electronics

Article

Improved Belief Propagation List Decoding for Polar Codes

Huan Li, Jingxuan Huang * and Ce Sun

School of Information and Electronics, Beijing Institute of Technology (BIT), Beijing 100081, China
* Correspondence: jxhbit@gmail.com

Abstract: Polar codes have become the channel coding scheme for control channel of enhanced mobile broadband in the 5G communication systems. Belief propagation (BP) decoding of polar codes has advantages of low decoding latency and high parallelism but achieves worse bit error ratio (BER) performance compared with the successive cancellation list (SCL) decoding scheme. In this paper, an improved BP list (IBPL) decoding algorithm is proposed with comparable BER performance to SCL algoritm. Firstly, the optimal permuted factor graph is analyzed for polar codes, which improves the performance of the BP decoder without path extension. Furthermore, based on the optimal graph, the bit metric and decoding path metric are proposed to extend and prune the decoding path. The proposed IBPL decoder is focused on not only the permutation of polar codes but also the reliabilities of decoded codewords during each iteration of BP decoding, which has a more accurate decoding path list. The simulation results show that the proposed IBPL decoder improves the BER performance compared with the original BP decoder significantly, and can approach the performance of the SCL decoder at low signal to noise ratio regions.

Keywords: polar codes; belief propagation list; belief propagation flip; permuted factor graph; path extension; path pruning

Citation: Li, H.; Huang, J.; Sun, C. Improved Belief Propagation List Decoding for Polar Codes. *Electronics* **2022**, *11*, 2458. https://doi.org/10.3390/electronics11152458

Academic Editor: Stefanos Kollias

Received: 18 July 2022
Accepted: 4 August 2022
Published: 7 August 2022

1. Introduction

Polar codes were proposed by Arikan in 2009 [1], which is the first channel capacity achieving family of error correction codes for binary-input discrete memoryless channels. Polar codes have structured coding construction and low decoding complexity. Due to those properties, polar codes were selected as the channel coding scheme for the control channel of enhanced mobile broadband (eMBB) in the fifth generation (5G) communication. In practice, the decoding algorithm has a significant influence on the performance of polar codes. Therefore, the investigation of decoding algorithms for polar codes is crucial for better serving the next generation mobile communication system [2] and the ultraviolet communication system [3].

Arikan proposed the successive cancellation (SC) decoder in [1], which has the advantage of low-complexity, but suffers from high latency and error propagation. Based on the SC decoder, the authors in [4–6] utilized the special node structure to perform parallel decoding of multiple bits to reduce the decoding latency. In the aforementioned literature, special nodes were constructed by the indexes of frozen bits and information bits, which were utilized to design the SC-based algorithms, such as the simplified SC (SSC) [4], the maximum likelihood SSC (MLSSC) [5], and the fast SSC (FSSC) [6] algorithms. However, such decoding algorithms change with different special node structures, which are not identical at different code lengths and rates. Thus, these algorithms have a lack of adaptability. To decrease the error propagation in SC decoders, the successive cancellation list (SCL) decoder was proposed in [7], which extended the decoding paths to L. The SCL algorithm can be regarded as a breadth-first search algorithm, which keeps "zero" and "one" decoding paths for each information bit, hence improving the decoding reliability of polar codes. Moreover, the path pruning is executed when the number of decoding paths reaches the maximum value, to reduce the high complexity introduced by path extension.

Compared with the SC-based decoders, the belief propagation (BP) algorithm has the unique advantage of low decoding latency and high parallelism, which suits well with hardware implementation. Although BP algorithm has particular advantages with respect to the decoding latency and throughput, it has poor performance compared to SCL algorithm. There are many efforts that have been spent on enhancing the performance of BP polar decoder, among which the BP decoding with path extension has become a prevailing approach.

The BP decoding with path extension can be classified into two kinds of methods, i.e., the path extension based on permutations of factor graph, and the path extension based on bit flip. The construction of polar codes was represented as the factor graph in [8], based on that, the performance of the BP decoding was investigated in [9,10]. The results show that for a polar code of length 2^n, there exists $n!$ different permutations of the factor graph, each of which corresponds to a different decoding performance. The authors proposed a series of decoding schemes based on permutations of factor graphs in [11–14], called BP list (BPL) algorithms. Especially, when the sum-product algorithm is utilized coupled with cyclic redundancy check (CRC) bits, the performance of BPL algorithms can be significantly improved at high signal to noise ratio (SNR) regions. Specifically, the decoding paths of BP decoder were extended based on a different stage-permuted factor graph in [13]. However, this scheme lacked the corresponding theoretical analyses. Another method to improve the performance of BP decoder is bit flipping (BPF), which has been successfully applied to SC decoder. Based on the critical set (CS) that is originally proposed for SC flipping decoder, a BP flipping decoder was proposed in [15], which revealed the relationship between CS and the incorrect BP decoding results. Specifically, the BPF decoder is oriented to the current decoded bits due to the use of nodes' information in the decoding progress, such as log likelihood ratios (LLRs) and parity check status. However, BPF algorithm suffers from the failure of identifing the error bits due to the inaccuracy of the CS. In [16], a generalized BPF (GBPF) decoding algorithm was proposed to enhance the performance of the BPF algorithm in [15], which defined the flipped bit as a frozen bit, thus fixing the bit value as the flipping one of the original BP decoding result. In addition, the enhanced BPF (EBPF) decoding algorithm [16] optimized the GBPF by reducing the search range of unreliable bits. The authors in [17] proposed an advanced BPF decoder, which utilized one critical bit in CS and the optimized sorting network to improve the decoding efficiency in the hardware level. The decoding errors of BP-based decoders can be categorized into detected and undetected errors according to the parity status [18]. Based on these two error types, a high-order GBPF decoder was proposed in [19]. The GBPF decoder in [19] presented two methods for generating flipped bits, which outperformed the BPF decoders in [16] by few decoding attempts. However, the existing BPF based algorithm has the shortcoming of selecting the CS in response to the SC decoder, instead of the BP decoder. In reality, there lies differences between the error pattern of the BP decoder and the error pattern of the SC decoder. Thus, the BP decoding algorithm with bit flipping based on CS can be further advanced.

In addition to the aforementioned BPL-based and BPF-based path extension methods, there still exists some improvements to BP decoders from other perspectives. In [20], an LLR-evolution-based algorithm was proposed to select the information and frozen bit sub-channels, which achieves a BER performance gain. A low-complexity BP decoder for polar codes was proposed in [21], where early termination of nodes was applied if the LLRs of nodes in graph message exceed the proposed LLR threshold. Though these approaches have improved the performances of BP-based polar decoders, however, the SCL decoding algorithm still outperforms the BP-based decoding algorithms significantly.

In this paper, an improved BPL (IBPL) decoder is proposed. Firstly, we analyze how the permutation of the factor graph influences the performance of the BP decoder and propose an optimal graph permutation for polar codes. Based on that graph, the bit parity-check metric (BPM) is defined to select the flipping bit. According to the BPM, the decoding path is extended based on the current codeword and current iteration of the BP algorithm. At the same time, in order to reduce the complexity of the algorithm, the path pruning is proposed with decoding path metric (DPM). Simulation results show that the proposed IBPL decoder can approach the performance of the SCL decoder at the low SNR region, and significantly reduce the performance gap between the SCL decoder and the BP-based decoders under high SNR. To conclude, the innovations of the proposed IBPL algorithm are as follows:

- We optimize the permutation of factor graph for message iteration to achieve higher reliability of the decoding, which also reduces the complexity of the algorithm.
- We propose the metrics of bit reliability and decoding path reliability for the current codeword and the current iteration, which make the path expansion and pruning more accurate.
- The proposed algorithm flips one bit at a time and employs the flip-iteration-metric-expansion mechanism to further improve the decoding performance.

The rest of this paper is organized as follows. In Section 2, we introduce the basis of polar codes and the BP algorithm. The optimal permutation of factor graph for polar codes is described in Section 3. Based on the optimal permutation, we design the path extension and pruning of polar codes in Section 4. In Section 5, we present simulation results. Finally, Section 6 concludes the paper.

The lower-case and upper-case boldface letters denote vector and matrix, respectively. $\mathbf{F}_2^{\otimes n}$ denotes the nth Kronecker product of the matrix $\mathbf{F}$. $\mathcal{F}^C$ represents the complement of set $\mathcal{F}$. $|\cdot|$ denotes the magnitude of a vector, or the size of a set. $sign(\cdot)$ denotes the function that takes the sign of a number (positive or negative), i.e., $sign(x) = 1$, for $x > 0$; $sign(x) = -1$, for $x < 0$; and $sign(x) = 0$, for $x = 0$.

2. Polar Codes and Belief Propagation Decoder

In this section, we introduce polar encoding and the conventional BP decoding algorithms as the basis of the proposed scheme.

2.1. Polar Codes

Polar codes are based on the channel polarization clearing that the partial polarized sub-channels tend to be noise-free channels and others tend to be full-noise channels as the code length N goes to infinity. Specifically, the fraction for the noise-free channels approaches the channel capacity. In this paper, we denote $I\left(W_N^{(i)}\right)$ as the channel capacity of $W_N^{(i)}$, where $W_N^{(i)}$ is the ith sub-channel of N sub-channels. $I\left(W_N^{(i)}\right)$ is also defined as the reliability of the channel.

As described above, the information bits are set in the sub-channel set $\Phi = \left\{W_N^{(i)} | I\left(W_N^{(i)}\right) \in (1-\delta, 1]\right\}$ and the frozen bits are set in the rest sub-channels to construct the information block u. The conventional algorithms to calculate the reliability $I(W_N^{(i)})$ include algorithms based on Bhattacharyya parameters, density evolution, Gaussian approximation, and extrinsic information transfer evolution. Polar encoding is denoted as $\mathbf{y}_1^N = \mathbf{u}_1^N \mathbf{G}_N$, where $\mathbf{y}_1^N = [y_1, y_2, \ldots, y_N]$ is the codeword, $\mathbf{u}_1^N = [u_1, u_2, \ldots, u_N]$ is the information block, and $\mathbf{G}_N$ is the generator matrix of order N. The recursive definition of $\mathbf{G}_N$ is given by:

$$\mathbf{G}_N = \mathbf{B}_N \mathbf{F}_2^{\otimes n}, \tag{1}$$

$$\mathbf{F}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \tag{2}$$

where $\mathbf{B}_N$ is the permutation matrix.

The construction of polar codes can be represented by Figure 1 using (1) with $n = 3$, $N = 2^n = 8$, where $[\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_8]$ denotes the message to be encoded, and $[y_1, y_2, \ldots, y_8]$ denotes the codeword by polar coding.

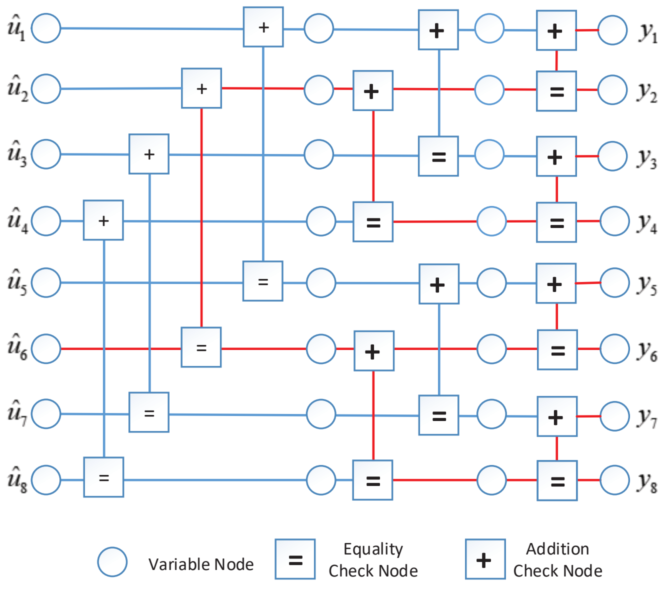

Figure 1. The processing of construction of polar codes.

2.2. *Belief Propagation Decoder*

The BP polar decoder is designed based on the polar coding construction that can be represented by a factor graph, as shown in Figure 1. The factor graph consists of $log_2 N + 1$ layers, where each stage has N variable nodes (VNs) and adjacent layers are connected with each other by N check nodes (CNs). The CNs are divided into the additional CNs and the equality CNs according to their functions "+" and "=", respectively. Four adjacent nodes in two layers form a 2-right-port basic computational block (2×2 BCB), as shown in Figure 2. Two types of messages are propagated through the 2×2 BCB: right-to-left messages L and left-to-right messages R. $L^t_{l,i}$ ($R^t_{l,i}$) denotes the right-to-left (left-to-right) message of the ith VN of layer l in the t-th iteration. Specifically, L and R are computed as:

$$\begin{cases} L^t_{l,i} & = g(L^{t-1}_{l+1,i}, L^{t-1}_{l+1,i+2^l} + R^{t-1}_{l,i+2^l}), \\ L^t_{l,i+2^l} & = g(R^{t-1}_{l,i}, L^{t-1}_{l+1,i}) + L^{t-1}_{l+1,i+2^l}, \\ R^t_{l+1,i} & = g(R^{t-1}_{l,i}, L^{t-1}_{l+1,i+2^l} + R^{t-1}_{l,i+2^l}), \\ R^t_{l+1,i+2^l} & = g(R^{t-1}_{l,i}, L^{t-1}_{l+1,i}) + R^{t-1}_{l,i+2^l}, \end{cases} \tag{3}$$

where:

$$g(x, y) = 2\tanh^{-1}(\tanh(x/2)\tanh(y/2)). \tag{4}$$

The left most VNs $R^0_{0,j}$ are initiated as:

$$R^0_{0,i} = \begin{cases} 0, & if \quad i \in \mathcal{A}, \\ \infty, & if \quad i \in \mathcal{A}^C, \end{cases} \tag{5}$$

where $i \in \mathcal{A}$ denotes the bit i is an information bit, and $i \in \mathcal{A}^C$ denotes the bit i is a frozen bit, respectively. Morever, messages of right most VNs $L^0_{L,j}$ are initiated as:

$$L^0_{log_2 N+1,i} = \ln \frac{P(y_i|\tilde{u}_i = 0)}{P(y_i|\tilde{u}_i = 1)}, \tag{6}$$

where $P(y_i|\tilde{u}_i)$ denotes the probability estimated by y_i during the decoding progress. In addition, messages of other nodes are initiated as zeros.

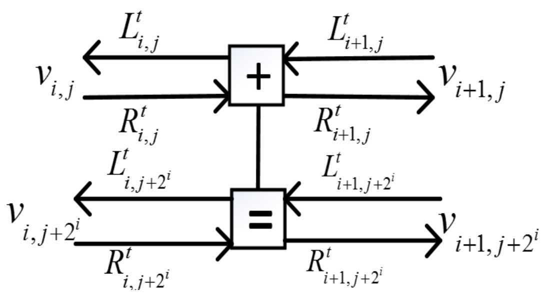

Figure 2. A 2-left-port 2-right-port basic computational block.

After the decoder reaches the maximum iteration number T_{max}, $\tilde{\mathbf{u}}_1^N = [\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_N]$ can be hard decided at the left most nodes as:

$$\tilde{u}_j = \begin{cases} 0, & if \quad R_{0,i}^{T_{max}} > 0, \\ 1, & otherwise, \end{cases} \tag{7}$$

where $j = 1, 2, \ldots, N$.

3. Design of Factor Graph Permutating for the IBPL Decoder

There are $n!$ permutations that can generate the same codeword for polar codes of length 2^n. Each permutation of the factor graph based on Arikan kernel [9] can be regarded as $\prod$, where each element in π_j represents a permutation in any random order, denoted as $(\prod)_{1\times n} = \text{randperm}(n)$. In the layer j of factor graph, the interval between two VNs that are linked in the same CN is denoted as s_j, where $s_j = 2^{\pi_j - 1}$ and $1 < j < n$. Figures 3 and 4 show a pair of permutations of (8, 3) polar codes, where the red nodes and green nodes denote frozen bits and information bits, respectively. The permutation of Figure 3 is denoted as $\prod = [3, 2, 1]$, with interval set $\mathbf{s} = [4, 2, 1]$, and the permutation of Figure 4 is denoted as $\prod = [2, 1, 3]$, with interval set $\mathbf{s} = [2, 1, 4]$.

During the iteration of BP decoding, the nodes with frozen bits are of vital significance. The value of these nodes can be determined before decoding, thus they help other nodes with error detection and correction. At the same time, observing a BCB, if the two left nodes of the BCB are frozen bits, then the two right nodes of this BCB are determined. Therefore, these right nodes also play the role of frozen bits in the decoding graph. We define such nodes as frozen nodes (FNs), which are indicated as orange circles in Figures 3 and 4.

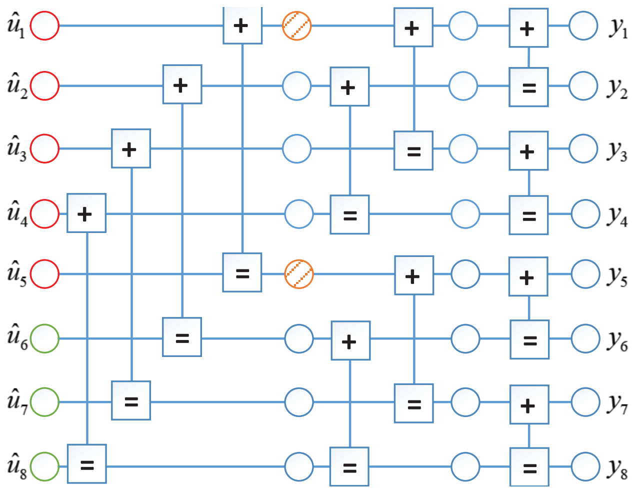

Figure 3. The factor graph of $(8, 3)$ polar codes with interval set $\mathbf{s} = [4, 2, 1]$.

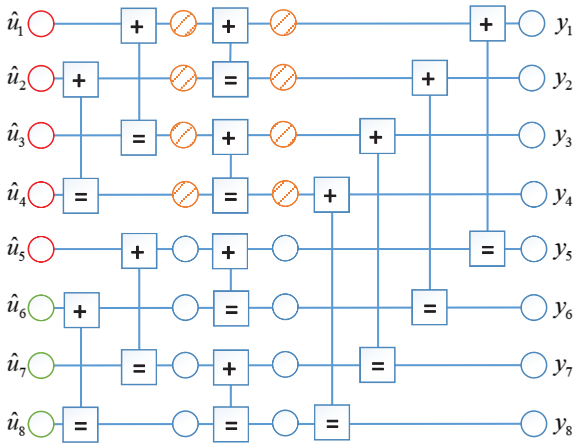

Figure 4. The factor graph of $(8,3)$ polar codes with interval set $\mathbf{s} = [2,1,4]$.

It can be further observed from Figures 3 and 4 that the number of FNs in the factor graph with different permutation is also different. In addition, according to the recursive structure shown as (1) of polar codes, changing the permutation of the factor graph does not affect the encoding result. Correspondingly, the permutation of the factor graph of the BP decoder can be arbitrary for the same codeword, which has no influence on the encoding process of polar codes. Based on the above observations, more FNs can provide more reliable messages to achieve better performance with no impact on the encoding process. Thus, we propose an optimal scheme to construct as many FNs as possible, which permutates the factor graph according to the increasing order of s. Specifically, in our proposed optimal s-ascending scheme, the interval set is permutated as $\mathbf{s} = [s_1, s_2, \dots s_n]$, where $s_j > s_k$ if $j > k$. Since the frozen bits are relatively concentrated, sorting the polarization layers in an increasing order of the $\mathbf{s}$ set enables the frozen bits to be embedded into the same BCB in the previous layer, thereby generating more FNs. As will be presented in Section 5, our proposed s-ascending scheme achieves the most FNs and better performance compared to other permutation methods.

In addition, LLRs of FNs can be assumed to be infinite and no update is required, since FNs always provide reliable messages. To be more specific, LLRs of FNs in the decoding process can be expressed as (5) and:

$$R^0_{i+1,j} = \begin{cases} \infty, & if \quad R^0_{i,j+2^i} = R^0_{i,j} = \infty, \\ 0, & if \quad otherwise, \end{cases} \tag{8}$$

which defines the LLRs of FNs produced by two left FNs of a BCB. Consequently, the complexity of the optimal permutation method can be decreased because there are more FNs in the factor graph.

Next, we propose the decoding path expansion and pruning method to improve the decoding reliability based on the proposed s-ascending permutation.

4. Design of Path Extension and Pruning for the IBPL Decoder

4.1. Design of Path Extension for the IBPL Decoder

In this section, decoding path expansion is designed aimed at each iteration of BP decoding by measuring the reliability of each encoded bit in the BP decoding process.

From the construction of polar codes, we can observe that polar codes are constructed by recursively polarizing BCBs, which can be shown as Figure 2. Specifically, a BCB satisfies the check relationship when encoding:

$$\begin{aligned} R_{l+1,i} &= L_{l,i} \oplus L_{l,i+2^{l-1}}, \\ R_{l+1,i+2^{l-1}} &= L_{l,i+2^{l-1}}. \end{aligned} \tag{9}$$

Similar to the construction of polar codes, the BP decoding also consists of BCBs recursively. More specifically, BP decoding contains totally $log(2N+1)$ polarization layers, and each layer contains $N/2$ BCBs, where the LLRs are transferred iteratively. The decoding result of the BP decoder is determined by the last iteration from right to left in the BCB, so the BCB check relationship of the last iteration from right to left represents the reliability of its output directly. Set the decision results of node (l,i) in the last iteration from right to left as $\tilde{u}_{l,i} = \left[1 - sign\left(L_{l.j}^{T_{\max}}\right)\right]/2$. The nodes pass the check if the decision results of nodes in one unit satisfy (10), which can be denoted as the parity-check passed nodes (PPNs).

$$\begin{aligned} &\tilde{u}_{l,i} = \tilde{u}_{l+1,i} \oplus \tilde{u}_{l+1,i+2^{l-1}}, \\ &\tilde{u}_{l,i+2^{l-1}} = \tilde{u}_{l+1,i+2^{l-1}}. \end{aligned} \tag{10}$$

For $\tilde{u}_i$, the units involved in the decoding calculation during the last right-to-left iteration constitute the bit decision path of $\tilde{u}_i$. For example, as shown in Figure 1, red lines constitute the bit decision path of $\tilde{u}_6$. The check results of the BCB in the bit decision path directly reflect the decoding accuracy of the bit. Then, the bit parity-check path metric (BPPM) is defined to measure the reliability of each decoded bit in the BP decoding process. The BPPM of $\tilde{u}_i$ (i.e., $\tilde{u}_{1,i}$) denoted as $BPPM_{1,i}$ is calculated as:

$$\begin{aligned} &BPPM_{1,i} \overset{\Delta}{=} -\ln[p(ppn_{1,i}|Y=y)] \\ &\overset{(a)}{=} -\ln[p(ppn_{1,i}|ppn_{2.i}, ppn_{2.i+1})p(ppn_{2.i}, ppn_{2.i+1}|Y=y)] \\ &\overset{(b)}{=} -\ln[p(ppn_{1,i}|ppn_{2.i}, ppn_{2.i+1})] - \ln[p(ppn_{2,i}|Y=y)] - \ln[p(ppn_{2.i+1}|Y=y)] \\ &\overset{(c)}{=} -\ln\left(\frac{1}{1+e^{-L_{1,i}^{t_{\max}}(-1)^{\tilde{u}_{1,i}}}}\right) + BPPM_{2,i} + BPPM_{2.i+1} \\ &\overset{(d)}{=} \ln\left(1+e^{-L_{1,i}^{t_{\max}}(-1)^{\tilde{u}_{1,i}}}\right) + BPPM_{2,i} + BPPM_{2.i+1} \\ &\overset{(e)}{=} \sum_{(l,j)\in\mathbf{B}} \ln\left(1+e^{-L_{l,j}^{t_{\max}}(-1)^{\tilde{u}_{l,j}}}\right), \end{aligned} \tag{11}$$

where $p(ppn_{1,i}|Y=y)$ represents the decoding accuracy of $\tilde{u}_i$ through the probability of passing the parity check, and set $\mathbf{B}$ contains all the nodes involved in the bit decision path. Equation (a) holds because in the bit decision path, if the left node of the BCB passes the check, the corresponding right node should also pass the check. Equation (b) follows from the independence between two right nodes of the BCB during the last iteration from right to left. In Equation (c), $L_{1,i}^{t_{\max}}$ is the LLR value of node $(1,i)$ in the t_{max}th iteration from right to left; that is, $L_{1,i}^{t_{\max}} = \ln[p(\tilde{u}_{1,i}=0)/p(\tilde{u}_{1,i}=1)]$.

The above calculation of $BPPM$ can be summarized by:

$$BPPM_{1,i} = \phi\left(BPPM_{2,i} + BPPM_{2,i+1}, L_{1,i}^{t_{\max}}, \tilde{u}_{1,j}\right), \tag{12}$$

where the function $\phi(\cdot)$ is defined as:

$$\phi(x,y,z) \overset{\Delta}{=} x + \ln\left(1+e^{-y(-1)^z}\right). \tag{13}$$

For the reason that:

$$\ln(1+e^x) \approx \begin{cases} 0 & x<0, \\ x & x \geq 0, \end{cases} \tag{14}$$

the function $\phi(\cdot)$ can be approximated as:

$$\phi(x,y,z) \overset{\Delta}{=} \begin{cases} x & z = \frac{1}{2}(1-sign(y)), \\ x+|y| & z \neq \frac{1}{2}(1-sign(y)). \end{cases} \tag{15}$$

By substituting (12) into (15), we can obtain that:

$$BPPM_{1,i} = \begin{cases} BPPM_{2,i} + BPPM_{2,i+1} & \tilde{u}_{1,j} = \frac{1}{2}\left(1 - sign\left(L_{1,i}^{t_{\max}}\right)\right), \\ BPPM_{2,i} + BPPM_{2,i+1} + \left|L_{1,i}^{t_{\max}}\right| & \tilde{u}_{1,j} \neq \frac{1}{2}\left(1 - sign\left(L_{1,i}^{t_{\max}}\right)\right). \end{cases} \tag{16}$$

where $\frac{1}{2}\left(1 - sign\left(L_{1,i}^{t_{\max}}\right)\right)$ is the decision value expressed by LLR, which represents the decision result of BP decoding. If the node in the bit decision path does not meet the BCB check after BP decoding, the bit decision path is a performed penalty with the penalty value approximated by $\left|L_{1,i}^{t_{\max}}\right|$.

Based on the BPPM, in the last iteration from right to left of BP decoding, the decoder also calculates the decision results of the nodes to check each BCB in addition to calculating the LLR values. Then, the BPPM of the information bit is obtained by (11). A larger BPPM means that more nodes fail the check in the decision path, thus the reliability is worse. After t_{max} iterations, the BPPM of the information bits are sorted. The information bit with the largest BPPM is flipped to expand the decoding path, and it is fixed as a frozen bit whose initial LLR value is set to infinity. Then, the two expanded decoding paths execute BP decoding for t_{max} iterations.

4.2. Design of Path Pruning for the IBPL Decoder

In this section, decoding path pruning is designed aimed at each iteration of BP decoding by measuring the reliability of each path in the BP decoding process.

Each round of path expansion to the bit with the largest BPPM doubles the number of decoding paths. If path pruning is not performed, the complexity of the algorithm will be impractical. Therefore, we define the decoding path metric, and propose a path pruning algorithm based on it to reduce the complexity of the algorithm.

In the SCL decoder, the path metric is decided by the decision results of the frozen bits. Because the frozen bits are determined previously by both the transmitter and the receiver, the decision results of which can be utilized to judge whether the decoding path is a failure or not. Specifically, the decoding path can be considered to be a failure, if the corresponding decision of frozen bits according to the soft information do not match the actual results, i.e.,

$$1 - sign(LLR_i) \neq u_i, \qquad i \in \mathcal{A}^C. \tag{17}$$

Similarly, frozen bits also play a vital role in the BP decoding algorithm. Before the iteration, the LLRs of the frozen bits need to be set to infinity to provide reliable information for the error correction of other nodes. Therefore, the decision accuracy of the frozen bit during the iteration will directly represent the reliability of the current decoding path. We define the frozen-bit error rate (FBER) $FBER$ as:

$$FBER^k = \frac{\frac{1}{2}\sum\limits_{i \in \mathcal{F}}\left(1 - sign\left(L_{1,i}^{k,t_{\max}}\right)\right)}{|\mathcal{F}|}, \tag{18}$$

where k represents the kth decoding path, and $\mathcal{F}$ denotes the set of frozen bits. FBER represents the ratio of frozen bit decision errors in the last iteration of BP decoding. It should be noted that in the iteration of the first $t_{max} - 1$ round, the frozen bits do not need to be decided, but participate in the calculation with infinite soft information. The decision of frozen bits is only necessary when calculating the decoding path metric. A smaller FBER means that this decoding path is more reliable.

Denote n_{path} as the total number of current decoding paths, and n_{max} as the given maximum number of decoding paths. After each time the path expansion is completed, if $n_{path} > n_{max}$, the FBER of each path is calculated, and the n_{max} paths with the smallest FBER are retained to continue BP iterative decoding. The path with the smallest FBER is selected as the final decoding result when the maximum iteration number for path extension T_{max} is reached. Note that different from the existing BPL algorithms, the proposed IBPL decoder not only utilizes new bit metric and path metric, but also flips 1 bit each path expansion time, and iteratively flips again. Cooperating with the proposed real-time bit decision metric, this method finds the wrong bits more accurately for path expansion. The complete algorithm is concluded as Algorithm 1, and the corresponding flow is shown in Figure 5.

Algorithm 1 The proposed IBPL decoder.

```
Input: The maximum iteration number for path extension $T_{max}$, the maximum iteration
    number for iterative BP decoding $t_{max}$, and the maximum number of decoding paths
    $n_{max}$.
 1: Initialize LLRs $R^0$ and $L^0$ according to (5) and (6), respectively.
 2: Set $n_{path} = 1$.
 3: for $T = 1$ to $T_{max}$ do
 4:     for $k = 1$ to $n_{path}$ do
 5:         Perform the s-ascending permutation of graph.
 6:         for $t = 1$ to $t_{max} - 1$ do
 7:             Calculate LLRs $L$ and $R$ according to (3).
 8:         end for
 9:         Calculating BPPM of bits in the last iteration according to (12), and filp the bit
    with the largest BPPM.
10:     end for
11:     Renew the total number of decoding paths $n_{path}$.
12:     for $k = 1$ to $n_{path}$ do
13:         for $i \in \mathcal{F}$ do
14:             Decide the frozen bits in the $t_{max}$th iteration.
15:         end for
16:         Calculate the FBER $FBER^k$ of $k$th decoding path according to (18). Specifically,
    FBER of the extended path can be calculated via (18) with the opposite value of $L_{1,i}^{k,t_{max}}$.
17:         Retain the $n_{max}$ paths with the smallest FBER.
18:     end for
19: end for
20: Select the path with the smallest FBER.
21: Perform decoding decision according to (7) for the selected path.
Output: The decoding result $\tilde{\mathbf{u}}$.
```

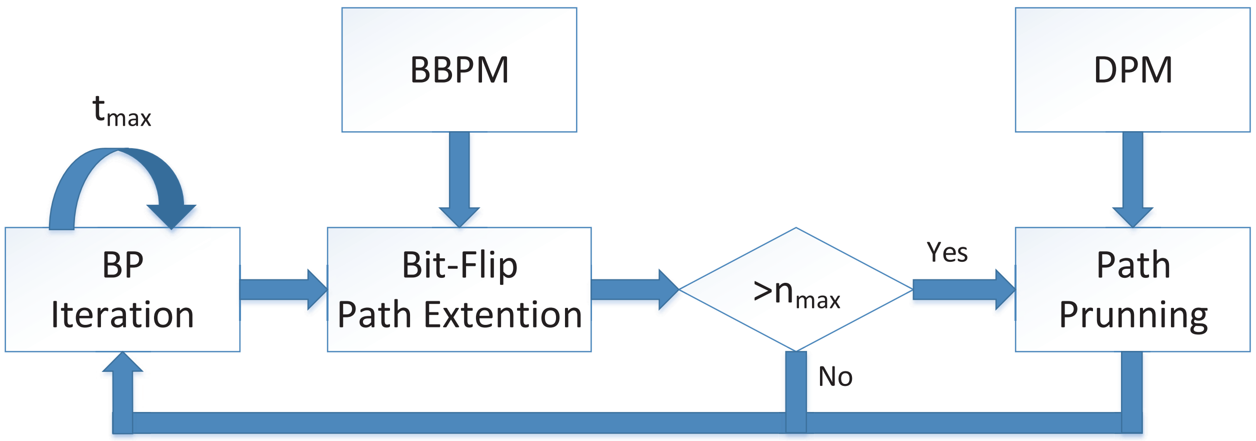

Figure 5. The algorithm flow of IBPL decoder.

5. Simulation Results

The simulation results and complexity analyses are presented in this section to verify the performance of the s-ascending permutation scheme of factor graph and the proposed IBPL decoder under the additive Gaussian white noise (AWGN) channel. The main simulation parameters are set as Table 1 unless otherwise stated.

Table 1. System Simulation Parameter Settings.

Parameter	Values
Channel model	AWGN
Modulation mode	BPSK
Length of codeword, N	128, 256
Code rate	1/2
Maximum number of decoding paths, n_{max}	4
Maximum iteration number for path extension of the proposed IBPL decoder, T_{max}	5
Maximum iteration number for inner BP decoding of the proposed IBPL decoder, t_{max}	20
Maximum iteration number for BP decoders of other schemes, $\hat{t}_{max}$	100

5.1. Performance Comparison for Different Permutations of Factor Graph

In this section, the block error rate (BLER) performance of polar codes with different permutations of factor graph are simulated, where the code length is 256 bits and the code rate is 1/2. The BPL decoder with cyclic-shift permutation method is simulated for comparison, which is obtained by cyclic shifting the polarization layers in factor graph [13]. The s-descending scheme, which permutates the factor graph according to the descending order of intervals, is also simulated. Moreover, the random permutation method of factor graph is presented as a baseline. It should be noted that the maximum iteration number of BP decoders in this subsection is set as $\hat{t}_{max} = 100$.

Table 2 compares the number of FNs among different permutation methods. From Table 2, we can find that the number of FNs in the factor graph in s-descending permutation is the least while the number of FNs in the factor graph in s-ascending permutation is the most. Figure 6 and Table 2 indicate that the BP decoding performance of polar codes increases with the number of FNs. The s-ascending permutation has the most FNs, thus achieving the best BLER performance.

Table 2. The comparison of FNs' number among different permutations of factor graph.

Permutation	The Number of FNs (Rate)
s-descending	216 (9.38%)
Random	298 (12.93%)
Cyclic-shift	328 (14.24%)
Proposed s-ascending	446 (19.36%)

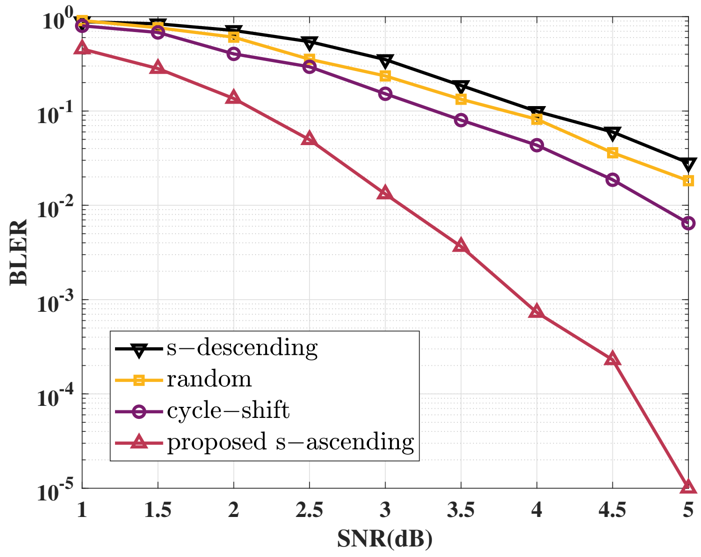

Figure 6. The comparison of BLER performance among different permutations of factor graph, where the code length is 256 bits and code rate is 1/2.

5.2. Performance Comparison of BP, BP-Flip, SCL and IBPL Decoding Algorithms

In this section, the BER and BLER performance of the proposed IBPL algorithm are simulated.

Figures 7 and 8 present the BER and BLER performance of the proposed IBPL polar decoder when the code length is 128 bits, and the code rate is 1/2. For comparison, the performance of BP-flip decoding algorithm based on CS in [15] is also simulated. The performance of SCL decoder [7] and BP decoder [11] is also simulated to better present the improvement of the proposed IBPL. The maximum numbers of iteration for BP-flip and BP decoders are set as $\hat{t}_{max} = 100$. It should be noted that the maximum number of extended paths of the SCL decoder and the proposed IBPL decoder are set as $n_{max} = 4$.

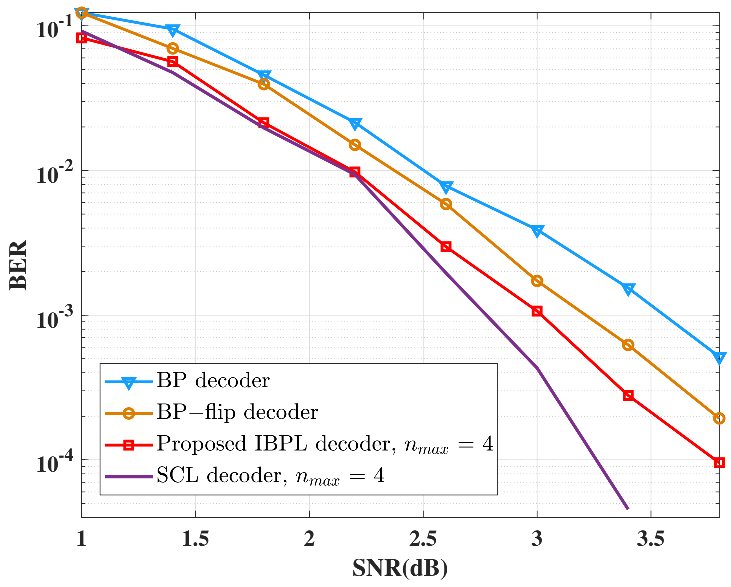

Figure 7. BER performance comparison among IBPL decoder, BP decoder, BP-flip decoder, and SCL decoder when code length is 128 bits, code rate is 1/2, and $n_{max} = 4$.

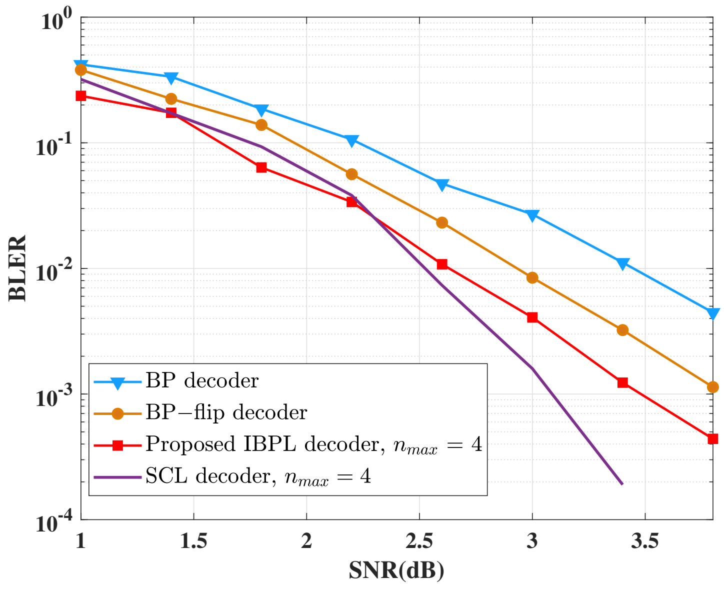

Figure 8. BLER performance comparison among IBPL decoder, BP decoder, BP-flip decoder, and SCL decoder when code length is 128 bits, code rate is 1/2, and $n_{max} = 4$.

It can be observed from Figures 7 and 8 that compared with the traditional BP decoding algorithm, the proposed IBPL algorithm has about a 0.7 dB gain. This gain comes from the expansion of the decoding path in the IBPL algorithm. Compared with the BP-flip algorithm, the proposed IBPL decoding algorithm achieves a 0.2 dB gain. This gain comes from the more accurate bit parity-check metric and decoding path metric of the proposed scheme. At the same time, unlike BP-flip that flips multiple bits at a time, the IBPL algorithm flips 1 bit each time, and then continues to the iteration-metric-expansion mechanism. The path extension is more accurate due to the metrics of reliability and further improves the performance of the BP decoder. In addition, the proposed IBPL algorithm can approach the performance of the SCL algorithm proposed in [7] at a low SNR region, and significantly reduce the performance gap between the SCL algorithm and BP-based algorithms under high SNR.

The complexity of the proposed IBPL decoder is mainly relevant to three loops in Algorithm 1, i.e., the outer-loop for path extension, the loop for the extended path to decode, and the inner-loop for BP decoding with the s-ascending permutation of the factor graph. Specifically, the complexity of BP decoding with the s-ascending permutation is composed of two parts, i.e., the complexity of the BP decoding and the complexity of the s-ascending sort of polarization layers. More specifically, the complexity of the BP decoder is $O(t_{max}NlogN)$ according to [22], and the complexity of the s-ascending sort can be represented as $O(logNlog(logN))$. Since the complexity $O(t_{max}NlogN)$ is significantly lager than $O(logNlog(logN))$, the complexity term $O(logNlog(logN))$ can be omitted. Therefore, we can conclude that the complexity of the BP decoding with s-ascending permutation in the third loop is $O(t_{max}NlogN)$. Consequently, the complexity of the proposed IBPL decoder can be concluded as $O(T_{max}n_{max}t_{max}NlogN)$. Moreover, the complexity of the proposed IBPL decoder is similar to the complexity of BP-flip and other BPL-based decoders in [13,23,24], which have the complexity as $O(n_{max}\hat{t}_{max}NlogN)$. Note that the maximum number of total iterations for BP decoding of the proposed IBPL decoder is calculated as $T_{max}t_{max} = \hat{t}_{max}$.

In Figures 9 and 10, we compare the performance of the proposed IBPL decoder under different maximum paths n_{max}. More specifically, we simulated the BER and BLER performance of the proposed IBPL decoder when the code length is set to 128 bits, and the code rate is set to 1/2. From Figures 9 and 10, we can see that as the maximum number of extended paths of BP decoding n_{max} increases, the reliability of the IBPL decoder also improves.

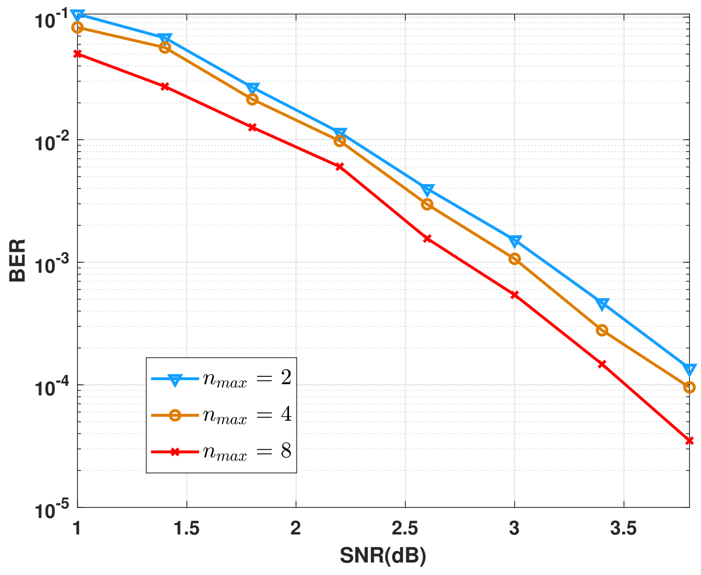

Figure 9. BER comparison of IBPL decoder under different number of maximum decoding paths when the code length is 128 bits and the code rate is 1/2.

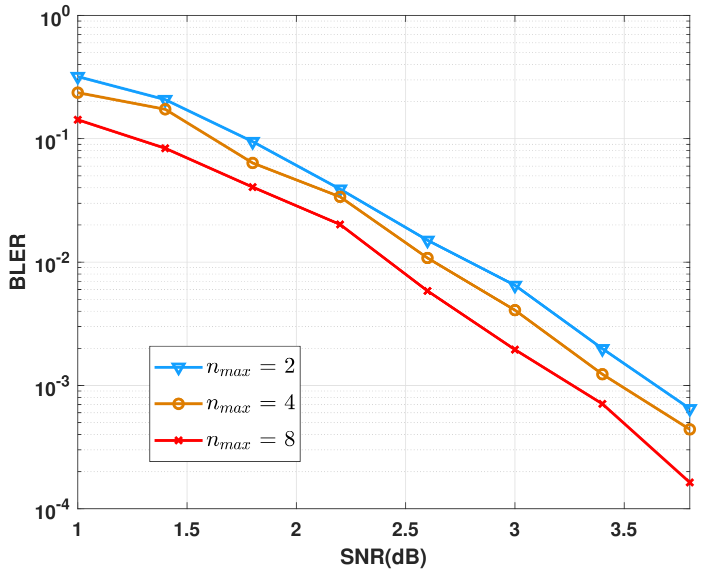

Figure 10. BLER comparison of IBPL decoder under different number of maximum decoding paths when the code length is 128 bits and code rate is 1/2.

6. Conclusions

In this paper, we proposed the IBPL decoding algorithm, which improves the BER and BLER performance of the exsited BP-based algorithms. The optimal graph permutation for polar codes was designed to maximize the number of FNs. Based on the optimal permutation, the decoding path expansion and pruning methods of the IBPL decoder were proposed to enhance the performance. For path extension and pruning, the bit parity-check metric and decoding path metric were defined respectively according to instant LLRs. The simulation results show that the proposed IBPL scheme outperforms the conventional BP decoder by 0.7 dB, and has a 0.2 dB gain compared to the existing BP flip algorithm.

Author Contributions: Methodology and writing—original draft preparation, H.L. and C.S.; software and writing—review and editing, J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Key R&D Program of China under grant 2020YFB1807203, in part by Ericsson, and in part by the National Natural Science Foundation of China under Grant No. 61871032.

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

Abbreviations

The following abbreviations are used in this manuscript:

BPM	Bit parity-check metric
DPM	Decoding path metric
2×2 BCB	2-right-port basic computational block
PPN	Parity-check passed node
BPPM	Bit parity-check path metric
FBER	Frozen-bit error rate

References

1. Arikan, 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. Liu, G.; Jiang, D. 5G: Vision and requirements for mobile communication system towards year 2020. *Chin. J. Eng.* **2016**, *2016*, 5974586. [CrossRef]
3. Hu, W.; Zhang, M.; Li, Z.; Popov, S.; Leeson, M.; Xu, T. High-dimensional feature based non-coherent detection for multi-intensity modulated ultraviolet communications. *J. Light. Technol.* **2022**, *40*, 1879–1887. [CrossRef]
4. Alamdar-Yazdi, A.; Kschischang, F.R. A simplified successive-cancellation decoder for polar codes. *IEEE Commun. Lett.* **2011**, *15*, 1378–1380. [CrossRef]
5. Sarkis, G.; Gross, W.J. Increasing the tThroughput of polar decoders. *IEEE Commun. Lett.* **2013**, *17*, 725–728. [CrossRef]
6. Sarkis, G.; Giard, P.; Vardy, A.; Thibeault, C.; Gross, W.J. Fast polar decoders: Algorithm and implementation. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 946–957. [CrossRef]
7. Tal, I.; Vardy, A. List decoding of polar codes. *IEEE Trans. Inf. Theory* **2015**, *61*, 2213–2226. [CrossRef]
8. Arikan, E. A performance comparison of polar codes and reed-mullercodes. *IEEE Commun. Lett.* **2008**, *12*, 447–449. [CrossRef]
9. Arikan, E. Polar codes: A pipelined implementation. In Proceedings of the 4th International Symposium on Broadband Communication (ISBC), Melaka, Malaysia, 11–14 July 2010; pp. 11–14.
10. Hussami, N.; Korada, S.; Urbanke, R. Performance of polar codes for channel and source coding. In Proceedings of the 2009 IEEE International Symposium on Information Theory, Seoul, Korea, 28 June–3 July 2009; pp. 1488–1492.
11. Elkelesh, A.; Ebada, M.; Cammerer, S.; ten Brink, S. Belief propagation decoding of polar codes on permuted factor graphs. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.
12. Doan, N.; Hashemi, S.A.; Mondelli, M.; Gross, W.J. 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.
13. Elkelesh, A.; Ebada, M.; Cammerer, S.; Ten Brink, S. Belief propagation list decoding of polar codes. *IEEE Commun. Lett.* **2018**, *22*, 1536–1539. [CrossRef]
14. Ren, Y.; Shen, Y.; Zhang, Z.; You, X.; Zhang, C. Efficient belief propagation polar decoder with loop simplification based factor graphs. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5657–5660. [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. Ji, H.; Shen, Y.; Song, W.; Zhang, Z.; You, X.; Zhang, C. Hardware implementation for belief propagation flip decoding of polar codes. *IEEE Trans. Circuits Syst.* **2021**, *68*, 1330–1341. [CrossRef]
18. Cavus, E.; Daneshrad, B. A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes. In Proceedings of the 2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, Berlin, Germany, 11–14 September 2005; Volume 4, pp. 2386–2390.
19. Shen, Y.; Song, W.; Ji, H.; Ren, Y.; Ji, C.; You, X.; Zhang, C. Improved belief propagation polar decoders with bit-flipping algorithms. *IEEE Trans. Commun.* **2020**, *68*, 6699–6713. [CrossRef]
20. Qin, M.; Guo, J.; Bhatia, A.; i Fàbregas, A.G.; Siegel, P.H. Polar code constructions based on LLR evolution. *IEEE Commun. Lett.* **2017**, *21*, 1221–1224. [CrossRef]
21. Zhang, Y.; Zhang, Q.; Pan, X.; Ye, Z.; Gong, C. A simplified belief propagation decoder for polar codes. In Proceedings of the IEEE International Wireless Symposium (IWS 2014), Xi'an, China, 24–26 March 2014; pp. 1–4.

22. Zhang, Y.; Liu, A.; Pan, X.; Ye, Z.; Gong, C. A modified belief propagation polar decoder. *IEEE Commun. Lett.* **2014**, *18*, 1091–1094. [CrossRef]
23. Marvin, G.; Ahmed, E.; Jannis, C.; Stephan, B. A polar subcode approach to belief propagation list decoding. *arXiv* **2022**, arXiv:2205.06631.
24. Geiselhart, M.; Elkelesh, A.; Ebada, M.; Cammerer, S.; ten Brink, S. CRC-aided belief propagation list decoding of polar codes. In Proceedings of the 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 21–26 June 2020; pp. 395–400.