in order to limit the number of required weights for NCPBP.

The NCPBP decoding algorithm starts by a right-to-left message update at iteration 1. At the i-th iteration and the s-th stage of the NCPBP decoder, $\boldsymbol{l}_s^i$ and $\boldsymbol{r}_s^i$ denote the LLR vectors of the right-to-left and left-to-right message updates computed by a polar code PE layer, respectively. Furthermore, the output LLR vector of the CRC layer is denoted as $\boldsymbol{r}_0^{i+1}$. The hard estimated values of all the stages in the polar code factor graph are obtained at the right-to-left message updates, denoted as $\hat{\boldsymbol{h}}_s^i$, while the hard estimated values derived from the CRC layer is denoted as $\hat{\boldsymbol{u}}_{\text{CRC}}^i$.

The weights of all the polar code and CRC computational layers are trained using a multiloss function defined as

$$
L = \sum_{i=1}^{I_{\max}} \sum_{s=0}^{n-1} H_{\text{CE}}(\hat{\boldsymbol{h}}_s^i, \boldsymbol{h_s}) + \sum_{i'=I_{\text{thr}}+1}^{I_{\max}} H_{\text{CE}}(\hat{\boldsymbol{u}}_{\text{CRC}}^{i'}, \boldsymbol{u}), \tag{15}
$$

where H_{CE} is the cross-entropy function, and $\boldsymbol{h_s}$ is the correct hard value vector at stage s of the polar code factor graph which is obtained from the training samples. Note that in the

testing phase, only the hard estimated values at stage 0 of the polar code factor graph, i.e. $\hat{\boldsymbol{h}}_0^i$ ($1 \leq i \leq I_{\max}$), and the hard estimated values at the CRC layer, i.e. $\hat{\boldsymbol{u}}_{\text{CRC}}^i$, ($I_{\text{thr}} < i \leq I_{\max}$), are required to obtain the decoded message bits.

We evaluate the proposed NCPBP decoder for $\mathcal{P}(128, 80)$ concatenated with a 16-bit CRC which is also used in Section III, and we compare the error-correction performance and latency of NCPBP with those of [4], [11], [14]. All the neural BP-based decoders in this section are trained using stochastic gradient descent with RMSPROP optimizer [16] and the learning rate is set to 0.001. We use Tensorflow [15] as our deep learning framework. Since all the considered neural BP-based decoders satisfy the symmetry conditions [17], we collect $100,000$ zero codewords at each E_b/N_0 value for training, where $E_b/N_0 \in \{4, 4.5, 5, 5.5\}$ dB. All the weights of all the neural BP-based decoders are initialized to one and all the LLR values are clipped to be in the interval of $[-20, 20]$. The mini-batch size is set to 64 and each neural decoder is trained for 40 epochs. To evaluate the error-correction performance, randomly generated codewords are used during the testing phase and each decoder simulates at least $10,000$ codewords until it obtains at least 50 frames in error.

Fig. 6 compares the error-correction performance of the proposed NCPBP decoder with state-of-the-art BP-based decoders in [4], [11], [14]. We use the NNMS-$I_{\max}$ decoder of [14] and the NNMS-RNN-$I_{\max}$ decoder of [11] for our comparisons. In all the decoders, we set $I_{\max} = 30$. At a target FER of 10^{-5}, the proposed NCPBP decoder provides about 0.5 dB gain with respect to [4], 0.4 dB gain with respect to [14], and 0.2 dB gain with respect to [11]. Compared to the CPBP decoder of Section III, the proposed NCPBP provides 0.25 dB FER performance improvement.

Fig. 7 illustrates the average latency requirements of the NCPBP decoder compared to the state-of-the-art decoders in [4], [11], [14]. It can be seen that while the average latency of the NCPBP decoder is similar to that of the decoders in [4], [14], it is always better than that of [11]. In addition, NCPBP incurs almost no latency overhead with respect to the
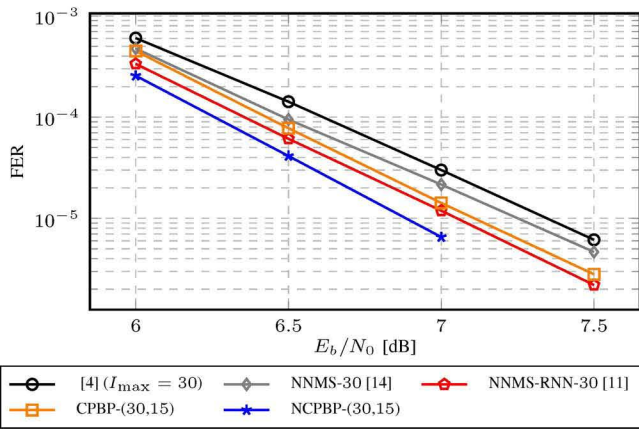
Fig. 6: FER performance of NCPBP decoding for $\mathcal{P}(128, 80)$ and a 16-bit CRC used in 5G.
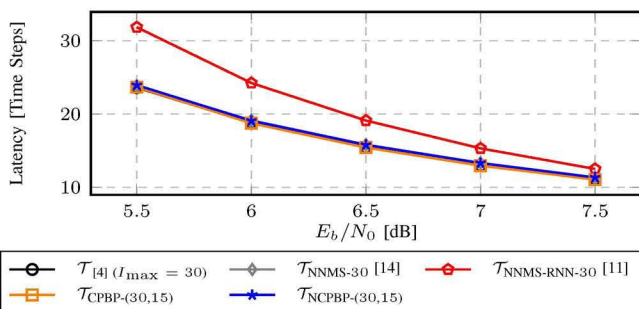

Fig. 7: Average decoding latency of NCPBP decoding for $\mathcal{P}(128, 80)$ and a 16-bit CRC used in 5G.

proposed CPBP decoder while having a notably smaller error probability.

Table I shows the number of weights required for the proposed NCPBP decoder in comparison with the neural BP decoders of [11], [14]. The proposed NCPBP decoder requires 28% fewer weights with respect to the decoder in [11]. The decoder in [14] requires 46% of the weights that is required by the proposed NCPBP decoder. However, the smaller number of weights in [14] results in significant error-correction performance loss as shown in Fig. 6.

V. CONCLUSION

In this paper, we first proposed a cyclic redundancy check (CRC)-polar belief propagation (BP) (CPBP) decoding algorithm which exploits concatenated factor graphs of polar codes and CRC, by passing the extrinsic information between the two factor graphs. We showed that the proposed CPBP decoding brings significant error-correction performance improvements in comparison with a conventional BP decoder when a large maximum number of BP iterations is used. We further proposed a neural CPBP (NCPBP) decoding algorithm which further improves the error probabilities of CPBP by assigning trainable weights to the edges of the CRC-polar concatenated factor graph. We showed that the NCPBP decoding algorithm can be used in applications which require stringent latency

TABLE I: Number of weights required by different neural BP decoders.

Decoder	Number of weights
NNMS-30 [14]	3840
NNMS-RNN-30 [11]	11520
NCPBP-(30,15)	8288

requirements and that it can benefit from the CRC which is present in 5G. Our results for a polar code of length 128 with 80 information bits concatenated with a CRC of length 16 show that the proposed NCPBP decoding algorithm obtains up to 0.4 dB error-correction performance improvement with respect to the state of the art, at a target frame error rate of 10^{-5}, while incurring negligible latency overhead.

REFERENCES

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

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

[3] 3GPP, "Multiplexing and channel coding (Release 10) 3GPP TS 21.101 v10.4.0." Oct. 2018. [Online]. Available: http://www.3gpp.org/ftp/Specs/2018-09/Rel-10/21_series/21101-a40.zip

[4] Y. Ren, C. Zhang, X. Liu, and X. You, "Efficient early termination schemes for belief-propagation decoding of polar codes," in *IEEE 11th Int. Conf. on ASIC*, Nov 2015, pp. 1–4.

[5] S. Sun, S. Cho, and Z. Zhang, "Post-processing methods for improving coding gain in belief propagation decoding of polar codes," in *2017 IEEE Glob. Commun. Conf.*, Dec 2017, pp. 1–6.

[6] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation decoding of polar codes on permuted factor graphs," in *IEEE Wireless Commun. and Net. Conf.*, April 2018, pp. 1–6.

[7] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "On the Decoding of Polar Codes on Permuted Factor Graphs," *ArXiv e-prints*, Jun. 2018. [Online]. Available: https://arxiv.org/abs/1806.11195

[8] J. Guo, M. Qin, A. G. i Fàbregas, and P. H. Siegel, "Enhanced belief propagation decoding of polar codes through concatenation," in *IEEE Int. Symp. on Inf. Theory*, June 2014, pp. 2987–2991.

[9] S. M. Abbas, Y. Fan, J. Chen, and C. Tsui, "Concatenated LDPC-polar codes decoding through belief propagation," in *IEEE Int. Symp. on Circuits and Systems*, May 2017, pp. 1–4.

[10] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Improving belief propagation decoding of polar codes using scattered exit charts," in *IEEE Inf. Theory Workshop*, Sept 2016, pp. 91–95.

[11] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. of Sel. Topics in Signal Process.*, vol. 12, no. 1, pp. 119–131, February 2018.

[12] E. Arıkan, "Polar codes: A pipelined implementation," in *Proc. 4th Int. Symp. on Broad. Commun.*, 2010, pp. 11–14.

[13] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *IEEE Int Symp. on Inf. Theory*, August 2017, pp. 1361–1365.

[14] W. Xu, Z. Wu, Y.-L. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *IEEE Int. Workshop on Signal Process. Syst.*, November 2017, pp. 1–6.

[15] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. on Operating Systems Design and Impl.*, ser. OSDI'16. USENIX Association, 2016, pp. 265–283.

[16] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." [Online]. Available: https://cs.toronto.edu/csc321/slides/lecture_slides_lec6.pdf

[17] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb 2001.