ISSN 13303651 (Print), ISSN 18486339 (Online)
Received: 3 September 2025; Accepted: 17 November 2025

https://doi.org/10.17559/TV-20250903002940
Original scientific paper

An Optimized Belief Propagation List Decoding for Polar Codes with Dynamic Flipping

Yinyou MAO, Wenxue TAN, Jianying LI, Lin NI*

Abstract: In the context of polar codes, belief propagation list (BPL) decoding has demonstrated a substantial enhancement in parallel decoding performance, achieving high throughput. Nevertheless, a performance gap still exists between the advanced BPL decoding and successive cancellation list (SCL) decoding methods. Moreover, existing bit-flipping strategies are inefficient in accurately identifying erroneous bit positions, leading to elevated computational complexity and limiting their practical applicability. This study introduces an optimized BPL decoding algorithm with dynamic flipping (OBPL-DF) aimed at bridging this performance gap while reducing computational demands. Initially, an efficient decoding scheme is proposed to further decrease computational complexity in practical scenarios. Subsequently, to improve the precision of error position detection, a partial cyclic redundancy check (CRC) code is employed on erroneous codewords. Finally, a dynamic flipping metric is developed within the bit-flipping strategy, allowing the selection of flipped positions to be guided by this novel metric rather than being confined to a predetermined set. Simulation results demonstrate that the OBPL-DF algorithm surpasses the performance of existing BPL flip (BPLF) decoding techniques and approaches that of enhanced SCL decoding, all while achieving significantly lower latency.

Keywords: bit-flipping; BPL decoding; polar codes

1 INTRODUCTION

Polar codes [4] represent the only channel-coding scheme that has been rigorously proven to achieve channel capacity while maintaining practical linear complexity in both encoding and decoding processes. Consequently, polar codes have been adopted as the coding scheme for 5G enhanced mobile broadband control channels [1]. Although successive cancellation (SC) decoding theoretically guarantees capacity achievement, its error-correction performance can be significantly compromised at practical code lengths due to incomplete polarization. To address this limitation, alternative decoding methods such as successive cancellation list (SCL) decoding [17] and successive cancellation flip (SCF) decoding [2] have been introduced. Furthermore, to improve the accuracy of correct path identification, polar codes have been concatenated with an outer cyclic redundancy check (CRC) code [14]. Leveraging the CRC, the error-correction performance of the concatenated codes under CRC-aided SCL (CA-SCL) decoding can rival that of other high-performance error-correcting codes, including Low-Density Parity-Check (LDPC) and Turbo codes. Nevertheless, these decoding algorithms are characterized by high latency, primarily due to their inherently serial nature. In contrast, belief propagation (BP) decoding has garnered considerable interest owing to its parallel structure [3].

Belief Propagation (BP) decoding operates as an iterative parallel decoding method, inherently offering the advantage of low latency and consequently achieving high throughput. Furthermore, BP decoding is more amenable to hardware implementation compared to Successive Cancellation (SC) decoding, which positions BP decoding as a promising candidate for supporting polar codes within the 5G standard in practical applications. Nevertheless, the error-correction performance of the original BP decoding algorithm remains suboptimal, particularly due to its inferior block error rate (BLER) relative to Successive Cancellation List (SCL) decoding. To address this limitation, several enhancements have been proposed. Drawing inspiration from SCL decoding, the concept of list decoding was adapted to BP decoding, resulting in the BP list (BPL) decoding approach [9]. BPL decoding can approximate the performance of SCL decoding with reduced latency by employing L parallel BP decoders. When combined with a cyclic redundancy check (CRC) code and decoded using the sum-product algorithm (SPA), the CRC-aided BPL (CA-BPL) decoding scheme achieves performance close to that of CRC-aided SCL (CA-SCL) decoding at high signal-to-noise ratios (SNRs) for short code lengths [12]. Additionally, methods such as noise perturbation [6] and two-level detection [11] have been introduced to further enhance BPL performance. Despite these improvements, a performance gap persists between BPL-based algorithms and CA-SCL decoding for medium to long code lengths, with the former also exhibiting higher computational complexity. Alternative strategies to improve BP decoding include BP flip (BPF) and BP correction (BPC) decoding. Analogous to SC flip (SCF) decoding, BPF decoding flips error-prone information bits by assigning $\pm\infty$ to their a priori information [21]. As a post-processing technique, BPC decoding adjusts the prior knowledge of identified code bits at the received codeword stage [24]. Although both BPF and BPC decoding improve upon the original BP decoding performance, they still fall short of CA-SCL decoding performance. Moreover, their enhanced variants [15, 16, 23] incur increased complexity and decoding latency due to additional decoding attempts. Recently, the efficient BPL flip (EBPLF) decoder has been proposed to approach the performance of advanced SCL decoding algorithms [25]. This method efficiently identifies error-prone patterns but contributes to elevated computational complexity. Consequently, achieving an optimal balance between decoding performance and computational complexity remains a critical challenge in current research.

This paper introduces an optimized belief propagation list (BPL) decoding algorithm for polar codes incorporating dynamic flipping (OBPL-DF) to enhance error-correction performance. The proposed method increases the number of flipped bits in an additional decoding attempt and demonstrates a significant improvement in block error rate (BLER), achieving performance comparable to that of cyclic redundancy check-aided successive cancellation list (CA-SCL) decoding at equivalent list sizes. Furthermore, OBPL-DF substantially reduces the number of flipping attempts

relative to existing belief propagation (BP) and BPL algorithms employing bit-flipping. Notably, this work is the first to introduce a dynamic bit-flipping metric within BP decoding. Compared to prior approaches based on bit-flipping (BPF) decoding, the dynamic construction of the flipping set (FS) yields superior performance.

The main contributions of this study are summarized as follows:

- Unlike the extended EBPLF decoding, which simultaneously utilizes all component decoders to compute the codeword, OBPL-DF employs a single component decoder for codeword calculation, while the remaining decoders serve as auxiliaries during flipping attempts. This architecture significantly reduces computational complexity compared to conventional BPL decoding.

- To enhance the precision in identifying erroneous bit locations, the algorithm employs segmentation of information bits. Distinct from existing segmented decoding methods, this approach integrates cyclic redundancy check (CRC) codes with submatrix verification, thereby improving error localization by concentrating CRC codes in regions more susceptible to errors.

- An efficient bit-flipping metric is introduced to guide flipping attempts. This metric enables dynamic selection of flipping positions rather than static choices, resulting in improved decoding performance and a reduction in the number of flipping attempts.

The structure of the paper is as follows: Section 2 provides a concise overview of polar codes, BP and BPL decoding algorithms, and the BPF decoding algorithm. Section 3 details the proposed OBPL-DF decoding algorithm, including the designed segmentation method and the efficient bit-flipping metric. Section 4 discusses the implementation aspects of the proposed algorithm. Simulation results are presented in Section 5, and Section 6 concludes the paper.

2 PRELIMINARIES

2.1 Polar Codes

The concept of polar codes is developed from the theory of channel polarization, in which N independent channels become N correlated channels after channel combining and channel splitting. Some of these N correlated channels have a capacity close to 1 and some close to 0. As N increases, this polarization phenomenon tends to become more prominent, i.e., more and more channels with a capacity approach to 1 or 0. In information transmission, we can choose those channels with better capacity to transmit information bits, and the remaining channels will transmit frozen bits by knowing of both transmitter and receiver, i.e., the zero bits. Hence it can improve the efficiency of transmitting information. Ually, the sets of information indices and frozen indices are defined as $\mathcal{A}$ and $\mathcal{A}^c$, respectively.

The concept of polar codes originates from the theory of channel polarization, wherein a set of N independent channels is transformed into N interrelated channels through the processes of channel combining and channel splitting. Among these N resultant channels, some exhibit capacities approaching unity, while others approach zero. As the value of N increases, this polarization effect becomes increasingly pronounced, resulting in a greater proportion of channels with capacities near either 1 or 0. In the context of information transmission, channels with higher capacities are selected for transmitting information bits, whereas the remaining channels are assigned frozen bits - known to both the transmitter and receiver - typically set to zero. This selective allocation enhances the efficiency of information transmission. Conventionally, the sets of indices corresponding to information bits and frozen bits are denoted as $\mathcal{A}$ and $\mathcal{A}^c$, respectively.

For brevity, polar codes are commonly represented as $P(N,K)$, where N denotes the code length and represents the number of information bits. The K information bits comprise $(K-M)$-bit data and M-bit CRC. The resulting codeword x_1^N is generated through the encoding process as defined by the polar coding scheme.

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

where u_1^N is the source vector which includes the information bits $u_{\mathcal{A}}$ and the frozen bits $u_{\mathcal{A}^c}$. The generator matrix $G_N = B_N F^{\otimes n}$, where B_N is a $N \times N$ bit-reversal permutation matrix and $F^{\otimes n}$ is the n-th Kronecker power of $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, note that here $N = 2^n$.

2.2 Belief Propagation and Belief Propagation List Decoding

The BP decoding is an iterative decoding and initiated from the received vector $y_1^N = \{y_1, y_2, \ldots y_N\}$. Then the iterative process is carried out in the factor graph based on the realization of G_N. Fig. 1 (replaced by Fig. 1) shows an example of a factor graph about N=16 [5]. Here (i, j) indicate the column index and the node index in the actor graph, respectively. It is worth noting that the rightmost side of the actor graph connects with the received vector y_1^N. Furthermore, two types of log-likelihood ratio (LLR) messages are introduced: the left-directed message $L_{i,j}^t$ and the right-directed message $R_{i,j}^t$. As shown in Fig. 1, these two messages are propagated in factor graph.

Meanwhile, the above two messages are updated in the process element (PE) which is shown in Fig. 2. A factor graph of N usually includes $\frac{N}{2}\log N$ PE modules. The rules of message updating are as follows:

$$\begin{cases} L_{i,j}^{t+1} = g\left(L_{i+1,j}^t, L_{i+1,j+N_i}^t + R_{i,j+N_i}^t\right) \\ L_{i,j+N_i}^{t+1} = L_{i+1,j+N_i}^t + g\left(L_{i+1,j}^t, R_{i,j}^t\right) \\ R_{i+1,j}^{t+1} = g\left(R_{i,j}^t, L_{i+1,j+N_i}^t + R_{i+1,j+N_i}^t\right) \\ R_{i+1,j+N_i}^{t+1} = R_{i,j+N_i}^t + g\left(R_{i,j}^t, L_{i+1,j}^t\right) \end{cases} \tag{2}$$

and

$$g(x,y)=\ln\frac{1+xy}{x+y} \tag{3}$$

where t is the number of iterations and $N_i=2^{i-1}$. $g(x,y)$ could also be approximately equal to $g(x,y)\approx 0.9375\times\text{sign}(x)\times\text{sign}(y)\times\min(|x|,|y|)$ in hardware implementation [22].

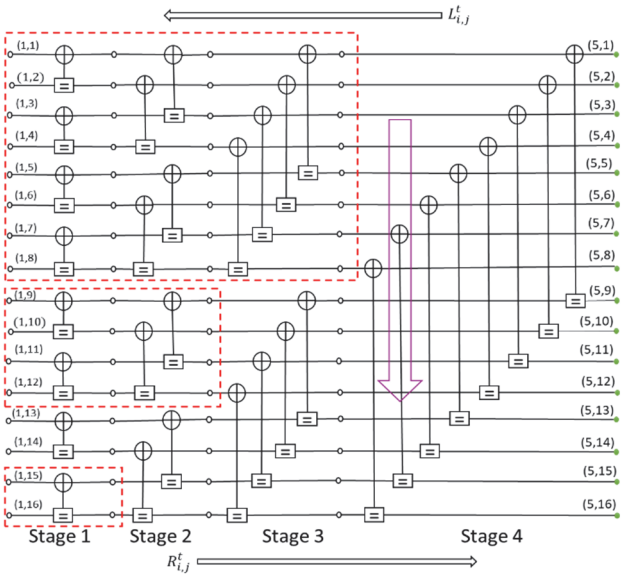

Figure 1 The factor graph for the transformation $F^{\otimes 4}$

In addition, the left-directed message $L^0_{n+1,j}$ at the rightmost side is obtained by the channel output $LLR(y_j)$ and the right-directed message $R^0_{n+1,j}$ (replaced by $R^0_{1,j}$) at the leftmost side is initialized by the priori information whether j is a frozen bit or an information bit.

$$L^0_{(n+1,j)}=LLR(y_j) \tag{4}$$

$$R^0_{1,j}=\begin{cases}0 & j\in A\\ \infty & j\in A^c\end{cases} \tag{5}$$

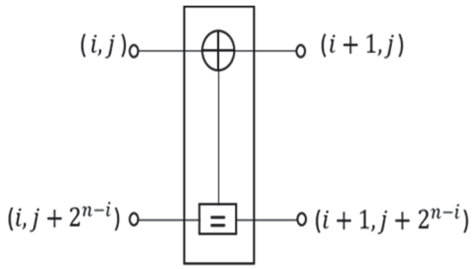

Figure 2 The basic processing element (PE) of BP decoder

The BPL decoding is proposed to improve the error-correction performance of BP decoding and maintain its parallel nature at the same time. BPL decoding consists of L different BP branches based on permuted factor graphs $\Pi_1, ..., \Pi_L$ [9]. The channel output $LLR(y_j)$ is calculated in these different decoders simultaneously. Then, all the sequences will be checked by the CRC code. As is shown in Fig. 3, if there is no sequence passing the CRC, the minimum code distance between the codeword and the received sequence will be found, and the codeword with minimum code distance as the output result.

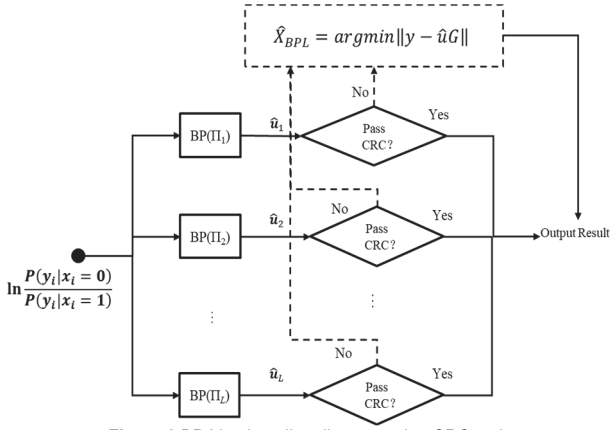

Figure 3 BP List decoding diagram using CRC code

2.3 Belief Propagation Flip Decoding

For polar codes, the strategy of bit-flipping was first proposed to improve the performance of SC-based algorithms, which is called SCF. SCF improved the decoding performance significantly due to it correcting the first incorrect bit. Inspired by this, the BPF decoding is firstly proposed to flip the error bits in [9], and the created way of FS is the same as the SC flipping decoding. Then, the flipping operation does not flip the error estimated bit directly, but assigns its priori information $R_{1,j}$ to $+\infty$ or $-\infty$.

In belief propagation (BP) decoding, erroneous estimation of information bits can induce error propagation, thereby adversely impacting the reliability of other bits. To mitigate this issue, a bit-flipping mechanism has been introduced. Specifically, for polar codes, the bit-flipping strategy was initially proposed to enhance the performance of successive cancellation (SC)-based algorithms, known as successive cancellation flip (SCF). SCF significantly improves decoding performance by correcting the first erroneous bit. Building upon this concept, bit-flipping decoding (BPF) was first introduced in [9] to flip erroneous bits, employing a flipping set (FS) construction analogous to that used in SC flip decoding. Notably, the flipping operation does not directly invert the incorrectly estimated bit; instead, it modifies its a priori information $R_{1,j}$ to $+\infty$ or $-\infty$.

It is well established that accurate bit-flipping can substantially improve the block error rate (BLER) performance of polar codes. However, the fixed flipping set, which is determined based on the polar code structure, does not consistently identify incorrectly estimated information bits with high precision. To enhance the accuracy of detecting erroneous bits, Ω order-based flipping set (FS) was proposed to flip multiple Ω bits simultaneously [11]. However, it increases the flipping attempts at the cost of 2^{Ω} times. Besides, the incorrectly estimated bits outside the FS are also excluded. Moreover,

incorrectly estimated bits outside the predefined FS remain unaddressed. To approximate the performance bound, the oracle-assisted BP (OA-BP) decoder was introduced in [12], wherein the decoder is assumed to know the exact positions of erroneous bits and adjusts their a priori information accordingly. Overall, existing BP decoding methods incorporating bit-flipping strategies fundamentally rely on a trial-and-error approach, sequentially flipping multiple bits. This sequential process undermines the inherent parallelism of BP decoding and yields only modest improvements in decoding performance.

Fig. 4 illustrates the procedure of the generalized BP flip decoding with order 2 (GBPF-2). As depicted, following each bit-flipping operation, an additional BP decoding attempt is executed sequentially. Furthermore, the bit-flipping operation of order Ω is built upon the flipping operation of order $\Omega - 1$. Consequently, BPF decoding incurs high latency in practical implementations, which is unacceptable for fifth-generation and future communication systems. To reconcile the trade-off between computational complexity and decoding performance, we propose the OBPL-DF decoding algorithm, which effectively balances these requirements.

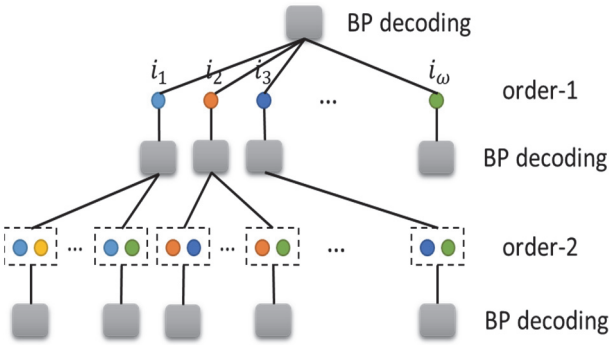

Figure 4 The procedure of GBPF-2 decoding

3 THE PROPOSED OPTIMIZED BPL DECODING WITH DYNAMIC FLIPPING

Within the context of 5G and prospective 6G applications, massive machine-type communications emphasize the importance of energy efficiency and the reduction of latency. Although belief propagation list (BPL) decoding enhances error-correction capabilities while preserving the parallelism inherent in belief propagation (BP) decoding, the concurrent operation of multiple component decoders results in substantial energy consumption. It is well established that under favourable channel conditions, the majority of codewords can be accurately decoded by a single decoder. Consequently, the remaining component decoders are primarily responsible for addressing a limited number of erroneous codewords during the decoding process. This observation motivates an investigation into strategies that involve simultaneously flipping multiple bits within the component decoders of BPL decoding.

3.1 Partial Segmentation with CRC Code

As outlined in [21], at stage m, there exist $2(n-m)$ sub-factor graphs. Fig. 1 provides an illustrative example of these sub-factor graphs for $N = 16$, where the numbers of sub-factor graphs at stages 1, 2, and 3 are 8, 4, and 2, respectively. As indicated by the arrow in Fig. 1, if the sub-factor graphs successfully pass the sub-matrix verification at stage $m - 1$, the smaller sub-factor graphs at stage m will no longer require examination. In this context, the sub-matrix verification corresponds to the G-Matrix check, and the length of the codeword subjected to verification is equivalent to the size of the respective sub-factor graph.

$$x_{N'} = u_{N'} G_{N'} \tag{6}$$

where N' denote the length of the sub-factor graph. Since the frozen bits are predominantly located in the first half of the entire code length, this segment exhibits a lower code rate relative to the second half. Furthermore, error occurrences are infrequent in the first half of the code length. Consequently, for sub-factor graphs of identical size, a lower code rate corresponds to improved block error rate (BLER) performance, as illustrated by Fig. 5 in [19].

Segmented multi-CRC-aided decoding algorithms have demonstrated significant advancements within successive cancellation flip (SCF) and successive cancellation list (SCL) decoding frameworks [10, 13]. This methodology, which partitions the codeword into multiple segments, facilitates more precise localization of erroneous bits. In the present study, a novel partial CRC segmentation strategy is introduced, informed by the error distribution characteristics observed in belief propagation (BP) decoding. The proposed segmentation integrates CRC codes with sub-matrix checks, wherein the CRC does not encompass all information bits but rather targets a subset. This selective application ensures that the CRC protects the portion of the codeword more susceptible to errors. Fig. 5 depicts the error distribution of information bits under BP decoding at a signal-to-noise ratio (SNR) of 1.5 dB with 60 iterations, considering a code length $N = 1024$ and a code rate of 0.5. Unlike SCF decoding [26], errors in BP decoding predominantly manifest in the second half of the codeword. This phenomenon can be partially attributed to the concentration of information bits in the latter half due to channel polarization. Moreover, BP decoding, characterized by its parallel and iterative nature, permits the propagation of noise-induced errors to nodes situated in the latter portion of the codeword during the iterative process.

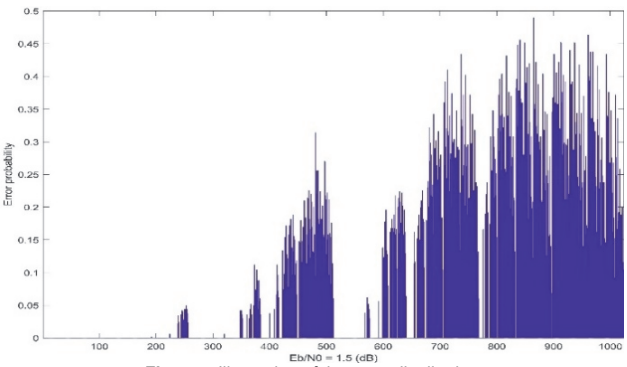

Figure 5 Illustration of the error-distribution

In this study, a multi-CRC code is incorporated into the latter half of the entire code length, as the information

bits are predominantly concentrated in this region. The segmentation approach is employed for constructing the polar code within this second half, leveraging the relationship between bit-channel polarization and error probability. Specifically, sub-block sizes are adjusted based on the degree of polarization: smaller sub-blocks correspond to bit-channels with insufficient polarization, whereas larger sub-blocks are assigned to those with sufficient polarization. To illustrate this method, Fig. 6 presents the proposed segmentation strategy. For instance, considering the $P(1024, 544)$ polar code, there are 400 information bits in the second half. Based on the polarization characteristics in this segment, the sub-block lengths are set to 20, 28, 36, and 44 for the insufficiently polarized bits. Conversely, the remaining bits in the second half, which exhibit sufficient polarization, are partitioned into segments of equal length.

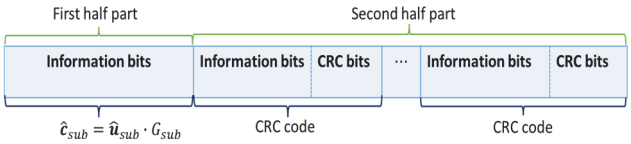

Figure 6 The diagram of the proposed method of segmentation

3.2 OBPL-DF Decoding

This paper introduces the OBPL-DF decoding algorithm as a means to reduce computational complexity. Unlike the EBPLF decoding method [25], which employs all decoders simultaneously during the decoding process, the OBPL-DF algorithm activates only after the initial belief propagation (BP) decoding fails. Specifically, decoding begins with a single active decoder, designated as the main decoder, while the remaining decoders function as auxiliary decoders. Should the main decoder fail to successfully decode the codeword, the auxiliary decoders then perform bit-flipping operations.

Fig. 7 illustrates the overall mechanism of the proposed OBPL-DF decoding approach. The decoder Π_0, corresponding to the original BP decoder, is selected as the main decoder. If the output codeword from this main decoder satisfies both the sub-matrix verification and the multi-CRC code checks, the decoding is deemed successful. Otherwise, the auxiliary decoders engage in multiple-bit-flipping operations. In contrast to the conventional BPL decoding algorithm, where all decoders operate concurrently for every received input, the OBPL-DF algorithm restricts the participation of auxiliary decoders to the bit-flipping phase only when the initial BP decoding is unsuccessful. This selective activation significantly reduces computational complexity and power consumption in integrated circuit implementations compared to traditional BPL decoding methods.

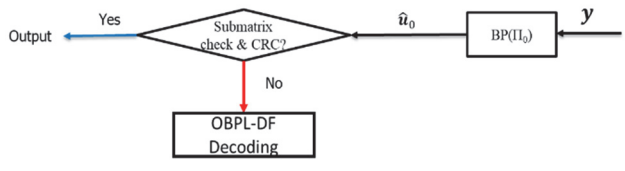

Figure 7 The overview of the proposed OBPL-DF decoding

The fundamental principle underlying the bit-flipping operation in belief propagation flipping (BPF) decoding is that prior knowledge of flipped bits can be utilized to rectify other erroneously propagated messages resulting from previous unsuccessful decoding attempts. Depending on the location of the flipped bit, three distinct outcomes may arise: correct position, incorrect position, and intermediate position. A correct position refers to a scenario in which flipping the bit leads to a successful belief propagation (BP) decoding in the subsequent attempt. Conversely, an incorrect position occurs when the flipped bit corresponds to a correctly decoded bit, causing the next BP decoding attempt to fail. The intermediate position is characterized by a flipped bit that is indeed correct, yet the following BP decoding attempt remains unsuccessful [18]. Although flipping a bit in an intermediate position does not immediately yield successful decoding, it enhances the likelihood of success by better aligning the prior information with the accurate bit value. However, distinguishing among these three cases poses significant challenges in practical decoding implementations. Motivated by the full list bit-flipping approach employed in successive cancellation list (SCL) decoding [7], this study adopts a strategy that considers all possible bit states for flipping. For instance, as illustrated in Fig. 8, when the list size L equals 16, the number of auxiliary decoders employed is 15.

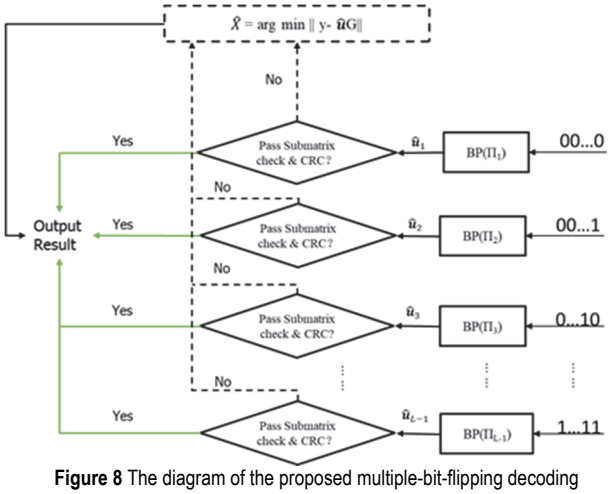

Figure 8 The diagram of the proposed multiple-bit-flipping decoding

Subsequent to the failure of the main decoder to satisfy the joint verification, the auxiliary decoders are engaged, with a flipping order of four. Assuming that the decision value at the flipped position in the main decoder is 0010, the auxiliary decoders assign right-directed messages to the flipped bits ranging from 0000 to 1111, excluding the state 0010. This approach ensures that at least one auxiliary decoder possesses the accurate prior information for the flipped bit, thereby enhancing the likelihood of successful codeword decoding in subsequent belief propagation decoding iterations.

3.3 Proposed Metric for Flipping Set

Motivated by the notable efficacy of the Dynamic Successive Cancellation Flip (DSCF) algorithm [8] within serial decoding frameworks, this study initially introduces the DSCF metric into Belief Propagation Flip (BPF) decoding to detect channel-induced errors. It is important

to highlight that, in DSCF decoding, the flipping set is iteratively updated throughout each decoding attempt. Moreover, the metric updating strategy is inherently based on the sequential nature of Successive Cancellation (SC) decoding. However, Belief Propagation (BP) decoding cannot directly replicate the DSCF metric, which is specifically designed to correct the decision trajectory of SC decoding [8]. To address this limitation, we propose an enhanced metric for updating the set of flipping indices within a sub-block of BPF decoding. The objective is to establish a combination of bit-flipping positions capable of correcting multiple channel-induced errors, informed by the metric provided by the decoder.

Initially, we assume the existence of a flipping set (FS) that contains the non-frozen indices designated for bit-flipping. The FS is denoted as $\varepsilon_\omega = \{j_1,\ldots,j_\omega\}$, where represents the number of channel-induced errors, also referred to as the flipping order. In DSCF decoding, all prior decisions corresponding to non-frozen indices are incorporated into the computation of the decision metric. Consequently, the success probability of SCF decoding $P(\varepsilon_\omega)$ can be expressed as follows:

$$P(\varepsilon_\omega) = \prod_{i\in\varepsilon_\omega} p_e(\hat{u}[\varepsilon_{\omega-1}]_i) \times \prod_{\substack{i<j_\omega \\ i\in\mathcal{A}\setminus\varepsilon_\omega}} (1-p_e(\hat{u}[\varepsilon_{\omega-1}]_i)) \quad (7)$$

where $p_e\left(\hat{u}\left[\varepsilon_{\omega-1}\right]_i\right)$ is the probability which an error incurred at index *i*.

$$p_e(\hat{u}[\varepsilon_{\omega-1}]_i) = P(\hat{u}[\varepsilon_{\omega-1}]_i \neq u_i \mid y, \hat{u}[\varepsilon_{\omega-1}]_1^{i-1} = u_1^{i-1}) \quad (8)$$

To elucidate Eq. (7), we consider a specific example. Assuming the decision bits $\hat{u}_4$ and $\hat{u}_7$ within the *P*(8, 4) polar codes are designated as the flipping bits, collectively referred to as the flipping set (FS) $\varepsilon_2 = \{4,7\}$. When a flipping attempt is performed, the bit $\hat{u}_4$ in the set is flipped first. The probability that flipping a particular bit leads to a correct decision corresponds to the probability that this bit $\hat{u}_4$ initially suffered a channel-induced error. Subsequently, the estimation of the following bit is influenced by the value of the flipped bit $\hat{u}_4$, denoted as $\hat{u}\left[\varepsilon_1\right]_6$. Consequently, the probability of correctly estimating this subsequent bit is represented as $\left(1-p_e\left(\hat{u}\left[\varepsilon_1\right]_6\right)\right)$. Analogously, the probability of an error occurring at the final index (index 7) is equivalent to the probability of a bit-flip at that position. Ultimately, the overall probability of successful decoding via flipping is obtained by taking the product of these individual probabilities associated with flipping the bits in FS.

It is important to highlight that the probability $p_e\left(\hat{u}\left[\varepsilon_{\omega-1}\right]_I\right)$ in Eq. (7) is a conditional probability, assuming that all preceding unfrozen bits have been decoded correctly. However, in practical decoding scenarios, determining the correctness of each unfrozen bit is a challenging task. Instead, one can compute $q_e\left(\hat{u}\left[\varepsilon_{\omega-1}\right]_i\right) = P(\hat{u}\left[\varepsilon_{\omega-1}\right]_i \neq u_i \mid y, \hat{u}\left[\varepsilon_{\omega-1}\right]_1^{i-1})$, which is a conditional probability dependent solely on the previously decoded bits, regardless of their correctness. This probability $q_e\left(\hat{u}\left[\varepsilon_{\omega-1}\right]_i\right)$ can be derived from the log-likelihood ratio (LLR) values as follows:

$$q_e(\hat{u}[\varepsilon_{\omega-1}]_i) = \frac{1}{1+\exp(\left|L[\varepsilon_{\omega-1}]_i\right|)}, \forall i \in \mathcal{A} \quad (9)$$

where $L\left[\varepsilon_{\omega-1}\right]_i$ is the LLR corresponding to the bit u_i. Then the $p_e\left(\hat{u}\left[\varepsilon_{\omega-1}\right]_i\right)$ will be approximated with the $q_e\left(\hat{u}\left[\varepsilon_{\omega-1}\right]_i\right)$, and Eq. (7) will be represented as follows.

$$\begin{aligned} M(\varepsilon_\omega) = & \prod_{i\in\varepsilon_\omega} \frac{1}{1+\exp(\left|L[\varepsilon_{\omega-1}]_i\right|)} \\ & \times \prod_{\substack{i<j_\omega \\ i\in\mathcal{A}\setminus\varepsilon_\omega}} \left(1-\frac{1}{1+exp\left(\left|L[\varepsilon_{\omega-1}]_i\right|\right)}\right) \end{aligned} \quad (10)$$

using the fact that $\frac{1}{1+\exp(x)} = \frac{\exp(-x)}{1+\exp(-x)}$, Eq. (10) could be rewritten as follows:

$$\begin{aligned} M(\varepsilon_\omega) = & \prod_{i\in\varepsilon_\omega} \exp(-\left|L[\varepsilon_{\omega-1}]_i\right|) \times \\ & \prod_{\substack{i<j_\omega \\ i\in\mathcal{A}\setminus\varepsilon_\omega}} \left(\frac{1}{1+\exp(-\left|L[\varepsilon_{\omega-1}]_i\right|)}\right) \end{aligned} \quad (11)$$

Furthermore, in order to bring $q_e\left(\hat{u}\left[\varepsilon_{\omega-1}\right]_i\right)$ closer to the approximated value, a perturbation parameter α was introduced. Such a perturbation parameter could be optimized by Monte-Carlo simulation [8], and Eq. (11) could be represented as follows.

$$\begin{aligned} M_\alpha(\varepsilon_\omega) = & \prod_{i\in\varepsilon_\omega} \exp(-\alpha\left|L[\varepsilon_{\omega-1}]_i\right|) \\ & \times \prod_{\substack{i<j_\omega \\ i\in\mathcal{A}\setminus\varepsilon_\omega}} \left(\frac{1}{1+\exp(-\alpha\left|L[\varepsilon_{\omega-1}]_i\right|)}\right) \end{aligned} \quad (12)$$

Especially, for a bit flip $\varepsilon_1 = \{j_1\}$ with order 1, the above formula can be written as:

$$M_\alpha(\varepsilon_1) = \exp(-\alpha\left|L_{j_1}\right|) \times \prod_{\substack{i<j_1 \\ i\in\mathcal{A}}} \left(\frac{1}{1+\exp(-\alpha\left|L_i\right|)}\right) \quad (13)$$

In Eq. (12), j_1 is a bit-flipping position with order 1 based on Eq. (12), we can constitute the flipped bits in SCF decoding with the highest probability metric $M_\alpha(\varepsilon_\omega)$. However, this metric is based on the serial nature of SC

decoding, as reflected in the condition $i < j_\omega$ and $i \in \mathcal{A} \setminus \varepsilon_\omega$ in Eq. (12). It is well established that belief propagation (BP) decoding operates with parallel characteristics, rendering this condition $i < j_\omega$ obviously not suitable for the BP decoding. Moreover, the metric in Eq. (12) depends solely on the magnitude of the LLR and does not account for the polarization of each information bit. Therefore, it is necessary to refine this metric to enhance its applicability to BP decoding.

Considering the bit-flipping operation within BP decoding, the a priori information processing of a flipped bit influences not only the decoding computations for bits positioned after the flipped bit but also those located before it. Consequently, we have modified the $M_\alpha(\varepsilon_\omega)$ in Eq. (12) into $M_k(\varepsilon_\omega)$ accordingly to better accommodate these characteristics.

$$M_k(\varepsilon_\omega) = \prod_{i \in \varepsilon_\omega} \exp(-\alpha \left| L[\varepsilon_{\omega-1}]_i \right|) \times \prod_{i \in \mathcal{A}_k \setminus \varepsilon_\omega} \left(\frac{1}{1+\exp(-\alpha \left| L[\varepsilon_{\omega-1}]_i \right|)} \right) \tag{14}$$

The location $\mathcal{A}_k$ of the information bits within the *k*-th segment is identified. The second component denotes the probability of correctly decoding all information bits in the *k*-th segment, excluding those within the set FS. This metric is evaluated based on data obtained from the most recent flipping attempt of a specified order $\omega - 1$. When performing multi-bit flipping, it becomes necessary to conduct cumulative flipping attempts in a sequential order. For instance, assuming the presence of 31 auxiliary belief propagation (BP) decoders in the list, it is possible to flip up to five bits during an additional BP decoding attempt, as outlined in subsection 3.2. However, according to Eq. (14), flipping attempts must be conducted in ascending order of flipping order less than five, specifically attempting orders $\omega = 1$, $\omega = 2$, $\omega = 3$, and $\omega = 4$ sequentially. This requirement increases decoding latency, which conflicts with the primary objective of the proposed algorithm design. Therefore, to reduce computational complexity and preserve the inherently low latency characteristic of BP decoding, this study focuses exclusively on the metric associated with bit flips of order one. Consequently, Eq. (13) can be reformulated as follows:

$$M_k(\varepsilon_1) = \exp(-\alpha \left| L_{j1} \right|) \times \prod_{i \in \mathcal{A}_k} \left(\frac{1}{1+\exp(-\alpha \left| L_i \right|)} \right) \tag{15}$$

Meanwhile, the polarization of each information bit is also considered. For each bit-channel $W_N^{(i)}$, its Bhattacharyya parameter $Z\left(W_N^{(i)}\right)$ [4] is introduced here, and for the additive white Gaussian noise (AWGN) channel, its Bhattacharyya parameter could be obtained by

$$Z(W_N^{(i)}) = \exp^{\frac{-1}{2(\sigma_N^{(i)})^2}} \tag{16}$$

where $\left(\sigma_N^{(i)}\right)^2$ is a noise variance of a bit-channel, and it could be calculated by the Gaussian approximation (GA) [20]. Due to Z(W) being an upper bound for the probability of maximum-likelihood (ML) decision error [4], we further denote $E_0\left(1, W_N^{(i)}\right)$ to measure the polarization extent of each bit-channel:

$$E_0(1, W_N^{(i)}) = 1 - \log_2(1 + Z(W_N^{(i)})) \tag{17}$$

which is also a lower bound of mutual information for a bit-channel. Up to this moment, we can generate the per-segment bit-flipping metric:

$$M'_k(j) = 1 - \log_2(1 + Z(W_N^{(i)})) + \exp(-\alpha \left| L_j \right|) \times \prod_{i \in \mathcal{A}_k} \left(\frac{1}{1+\exp(-\alpha \left| L_i \right|)} \right) \tag{18}$$

Due to $0 \le E_0\left(1, W_N^{(i)}\right) \le 1$ and $0 \le M_k(\varepsilon_1) \le 1$, it is worth noting that $0 \le M_k^{'}(j) \le 2$. Although the maximum value in Eq. (18) exceeds 1, this does not adversely impact the selection of flipping bits. This is because the first term in Eq. (18) quantifies the polarization degree of an information bit, while the second term represents the probability that the segment is correctly decoded after flipping this bit. Consequently, a higher value in Eq. (18) corresponds to greater bit reliability. In practical applications, it is necessary to select bits with low reliability, which corresponds to choosing bits with lower metric values as determined by Eq. (18).

4 IMPLEMENTATION OF PROPOSED ALGORITHM

The procedure of the OBPL-DF decoding is described in Algorithm 1. The maximum number of flipping attempts $T_{\max}$ and the number of all decoders *L*. Due to the number of decoders deciding the flipping order ω, the *L* is larger, the flip orders are more. To proceed, t is the number of flipping attempts and V_l represents the *l*-th binary vector with the length ω. Note that the function "dec2bin(ω, l)" represents the integer l is converted to the corresponding binary expression with length ω (line 2). For example, letting L = 32 and $\omega = \log_2 32 = 5$, then V_0 is equal to 00000 and V_{31} is equal to 11111. Here, a special data structure (*F*) and (*S*) is utilized to include a flag (*f*) that indicates whether a sub-block is decoded correctly, bit-flipping indices (*idc*) and the flipping metric of the *j*-th bit in the *k*-th sub-block ($s^{M_k^{'}}$).

It is worth noting that the flipping metric $0 \le M_k^{'}(j) \le 2$ as mentioned in the above subsection. Therefore, ($s^{M_k^{'}}$) is set to 2 at initialization (line 4).

Algorithm 1 PBPL-MBPF decoding

Input: y, $T_{\max}$, L

Output: The estimated information bits $\hat{u}_1$.

1 initialization: $\omega=\left(\log_2 L\right)$, $t=0$;

for $1=0$; $1<L$; $l++$ do

2 $V_l=$ dec2bin(ω, 1);

3 for $k=1$; $k<m$; $k++$ do

4 $F_k=S_k=\left\{f\leftarrow 0, idc\leftarrow\varnothing, s^{M_k^{'}}\leftarrow 2\right\}$

5 $\hat{u}_1\leftarrow$ BP decoding by main decoder;

if a ll segments pass through the submatrix check and the check by CRC code then

6 return $\hat{u}_1$; //Output Result

7 else

8 for $k=1$; $k<m$; $k++$ do

9 if (CRC.decode$\left(k\right)==$ true) then

10 $R_{1,k}=\left(1-2\hat{u}_{1,k}\right)\times\infty$;

11 else

12 compute the flipping metric $M_k^{'}\left(j\right)$ of Each bit by (Eq. (18));

$S_k\leftarrow \text{push}\left(1,\{j\},M_k^{'}\left(j\right)\right)$;

$F_k\leftarrow \text{sort}\left(S_k\rightarrow s^{M_k^{'}},\text{ascending}\right)$;

13 // decoding with bit-flipping order ω

14 for $k=1$; $k<m$; $k++$ do

15 if ($f==0$) then

16 continue;

17 else

18 while $t<T_{\max}$ do

19 $t++$;

$\varepsilon_\omega\leftarrow \text{pop}\left(F_k\left(idc\right)\right)$ with number ω;

$V'\leftarrow V\setminus\hat{u}_{1,\varepsilon_\omega}$;

for $l=1$; $l<L-1$; $l++$ do

20 $R_{l,0,\varepsilon_\omega}\leftarrow\left(1-2\left[V_{l,\varepsilon_\omega}\right]\right)\times\infty$;

21 $\hat{u}_1\leftarrow$ BPL decoding by auxiliary decoders;

if all segments pass through the submatrix check and the check by CRC code then

22 return $\hat{u}_1$; //Output Result

23 else

24 for $l=1$; $l<L$; $l++$ do

25 for $k=1$; $k<m$; $k++$ do

26 if (CRC.decode$\left(k\right)==$ true) then

27 $R_{1,k}=\left(1-2\hat{u}_{1,k}\right)\times\infty$;

$F_k\left(f\leftarrow 0\right)$;

28 else

29 continue;

Following the execution of belief propagation (BP) decoding by the main decoder, if a sub-block is identified as incorrectly decoded, the calculation of the flipping metric for each bit within this sub-block is initiated. The resulting structure, denoted as *S*, is then sorted in ascending order based on the flipping metric values. Subsequently, the sorted metrics along with their corresponding indices are stored in a separate data structure, *F* (refer to line 12). The correctness of sub-block decoding is assessed using CRC.decode(), with particular emphasis on the fact that the information bits in the first half are determined by the submatrix check, satisfying the specified condition $\left(\hat{u}_{1,N/2}G_{N/2}=\hat{x}_{n,N/2}\right)$. Thereafter, a sequence of additional decoding attempts employing bit-flipping strategies is conducted contingent upon the CRC output and the condition $T_{\max}$ (line 18). During each decoding attempt, bit-flipping indices are sequentially extracted from structure *F* and aggregated into a set *FS* ε_ω (line 19). Concurrently, one binary vector corresponding to the decoded bits from the main decoder is removed from the set *V*, yielding a new set *V'*. The elements of each binary vector in *V'* are then designated as flipping bits corresponding to the positions in ε_ω, and the prior information associated with these bits is reset accordingly (line 20). Finally, auxiliary decoders perform belief propagation list (BPL) decoding and verify the correctness of the decoding outcome. If the decoding remains unsuccessful, correctly decoded sub-blocks are identified, and their corresponding prior information is updated (lines 26-27). It is noteworthy that the computational complexity is reduced by restricting the sorting of the flipping metric to a single sub-block rather than the entire set of information bits, as was the case in previous successive cancellation flip (SCF) decoding approaches. Furthermore, the presence of auxiliary decoders obviates the need for incrementally increasing the order ω of additional flipping attempts; instead, bit flips $\omega=\log_2 L$ are applied directly.

5 SIMULATION RESULTS

This section presents the simulation results evaluating the decoding performance of the proposed algorithm. A comparative analysis is conducted against other decoding algorithms, including the current state-of-the-art method for polar codes. The communication channel is modelled as a Binary Input Additive White Gaussian Noise (BI-AWGN) channel, with Binary Phase Shift Keying (BPSK) employed as the modulation scheme. The proposed algorithm utilizes concatenated Cyclic Redundancy Check (CRC) codes, specifically a 4-bit CRC characterized by a given generator polynomial $g\left(x\right)=x^4+x^3+x^2+x+1$. Information bits are optimized for each signal-to-noise ratio (SNR) level and selected using the Gaussian Approximation method [20]. Corresponding parameters $Z\left(W_N^{(i)}\right)$ are derived according to Eq. (16). For a polar code of length 1024, the first half contains 144 information bits, which are subjected to submatrix verification. The second half comprises 400 information bits, inclusive of CRC bits, partitioned into eight segments with CRC protection applied. Among these, segments corresponding to information bits in the second half with their polarization have lengths of 20, 28, 36, and 44 bits, arranged in ascending order of polarization. Conversely, the remaining information bits in the second half are partitioned uniformly with the length of 68 bits. Additionally, the total

number of bits designated for flipping during decoding is fixed at 30.

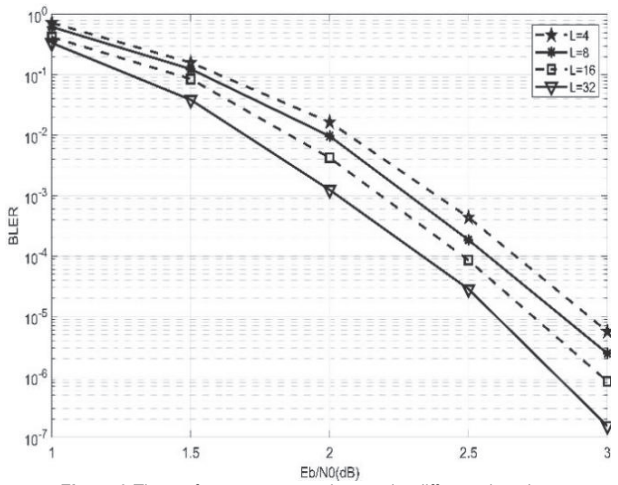

Figure 9 The performance comparison at the different decoders

Fig. 9 presents a comparative analysis of the decoding performance of the proposed algorithm across varying numbers of decoders. The code length is fixed at 1024, with 544 information bits inclusive of the CRC code. As depicted, an increase in the number of decoders corresponds to a gradual improvement in decoding performance. Notably, the performance differential between decoder quantities is less pronounced than that observed in CA-SCL decoding, particularly when the number of decoders is fewer than 16. This characteristic offers a viable alternative for practical implementations where hardware costs are a significant constraint.

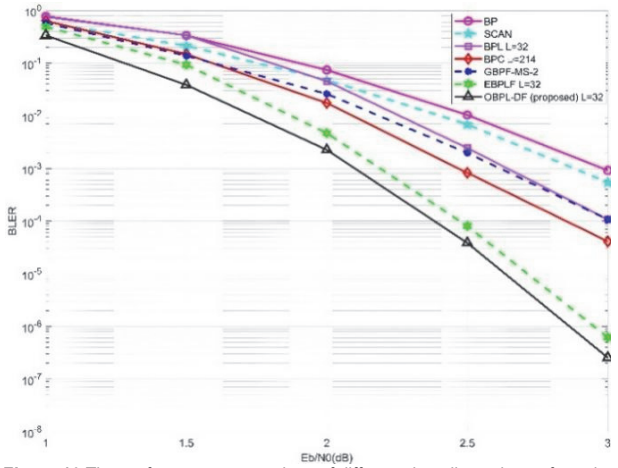

Figure 10 The performance comparison of different decoding scheme for polar codes (1024, 0.5)

Figs. 10 and 11 provide a comparative evaluation of different decoding schemes applied to polar codes of medium to long lengths (1024 and 2048). For the polar code with a length of 2048, the total number of information bits, including the CRC code, is 1088. The first half contains 313 information bits, while the second half is divided into 16 segments. Consistent with the segmentation approach used for the 1024-length polar code, the segmentation in the latter half is based on the polarization levels of the corresponding information bits. Specifically, for bits exhibiting insufficient polarization, segment lengths increase in ascending order of polarization, with sizes of 20, 28, 36, and 43, respectively. Conversely, segments comprising sufficiently polarized bits uniformly have a length of 54. As illustrated in Fig. 10, the proposed decoding scheme achieves performance comparable to that of the EBPLF decoding under identical conditions. Moreover, the proposed method significantly outperforms BPC decoding, which is configured with a maximum of 214 flipping bits, whereas the proposed decoding utilizes only 128 flipping bits. Additionally, Fig. 11 demonstrates that the proposed decoding approach surpasses the performance of NPBPL decoding and exceeds that of the state-of-the-art decoding method referenced in [25], given the same list size.

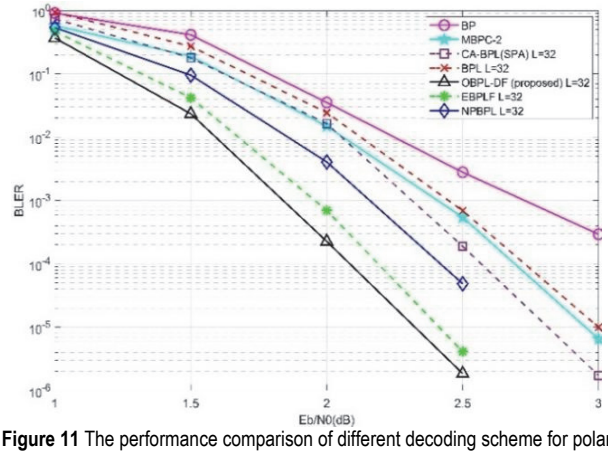

Figure 11 The performance comparison of different decoding scheme for polar codes (2048, 0.5)

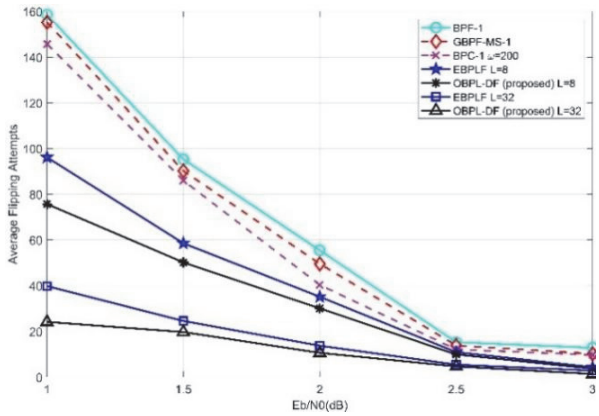

Figure 12 Comparison of average flipping attempts under different SNRs for polar codes (1024, 0.5)

It is well established that decoding latency is significantly affected by the number of flipping attempts in bit-flipping (BPF) decoding. In this study, we analyze the average number of flipping attempts across varying signal-to-noise ratios (SNRs). As illustrated in Fig. 12, all observed trajectories demonstrate a pronounced decline in average flipping attempts as the SNR increases. Moreover, the proposed decoding method consistently achieves a lower average number of flipping attempts compared to the BPF-1, GBPF-MS-1, BPC, and EBPLF decoding algorithms. Notably, the average flipping attempts associated with the proposed decoding decrease further with an increase in list size, particularly under low SNR conditions. This finding highlights that the proposed decoding approach requires substantially fewer flipping attempts than existing bit-flipping algorithms employed in belief propagation (BP) decoding.

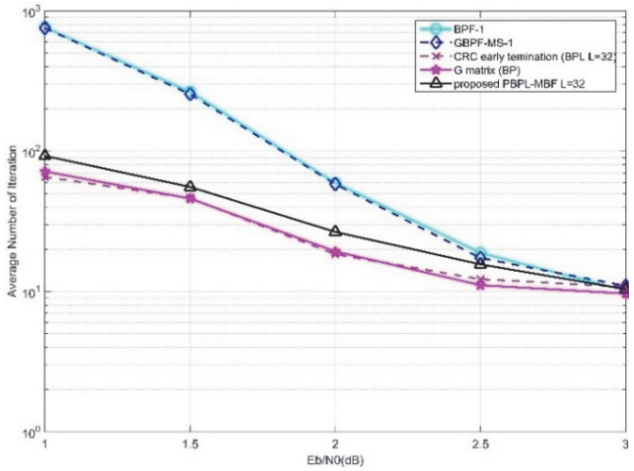

Figure 13 The average iteration of different decoding scheme with early stopping criteria for polar codes (1024, 0.5)

Subsequently, Fig. 13 presents a comparison of the average number of iterations among several prominent decoding algorithms. It is important to note that these algorithms incorporate early stopping criteria based on cyclic redundancy check (CRC) codes, while the traditional BP decoding algorithm employs early stopping criteria based on the G-matrix check. The BPF-1 algorithm exhibits an average iteration count comparable to that of the GBPF-MS-1 algorithm. In contrast, the proposed algorithm demonstrates a reduced number of iterations relative to both algorithms. This improvement can be attributed to the proposed algorithm's capability to minimize flipping attempts by concurrently flipping multiple bits. Additionally, Fig. 13 reveals that the average number of iterations for the proposed algorithm closely approximates that of the BPL decoding algorithm when utilizing the CRC-based early stopping criterion. This observation further corroborates that the proposed algorithm necessitates a minimal number of flipping attempts.

Table 1 Computational complexity for polar codes under different decoding algorithms

Decoding algorithms	Computational complexity
SC	$\mathcal{O}(N\log N)$
SCL	$\mathcal{O}(LN\log N)$
BP	$\mathcal{O}(2I_{\text{ave}}N\log N)$
BPL	$\mathcal{O}(2LI_{\text{ave}}N\log N)$
BPF	$\mathcal{O}(2M_{\text{iter}}N\log N+2TM_{\text{iter}}N\log N)$
OBPL-DF	$\mathcal{O}\begin{pmatrix}2M_{\text{iter}}N\log N\\+2T(L-1)M'_{\text{iter}}N\log N\end{pmatrix}$

Tab. 1 provides a comparative analysis of the computational complexity associated with various decoding algorithms, where I_{ave} denotes the average number of iterations and T represents the maximum number of flipping attempts. It is important to highlight the parameters M_{iter} and M'_{iter}, which correspond to the average iterations during successful belief propagation (BP) decoding and the additional decoding attempts, respectively. Although the proposed algorithm exhibits a higher computational complexity relative to the bit-flipping (BPF) and bit-flipping with list (BPL) decoding algorithms, Fig. 12 illustrates that the number of additional flipping attempts is substantially lower than that observed in BPF decoding. Furthermore, it is infrequent for $L-1$ decoders to be activated for supplementary flipping attempts, as the majority of codewords are successfully decoded using a single decoder. Consequently, the proposed algorithm demonstrates significantly reduced energy consumption compared to BPL decoding in practical scenarios.

Table 2 An example of a table

Decoding algorithms	Latency Analysis (clock cycles)
BP	$2(n+1)\times I_{\text{ave}}$
BPL	$2(n+1)\times I_{\text{ave}}$
SCL	$2N+K-2$
GBPF-MS	$2(n+1)\times I_{\text{ave}}+2$
OBPL-DF	$2(n+1)\times I_{\text{ave}}+2$

Tab. 2 presents a comparison of decoding latency among BP, BPL, successive cancellation list (SCL), generalized bit-flipping with merged sets (GBPF-MS) [15], and the proposed decoding algorithms. The latency of these decoding methods is quantified by the number of clock cycles required to complete the decoding process. Specifically, BP decoding requires $2(n+1)$ clock cycles per iteration, with n cycles allocated for each message-passing step and an additional cycle for assigning channel output log-likelihood ratios (LLRs) and a priori information. The BPL decoding process is recognized to demand an equivalent number of clock cycles as BP decoding, since all decoders in BPL operate concurrently. Notably, the decoding clock cycles for cyclic redundancy check-aided SCL (CA-SCL) decoding are comparable to those of SCL decoding, excluding the overhead associated with CRC decoding [17]. In the GBPF-MS decoding algorithm, following successful CRC verification of the decoding result, the BP decoder must re-access the received LLRs, resulting in a two-clock cycle interval between consecutive frames [15]. This characteristic is also present in the proposed algorithm. As depicted in Fig. 13, the proposed algorithm achieves a markedly lower average number of iterations compared to the GBPF-MS decoding algorithm, particularly in low signal-to-noise ratio (SNR) regions. This finding suggests that the proposed algorithm requires substantially fewer clock cycles than the GBPF-MS decoding algorithm and exhibits latency performance comparable to that of BP or BPL decoding algorithms.

6 CONCLUSION

This paper introduces an OBPL-DF decoding algorithm incorporating a multiple-bit-flipping strategy. Firstly, the proposed algorithm addresses the limitation of conventional bit-flipping (BPF) decoding, where sequential flipping of selected bits leads to increased latency due to repeated decoding attempts. Secondly, it mitigates the high energy consumption of traditional bit-parallel list (BPL) decoding, which requires L decoders to perform decoding operations for each codeword. Additionally, an enhanced metric for selecting bits to be flipped is developed, which accounts for both the

polarization and magnitude of each information bit. A dynamic flipping set (*FS*) is then determined, making the approach more compatible with the proposed decoding scheme. Experimental results demonstrate that the proposed algorithm achieves a lower average number of flipping attempts compared to state-of-the-art BPF decoding methods, while also delivering improved block error rate (BLER) performance relative to leading BPL decoding techniques.

Acknowledgments

This work has been funded by the National Natural Science Foundation of China (Grant No. 62273142), Hunan Provincial Natural Science Foundation of China (Grant No. 2022JJ50253), and the Research Foundation of the Education Bureau of Hunan Province, China (Grant Nos. B08007054 and 22A0490), No 23BSQD30.

7 REFERENCES

[1] 3GPP. Technical specification group radio access network. http://www.3gpp.org/ftp/Specs/archieve/38_series/38.212/. 2018.TS.38.212V.15.1.1.

[2] Afisiadis, O., Balatsoukas-Stimming A., & Brug A. (2014). A low complexity improved successive cancellation decoder for polar codes. *48th Asilomar Conference on Signals, Systems and Computers*, 2116-2120. https://doi/org/10.1109/acssc.2014.7094848

[3] Arikan, E. (2008). A performance comparison of polar codes and reed-muller codes. *IEEE Communications Letters*, *12*(6), 447-449. https://doi/org/10.1109/lcomm.2008.080017

[4] Arikan, E. (2009). Channel polarization: A method of constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, *55*(7), 3051-3073. https://doi/org/10.1109/tit.2009.2021379

[5] Arikan, E. (2010). Polar codes: A pipelined implementation. *4th International Symposium Broadband Communication (ISBC)*, 11-14.

[6] Arili, A. C. & Gazi, O. (2019). Noise-aided belief propagation list decoding of polar codes. *IEEE Communications Letters*, *23*(6), 1285-1288. https://doi/10.1109/lcomm.2019.2918535

[7] Cao, A., Zhang, L., Qiao, J., & He, Y. (2019). An LLR-based segmented flipped SCLdecoding algorithm for polar codes. *IEEE/CIC International Conference on Communications in China (ICCC)*, 724-729. https://doi/org/10.1109/iccchina.2019.8855853

[8] Chandesris, L., Savin, V., & Declercq, D. (2018). Dynamic-scflip decoding of polar codes. *IEEE Transactions on Communications*, *66*(6), 2333-2345. https://doi/org/10.1109/ tcomm.2018.2793887

[9] Elkelesh, A., Ebada, M., & Ten Brink, S. (2018). Belief propagation list decoding of polar codes. *IEEE Communications Letters*, *22*(8), 1536-1539. https://doi/org/10.1109/lcomm.2018.2850772

[10] Ercan, F., Condo, C., Hashemi, S. A., & Gross, W. J. (2018). Partitioned successive cancellation flip decoding of polar codes. *IEEE International Conference on Communications (ICC)*, 1-6. https://doi/org/10.1109/ICC.2018.8422464

[11] Feng, B., Liu, R., & Tian, K. (2021). A novel post-processing method for belief propagation list decoding of polar codes. *IEEE Communications Letters*, *25*(8), 2468-2471. https://doi/org/10.1109/lcomm.2021.3086479

[12] Geiselhart, M., Elkelesh, A., & Ten Brink, S. (2020). Crc-aided belief propagation list decoding of polar codes. *IEEE International Symposium Information Theory (ISIT)*, 395-400. https://doi/org/10.1109/ISIT44484.2020.9174249

[13] Guo, J., Shi, Z., Liu, Z., & Liu, Q. (2016). Multi-crc polar codes and their applications. *IEEE Communications Letters*, *20*(2), 212-215. https://doi/org/10.1109/lcomm.2015.2508022

[14] Niu, K., Chen, K. (2012). CRC-aided decoding of polar codes. *IEEE Communications Letters*, *16*(10), 1668-1671. https://doi/org/10.1109/lcomm.2012.090312.121501

[15] Shen. Y., Song, W., You, X., & Zhang, C. (2020). Improved belief propagation polar decoders with bit-flipping algorithms. *IEEE Transactions on Communications*, *68*(11), 6699-6713. https://doi/org/10.1109/tcomm.2020.3017656

[16] Shen. Y., Song, W., & Zhang, C. (2020). Enhanced belief propagation decoder for 5g polar code with bit-flipping. *IEEE Transactions on Circuits and Systems II*, *67*(5), 901-905. https://doi/org/10.1109/tcsii.2020.2984536

[17] Tal, I. & Vardy, A. (2015). List decoding of polar codes. *IEEE Transactions on Information Theory*, *61*(5), 2213-2226. https://doi/org/10.1109/tit.2015.2410251

[18] Teng, C. F. & Wu, A. Y. A. (2021). Convolutional neural network-aided tree-based bit-flipping framework for polar decoder using imitation learning. *IEEE Transactions on Signal Processing*, *69*, 300-313. https://doi/org/10.1109/tsp.2020.3040897

[19] Wang, X., Li, J., & Li, Z. (2019). Belief propagation bit-strengthening decoder for polar codes. *IEEE Communications Letters*, *23*(11), 1958-1961. https://doi/org/10.1109/lcomm.2019.2939801

[20] Wu, D., Li, Y., Sun, Y. (2014). Construction and block error rate analysis of polar codes over AWGN channel based on gaussian approximation. *IEEE Communications Letters*, *18*(7), 1099-1102. https://doi/org/10.1109/lcomm.2014.2325811

[21] Yu, Y., Pan, Z., & You, X. (2019). Belief propagation bit-flip decoder for polar codes. *IEEE Access*, *7*, 10937-10946. https://doi/org/10.1109/access.2019.2891951

[22] Yuan, B. & Parhi, K. (2014). Early stopping criteria for energy efficient low-latency belief propagation polar code decoder. *IEEE Transactions on Signal Processing*, *62*(24), 6469-6506. https://doi/org/10.1109/tsp.2014.2366712

[23] Zhang, M., Li, Z., & Liao, X. (2022). Higher-order belief propagation correction decoder for polar codes. *Entropy*, *24*, 534-545. https://doi.org/10.3390/e24040534

[24] Zhang, M., Li, Z., & Xing, L. (2021). An enhanced belief propagation decoder for polar codes. *IEEE Communications Letters*, 25(10), 3161-3165. https://doi.org/10.1109/lcomm. 2021.3104152

[25] Chen, W. et al. (2023). An Efficient Belief Propagation List Flip Decoder for Polar Codes. *IEEE Communications Letters*, *27*(9), 2264-2268. https://doi.org/10.1109/lcomm. 2023.3292268

[26] Ercan, F., Condo, C., & Gross, W. J. (2019). Improved Bit-Flipping Algorithm for Successive Cancellation Decoding of Polar Codes. *IEEE Transactions on Communications*, *67*(1), 61-72. https://doi/org/10.1109/tcomm.2018. 2873322

Contact information:

Yinyou MAO
School of Computer and Electrical Engineering,
Hunan University of Arts and Science, Changde, China

Wenxue TAN
School of Computer and Electrical Engineering,
Hunan University of Arts and Science, Changde, China

Jianying LI
Key Laboratory of Hunan Province for Control Technology of Distributed Electric Propulsion Air Vehicle, Changde, China

Lin NI
(Corresponding author)
Information Support Force Engineering University, China
E-mail: nilin@sina.com