entropy

MDPI

Article

Higher-Order Belief Propagation Correction Decoder for Polar Codes

Meng Zhang, Zhuo Li *, Lijuan Xing and Xin Liao

The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China; mzhang_218@stu.xidian.edu.cn (M.Z.); ljxing@mail.xidian.edu.cn (L.X.); lxin2526@126.com (X.L.)
* Correspondence: lizhuo@xidian.edu.cn

Abstract: Belief propagation (BP) decoding for polar codes has been extensively studied because of its inherent parallelism. However, its performance remains inferior to that of successive cancellation list decoding (SCL) due to the structure of the decoding graph. To improve the block error rate (BLER) performance, the BP correction (BPC) decoding, a post-processing scheme that corrects prior knowledge of the identified code bit, improves convergence by executing additional iterations on the failed BP decoder. Moreover, the BPC decoder demonstrates a better decoding performance than the BP-based bit-flipping decoder. Nevertheless, the additional decoding attempts lead to increased latency. In this article, a modified BPC decoder is proposed to reduce the number of decoding attempts by redefining the correction rules. A new metric is designed to effectively identify the corrected location. Numerical results show that the proposed modified BPC decoder achieves a slight improvement in BLER compared with the original BPC, with a dramatic reduction in average complexity. Furthermore, a higher-order version, named MBPC-Ω, is extended to further improve the performance, where the Ω is the maximum correction order. Numerical results show that the higher-order modified BPC achieves a similar BLER performance to existing multiple bit-flipping BP decoders but has around half the latency overhead. In addition, the proposed MBPC-2 decoder performs better than the cyclic redundancy check-aided SCL (CA-SCL) decoder with list size 4 and is slightly worse than the CA-SCL with list size 8 in high signal-to-noise ratio (SNR) regions but with significant decoding latency reduction.

Keywords: polar code; belief propagation decoding; post-processing; higher-order correction

Citation: Zhang, M.; Li, Z.; Xing, L.; Liao, X. Higher-Order Belief Propagation Correction Decoder for Polar Codes. *Entropy* **2022**, *24*, 534. https://doi.org/10.3390/e24040534

Academic Editors: Predrag Ivanis and Goran Djordjević

Received: 11 February 2022
Accepted: 8 April 2022
Published: 11 April 2022

1. Introduction

Polar code [1] is a mathematically proven capacity-achieving coding scheme and has specific construction methods and linear encoding and decoding complexity, which have been selected as the coding scheme in fifth-generation enhanced mobile broadband (eMBB) control channels [2]. A successive cancellation list (SCL) decoder with a cyclic redundancy check (CRC) (CA-SCL) [3,4] obtains comparable error performance with that of the state-of-the-art LDPC code. However, it suffers from high latency due to its intrinsic serial structure. Conversely, the belief propagation (BP) decoder [5,6] performs the decoding process in a parallel manner, therefore garnering more and more attention as a low-latency implementation method.

However, the original BP decoder has a worse block error rate (BLER) performance than that of the SCL decoder. Therefore, several BP-based decoding algorithms have been proposed to enhance BLER performance. By considering different representations obtained using graph permutations, the multi-trellis scheme, first proposed in [7], was further improved in [8–10] with different permuted patterns, resulting in better BLER performance, but its throughput subsequently dropped. In [11], a BP list (BPL) decoder performed L independent BP decoders on different permuted versions of the factor graph to yield a valid estimate. Additionally, the selection strategy of L permuted factor graphs which have the least number

of girths was designed in [12], resulting in better performance and lower average decoding latency. Post-processing is another effective strategy to improve the BLER performance. A random perturbation-based post-processing method was developed in [13] to address different types of BP errors and to improve convergence. The noise-aided BPL decoder, combined with random noise injection and list decoding, introduced in [14], has a much better performance than the SCL and is a bit behind the CA-SCL decoder in terms of BLER performance. Inspired by the SC bit-flipping decoder, the BP flipping (BPF) decoder, as a post-processing method, was first proposed in [15] and was then improved in [16–18]. The BPF decoder, performing bit-flipping on error-prone information bits, achieved a comparable BLER performance with that of the SCL decoder with a small list size. The flipping set which is constructed by a convolutional neural network-based imitation learning scheme was presented in [19], such that useless flipping attempts are avoided. [20] proposed a noised-aided BPL bit-flipping decoder which is superior to the CA-SCL decoder at the cost of numbers of decoding iterations. In [21], a low-complexity two-level post-processing algorithm was proposed to modify the false converged error efficiently in the BPL decoder. In [22], the correction performed on code bits was studied. When the BP decoder failed, the correction operation performed as an external perturbation used to improve the convergence, named the BPC decoder. By correcting the unreliable code bits, i.e., assigning $\pm\infty$ to their log-likelihood rate (LLR) and performing additional iterations, the decoder is expected to converge correctly. For identical additional decoding attempts, the BPC decoder shows better BLER performance than the BPF decoder. However, without benefiting from prior knowledge about the identified code bit, assignments on both sides lead to excessive decoding attempts, resulting in relatively high decoding latency overhead.

In this article, the BPC decoder is modified by redefining the correction set based on a new metric and the correction operation based on the predefined threshold, abbreviated as the MBPC decoder. Then, the proposed MBPC decoding is extended to a higher-order scheme, named MBPC-Ω, where Ω code bits are corrected in one decoding attempt. The simulation results show that the MBPC decoder shows a slight performance gain but has less complexity and latency than the original BPC decoder. The higher-order MBPC decoders are verified to considerably improve the BLER performance. For polar codes of length $N = 2048$ with 24-bit CRC termination, when BLER $= 10^{-3}$, the MBPC-2 decoder achieves around a 0.9 dB performance gain compared with the original BP decoder. Compared with existing BP-based multiple flipping decoders, the MBPC-2 performs a similar BLER but approximately half iterations. At BLER $= 10^{-5}$, the MBPC-2 decoder achieves a gain of 0.2 dB compared with the CA-SCL decoder with $L = 4$, at a latency overhead saving of 95%. The contributions of this article are summarized as follows.

- Different from using the stopping tree to reduce the searching range in the original BPC decoder, a new metric is proposed, which combines the reliability of code bits and the size of the stopping tree to which it belongs, to construct the correction set effectively.
- We refine the correction operation in the original BPC decoder. Setting a threshold $\mathcal{V}$, an identified code bit is only corrected to its opposite side when its reliability is lower than $\mathcal{V}$. The decoding latency is reduced by avoiding invalid decoding attempts, especially in low signal-to-noise ratio (SNR) regions. In addition, the refresh operation is modified by re-initializing the massage to the initial state of the original BP rather than the end state to remove the dispensable store procedures.
- The MBPC-Ω decoder is proposed to further improve the BLER performance, where the decoder is executed on correction sets of increasing order. The correction set $\mathcal{S}_\Omega$, in which each element contains Ω code bits, is established nested based on the preorder set $\mathcal{S}_{\Omega-1}$ and the corresponding decoding result.

The remainder of this article is organized as follows. Section 2 provides brief statements of the polar code, the BP decoder, and the BPC decoding algorithm. Section 3 analyzes the metric and provides the proposed MBPC decoding algorithm. The higher-order MBPC decoder is presented. The simulation results are presented in Section 4. Finally, Section 5 concludes the article.

2. Preliminaries

2.1. Polar Codes

Polar codes are linear block codes based on the channel polarization phenomenon. After channel combination and splitting, N channel copies are converted to N polarized sub-channels, where the capacity of each sub-channel approaches 1 or 0 as N increases. The polar codes select K reliable sub-channels to transmit information bits, and the remaining sub-channels transmit frozen bits. The sets of information indices and frozen indices are denoted by $\mathcal{I}$ and its complement $\mathcal{I}^C$, respectively. For shorthand, we use a_i^j to denote the vector $(a_i, a_{i+1}, ..., a_j)$. The codeword x_1^N can be mapped by the generator matrix G_N:

$$x_1^N = u_1^N G_N, \tag{1}$$

where $N = 2^n$, $G_N = R_N F^{\otimes n}$. The $F^{\otimes n}$ is the n-th Kronecker power of $F = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ and R_N is a $N \times N$ bit-reversal permutation matrix. The source vector $u_1^N \in \{0,1\}^N$ includes both information and frozen blocks.

2.2. Belief Propagation Decoder

BP decoding for polar codes is an iterative decoding algorithm based on the $(n+1)$-stage factor-graph realization of G_N. Two types of LLR messages are involved: right-propagating $R_{i,j}$ and left-propagating $L_{i,j}$, where i denotes the stage and j denotes the row index. The four nodes connecting two stages constitute the processing elements (PEs). The architecture of a PE is shown in Figure 1. Thus, the iteration rule can be represented by Equation (2):

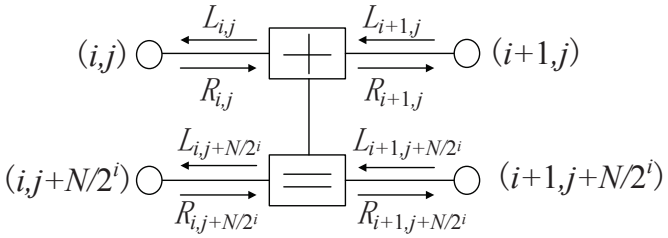

Figure 1. The processing element (PE) in BP decoder.

$$\begin{cases} L_{i,j} = f(L_{i+1,j}, L_{i+1,j+N/2^i} + R_{i,j+N/2^i}) \\ L_{i,j+N/2^i} = f(L_{i+1,j}, R_{i,j}) + L_{i+1,j+N/2^i} \\ R_{i+1,j} = f(R_{i,j}, L_{i+1,j+N/2^i} + R_{i,j+N/2^i}) \\ R_{i+1,j+N/2^i} = f(L_{i+1,j}, R_{i,j}) + R_{i,j+N/2^i} \end{cases}, \tag{2}$$

where the function $f(x,y) = \ln(1 + e^{x+y})/(e^x + e^y)$. The $L_{n+1,j}$ messages at the rightmost stage are obtained by the channel output $\mathrm{LLR}(y_j)$. Additionally, the $R_{1,j}$ messages at the leftmost stage are initialized by a priori known information from the information bits and frozen bits, as shown in Equations (3) and (4).

$$L_{n+1,j} = \mathrm{LLR}(y_j), \tag{3}$$

$$R_{1,j} = \begin{cases} 0, & \text{if } j \in \mathcal{I} \\ +\infty, & \text{if } j \in \mathcal{I}^C \end{cases}. \tag{4}$$

Conventional flooding scheduling is applied [23]. The messages, starting from the rightmost stage, are iteratively updated column by column from right to left and then from left to right until a maximum iteration number I_{max} is reached or a predefined stopping

condition is valid. The BP decoder outputs the estimated vector $\hat{u}_1^N = (\hat{u}_1, \hat{u}_2, ..., \hat{u}_N)$ through $\hat{u}_j = \text{sgn}(L_{1,j} + R_{1,j})$, where $\text{sgn}(x) = 0$ when $x \geq 0$, and $\text{sgn}(x) = 1$ otherwise.

2.3. Belief Propagation Correction Decoder

The BPC decoder, proposed in [22], considers the perturbation on code bits located at the rightmost stage of the factor graph, which are corrupted by channel noise. When a decoding failure occurs in the original BP decoder, the decoding attempts after correction are performed to generate a valid estimate. Specifically, the correction operations assign both $+\infty$ and $-\infty$ to the $L_{n+1,j}$ of identified code bits in turn, and additional decoding attempts are executed until the estimate fulfills the error detection and termination check (EDTC) criterion. Each decoding attempt involves a one-bit correction. Compared with the flipping on information bit, the simulation results show that the correction on code bit has a greater impact on the decoding result. The correction set $\mathcal{S}$ is established based on the reliabilities of the left-propagating message determined by the channel LLRs. The decoder selects T indices with the lowest magnitude of $L_{n+1,j}$, i.e., $\mathcal{S} \leftarrow j \in \{1, 2, ..., N\}$ of T smallest $|L_{n+1,j}|$. The stopping tree [22] is utilized to reduce the sorting range. The CRC is selected as the EDTC. Let the symbol in bold denote the matrix for LLR messages. Algorithm 1 gives a brief description of the BPC decoding.

Algorithm 1 The BPC decoding algorithm.

Input: LLR(y_1^N), I_{max}, T, $\mathcal{I}$.
1: Initialize **L** and **R** by Equations (3) and (4)
2: $\hat{u}_1^N \leftarrow$ conventional BP decoding with CRC termination and I_{max}.
3: **if** CRC$(\hat{u}_1^N)$ = true **then**
4: **return** $\hat{u}_1^N$.
5: **else**
6: Store the messages of current state in vectors $\mathbf{L}'$ and $\mathbf{R}'$.
7: Obtain the correction set $\mathcal{S} = \{j_1, j_2, ..., j_T\}$.
8: **for** $k = 1$ to T **do**
9: Refresh the **L** and **R** by $\mathbf{L}'$ and $\mathbf{R}'$. And assign the $\pm\infty$ to L_{n+1,j_k} in turn.
10: Perform additional BP iterations with CRC termination and get $\hat{u}_1^N$.
11: **if** CRC$(\hat{u}_1^N)$ = true **then**
12: **return** $\hat{u}_1^N$.
13: **end if**
14: **end for**
15: **end if**
Output: $\hat{u}_1^N$.

3. Design of Higher-Order Belief Propagation Correction Decoder

As the SNR increases, the decoding failure is gradually dominated by the graph structure, and the scheme for selecting code bits based on the value of LLR alone is flawed. This section first proposes a metric to effectively select the corrected code bit. Differently from the reduction of the searching range in [22], the stopping tree is selected as a part of the metric. Then, to solve the issue of additional latency caused by bilateral attempts, the original BPC is modified by refining the correction operation with a threshold. Then, a high-order modified BPC decoder is designed to further improve the BLER performance.

3.1. A Metric to Establish the Correction Set $\mathcal{S}$

Inspired by the fact that the channel noise and graph structure dominate the errors in BP decoding, we propose a new metric, which combines the reliability of the code bit and the characteristic of the stopping tree where the code node is located. Specifically, eruption by channel noise is the leading cause of decoding errors in low to medium SNR regions,

while the graph structure dominates the decoding result in high SNR regions. Thus, the metric combines the two factors to identify the valid corrected node.

A stopping set, defined as a subset of all variable nodes in the Tanner graph realization, is one of the critical factors affecting the BP decoding result and causing error floor [24]. In the stopping set, the adjacent check nodes of each variable node are connected to the set at least twice. For polar codes, a stopping set can be shaped like a tree rooted at a single information bit with leaves at code bits and includes at least one variable node from each middle column of the graph. A stopping tree rooted with it can be easily found for each information bit. An example of such a stopping tree is shown in Figure 2 with black nodes. Conversely, for each code bit x_j, we can find the stopping trees with that specific code bit as a leaf node. Let $ST(j)$ denote the number of stopping trees that the code bit x_j belongs to. According to [25], the chance of recovery for the code bit with small $ST(j)$ is higher than others by iterations. In other words, errors in the code bits with large $ST(j)$ have more impact on the decoding results. Therefore, we select the corrected code bit according to $ST(j)$ combined with the reliability $\mathcal{L}(j)$ of code bit.

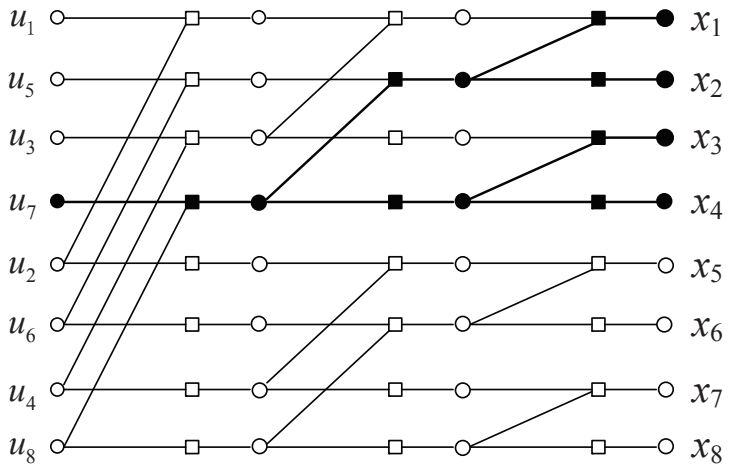

Figure 2. Normal realization of the encoding graph for $N = 8$. An example of the stopping tree rooted at u_7 is shown in the black nodes.

Denoting the reliability of j-th code bit as $\mathcal{L}(j) = R_{n+1,j} + L_{n+1,j}$, a new metric $M(j)$ for code bit x_j is defined in Equation (5), where α is used to select the unreliable code bit and β is a penalty indicated by a stopping tree.

$$M(j) = \alpha \cdot |\mathcal{L}(j)| + \beta \cdot (n+1) \cdot (1/ST(j)). \tag{5}$$

By sorting the resulting metrics in ascending order, i.e., $M(j_1) \leq M(j_2) \leq ... \leq M(j_N)$, the nodes with the smallest metrics are selected to establish the set $\mathcal{S}$, i.e., $\mathcal{S} = \{j_1, j_2, ..., j_T\}, |\mathcal{S}| = T$. The values of α and β, related to the dimension and channel noise, can be optimized by off-line Monte Carlo simulation.

3.2. The Modified BPC Decoder

In the BPC decoder, the correction operation assigns $\pm\infty$ to the selected code bits separately and performs decoding attempts, bringing numerous latencies overhead. Since the code bits contain prior information, the correction operation is reconsidered by the parameter $\mathcal{V}$, a threshold for the magnitude of $\mathcal{L}(j)$, which is set to identify the current state of x_j. When $|\mathcal{L}(j)| \geq \mathcal{V}$, the code bit is considered reliable so that both positive and negative assignments were executed. Otherwise, the node is corrected with its opposite side based on the sign of $\mathcal{L}(j)$.

The modified BPC decoder based on the proposed metric and refined correction, abbreviated as MBPC, is detailed in Algorithm 2. The function sign(x) marks the sign of

x, $\text{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$. As shown in Algorithm 2, the enhanced correction operation in lines 9–14 avoids unnecessary attempts to reduce the complexity and latency overhead. Additionally, in lines 10 and 14, a predefined finite value τ is set, rather than ∞, which can mitigate the impact caused by a false assignment. Here, the value of τ is set to 8. Note that, in Algorithm 1, the store operation in line 6 and refresh operation in line 9 ensure that the correction is post-processed on the conventional BP and avoid error propagation caused by previous false attempts. Based on the empirical fact that it converges quickly if the input is valid in the BP decoder, we modify the refresh operation, where the **L** and **R** messages are reset to the initial state before each attempt, as shown in Algorithm 2 line 8, resulting in avoiding complexity and memory consumption. Moreover, the modified refresh operation enables the post-processing scheme to operate in T independent factor graphs in parallel.

Algorithm 2 Modified BPC decoding algorithm.

Input: LLR(y_1^N), I_{max}, $\mathcal{I}$, T.
1: Initial **L** and **R** by Equations (3) and (4).
2: $\hat{u}_1^N \leftarrow$ conventional BP decoding with CRC termination and I_{max}.
3: **if** CRC$(\hat{u}_1^N)$ = true **then**
4: **return** $\hat{u}_1^N$.
5: **else**
6: Construct the set $\mathcal{S} = \{j_1, j_2, ..., j_T\}$ based on $M(j)$ in Equation (5).
7: **for** $k = 1$ to T **do**
8: Initial **L** and **R** by Equations (3) and (4).
9: **if** $|\mathcal{L}(j_k)| < \mathcal{V}$ **then**
10: $L_{n+1,j_k} = -\text{sign}(L_{n+1,j_k}) \times \tau$
11: $\hat{u}_1^N \leftarrow$ conventional BP decoding with CRC termination.
12: **else**
13: **for** $a = 0 : 1$ **do**
14: $L_{n+1,j_k} = (1 - 2a) \times \tau$.
15: $\hat{u}_1^N \leftarrow$ conventional BP decoding with CRC termination.
16: **end for**
17: **end if**
18: **if** CRC$(\hat{u}_1^N)$ = true **then**
19: **return** $\hat{u}_1^N$.
20: **end if**
21: **end for**
22: **end if**
Output: $\hat{u}_1^N$

3.3. Higher-Order Belief Propagation Correction Decoder

In this section, we generalize the MBPC decoder to a higher-order version, called MBPC-Ω. When the conventional BP fails to pass the CRC, the decoder executes MBPC-Ω in increasing order consecutively. Similar to the SCF-Ω decoder defined in [26], the proposed MBPC-Ω decoding is performed on nested correction sets, each of which contains T_k elements, denoted as $\mathcal{S}_k$, $1 \leq k \leq \Omega$. Each element consists of k code bits. The nested rule is designed to construct the set. Concretely, the set $\mathcal{S}_k$ is established based on the preorder set $\mathcal{S}_{k-1}$ and its corresponding decoding results. In other words, each element in $\mathcal{S}_k$ contains the element in $\mathcal{S}_{k-1}$ and an additional code bit, which was selected based on the metric of the decoding result in the decoding attempt by $\mathcal{S}_{k-1}$.

Denoting the r-th element in S_{k-1} by $S_{k-1,r}$, we redefine $M(j)^{k,r}$ as the metric of each code bit in the decoding result of a decoding attempt with $S_{k-1,r}$.

$$M(j)^{k,r} = \alpha \cdot |\mathcal{L}(j)^{k-1,r}| + \beta \cdot (n+1) \cdot (1/ST(j)), \tag{6}$$

where $\mathcal{L}(j)^{k-1,r}$ represents the reliability of code bits corresponding to the decoding result by $S_{k-1,r}$ and $\mathcal{L}(j)^{0,r}$ represents the initial message LLR(y_j). For limited computational complexity, we extract $T_{k,k-1}$ elements from S_{k-1}, each of which selects $T_{k,k}$ code bits with the smallest metric based on Equation (6) as the k-th identified bit. Thus, the size of set $\mathcal{S}_k$ is $T_k = T_{k,k-1} \times T_{k,k}$, and the maximum number of decoding attempts of order-k is $2^k \times T_k$. The rule to generate the correction set can be summarized as $generate\_S_k$, described in Algorithm 3.

The MBPC-Ω decoder is presented in Algorithm 4. For the convenience of description, we describe the enhanced correction operation in the procedure as $correction\_operation(\mathcal{S}_{k,r})$ in Algortihm 5. For each attempt, k identified code bits are corrected simultaneously. The "flag" indicates the avoidable decoding attempt. The maximum number of decoding attempts is $T = \sum_{k=1}^{\Omega} 2^k \times T_k$. Note that, in the case of $|\mathcal{L}(j_l)| \geq \mathcal{V}$, if the decoding attempts of both sides fail the CRC, the element to which j_l belongs is repeatedly recorded in the next order of the correction set. If none of the decoding attempts meet the requirements of the CRC until the order exceeds Ω, the decoder declares a failure.

Algorithm 3 The procedure of $generate\_S_k$.

for $r = 1$ to $T_{k,k-1}$ **do**
  $\mathcal{S}_{tmp} \leftarrow T_{k,k}$ bits with smallest $M(j)^{k,r}$.
    // generate a size-$T_{k,k}$ set of k-th corrected bit
  for $j = 1$ to $T_{k,k}$ **do**
    $\mathcal{S}_{k,(r-1)\cdot T_{k,k-1}+j} \leftarrow \mathcal{S}_{k-1,r} \cup \mathcal{S}_{tmp}, j$.
  end for
end for

Algorithm 4 The proposed MBPC-Ω decoding algorithm.

Input: LLR(y_1^N), I_{max}, $\mathcal{I}$, T, Ω.
Initial **L** and **R** by Equations (3) and (4).
Perform the original BP decoder with CRC termination.
if $\hat{u}_1^N$ passes the CRC **then**
  return $\hat{u}_1^N$.
else
  Generate the set $\mathcal{S}_1$ of size T_1.
  for $k = 1$ to Ω **do**
    for $r = 1$ to T_k **do**
      // each code bit in r-th element expressed by j_l: $(j_1, j_2, ...j_k)$.
      $a = 0$, flag $= 0$.
      while $a < 2^k$ **do**
        Initialize **L** and **R** by Equations (3) and (4).
        Perform the $correction\_operation(\mathcal{S}_{k,r}, a,$ flag$)$; a++.
        if flag $= 1$ **then**
          continue.
        end if
        $\hat{u}_1^N \leftarrow$ conventional BP decoding with CRC termination.
        if CRC$(\hat{u}_1^N)$ = true **then**
          return $\hat{u}_1^N$.
        else
          Construct the set $\mathcal{S}_{k+1}$ by $generate\_S_k$.
        end if
      end while
    end for
  end for
end if
Output: $\hat{u}_1^N$

Algorithm 5 The procedure of *correction_operation*($\mathcal{S}_{k,r}, a$, flag).

Binary extension for $a = (b_1, b_2, ..., b_k)$.
for $l = 1$ to k **do**
  if $|\mathcal{L}(j_l)| < \mathcal{V}$ and $b_l = 1$ **then**
    flag $= 1$; **break**.
  else
    $L_{n+1,j_l} = (2b_l - 1) \times \text{sign}(L_{n+1,j_l}) \times \tau$; flag $= 0$.
  end if
end for
Output: flag.

4. Numerical Results

In this section, the simulation results are verified on a binary-input additive white Gaussian noise (BI-AWGN) channel in terms of the BLER performance and decoding latency for polar codes based on the proposed schemes. Information set $\mathcal{I}$ is selected based on the Monte Carlo approach [1] at 2.0 dB.

First, the BLER performance of the proposed MBPC-Ω scheme is compared with that of the BPC decoder in [22] and the EBPF decoder in [16]. MBPC-Ω with order $\Omega = 1$ is a one-bit correction scheme, the same as MBPC. The parameters of the proposed metric are set to $\alpha = 1.0, \beta = 0.75$ for code length $N = 512$ and rate $R = 1/2$. To ensure a fair comparison, the sizes of the correction set are set to $T = 20$ in the MBPC-1 and the BPC, while the size of the flipping set in the EBPF is set to $T = 40$. The parameters of correction sets in the MBPC-2 and the MBPC-3 decoders are set to $T_1 = 20$, $T_{2,1} = T_{3,2} = 20$ and $T_{2,2} = T_{3,3} = 20$, respectively. As seen in Figure 3, the MBPC-1 decoder has a better BLER performance than the EBPF decoder and achieves a slight performance gain than that of the BPC decoder in high SNR regions. The proposed higher-order algorithm can evidently effectively improve the BLER performance, e.g., at BLER $= 10^{-3}$, the MBPC-2 decoder performs 0.9 dB better than the original BP 0.45 dB and better than the MBPC-1. As Ω increases, the maximum number of decoding attempts increases dramatically, bringing challenges to the complexity, but the performance gain gradually decreases. Accordingly, in the further comparisons with the existing multiple flipping scheme, only MBPC-1 and MBPC-2 are studied.

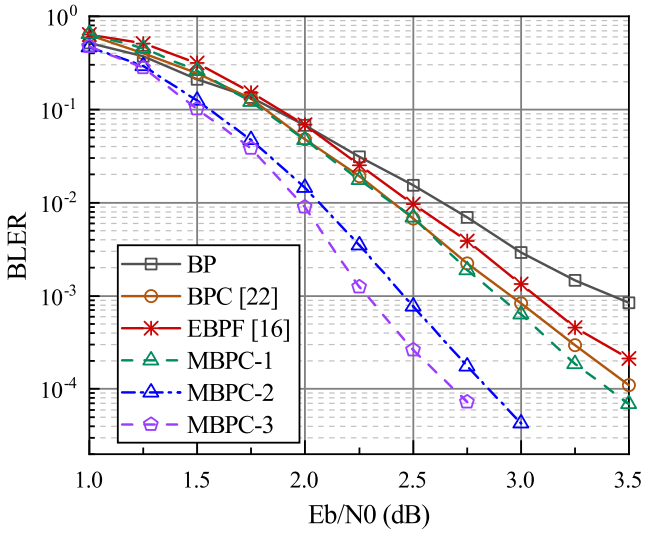

Figure 3. The BLER performance comparisons in polar code with length $N = 512$, $R = 1/2$, and $I_{max} = 60$. A 16-bit CRC is used.

The BLER performance and average number of iterations of the proposed schemes for $N = 2048$, $R = 1/2$ are simulated for a comparison with the original BP, BPC, higher-order GBPF proposed in [16], and multiple BPF decoder designed based on the critical set (CS) in [15] as shown in Figures 4 and 5. The 24-bit CRC is used. The parameters in Equation (6) are set to $\alpha = 0.65$ and $\beta = 3.5$. The size of the correction set in the MBPC-1 is $T = 20$. In the higher-order scenario, the number of maximum flipping attempts in the GBPF-2 decoder and the maximum correction operations in the MBPC-2 decoder are set to $T = 80 + 20 \times 40$ and $T = 40 + 4 \times 10 \times 20$, respectively, to keep the maximum decoding attempts the same as those for the BPF-2, i.e., $T = 4 \times |CS|$, $|CS| = 210$. As seen in Figure 4, the performance of the proposed MBPC-2 decoder is significantly improved compared with that of the conventional BP decoder. When compared with the BPF-2 and GBPF-2 decoders, the MBPC-2 decoder achieves a similar performance in low to medium SNR regions but has a 0.1 dB performance gain when BLER $= 10^{-5}$. Compared with the CA-SCL decoder with $L = 4$, the gain is around 0.2 dB in almost all simulated SNR regions. In addition, the BLER of the MBPC-2 decoder can compete with the CA-SCL decoder with $L = 8$ when SNR $\leq$ 2.2 dB, but the performance degrades at high SNR regions.

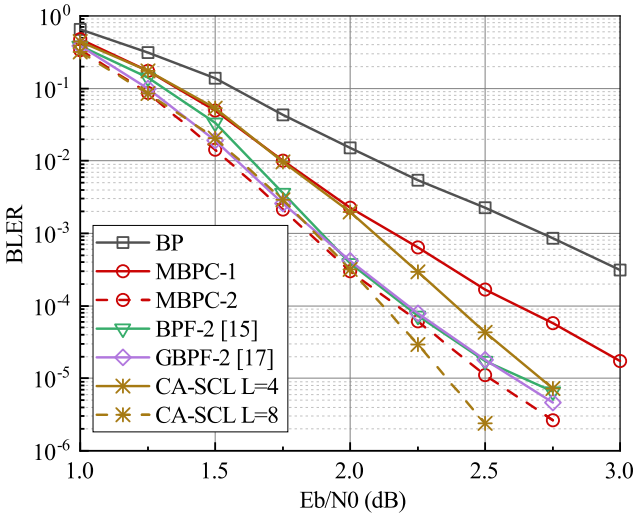

Figure 4. The BLER performance comparisons in the polar code with length $N = 2048$, $R = 1/2$, and $I_{max} = 200$. A 24-bit CRC is used.

The BP-based decoding complexity and latency can be reflected as the number of iterations. Figure 5 presents a comparison of the average number of iterations. As expected, the number of average iterations of the MBPC-1 decoder is approximately half that of the BPC and EBPF decoders, while the MBPC-2 decoder also reduces the complexity and latency by half compared with the BPF-2 and GBPF-2 decoders at a similar BLER performance. Specifically, the proposed MBPC-2 decoder saves around 45% of the average number of iterations at 2.0 dB than the BPF-2 decoder; also observed is that, with the increase in SNR, the average number of iterations of the proposed MBPC decoder can approach that of the conventional BP decoder.

Furthermore, the average decoding latency in the MBPC decoder defined by the clock cycles can be expressed by Equation (7) and compared with that of the CA-SCL decoder.

$$L_{ave} = I_{ave} \times 2 \log N, \tag{7}$$

where the I_{ave} represents the average number of iterations. Without considering the consumption of CRC, the decoding clock cycles of the CA-SCL decoder can be represented by the conventional SCL decoder in [27] as $L_{SCL} = 2N + K - 2$. Observed from Table 1, the MBPC-1 decoder has a low latency overhead at all simulated points, while the MBPC-2 decoder shows a lower decoding latency than that of the CA-SCL decoder in medium to high SNR regions.

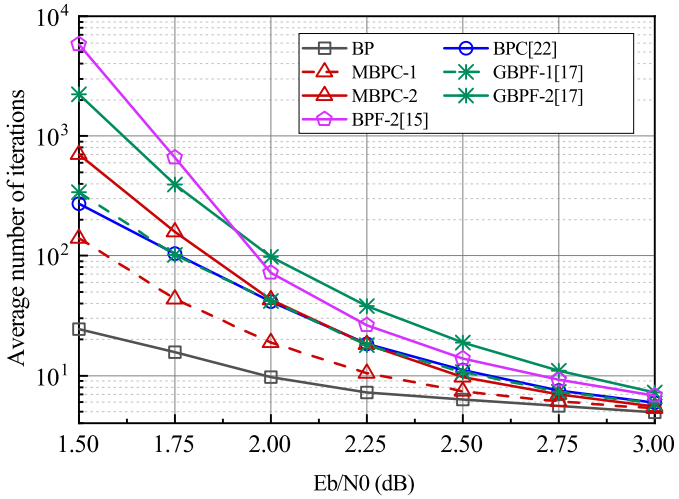

Figure 5. The average iteration number comparisons between the proposed MBPF and existing BP-based decoders. The code length $N = 2048$, $R = 1/2$ and $I_{max} = 200$.

Table 1. Average decoding clock cycle comparisons between the proposed MBPC decoder and the CA-SCL decoder.

Decoder	**SNR (dB)**						
	1.5	1.75	2.0	2.25	2.5	2.75	3.0
MBPC-1	3080	946	396	220	165	133	115
MBPC-2	15,466	3476	924	396	212	154	121
	List size						
CA-SCL	$L = 4$				$L = 8$		
	5142				5142		

5. Conclusions

In this article, a new metric, combined reliability with stopping tree, was defined to effectively identify corrected code bits in the BPC decoder. A rule for the correction operation defined by a threshold is designed to reduce the number of additional decoding attempts, especially in low to medium SNR regions. A higher-order version is designed based on the nested correction set to better decrease the gap with the performance of the CA-SCL decoder. The simulation results illustrate that the proposed higher-order MBPC decoding algorithm can effectively improve BLER performance with lower average latency in medium to high SNR regions and ensure better applicability than existing BPF decoders. Compared with the CA-SCL decoder, the proposed MBPC-2 decoder achieves around a 0.2dB performance gain with $L = 4$ in all simulated SNR regions and competes with the CA-SCL with $L = 8$ when SNR $\leq$ 2.2 dB. Further research directions involve latency reduction in low SNR regions and BLER performance improvement in high SNR regions. For the latency overhead caused by massive increases in the iterations at low SNR regions, the modified refresh operation enables the decoder to be implemented in a parallel version to reduce the latency for further research, in which T_1 decoders with the same structure can be executed independently with different initializations. In addition, the method for identifying the corrected locations by deep learning is also an attractive direction of study.

Author Contributions: Conceptualization, M.Z.; methodology, M.Z. and Z.L.; validation, Z.L.; investigation, M.Z.; writing—original, M.Z.; writing—review and editing, L.X. and X.L. All authors have read and agreed to the published version of the manuscript

Funding: This research was funded by the National Natural Science Foundation of China (NSFC) (61372072), by the Overseas Expertise Introduction Project for Discipline Innovation (111 Project) (B08038), and by the Fundamental Research Funds for the Central Universities.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Arıkan, E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inf. Theory* **2009**, *55*, 3051–3073. [CrossRef]
2. Final Report of 3GPP TSG RAN WG1 #87 v1.0.0; Reno, NV, USA. November 2016. Available online: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_87/Report/Final_Minutes_report_RAN1%2387_v100.zip (accessed on 10 November 2021).
3. Tal, I.; Vardy, A. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory* **2015**, *61*, 2213–2226. [CrossRef]
4. Niu, K.; Chen, K. CRC-Aided Decoding of Polar Codes. *IEEE Commun. Lett.* **2012**, *16*, 1668–1671. [CrossRef]
5. Arıkan, E. Polar Codes: A Pipelined Implementation. In Proceedings of the 4th International Symposium on Broadband Communication (ISBC 2010), Melaka, Malaysia, 11–14 July 2010; pp. 1–3.
6. Arıkan, E. A performance comparison of polar codes and reed-muller codes. *IEEE Commun. Lett.* **2008**, *12*, 447–449. [CrossRef]
7. Elkelesh, A.; Ebada, M.; Cammerer, S.; Brink, S.T. Mitigating clipping effects on error floors under belief propagation decoding of polar codes. In Proceedings of the 2017 International Symposium on Wireless Communication Systems (ISWCS), Bologna, Italy, 28–31 August 2017; pp. 384–389.
8. Doan, N.; Hashemi, S.A. On the decoding of polar codes on permuted factor graphs. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
9. Li, L.; Liu, L. Belief propagation with permutated graphs of polar codes. *IEEE Access* **2020**, *8*, 17632–17641. [CrossRef]
10. Ranasinghe, V.; Rajatheva, N. Partially permuted multi-trellis belief propagation for polar codes. In Proceedings of the 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
11. Elkelesh, A.; Ebada, M.; Cammerer, S.; Brink, S.T. Belief propagation list decoding of polar codes. *IEEE Commun. Lett.* **2018**, *22*, 1536–1539. [CrossRef]
12. Ren, Y.; Shen, Y.; Zhang, Z.; You, X. Efficient belief propagation polar decoder with loop simplification based factor graphs. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5657–5660. [CrossRef]
13. Sun, S.; Cho, S.G.; Zhang, Z. Post-processing methods for improving coding gain in belief propagation decoding of polar codes. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Singapore, 4–8 December 2017; pp. 1–6.
14. Arlı, A.C.; Gazi, O. Noise-aided belief propagationlist decoding of polar codes. *IEEE Commun. Lett.* **2019**, *23*, 1285–1288. [CrossRef]
15. Yu, Y.; Pan, Z.; Liu, N.; You, X. Belief propagation bit-flip decoder for polar codes. *IEEE Access* **2019**, *7*, 10937–10946. [CrossRef]
16. Shen, Y.; Song, W.; Ren, Y.; Ji, H.; You, X.; Zhang, C. Enhanced belief propagation decoder for 5G polar codes with bit-flipping. *IEEE Trans. Circuits Syst. II Exp. Briefs* **2020**, *67*, 901–905. [CrossRef]
17. Shen, Y.; Song, W.; Ji, H. Improved Belief Propagation Polar Decoders With Bit-Flipping Algorithms. *IEEE Trans. Commun.* **2020**, *68*, 6699–6713. [CrossRef]
18. Zhang, J.; Wang, M. Belief Propagation Decoder With Multiple Bit-Flipping Sets and Stopping Criteria for Polar Codes. *IEEE Access* **2020**, *8*, 83710–83717. [CrossRef]
19. Teng, C.; Yeu, A.; Wu, A. Convolutional neural network-aided tree-based bit-flipping framework for polar decoder using imitation learning. *IEEE Trans. Signal Process.* **2020**, *69*, 300–313. [CrossRef]
20. Yang, Y.; Yin, C.; Jan, Q.; Hu, Y.; Pan, Z.; Liu, N.; You, X. Noise-aided belief propagation list bit-flip decoder for polar codes. In Proceedings of the 2020 International Conference on Wireless Communications and Signal Processing(WCSP), Wuhan, China, 21–23 October 2020; pp. 807–810.
21. Feng, B.; Liu, R.; Tian, K. A Novel Post-Processing Method for Belief Propagation List Decoding of Polar Codes. *IEEE Commun. Lett.* **2021**, *25*, 2468–2471. [CrossRef]
22. Zhang, M.; Li, Z.; Xing, L. An Enhanced Belief Propagation Decoder for Polar Codes. *IEEE Commun. Lett.* **2021**, *25*, 3161–3165. [CrossRef]
23. Hussami, N.; Korada, S.B.; Urbanke, R. Performance of polar codes for channel and source coding. In Proceedings of the IEEE International Symposium on Information Theory (ISIT), Seoul, Korea, 28 June–3 July 2009; pp. 1488–1492.
24. Eslami, A.; Pishro-Nik, H. On bit error rate performance of polar codes in finite regime. In Proceedings of the 48th Annual Allerton Conference on Communication, Control, and Computing(Allerton), Monticello, IL, USA, 29 September–1 October 2010; pp. 188–194.

25. Eslami, A.; Pishro-Nik, H. A practical approach to polar codes. In Proceedings of the IEEE International Symposium on Information Theory (ISIT), Saint Petersburg, Russia, 31 July–5 August 2011; pp. 16–20.
26. Chandesris, L.; Savin, V.; Declercq, D. An improved SCFlip decoder for polar codes. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
27. Hashemi, S.A.; Condo, C.; Gross, W.J. Fast and Flexible Successive-Cancellation List Decoders for Polar Codes. *IEEE Trans. Signal Process.* **2017**, *65*, 5756–5769. [CrossRef]