An Enhanced Belief Propagation Flipping Decoder for Polar Codes with Stepping Strategy

Xiaojun Zhang 1,2,* , Yimeng Liu 1, Chengguan Chen 1, Hua Guo 1 and Qingtian Zeng 1

1 College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China; 202082130008@sdust.edu.cn (Y.L.); 201982130005@sdust.edu.cn (C.C.); stonestrong@sdust.edu.cn (H.G.); qtzeng@sdust.edu.cn (Q.Z.)

* Correspondence: zhangxiaojun@sdust.edu.cn

Abstract: The Belief Propagation (BP) algorithm has the advantages of high-speed decoding and low latency. To improve the block error rate (BLER) performance of the BP-based algorithm, the BP flipping algorithm was proposed. However, the BP flipping algorithm attempts numerous useless flippings for improving the BLER performance. To reduce the number of decoding attempts needed without any loss of BLER performance, in this paper a metric is presented to evaluate the likelihood that the bits would correct the BP flipping decoding. Based on this, a BP-Step-Flipping (BPSF) algorithm is proposed which only traces the unreliable bits in the flip set (FS) to flip and skips over the reliable ones. In addition, a threshold $\beta$ is applied when the magnitude of the log–likelihood ratio (LLR) is small, and an enhanced BPSF (EBPSF) algorithm is presented to lower the BLER. With the same FS, the proposed algorithm can reduce the average number of iterations efficiently. Numerical results show the average number of iterations for EBPSF-1 decreases by 77.5% when $N = 256$, compared with the BP bit-flip-1 (BPF-1) algorithm at $E_b/N_0 = 1.5$dB.

Keywords: polar code; belief propagation; bit-flipping; stepping

1. Introduction

Polar code is the first error-correcting code [1] which has achieved the Shannon limit. It has been adopted as the fifth generation (5G) wireless communications standard [2]. Not only does it have a strong error correction capability, but its encoding and decoding complexity is also affordable compared with the Low Density Parity Check (LDPC) [3] and Turbo codes [4].

The Successive Cancellation (SC) algorithm, which was proposed by Arıkan [5], is one of the most common decoding methods for polar codes and has attracted widespread attention. The original SC algorithm has been optimized by the relevant scholars, producing optimizations such as the SC list (SCL) [6], SC flip (SCF) [7] and dynamic SC flipping (DSCF) [8] algorithms. The Cyclic redundancy check (CRC)-aided SC list (CA-SCL) [9] decoder was introduced to improve the BLER of polar codes and has become a baseline algorithm used in the standardization process. Compared with the SC and its other optimized algorithms, the Belief Propagation (BP) algorithm [10] with parallel decoding has great advantages in terms of its throughput and decoding latency. In addition, BP decoding is expected to support polar codes of 5G standard in practical applications with a set of hardware units. However, the BP decoder will not terminate until the maximum number, the termination scheme suffers from lack of flexibility, and introduces great computational complexity. Moreover, the BLER performance of BP is uncompetitive.

To improve the original BP decoding performance, many methods to improve the BP-based algorithm have been proposed—for example, the BP list (BPL) [11] decoder. When using the standard polar code decoding factor graph (DFG), the original BP algorithm may generate incorrect decoding results because the transmission process of the messages is
modified. However, the BPL decoder uses the permuted version of the standard graph and can produce the correct estimation result. A permuted factor graph is introduced in [12,13] based on the BPL decoder. The guessing algorithm [14] was first proposed in the LDPC codes and then introduced to the BP decoding, which guesses the oscillating bits by assigning the priority log-likelihood ratios (LLRs), the maximum LLR magnitude, and repeating the original BP decoding to achieve good decoding results. The BP Bit-Flip (BPF) [15] algorithm, which is different from other BP-based evolutionary algorithms, utilizes the information in the decoding process to make it more targeted to the codeword being decoded.

BP flipping algorithms identify and then flip error-prone positions to improve decoding performance. For polar codes, bit-flipping strategies have been applied to both SC and BP decoding. However, the principles of flipping strategies on SC and BP decoding are different due to their different decoding schedules. The key to the BP flipping algorithm is to construct a flip set (FS), which consists of the error-prone bits. FS can accurately indicate the error bits, thus flipping the bits within FS can narrow the search space. Initially, the FS was constructed in SC [7], which proposed a progressive multi-bit-flipping algorithm. Under the Gaussian approximation (GA) construction, it was generated according to the Rate-1 nodes called the critical set (CS). The bits identified in the CS may exhibit a high possibility of error in the BP decoding. Thus, the CS is introduced to BP and the constructed CS with order $T$ (CS-T) [15], where $T$ denotes the size of the CS. By analyzing the behavior of the incorrect decoding results of the bit-flipping BP decoder with CS, a BP decoder with multiple bit-flipping sets (BFSs) and stopping criteria (BP-MF-MC) was proposed [16]. The CS is relatively static and the multiple BFSs have more complexity, so the Generalized BP Bit-flipping (GBPF) Decoder [17] with a redefinition of BP bit-flipping was proposed. The error-prone bits in the FS consist of the unflipped bits with small LLR magnitudes from the outputs, which contribute to a better performance. Subsequently, the GBPF decoding was extended to a higher-order GBPF algorithm (GBPF-$\Omega$) [18], where the maximum bit-flipping order is $\Omega$. To narrow the BLER performance gap between BP-based decoders and CA-SCL decoder, the bit-flip method was also introduced into the BPL decoder and the noise-aided BPL (NA-BPL) decoder [19] was proposed. Above all, although flipping error-prone positions after the failure of the first decoding trail will improve the performance of BP-based decoder, the characteristics of the bits in FS are selected to be flipped in turn and many of them have no contribution to correcting the error frame, which will result in a huge amount of invalid repeated decoding attempts, giving rise to more latency.

This paper aims to reduce the number of decoding attempts in the BP flipping algorithms. The main contributions are summarized as follows:

- A stepping strategy is proposed. We first analyze the behavior of FS and find that only a few bits in the FS could correct the error frame. Therefore, a concept to evaluate the likelihood of bits correcting the trajectory of BP decoding is presented to judge whether the bits in FS should be flipped or not. The judgement condition determines whether flipping the bits in the FS is helpful in correcting error frames.

- Based on the stepping strategy, an optimization algorithm for the BP flipping algorithm, the BP step-flipping (BPSF) algorithm, is proposed. The algorithm flips only unreliable bits in FS and steps reliable bits to shrink the number of flipping attempts necessary. Similarly, the stepping strategy is also added into the GBPF-$\Omega$ algorithm [17,18] to reduce the number of flipping attempts.

- In addition, we notice that some effective flipping bits may be skipped over when the LLR magnitude is small. We further propose the enhanced BPSF-$\Omega$ (EBSF-$\Omega$) algorithm, which adopts a threshold to identify the unreliable bits and lowers the block error rate (BLER). The numerical results obtained indicate that the average number of iterations can be significantly reduced for the EBSF-1 and EBSF-2 algorithms at the low $E_b/N_0$, compared with the BPF-1 and BPF-2 flipping algorithms when the code length is 256.
2. Preliminary

In this paper, we use calligraphic characters, such as $\mathcal{R}$, to denote sets. We write $r$, $r$, and $R$ to denote a scalar, a vector, and a matrix, respectively. In this section, we first describe the polar codes. Then, we briefly introduce the original BP algorithm. Finally, the BP flipping algorithm is presented.

2.1. Polar Code

After channel combining and channel splitting, $N$ independent copies of binary-input discrete memoryless channels are converted to $N$ split channels with different capacities [18]. Some of these have a high channel capacity, which means that the channel is more reliable for transmitting information, and some of them have a low capacity. Polar codes use high-capacity channels to transmit information bits and CRC bits, and the rest of the channels are used to transmit frozen bits. In this paper, the frozen bits are fixed to zero.

Polar code can be represented as $P(N, K)$, where $N$ is the code length of the polar code and $K$ represents the length of the information bits. Meanwhile, $(N - K)$ represents the length of frozen bits. The code rate is $R = K/N$. The $K$ information bits comprise $(K - r)$-bit data and $r$-bit CRC. The set of information bits and frozen bits are denoted as $\mathcal{A}$ and $\mathcal{A}^C$, respectively. The encoding process can be expressed as

$$\mathbf{x}_N^N = \mathbf{u}_N^N \mathbf{G}_N,$$

where $\mathbf{x}_N^N = \{x_1, x_2, \ldots, x_N\}$ represents the codeword and $\mathbf{u}_N^N = \{u_1, u_2, \ldots, u_N\}$ denotes the source vector which is mixed with the information bits $u_\mathcal{A}$ and the frozen bits $u_{\mathcal{A}^C}$. The generator matrix is represented as $\mathbf{G}_N = \mathbf{B}_N \mathbf{F}^\otimes n$, where $\mathbf{B}_N$ is the bit-reversal permutation matrix [20], and $\mathbf{F}^\otimes n$ denotes the $n$-th Kronecker power of $n = \log_2 N$ and

$$\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \quad (2)$$

The BP decoding is initiated from the received value $y_N^N = \{y_1, y_2, \ldots, y_N\}$. The decoder generates an estimation $\hat{u}_i$ of $u_i$ based on the received $y_N^N$ as

$$\hat{u}_i = \begin{cases} 0, & \text{if } i \in \mathcal{A}^C, \\ 0, & \text{if } i \in \mathcal{A} \text{ and } \frac{W_{N}^{(i)}(y_{N}^{i}, \hat{u}_{i}^{-1} | u_i = 0)}{W_{N}^{(i)}(y_{N}^{i}, \hat{u}_{i}^{-1} | u_i = 1)} \geq 1, \\ 1, & \text{if } i \in \mathcal{A} \text{ and } \frac{W_{N}^{(i)}(y_{N}^{i}, \hat{u}_{i}^{-1} | u_i = 0)}{W_{N}^{(i)}(y_{N}^{i}, \hat{u}_{i}^{-1} | u_i = 1)} < 1. \end{cases} \quad (3)$$

A binary input memoryless channel $W_n$ generates $N$ sub-channels by channel splitting, defining $W_n^{(i)}(y_{1}^{N}, \hat{u}_{1}^{-1} | u_i), i = 1, 2, \ldots, N$. Let the LLR of $W_n^{(i)}(y_{1}^{N}, \hat{u}_{1}^{-1} | u_i)$ be defined as

$$L_n^{(i)}(y_{1}^{N}, \hat{u}_{1}^{-1}) = \ln \left( \frac{W_n^{(i)}(y_{1}^{N}, \hat{u}_{1}^{-1} | u_i = 0)}{W_n^{(i)}(y_{1}^{N}, \hat{u}_{1}^{-1} | u_i = 1)} \right). \quad (4)$$

2.2. Original BP Decoding Algorithm

The BP algorithm for polar codes is based on a DFG. A polar code with code length $N$ is represented by an $n$-stage DFG. We use ($i$, $j$) to indicate the nodes of the DFG, where $i$ indicates node index and $j$ indicates column index. The leftmost nodes in the DFG mean $j = 0$, such as the blue and black nodes in the Figure 1. Similarly, the rightmost nodes in the DFG mean $j = n$, as shown in the grey node column.
The classic BP DFG is depicted in Figure 1. Each stage consists of \(N/2\) processing elements (PE), where a fundamental PE is shown in Figure 2. Figure 1 consists of three stages and each stage has four PEs. One PE has four nodes and each node is associated with two types of messages, a right-to-left message \(L_{ji}\) and a left-to-right message \(R_{ji}\). \(L_{ji}\) and \(R_{ji}\) are in the form of LLR. The message propagation rules are as follows:

\[
\begin{align*}
L_{ji} &= g(L_{j+1,i}, R_{j+1,i} + L_{j+1,i+2}), \\
L_{j+1,i} &= g(L_{j,i+1}, R_{j,i+1} + L_{j+1,i+2}), \\
R_{j+1,i} &= g(R_{j,i}, L_{j,i+1} + R_{j+1,i+2}), \\
R_{j,i+1} &= g(R_{j+1,i}, L_{j+1,i} + R_{j,i+2}).
\end{align*}
\]

(5)

where

\[
g(x, y) = \alpha \cdot \text{sign}(x) \cdot \text{sign}(y) \cdot \min(|x|, |y|),
\]

(6)

where \(\alpha = 0.9375\) follows from the scaling factor used in [21]. \(L_{ji}\) and \(R_{ji}\) need to be initialized as follows

\[
L_{ji} = \begin{cases} 
0, & i \neq n + 1, \\
\mathcal{L}_n^{(i)}, & i = n + 1.
\end{cases}
\]

(7)

\[
R_{ji} = \begin{cases} 
0, & j \in \mathcal{A}, \\
+\infty, & i = 1, j \in \mathcal{A}^C,
\end{cases}
\]

(8)

where \(\mathcal{L}_n^{(j)}\) denotes the LLR of the \(j\)-th received bit. The \(+\infty\) in the first column of \(R_{j,1}\) indicates the prior knowledge carried by the frozen bits.

**Figure 1.** Decoding factor graph with \(N = 8\).
The maximum number of iterations $I_{\text{max}}$ is preset, and the decoding is terminated when the number of iterations is equal to $I_{\text{max}}$ or when the CRC check is satisfied. The hard decisions of $\hat{u}_i$ and $\hat{x}_i$ are estimated as

$$\hat{u}_i = \begin{cases} 1, & L_{j,1} + R_{j,1} < 0, \\ 0, & L_{j,1} + R_{j,1} \geq 0. \end{cases}$$

$$\hat{x}_i = \begin{cases} 1, & L_{j,n+1} + R_{j,n+1} < 0, \\ 0, & L_{j,n+1} + R_{j,n+1} \geq 0. \end{cases}$$

2.3. BP Flipping Decoding Algorithm

The bit-flipping strategy is a feasible method with which to improve the performance of BP-based algorithms. Due to the parallelism of BP decoding, there may be more than one bit that could correct the error frame by flipping it. Not only can the real error bits correct the error frames, but there are still other bits that can correct the error frames in the process of the iterative computation of the DFG [18]. However, there exist more bits that provide no assistance in error frame correction, and the invalid flipping of these will cause much latency. Therefore, the study of the strategy used for locating the flipping bits which can effectively correct the error frames is essential.

Initially, the CS is used to identify unreliable bits in the SC flipping decoder, which is composed of the first bit index of each Rate-1 node [15]. As shown in Figure 3, blue nodes are referred to as Rate-1 nodes because all the leaf nodes are information bits, white node means that all its leaf nodes are frozen bits and grey node denotes that its leaf nodes include both information and frozen bits. In Figure 3, $\text{CS} = \{8, 10, 11, 13\}$ is shown as striped blue nodes. It can be noticed that the size and elements of CS are fixed for a certain polar code, which means that CS is static in decoding. Because of this characteristic, no latency will be caused by FS construction. The BPF algorithm [15] employs CS in bit-flipping and the flipping operation is as follows

$$R_{i,0} = \begin{cases} +\infty, & i \in \text{CS and } \hat{u}_i = 0, \\ -\infty, & i \in \text{CS and } \hat{u}_i = 1. \end{cases}$$

For the GBPF decoding [17,18] algorithm, the FS is constructed dynamically with the smallest LLR magnitude. A sorting network is required to select information bits to constitute the FS, $F_1^\Gamma = \{F_1, F_2, \ldots, F_\Gamma\}$, where $\Gamma$ denotes the length of the FS. Before the next-round of BP decoding attempts, the FS is generated by the smallest LLR magnitude and the bit-flipping operation is performed with the FS. The rule used to generate the FS is defined by

$$FS \leftarrow j \in \mathcal{A} \text{ and smallest } |L_n^{(j)}|. \quad (12)$$
Specifically, the GBPF flipping operation can be written as

$$R_{0,i} = \begin{cases} +\infty, & i \in \text{FS} \quad \text{and} \quad \hat{u}_i = 1, \\ -\infty, & i \in \text{FS} \quad \text{and} \quad \hat{u}_i = 0, \\ 0, & i \in A / \text{FS}. \end{cases} \quad (13)$$

The oracle-assisted BP (OA-BP) decoder [18] knows which bit estimates make the frame mistakes after the original BP decoding and then re-decodes the incorrect frame by flipping the erroneous bit in turn into the correct value. The incorrect codeword set can be expressed as $A_{\tau} = \{A_1, A_2, \ldots, A_\tau\}$, where $\tau$ is the count of erroneous bits.

The differences between BPF algorithms are how to choose the flipping set. The BPF algorithm generates CS before decoding with Rate-1 nodes and stays static in decoding. The GBPF algorithm generates FS dynamically in decoding, which will lead to extra costs of sorting the LLR magnitude in generating FS. However, the GBPF algorithm also provides greater possibility of error-correction by setting larger sizes of FS. The generalized procedure for BPF-1 algorithm [15] and GBPF-1 algorithm [17] can be summarized as Algorithm 1.

**Algorithm 1 BP flipping algorithm.**

1. **Input:** $y^N_1, \Gamma, \rho_1$
2. **Output:** $\hat{u}^N_1$
3. $\{\hat{u}_i^N\} \leftarrow \text{BP} (y^N_1)$
4. if CRC($\hat{u}_i^N$) fail
5. for $\rho_1 \leftarrow 1$ to $\Gamma$ do
6. $\{\hat{u}_1^N\} \leftarrow \text{BP} (y^N_1, \rho_1)$;
7. if CRC($\hat{u}_1^N$) succeed
8. break;
9. end if
10. end for
11. end if
12. **return** $\hat{u}_1^N$

The bits needing to be flipped by the BP flipping algorithms are listed in Table 1. The BPF algorithm has a significantly higher number of flipping bits than the GBPF algorithm. The BLER performance and the average number of iterations for the existing algorithms are shown in Figure 4. It can be seen that the BLER performance of the BPF-$\Omega$ and GBPF-$\Omega$ algorithms is competitive. However, the average number of iterations for the BPF and GBPF algorithms increases exponentially with the rise of $\Omega$. The average number
of iterations of the BPF-2 algorithm is more than two thousand at $E_b/N_0 = 1$. Therefore, we propose a step-flipping strategy to reduce the average number of iterations in Section 3.

| Algorithm         | One-Bit-Flipping Digits | Two-Bit-Flipping Digits |
|-------------------|-------------------------|-------------------------|
| BPF [15]          | $2 \times T$           | $2^2 \times C_T^2$      |
| GBPF [17,18]      | $\Gamma$               | $C_T^2$                 |
| OA-BP [18]        | $\tau$                 | $C_T^2$                 |

Table 1. Statistics for the flipping bits.

![Figure 4](image)

Figure 4. The BLER performance and the average number of iterations for the BP [10] (Hussami, N.; Korada, S.B.; Urbanke, R. 2009), BPF-1, BPF-2 [15] (Yu, Y.; Pan, Z.; Liu, N.; You, X. 2019), GBPF-1 [17] (Shen, Y.; Song, W.; Ren, Y.; Ji, H.; You, X.; Zhang, C. 2020), GBPF-2, OA-BP-1 [18] (Shen, Y.; Song, W.; Ji, H.; Ren, Y.; Ji, C.; You, X.; Zhang, C 2020) and CA-SCL [9] (Niu, K.; Chen, K. 2012) algorithm.

(a) The BLER comparison of existing algorithms with $N = 256$. (b) The average number of iterations for the existing algorithms with $N = 256$. 3. The Proposed BPSF Algorithm

In this section, we first analyze the behavior of the CS. Then, the BPSF algorithm is proposed to reduce the average number of iterations. Finally, the threshold factor is applied to the BPSF algorithm to lower the BLER, and the pseudocode of the two-bit-flipping algorithm is presented.

3.1. Analysis of Critical Set

According to Section 2.3, the size of the critical set is determined by the Rate-1 nodes. For $N = 256$, the size of CS is $T = 39$. Similarly for $N = 512$ and $N = 1024$, $T$ equals 60 and 116 respectively. There are many elements in CS that could correct the error frames by flipping them and one flipping means one attempt. The BPF-1 and GBPF-1 decoders are analyzed in Figure 5 when $N = 256$ and $N = 1024$. It illustrates the number of successful flipping attempts in the CS-T that could correct the error frames by flipping them at 2 dB.

As shown in Figure 5a, there are nine attempts of BPF-1 decoding and three attempts of GBPF-1 decoding that could correct the error. Frame 10 is marked with a blue rectangle, but the other decoding attempts are failed to correct this error frame. Those successful decoding attempts can be expressed as $r_M^\Omega = \{r_1^\Omega, r_2^\Omega, \cdots, r_M^\Omega\}$, where $\Omega$ is the maximum
bit-flipping order and $M$ is the number of successful BP-flipping decoding attempts. When $Ω = 1$, $M$ is less or equal to the number of BP-flipping decoding attempts ($C_1 = 39$). When $Ω = 2$, $M$ is less or equal to the number of BP-flipping decoding attempts ($C_1 + C_2 = 780$). The decoding attempts are similar when $N = 1024$, as shown in Figure 5b.

![Graph showing decoding attempts](image)

**Figure 5.** The number of successful BP flipping decoding attempts by the BPF-1 [15] (Yu, Y.; Pan, Z.; Liu, N.; You, X. 2019), GBPF-1 algorithms [17] (Shen, Y.; Song, W.; Ren, Y.; Ji, H.; You, X.; Zhang, C. 2020) at 2 dB. (a) The number of successful BP flipping decoding attempts with $N = 256$. (b) The number of successful BP flipping decoding attempts with $N = 1024$.

The distribution of $M$ is listed in Table 2. In the BPF-1 and GBPF-1 decoders, 87.9% and 79.3% of the error frames can be corrected with $M \leq 9$. It means that the number of successful decoding attempts is not larger than 9 for most error frames. Likewise, 79.1% and 78.3% of the error frames can be corrected with $M \leq 99$ in the BPF-2 and GBPF-2 decoders. It can be observed that only a few BP-flipping decoding attempts can correct the error frames while the others are not helpful. In addition, the BP-flipping decoding attempts are different for each error frame. Thus, we propose a method to detect ineffective decoding attempts and step them to reduce the average number of decoding attempts.

| $M$  | $M \leq 9$ | $10 \leq M \leq 19$ | $20 \leq M \leq 39$ |
|------|-------------|---------------------|---------------------|
| BPF-1| 87.9%       | 11.7%               | 0.4%                |
| GBPF-1| 79.3%       | 17.6%               | 3.1%                |

| $M$  | $M \leq 99$ | $100 \leq M \leq 399$ | $400 \leq M \leq 780$ |
|------|-------------|------------------------|------------------------|
| BPF-2| 79.1%       | 17.7%                  | 3.2%                   |
| GBPF-2| 78.3%       | 18.1%                  | 3.6%                   |

### 3.2. Proposed BPSF Algorithm

A flipping set of FS-$T$ can be expressed as $ε_T = \{ω_1, \cdots, ω_T\}$ and $ω_q$ ($1 \leq q \leq T$) indicates the $q$-th flipping position. $\hat{u}[ω_q]$ denotes the hard decision estimate of the bit $u_q$ after flipping the $ω_q$-th bit. Let $P(ω_q)$ be the probability of $ω_q$ correcting the trajectory of BP decoding, where

$$P(ω_q) = \Pr(\hat{u}[ω_q]^N = u_j^N|y) \quad (14)$$

Experiments show that only a few bits in the FS-$T$ can correct the error frames, and flipping the rest will not facilitate successful decoding. Thus, this paper proposes a stepping
scheme to step the flipping bits with low $P(\omega_j)$. Let $L_{j,0}^{(\omega_j)}$ denote the LLR of the bit $u_j$ after flipping the $\omega_j$-th bit. The LLR magnitude can be intuitively used as a metric of the reliability [21]. If the $L_{\omega q,0}^{(\omega_q)}$ magnitude is smaller, the bit is considered as an unreliable bit and then the $P(\omega_q)$ of the bit is deemed to be higher. Thus, if the $L_{\omega q,0}^{(\omega_q)}$ magnitude is smaller than the $L_{\omega q+1,0}^{(\omega_q+1)}$ magnitude, $P(\omega_q)$ is deemed to be higher than $P(\omega_q+1)$ and there is no need to perform a useless flipping for the reliable $\omega_q+1$ bit. Let $\rho=\{\rho_1,\rho_2,\cdots,\rho_T\}$ denote the index of FS. We assume one decoding trial of flipping $\omega_{\rho_i}$ has failed. Additionally, we obtain the new LLR of $L_{\omega_{\rho_i},0}^{(\omega_{\rho_i})}$ and $L_{\omega_{\rho_k},0}^{(\omega_{\rho_k})}$, $i < k \leq T$. The stepping decision is as (15). Using the stepping decision, the flipping sequence of the original BPF is modified, as shown in Figure 6.

$$\rho_k \text{ is skipped, if } |L_{\omega_{\rho_k},0}^{(\omega_{\rho_k})}| < |L_{\omega_{\rho_k},0}^{(\omega_{\rho_k})}|.$$  

Figure 6. The flipping sequence of original BPF and BPF with stepping strategy.

The original flipping strategy used in BPF [15] is to flip bits within CS-T in turn. We propose the use of the stepping decision in the BPSF algorithm to decide which bit can be skipped. The flipping operation in the proposed algorithm is as (16), which flips the left message $L_{j,n+1}$ of the rightmost nodes.

$$L_{j,n+1} = \begin{cases} 
-\infty, & \text{if } L_n^{(j)} \geq 0, \\
+\infty, & \text{if } L_n^{(j)} < 0. 
\end{cases}$$  

Similarly, inspired by the GBPF-\(\Omega\) algorithm [17,18], we design a generalized bit step-flipping (GBPSF-\(\Omega\)) algorithm. There are two differences between the GBPF-\(\Omega\) and GBPSF-\(\Omega\). Firstly, GBPSF-\(\Omega\) flips the left message of the rightmost nodes according to (16). Secondly, the stepping decision is used to step over bits, as shown in Figure 6, after the construction of FS.

3.3. Enhanced BPSF Algorithm

The BPSF-\(\Omega\) algorithm can significantly decrease the average number of iterations, but some flipping bits that could correct the frame in FS are skipped when their LLR magnitude is small, which will cause performance degradation. In Figure 7a,c, the “blue line” denotes the LLR magnitude of $\omega_j$ ($1 \leq j \leq M$), which cannot correct the error frame after one BP decoding attempt. The “Purple line” denotes the LLR magnitude of $\omega_k$ ($j < k \leq M$), which is the first bit after $\omega_j$ that can correct the error frame. The LLR magnitude gap between $\omega_j$ and $\omega_k$, called $\beta$, is shown in Figure 7b,d. It indicates that $\omega_k$ can still be an unreliable bit even if it satisfies (15).
Figure 7. The LLR magnitude and the gap $\beta$ between the error bits and the correct bits at 2 dB. (a) The LLR magnitude for $P(512, 256)$. (b) The gap $\beta$ between the error bits and the correct bits for $P(512, 256)$. (c) The LLR magnitude for $P(1024, 512)$. (d) The gap $\beta$ between the error bits and the correct bits for $P(1024, 512)$.

Table 3 illustrates that the $\beta$ value mostly concentrates on the range [1, 2] and [0, 2]. Using the $\beta$, the stepping decision of (15) is modified to (17) to skip the unreliable bits more accurately. Then, the enhanced BPSF-\(\Omega\) algorithm (EBPSF-\(\Omega\)) and the enhanced GBPSF-\(\Omega\) algorithm (EGBPSF-\(\Omega\)) with the threshold $\beta$ are developed, whose stepping decision is the same as (17). $\beta$ can be determined by a Monte-Carlo simulation. We detail the EBPSF-2 algorithm in Algorithm 2, and the EBPSF-\(\Omega\) algorithm is also similar to it when $\Omega > 2$.

$$\rho_k \text{ is skipped, } \text{if } |L_{\omega_{\rho_k},\rho_k}| - \beta < |L_{\omega_{\rho_k},\rho_k}|. \quad (17)$$

| $N = 512$ | 0–1 | 1–2 | 2–4 | >4 |
|------------|-----|-----|-----|----|
| The distribution of $\beta$ | 26.9% | 46.9% | 19.7% | 6.2% |

| $N = 1024$ | 0–2 | 2–4 | 4–6 | >6 |
|------------|-----|-----|-----|----|
| The distribution of $\beta$ | 48.1% | 31.9% | 11.7% | 8.3% |
Algorithm 2: EBPSF-2 decoding.

1: **Input:** \( y_N \), \( \omega_{\rho} \), \( \beta \), \( T \)
2: **Output:** \( \hat{u}_1^N \)
3: \{\( \hat{u}_1^N \)\} \( \leftarrow \) BP \((y_N^N)\);
4: **if** CRC\((\hat{u}_1^N)\) fail
5: **while** \( \rho_1 < T \) do
6: \( \text{flip} \,(L_n^{(\omega_{\rho_1})}) \text{ based } (16); \)
7: \{\( \hat{u}_1^N \)\} \( \leftarrow \) BP \((y_N^N)\);
8: **if** CRC\((\hat{u}_1^N)\) fail
9: update \( \rho_1 \) according to (17);
10: **else**
11: \( \text{break}; \)
12: **end if**
13: **end while**
14: **if** CRC\((\hat{u}_1^N)\) fail
15: \( \rho_2 = \rho_1 + 1; \)
16: **while** \( \rho_1 < T \) do
17: **while** \( \rho_2 < T \) do
18: \( \text{flip} \,(L_n^{(\omega_{\rho_1})},L_n^{(\omega_{\rho_2})}) \text{ based } (16); \)
19: \{\( \hat{u}_1^N \)\} \( \leftarrow \) BP \((y_N^N)\);
20: **if** CRC\((\hat{u}_1^N)\) fail
21: update \( \rho_1 \) according to (17);
22: update \( \rho_2 \) according to (17);
23: **else**
24: \( \text{break}; \)
25: **end if**
26: **end while**
27: **if** CRC\((\hat{u}_1^N)\) succeed
28: \( \text{break}; \)
29: **end if**
30: **end while**
31: **end if**
32: **end if**
33: return \( \hat{u}_1^N \)

4. Numerical Results

In this section, we compare the proposed step-flipping algorithm and the existing flipping algorithm in terms of \( \beta \), the average number of iterations, and the BLER with different code lengths. Simulations are performed with additive white Gaussian noise (AWGN) channel and binary-phase shift keying (BPSK) modulation. The additional simulation parameters are listed in Table 4. The simulations for the \((256, 128)\), \((512, 256)\), and \((1024, 512)\) polar codes are concatenated with 24-bit CRC and for \((64, 32)\) polar codes with 11-bit CRC. The \( m \) CRC bits are attached to the information block, where \( m \) is the CRC remainder length, and all the \( K \) bits are sent into the error-correcting encoders.
4.1. Analysis of the Threshold $\beta$

To verify the effectiveness of the threshold $\beta$, the BLERs of the BPSF-1, EBPSF-1, GBPSF-1 and EGBPSF-1 algorithms with different threshold $\beta$ are compared in Figure 8. Additionally, the GBPF-1 and BPF-1 algorithms are provided as references.

With the assistance of $\beta$, the EBPSF-1 algorithm outperforms the BPSF-1 algorithm in BLER under the same parameters for both $N = 256$ in Figure 8a and $N = 1024$ in Figure 8b. For $\beta = 0.5$ and $\beta = 1$ when $N = 256$ and $T = 128$, the EBPSF-1 algorithm achieves 0.09 dB and 0.13 dB gain with the BPSF-1 algorithm at BLER $= 1 \times 10^{-2}$, respectively. Furthermore, for $\beta = 1$ and $\beta = 10$ when $N = 1024$ and $T = 116$, the EBPSF-1 algorithm obtain the gain of 0.04 dB and 0.23 dB for BLER $= 10^{-3}$ compared with the BPSF-1 algorithm. Therefore, the EBPSF-1 algorithm has more accuracy in locating the reliable bits among the CS-T.

With the increase of $\beta$, the BLER performance of the EBPSF algorithm is continuously

---

**Table 4. Simulation Parameters.**

| Name                                      | Parameters                                                                 |
|-------------------------------------------|---------------------------------------------------------------------------|
| Polar Codes                               | (64, 32), (256, 128), (512, 256) and (1024, 512)                           |
| Signal Channel                            | AWGN                                                                      |
| Modulation Method                         | BPSK                                                                      |
| Construction Method                       | GA                                                                        |
| Frame Number                              | 500,000                                                                   |
| Rate                                      | 1/2                                                                       |
| Maximum Iterations                        | 40                                                                        |
| CRC Generator Polynomial [22]             |                                                                           |
| $g(x) = x^{11} + x^{10} + x^9 + x^5 + 1$ |                                                                           |
| $g(x) = x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1$ |
| Horizontal Coordinate                     |                                                                           |
| Design SNR of Construction                | 1 dB                                                                      |
| Efficient Information Bits                | $K - m$                                                                   |

---

**Figure 8.** The BLER of BPF-1 [15] (Yu, Y.; Pan, Z.; Liu, N.; You, X. 2019), GBPF-1 [17] (Shen, Y.; Song, W.; Ren, Y.; Ji, H.; You, X.; Zhang, C. 2020), OA-BP-1 [18] (Shen, Y.; Song, W.; Ji, H.; Ren, Y.; Ji, C.; You, X.; Zhang, C 2020), BPSF-1, EBPSF-1, GBPSF-1 and EGBPSF-1 with different threshold $\beta$. (a) The BLER comparison of different algorithms for $P(256, 128)$. (b) The BLER comparison of different algorithms for $P(1024, 512)$. 
optimized, but the average number of iterations also increases. Therefore choosing a proper $\beta$ is essential to optimizing the BLER performance and can lead to a negligible increase in the average number of iterations.

4.2. Analysis of the Average Number of Iterations

The stepping strategy is used to skip some bits in FS. Thus, the number of flippings is smaller than the original flipping algorithm without the stepping strategy. To verify this point, some simulations are performed. Figures 9 and 10 compare the average number of iterations among the BPF-$\Omega$, GBPF-$\Omega$, EBPSF-$\Omega$ and EGBPSF-$\Omega$ algorithms. The average number of iterations for the EBPSF-$\Omega$ algorithm is lower than the BPF-$\Omega$ algorithm, while the EGBPSF-$\Omega$ algorithm is lower than that of the GBPF-$\Omega$ algorithm. That’s because the step-flipping strategy applied in the EBPSF-$\Omega$ and EGBPSF-$\Omega$ algorithms reduces flipping attempts by skipping the reliable flipping bits.

It can be observed from Figures 9 and 10 that there is a significant decrease in the average number of iterations when applying the step-flipping strategy. At high $E_{b}/N_0$, the EBPSF-$\Omega$ ($T = 256$) algorithm is close to the BPF-$\Omega$ ($T = 39$) algorithm in the average number of iterations. For $T = 39$, the EBPSF-1 ($\beta = 1$) algorithm reduces the average number of iterations by 14.1% and 62.2% at 1.5 dB compared with the GBPF-1 and BPF-1 algorithms, respectively. In comparison with the GBPF-1 algorithms, the EGBPSF-1 ($T = 116$, $\beta = 0$) algorithm reduces the average number of iterations by 16.28% at 1.5 dB.

![Figure 9. The average number of iterations for GBPF-1 [17] (Shen, Y.; Song, W.; Ren, Y.; Ji, H.; You, X.; Zhang, C. 2020), BPF-1 [15] (Yu, Y.; Pan, Z.; Liu, N.; You, X. 2019), EBPSF-1 and EGBPSF-1. (a) The average number of iterations for different algorithms with $N = 256$. (b) The average number of iterations for different algorithms with $N = 1024$.](image1)

Similarly, for the two-bit flipping, the average number of iterations is shown in Figure 10. Under the same $T$, the average number of iterations for the EBPSF-2 algorithm is significantly lower than that for the BPF-2 and GBPF-2 algorithms. In the case of 0 dB and $T = 12$, the EBPSF-2 algorithm is inferior to BPF-2 and GBPF-2 algorithms by 40.54% and 47.62%, as shown in Figure 10a, respectively. In Figure 10b, with $T = 128$ and $T = 256$, the EBPSF-2 algorithm reduces the average number of iterations by 77.4% and 95.9% at 1.5 dB against the BPF-2 algorithm, respectively. Consequently, the step-flipping strategy that we propose for the BP one-bit flipping and multi-bit flipping algorithms is effective in reducing the average number of iterations.
4.3. Analysis of the BLER Performance

Using the stepping strategy, some bits in FS are skipped during the procedure of flipping with negligible performance loss. The BLER performance of the EBPSF-Ω and EGBPSF-Ω algorithms is depicted in Figures 11 and 12. It can be observed that the algorithms which apply the proposed step-flipping strategy are better in terms of the BLER performance. Unlike the BPF-Ω and GBPF-Ω algorithms, which flip the right information of the leftmost nodes in the DFG, we flip the left information of the rightmost nodes by (16). In contrast to the CA-SCL decoding, the EBPSF-Ω and EGBPSF-Ω algorithms can achieve comparable decoding performance.

When \( T = 39 \), Figure 11a indicates that the EBPSF-1 (\( \beta = 1 \)) algorithm performs similarly to the BLER of GBPF-1 algorithm. Additionally, the EBPSF-1 algorithm has 0.23 dB gain at BLER = \( 4 \times 10^{-3} \) against the OA-BP decoder when \( T = 256 \). The BPF-Ω and GBPF-Ω algorithms flip the right information of the leftmost nodes in the DFG. However, the proposed algorithm flips the left information of the rightmost nodes, and the EBPSF algorithm obtains a gain over the OA-BP decoder. With the same parameters, the EBPSF-1 (\( T = 256 \)) outperforms the CA-SCL (\( L = 4 \)) decoder by 0.13 dB when BLER = \( 1 \times 10^{-2} \). The one-bit flipping BLER performance for \( N = 512 \) is depicted in Figure 11b. When \( T = 60 \) and \( \beta = 0.5 \), the EBPSF-1 algorithm for BLER compared to the BPF-1 algorithm shows a gain of 0.09 dB at BLER = \( 2 \times 10^{-3} \). At \( T = 60 \), the performance of the EGBPSF-1 (\( T = 60 \), \( \beta = 0.5 \)) algorithm is approaching the BLER of the CA-SCL (\( L = 2 \)) decoder.

The two-bit flipping performance for \( N = 64 \) is illustrated in Figure 12a. With \( T = 32 \), the EBPSF-2 (\( \beta = 1 \)) algorithm has 0.47 dB, 0.53 dB and 0.41 dB gain compared with the BPF-2, GBPF-2 and OA-BP-2 algorithms at BLER = \( 5 \times 10^{-2} \), respectively. Furthermore, the EBPSF-2 (\( T = 32 \), \( \beta = 1 \)) algorithm achieves the BLER performance between CA-SCL (\( L = 4 \)) and (\( L = 8 \)). Figure 12b presents the two-bit flipping performance for \( N = 256 \). The EBPSF-2 (\( \beta = 1 \)) algorithm, when \( T = 256 \), shows an improvement of 0.21 dB compared with the OA-BP-2 algorithm at BLER = \( 5 \times 10^{-4} \). The EBPSF-2 (\( T = 256 \), \( \beta = 1 \)) algorithm outperforms the CA-SCL (\( L = 16 \)) decoder by 0.02 dB at BLER = \( 1 \times 10^{-3} \).
Figure 11. The BLER comparison of BP [10] (Hussami, N.; Korada, S.B.; Urbanke, R. 2009), OA-BP-1 [18] (Shen, Y.; Song, W.; Ji, H.; Ren, Y.; Ji, C.; You, X.; Zhang, C 2020), CA-SCL [9] (Niu, K.; Chen, K. 2012), GBPF-1 [17] (Shen, Y.; Song, W.; Ren, Y.; Ji, H.; You, X.; Zhang, C. 2020), BPF-1 [15] (Yu, Y.; Pan, Z.; Liu, N.; You, X. 2019), EGBPSF-1 and EBPSF-1. (a) The BLER comparison of different algorithms for \( P(256, 128) \). (b) The BLER comparison of different algorithms for \( P(512, 256) \).

Figure 12. The BLER comparison of OA-BP-2 [18] (Shen, Y.; Song, W.; Ji, H.; Ren, Y.; Ji, C.; You, X.; Zhang, C 2020), CA-SCL [9] (Niu, K.; Chen, K. 2012), GBPF-2 [18] (Shen, Y.; Song, W.; Ji, H.; Ren, Y.; Ji, C.; You, X.; Zhang, C 2020), BPF-2 [15] (Yu, Y.; Pan, Z.; Liu, N.; You, X. 2019) and EBPSF-2. (a) The BLER comparison of different algorithms for \( P(64, 32) \). (b) The BLER comparison of different algorithms for \( P(256, 128) \).
5. Conclusions

To reduce the average number of iterations in the BP flipping algorithm, this paper proposes a BP flipping algorithm with the stepping strategy. To narrow the search space, the reliable bits are skipped by the stepping strategy to improve the accuracy of flipping bits. The magnitude of LLRs are used to determine whether the bits are reliable or not, skip over the reliable bits in the FS, and flip the unreliable bits to reduce the average number of iterations. Furthermore, to make the algorithm more robust, we propose the threshold $\beta$ to reduce the BLER. Simulation results show that the proposed algorithm with one-bit flipping and two-bit flipping can achieve BLER for CA-SCL decoder with list size 4 and 8, separately. Compared with the BPF decoder and the GBPF decoder, the EBPSF-$\Omega$ significantly reduces the average number of iterations.

Author Contributions: Conceptualization, X.Z., Y.L. and C.C.; Software, X.Z. and Y.L.; Validation, X.Z., Y.L. and C.C.; Supervision, Q.Z.; Writing-original draft preparation, X.Z., Y.L. and C.C.; Writing-review and editing, H.G. and Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Joint Fund for Smart Computing of Natural Science Foundation of Shandong Province (No.ZR2019LZH001), the Shandong University Youth Innovation Supporting Program (No.2019KJN020, No.2019KJN024) and Shandong Chongqing Science and technology cooperation project (No.cstc2020jscx-lysX0008).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Not applicable.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Abbreviations

The following abbreviations are used in this manuscript:

- **BP**: Belief propagation
- **BLER**: Block error rate
- **5G**: Fifth generation
- **SC**: Successive cancellation
- **FS**: Flip set
- **CS**: Critical set
- **LLR**: Log-likelihood ratio
- **CRC**: Cyclic redundancy check
- **PE**: Processing elements
- **DFG**: Decoding factor graph
- **BPSK**: Binary phase shift keying
- **AWGN**: Additive white gaussian noise

References

1. Arıkan, E. Channel polarization: A method for constructing capacity-achieving codes. In Proceedings of the 2008 IEEE International Symposium on Information Theory, Toronto, ON, Canada, 6–11 July 2008; pp. 1173–1177.
2. Final Report of 3GPP TSG RAN WG1 #87 v1.0.0. Reno, 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 5 January 2021).
3. Gallager, R. Low-density parity-check codes. *IRE Trans. Inf. Theory.* 1962, 8, 21–28. [CrossRef]
4. Balatsoukas-Stimming, A.; Giard, P.; Burg, A. Comparison of Polar Decoders with Existing Low-Density Parity-Check and Turbo Decoders. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), San Francisco, ON, USA, 19–22 March 2017, pp. 1–6.
5. 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]
6. Tal, I.; Vardy, A. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory.* 2015, 61, 2213–2226. [CrossRef]
7. Zhang, Z.; Qin, K.; Zhang, L.; Chen, G.T. Progressive bit-flipping decoding of polar codes over layered critical sets. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Singapore, 4–8 December 2017; pp. 4–8.
8. Chandresris, L.; Savin, V.; Declercq, D. Dynamic-SCFlip decoding of polar codes. *IEEE Trans. Commu.* 2018, 66, 2333–2345. [CrossRef]
9. Niu, K.; Chen, K. CRC-aided decoding of polar codes. *IEEE Commun. Lett.* 2012, 16, 1668–1671. [CrossRef]
10. Hussami, N.; Korada, S.B.; 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 List Decoding of Polar Codes. *IEEE Commun. Lett.* 2018, 22, 1536–1539. [CrossRef]
12. Elkelesh, A.; Ebada, M.; Cammerer, S.; ten Brink, S. Mitigating clipping effects on error floors under belief propagation decoding of polar codes. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.
13. Doan, N.; Hashemi, S.A.; Mondelli, M.; Gross, W.J. On the Decoding of Polar Codes on Permutted Factor Graphs. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
14. Elkelesh, A.; Cammerer, S.; Ebada, M.; ten Brink, S. 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.
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. 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]
17. 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]
18. 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]
19. 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), Nanjing, China, 21–23 October 2020; pp. 807–810.
20. Tal, I.; Vardy, A. How to construct polar codes. *IEEE Trans. Inf. Theory.* 2013, 59, 6562–6582. [CrossRef]
21. Yuan, B.; Parhi, K.K. Early Stopping Criteria for Energy-Efficient Low-Latency Belief-Propagation Polar Code Decoders. *IEEE Trans. Signal Process.* 2014, 62, 6496–6506. [CrossRef]
22. Technical Specification Group Radio Access Network. Valbonne, France. June 2021. Available online: https://www.3gpp.org/ftp/Specs/archive/38_series/38.212/38212-g60.zip (accessed on 10 June 2021).