Optimization of Polar Code BP Bit-Flipping Decoding Based on an Adaptive Genetic Algorithm

Xiaojie Liu,
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: 2719130786@qq.com

Guiping Li,
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: liguiping@xatu.edu.cn

***Abstract*—Polar codes, as capacity-achieving error-correcting codes, have become a cornerstone of modern communication systems due to their excellent theoretical performance. Compared with the Successive Cancellation (SC) decoding algorithm, the Belief Propagation (BP) decoding algorithm for polar codes offers advantages such as parallel output and ease of hardware implementation. However, the bit-flipping decoding schemes based on BP still exhibit a significant performance gap in frame error rate (FER) compared to the Successive Cancellation List (SCL) decoding. To address the demand for high reliability and low power consumption in practical applications, this paper proposes an optimized bit-flipping scheme in which the flipping set is constructed using an adaptive genetic algorithm. The proposed method first reduces the computational complexity of the initial BP decoding process by adopting the Offset Min-Sum (OMS) approximation. During the construction of the flipping set, an adaptive mechanism dynamically adjusts the crossover and variational probabilities based on the fitness of individuals in the population. The indices of the information bits are used as individuals in the genetic algorithm, enabling the fitness values to gradually evolve from local optima toward a global optimum. This approach allows for more accurate identification of bit positions prone to decoding errors. For a polar code with a length of 1024 and a code rate of 0.5, the proposed AGA-OMS-BPF decoder achieves approximately 1.3 dB BER performance gain at a BER of 10^{-5} compared with the conventional BPF decoder. Simulation results demonstrate that the proposed method effectively reduces the number of unsuccessful BP decoding attempts by constructing a more efficient flipping set, thereby achieving performance gains with reduced computational complexity.**

Keywords-Polar Codes; Belief Propagation; Genetic Algorithm; Adaptive mechanism; Bit-flip; Flipping Set

I. INTRODUCTION

In 2008, polar codes were introduced by Professor Arikan [1], representing the first category of channel codes theoretically guaranteed to reach channel capacity. The primary decoding approaches for polar codes include Successive Cancellation (SC) decoding and Belief Propagation (BP) decoding. Compared with SC decoding, BP decoding offers inherent parallelism, resulting in lower decoding latency, which makes it well suited for applications requiring low latency and high throughput. Therefore, this paper focuses on optimizing BP decoding. However, the initial BP decoding algorithm exhibits unsatisfactory frame error rate (FER) performance, necessitating further improvement. To this end, several enhancements to BP decoding have introduced several improvements to BP decoding, including BP List (BPL) [2], BP Bit-Flip (BPF) [3], Enhanced-BPF (EBPF) [4], and BP Correction (BPC) [5].Among these, the BPF decoder offers advantages in terms of computational and memory complexity, making it more suitable for power-constrained application scenarios such as 5G-based vehicular networks, the Internet of Things, and integrated air-space-ground communications [6,7,8,9,10]. Nevertheless, despite these benefits, BPF decoding is generally less stable and harder to implement than the structurally simple and robust BPL and BPC decoders. Therefore, optimizing the BPF decoder is both necessary and promising.

This paper focuses on the Belief Propagation Bit-Flip (BPF) decoding algorithm, which serves as an auxiliary mechanism to the BP decoding

process. BPF combines the high throughput of BP decoding with the performance enhancement capabilities of bit-flipping techniques. Before restarting decoding, it predicts and flips potentially incorrect bits. Constructing an accurate flipping set is thus essential to improving the BLER performance. In recent years, many researchers have proposed improvements for more precise flipping set construction. In 2019, Yu et al. proposed a BPF decoder [4] that constructs flipping set (FS) by selecting bits with high error probabilities based on their Log-Likelihood Ratio (LLR) values. However, the performance of this method is limited by its insufficient accuracy in identifying erroneous bits. Furthermore, it does not perform direct bit-flipping but rather attempts to guess each bit as either '0' or '1', resulting in limited performance gains. In [11], a CNN-based BPF decoding algorithm (CNNBPF) was introduced. It employs CRC to verify the BP decoding output and extracts residual information from the final BP iteration that fails the CRC check. This information is mapped into images and fed into a Convolutional Neural Network (CNN) to classify bits for flipping, significantly improving decoding performance. However, the training process of the CNN introduces high computational complexity. In [4], an EBPF decoding scheme was proposed to reduce decoding complexity by narrowing the range of unreliable bits, achieving better performance than [3]. Building on this, [12] introduced a high-order GBPF decoding method capable of flipping multiple bits in a single decoding attempt. The flipping set is refined using a reliability metric (RM) based on the reliability of polarized subchannels. While this approach improves decoding performance, the large number of flipping attempts leads to increased complexity. Moreover, erroneous bits are still measured only by their LLR magnitude or RM value, limiting decoding efficiency. In [13], a new weighted metric (WM) was introduced to identify potential error locations by leveraging the reliability and divergence of bit estimations. With the same number of flipping attempts, the decoding outperforms the GBPF decoding and the BP correction (BPC) decoding.

Although the above optimization algorithms and deep learning methods have enhanced the performance and efficiency of BPF decoding, further improvements are needed in flipping set construction. Designing a flipping set can be viewed as an optimization problem, for which Genetic Algorithm (GA) are well suited. GA can be employed to optimize the flipping set construction process in BPF decoding. However, as code length and decoding iterations increase, the parallel nature of BP decoding introduces higher computational complexity. To address this, the proposed method incorporates the OMS approximation to reduce the complexity of early BP decoding stages. When BPF decoding fails, a adaptive GA is applied to optimize the flipping set. Experimental results verify that the proposed AGA-OMS-BPF decoding scheme maintains competitive decoding performance while significantly reducing computational complexity under the same code length conditions.

II. BASIC THEORY

A. Belief Propagation Decoding and Bit-Flipping

The BP decoding algorithm is a widely used message-passing scheme, distinguished from other decoding algorithms by its inherent parallelism, which makes it particularly effective for addressing latency constraints. It performs decoding through iterative message exchange on a factor graph. During each iteration, the algorithm adaptively updates the parity-check matrix based on a reliability-based ordering of messages, thereby enhancing the reliability of the decoded information and improving the accuracy of the final decision.

The factor graph of an (N,K) polar code consists of $n=\log_2 N$ stages and $N\times(n+1)$ nodes. Each node carries two types of information: the left message $L_{i,j}^{(t)}$, which propagates from right to left, and the right message $R_{i,j}^{(t)}$, which propagates from left to right. The initialization process of BP decoding is defined as:

$$L_{n+1,j}^{(1)} = \ln \frac{P(y_j | \ x_j = 0)}{P(y_j | \ x_j = 1)} \tag{1}$$

$$R_{i,j}^{(1)} = \begin{cases} 0, & \text{if } j \in A \\ +\infty, & \text{if } j \in A^c \end{cases} \tag{2}$$

In a processing element, the outgoing message of a given node is determined by the incoming messages from the other three connected nodes. The message update rules are given by the following equations:

$$\begin{cases} L_{i,j}^t = g\left(L_{i+1,2j-1}^t, L_{i+1,2\mathrm{j}}^t + R_{i,j+N/2^{\mathrm{i}}}^t\right) \\ L_{i,j+N/2^i}^t = g\left(R_{i,j}^t, L_{i+1,2j-1}^t\right) + L_{i+1,2j}^t \\ R_{i+1,2j-1}^t = g\left(R_{i,j}^t, L_{i+1,2j}^t + R_{i,j+N/2^i}^t\right) \\ R_{i+1,2j}^t = g\left(R_{i,j}^t, L_{i+1,2j-1}^t\right) + R_{i,j+N/2^i}^t \end{cases} \tag{3}$$

Where:

$$g(x,y) = \ln\left(\frac{1+xy}{x+y}\right) \tag{4}$$

To reduce the computational complexity, the Min-Sum approximation is applied, in which the function is approximated as:

$$g(x,y) \approx \mathrm{sign}(x)\mathrm{sign}(y)\min(|x|,|y|) \tag{5}$$

After T iterations, the estimate of the j-th information bit can be obtained by hard decision on the left-propagated LLR $L_{i,j}^{(T)}$ from the final iteration:

$$\hat{u}_j^N = \begin{cases} 0, & \text{if } L_{i,j}^T \geq 0 \\ 1, & \text{if } L_{i,j}^T < 0 \end{cases} \tag{6}$$

When the original BP fails the CRC check, the algorithm flips the least reliable bits in the flipping set and restarts the BP decoding process. This procedure continues until either the maximum number of bit flips is reached or the decoded output passes the CRC verification. In this work, the BPF algorithm performs single-bit flipping by inverting the first bit in the flipping set-that is, the information bit with the highest estimated error probability-by assigning its a priori LLR value an infinite magnitude.

The bit-flipping mechanism inverts the previously estimated value of u_j^N and sets the priori information of $\hat{u}_j^N$ to infinity. So, the initial value of the right-propagated log-likelihood ratio $R_{i,j}^{(1)}$ in Equation 3 is updated as follows:

$$R_{1,j}^{(1)} = \begin{cases} 0, & \text{if } j \in \{A \setminus F\} \\ \infty \times (2\hat{u}_j^N - 1), & \text{if } j \in F \\ +\infty, & \text{if } j \in A^c \end{cases} \tag{7}$$

Here, F denotes the set of flipping positions, for which the values are set to infinity.

B. Genetic Algorithm

Genetic Algorithm (GA) is an intelligent heuristic optimization method originally proposed by Professor Holland [14]. GA treats each candidate solution of an optimization problem as an individual in a population and evaluates its fitness using a suitable fitness function. Through genetic operations such as selection, crossover, and variational, the population is iteratively optimized. In the context of polar code decoding, GA can be employed to optimize the selection of bit-flipping positions, flipping strategies, and control parameters, providing global search capability and a fitness-driven mechanism. This effectively enhances the performance of the BPF decoding algorithm, particularly for short to moderate code lengths. Figure 1 illustrates the specific iterative process of the genetic algorithm.

C. Offset Min-Sum Approximation

As the code length increases, the inherent parallelism of the BP decoding algorithm leads to a rise in computational complexity during the GA-based BPF decoder. Moreover, achieving optimal decoding performance within a limited number of iterations becomes challenging.

Inspired by the OMS approximation proposed in [15], the algorithm presented in this work replaces the conventional Min-Sum approximation with the OMS method. Accordingly, Equation (5) is updated as follows:

$$g(x,y) \approx \text{sign}(x)\text{sign}(y) \times \max(\min(|x|,|y|) - \beta, 0) \tag{8}$$

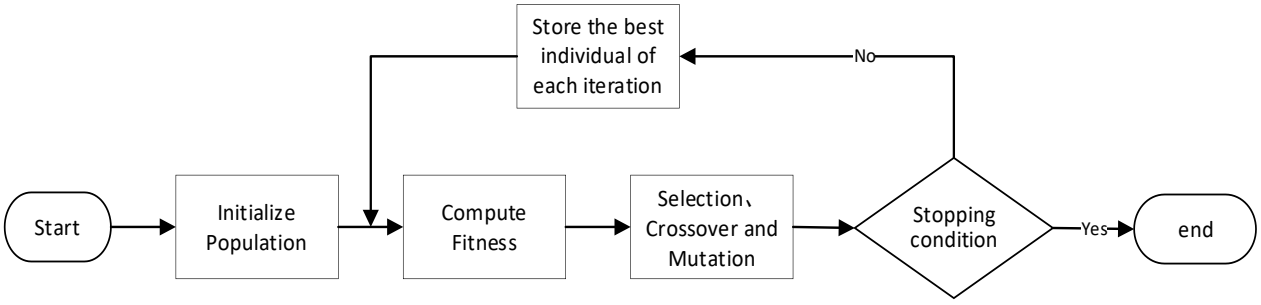

Figure 1 Framework of the Genetic Algorithm.

Here, the role of the max function is to exclude contributions from LLRs with a magnitude smaller than the offset factor. The computation of max(x,0), implemented by a switch that selects based on the sign bit of x, adds virtually no extra computational burden. Compared to the Min-Sum approximation, the OMS approximation computes the output of the processing node using only a subset of the inputs from adjacent nodes and introduces an offset parameter. This approach further improves the overall algorithm performance. Accordingly, Equation (3) is updated as follows:

$$\begin{cases} L_{i,j}^{t} = g\left(L_{i+1,j}^{t-1}, L_{i+1,j+N/2}^{t-1} + R_{i,j+N/2^{i}}^{t}, \beta_{L_{i,j}^{t}}\right) \\ L_{i,j+N/2^{i}}^{t} = g\left(R_{i,j}^{t}, L_{i+1,j}^{t-1}, \beta_{L_{i,j+N/2^{t}}^{t}}\right) + L_{i+1,j+N/2}^{t-1} \\ R_{i,j}^{t} = g\left(R_{i,j}^{t}, L_{i+1,j+N/2^{t}}^{t-1} + R_{i,j+N/2^{i}}^{t}, \beta_{R_{i+1,j}^{t}}\right) \\ R_{i,j+N/2^{i}}^{t} = g\left(R_{i,j}^{t}, L_{i+1,j}^{t+1}, \beta_{R_{i+1,j+N/2^{i}}^{t}}\right) + R_{i,j+N/2^{t}}^{t-1} \end{cases} \tag{9}$$

Here, $\beta_{L_{i,j}^{t}}$ and $\beta_{R_{i,j}^{t}}$ refer to the offset parameters introduced during the t-th iteration when calculating $L_{i,j}^{t}$ and $R_{i,j}^{t}$, respectively. This method replaces the original multiplications with simple sign operations, comparisons, and subtractions, thereby significantly improving computational efficiency and reducing algorithmic complexity.

III. Proposed Construction For Flip Set By Using Genetic Algorithm

A. Daptive Crossover and Variational Operations

In the optimization of the BPF decoding algorithm, the genetic algorithm is applied to the construction of the flipping set. However, the settings of crossover and variational probabilities during the genetic operations have a significant impact on the decoding efficiency of the BPF algorithm. Similar to variational probability, both excessively high and low crossover probabilities can hinder performance improvement. To address this, an adaptive probability adjustment mechanism driven by fitness values is introduced within the BPF decoding optimization framework. Rather than being fixed, the crossover and variational probabilities are dynamically adjusted based on the fitness of individuals, enabling a better balance between exploration and exploitation during the search process.

The adaptive crossover probability is formulated as Equation (10), where f_c denotes the higher fitness value of the two individuals undergoing crossover, f_{max} is the maximum

fitness value within the population, and f_{avg} is the average fitness of all individuals. Constants k_1 is set to 0.8 according to the traditional adaptive mechanism. During the dynamic adjustment of the crossover probability, if f_c is less than or equal to the population average fitness f_{avg} , it indicates relatively poor performance. In this case, a higher crossover probability k_1 is assigned to encourage more recombination with other individuals to generate better offspring. Conversely, if f_c exceeds the average fitness f_{avg} , particularly when approaching the population's optimum, it suggests that the individual possesses a superior genetic structure. Therefore, the crossover probability is reduced to minimize the risk of disrupting these high-quality individuals, thereby protecting elite solutions.

$$P_c = \begin{cases} k_1, & f_c \le f_{avg} \\ k_1 \dfrac{f_{max} - f_c}{f_{max} - f_{avg}}, & f_c > f_{avg} \end{cases} \quad (10)$$

The adaptive variational probability is given by Equation (11), where f_m represents the fitness of the individual undergoing variational, $f_{\max}$ is the maximum fitness value within the population, and f_{avg} is the average fitness of all individuals. Constants k_2 is set to 0.01 based on the traditional adaptive mechanism. During the dynamic adjustment of variational probability, if the fitness f_m of the mutated individual is less than or equal to the population average f_{avg} , it indicates that the individual may be trapped in a local optimum or lacks competitiveness. In this case, a higher variational probability k_2 is assigned to encourage genetic variation, thereby providing a chance to escape the current state and explore new solution spaces. Conversely, when f_m is relatively high, the variational probability is reduced accordingly to preserve the individual's superior genetic structure and prevent random variational from damaging its advantageous traits.

$$P_m = \begin{cases} k_2, & f_m \le f_{avg} \\ k_2 \dfrac{f_{max} - f_m}{f_{max} - f_{avg}}, & f_m > f_{avg} \end{cases} \quad (11)$$

The pseudocode for the adaptive crossover probability function is presented in Algorithm 1:

Algorithm 1 Adaptive crossover probabilities Function

Input: population **P**, fitness list **Fit**
Output: Adaptive crossover probabilities **p_c**
1: fmax = max (Fit)
2: favg = sum (Fit) / len (Fit)
3: f_c= max (Randomly select two individuals and obtain their fitness values f1, f2)
4: **if** fc <= favg:
5: **p_c** = 0.8
6: **else**: **p_c** = 0.8 * ((fmax - fc) / (fmax - favg))

B. Construction of Belief Propagation Bit-Flipping Sets Based on an Adaptive Genetic Algorithm

1) Population Initialization and Encoding

For polar code bit-flipping decoding algorithms, each flipping position is identified by the index of an information bit, making the information bit indices suitable as candidate solutions for the flipping problem. Therefore, in this work, the initial population P of the genetic algorithm is initialized using the information bit indices, with the population size set equal to K. Let $P = \{P_i\}, i = 1, 2, \ldots, K$ denote the entire population, where P_i represents each individual. Considering the operability of crossover and variational, binary encoding is employed. Since binary encoding corresponds to decimal numbers starting from 0, whereas information bit indices in polar codes start from 1, the original indices must be decremented by 1 before encoding to ensure correct mapping within the genetic algorithm. For example, if an individual in the population is P(i)=78, its corresponding binary encoding is V(i)={1,0,0,1,1,0,1}.

2) Fitness Function

In this work, the fitness is calculated directly using the objective function, where the objective function is denoted as y(x), and the fitness function is represented as Fit(x).

$$Fit(y(x)) = \begin{cases} y(x), & \text{Maximize} \\ -y(x), & \text{Minimize} \end{cases} \quad (12)$$

Since the channel reliability estimation in this work adopts the Gaussian approximation method, the $P_e(i)$ of each subchannel can be calculated using Equation (13) once the mean $m_N^{(i)}$ of the LLR distribution for each subchannel is obtained. This enables the reliability estimation of the subchannels.

$$P_e(i) = \int_{-\infty}^{0} \frac{1}{2\sqrt{\pi m_N^{(i)}}} \cdot \exp\left(\frac{-(x - m_N^{(i)})^2}{4m_N^{(i)}} \right) dx \quad (13)$$

Since the subchannel error probability $P_e(i)$ reflects the reliability of the subchannel, a larger $P_e(i)$ indicates lower reliability. For the problem studied in this paper, the optimal solution obtained by the genetic algorithm should include the unreliable subchannels to guide the flipping of bits in the flipping set constructed by the genetic algorithm, thereby improving decoding performance. Therefore, the subchannel error probability $P_e(i)$ is chosen as the objective function, and the error probability of each subchannel P_e is used as the fitness value. A higher fitness value indicates a less reliable corresponding subchannel.

The construction method of the belief propagation bit-flipping set based on the adaptive genetic algorithm is described as follows. Figure 2 illustrates the flowchart of the belief propagation bit-flipping set construction based on the adaptive genetic algorithm.

Step 1: Initialization of Fitness Vector and Population. First, the mean $m_N^{(i)}$ of the LLR distribution is obtained via Gaussian approximation, and the $P_e(i)$ of each subchannel is calculated using Equation 13. This value is then used to initialize the fitness vector Fit. Subsequently, the population is initialized based on the indices of the information bits, with the population size equal to the K.

Step 2: Selection Operation. The initial population undergoes selection using the tournament selection method to form a new population P_s. The mechanism of tournament selection involves randomly picking k individuals from P_s, comparing their fitness, and keeping the one with the highest fitness. This procedure repeats until the new population is finalized once it reaches the size of the original one.

Step 3: Crossover and Variational Operations. The population P_s is encoded in binary. Following the calculation of adaptive crossover probabilities as described in Algorithm 1, single-point crossover is performed. Variational is carried out similarly. The individual with the lowest fitness resulting from this genetic operation is stored in the path vector.

Step 4: Iteration. The algorithm cycles through Steps 2 and 3 until the maximum generation count is met.

Step 5: Processing of the Path Vector. The occurrences of each individual in the path vector are counted. Indices with nonzero occurrences are used to construct the polar code flipping set. These indices are then sorted in ascending order according to their fitness values.

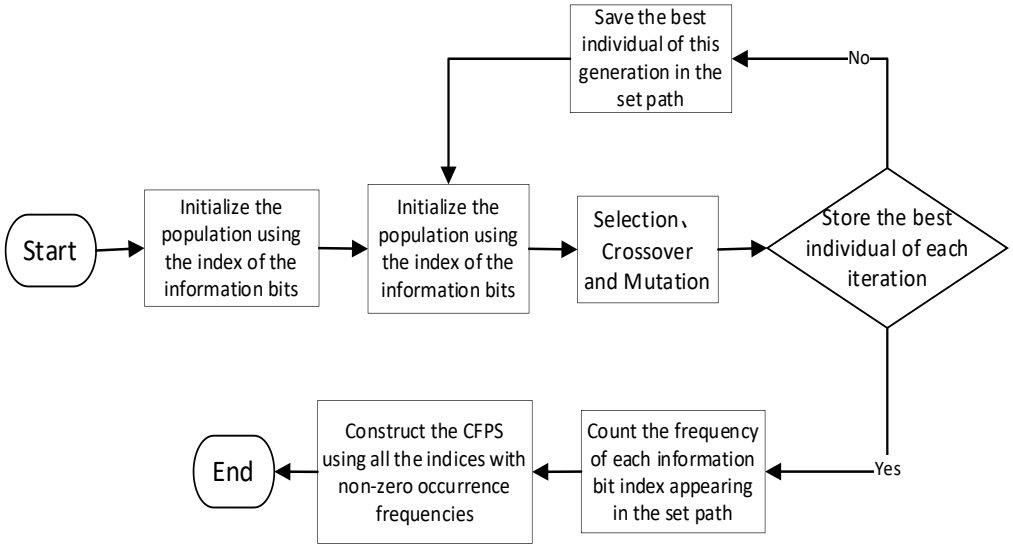

Figure 2 Flowchart of Belief Propagation Flipping Set Construction Based. on Adaptive Genetic Algorithm.

C. BPF Decoding Algorithm Based on Adaptive Genetic Algorithm

After obtaining a new flipping set through the adaptive genetic algorithm, this section introduces the BPF based on the adaptive genetic algorithm (AGA-OMS-BPF). First, the BP decoding algorithm optimized by the OMS approximation is employed. If the decoded output passes the CRC, decoding is considered successful; otherwise, the BPF decoding algorithm is invoked. During the bit-flipping process, individual bits indexed in the flipping set are flipped by modifying the initial value of the right-propagated LLR according to Equation (7). After each flip, BP decoding is re-executed followed by CRC verification. If decoding succeeds, the BER and FER are computed. Otherwise, further bits from the flipping set are flipped iteratively until either the maximum number of flips is reached or decoding succeeds. The framework of this algorithm is illustrated in Figure 3, and its pseudocode is presented in Algorithm 2.

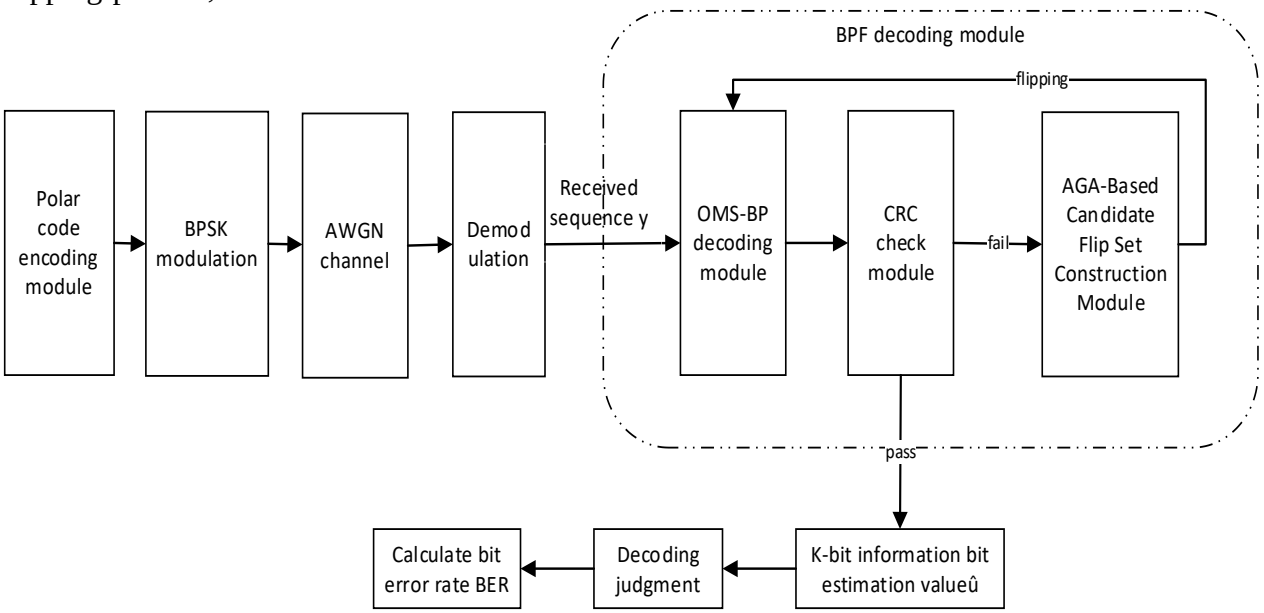

Figure 3 Framework Diagram of the AGA-OMS-BPF Decoding Algorithm

Algorithm 2 AGA-OMS-BPF Decoding algorithm

Input: The sequence after decoding $\mathbf{u}_1^N$, Flip set **path[]**

Output: The sequence after flipping $\mathbf{u}_1^N$

1: $\mathbf{u}_1^N \leftarrow$ **OMS-BP(** y_1^N **,** A **)**

2: if CRC($\mathbf{u}_1^N$) != success

3: FS_sorted ← sorting $\boldsymbol{i} \in \boldsymbol{FS} - \boldsymbol{AGA}$ in an increasing order

4: **while** Flip_num <= FNUM:

5: Bit-Flipping

6: BPDecoder(y_1^N , A)

7: **if** CRC($\mathbf{u}_1^N$) != success

8: Flip_num += 1

9: return $\mathbf{u}_1^N$

IV. Simulation Results And Analysis

A. Decoding Performance Analysis

Figure 4 presents a comparative analysis of the BER performance among various algorithms with length N=1024 and R=0.5, over a SNR range of 1.0-3.0dB. The AGA-OMS-BPF decoding algorithm is compared with the BP decoding algorithm, BPF-CS decoding algorithm, and other reference schemes. Specifically, BPF-CS represents the BPF based on the CS framework, while CRC-SCL2 and CRC-SCL4 denote the CRC-aided successive cancellation list decoding methods with list sizes of 2 and 4, respectively.

As evidenced by the results, the proposed AGA-OMS-BPF algorithm demonstrates significant performance improvements compared to both the BP and BPF-CS decoding algorithms. When compared specifically with the BPF-CS algorithm, AGA-OMS-BPF exhibits higher efficiency and superior performance in decoding tasks under high SNR conditions. In comparison with the CRC-SCL2 algorithm, the proposed AGA-OMS-BPF shows slightly inferior BER performance at SNRs below 2.4dB. However, when the SNR exceeds 2.4 dB, the AGA-OMS-BPF algorithm demonstrates notably superior BER performance, highlighting its enhanced decoding capability in moderate to high SNR regimes.

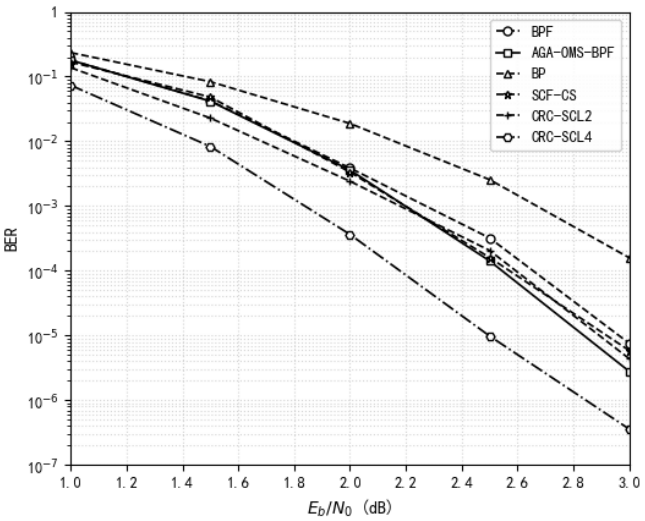

Figure 4 BER Performance of Different Algorithms for N=1024.

B. Decoding Complexity Analysis

1) Computational Complexity Analysis

Figure 5 presents a comparison of the average normalized complexity of different decoding algorithms under N=1024 and R=0.5. Here, the normalization is performed with respect to the standard BP decoder; that is, one unit of normalized complexity corresponds to the computational cost of one full BP decoding. For bit-flipping-based decoding algorithms, the average normalized complexity reflects the average number of BP decoding attempts required for successful decoding. This metric is widely used to evaluate the computational burden of flipping-based decoding schemes.

To quantify the complexity of each algorithm, we provide the computational complexity formulas as follows:

a) Bp: *The complexity per iteration is* $O(N\log N)$ *, and the total complexity is proportional to the number of iterations* I_{BP} *, yielding:*

$$C_{BP} = I_{BP} \cdot NlogN \tag{14}$$

b) Bpf*: The complexity includes the standard BP iterations plus additional flipping attempts. Let* F *denote the average number of flipping*

trials and the C_{flip} represents the cost per flip operation, typically $O(1)$ per bit. Then:

$$C_{BPF} = I_{BP} \cdot NlogN + F \cdot C_{flip} \quad (15)$$

c) Aga-oms-bpf*: Let P be the population size, G the number of generations, and C_{OMS} the cost of OMS processing. The complexity can be expressed as:*

$$C_{AGA-OMS-BPF} = G \cdot P \cdot (I_{BP} \cdot NlogN + C_{OMS}) \quad (16)$$

d) Scf-cs*: As a CRC-aided SCF with critical set (CS), its complexity combines the SCF structure and CS evaluation. Let T be the average number of flipping attempts, C_{CS} the cost of critical set identification and I_{SC} is the number of SC steps. The complexity can be expressed as:*

$$C_{SCF-CS} = I_{SC} \cdot NlogN + T \cdot C_{flip} + C_{CS} \quad (17)$$

e) Cro-scl2 / cro-scl4*: These are CRC-aided SCL decoders with list sizes L=2 and L=4. The complexity is:*

$$C_{CRO-SCL} = L \cdot NlogN \quad (18)$$

As shown in Figure 5, which illustrates the average normalized complexity with BP decoding complexity taken as the baseline, the BPF decoding algorithm exhibits slightly higher complexity than the BP algorithm at low SNR due to the introduced flipping mechanism. However, as the SNR increases, the complexity gradually decreases, eventually falling below that of standard BP decoding. This indicates that the algorithm can effectively reduce computational overhead under favorable channel conditions. The complexity of the AGA-OMS-BPF decoding algorithm is considerably higher than that of BP at low SNRs, primarily owing to the substantial additional computations required for population iterations. Nevertheless, as the SNR rises, the dynamic optimization and early stopping mechanisms of the adaptive genetic algorithm contribute to a reduction in complexity, bringing it close to or even below that of the BP decoding algorithm. In low-SNR environments, the AGA-OMS-BPF algorithm demonstrates a higher level of average normalized complexity compared to both CRC-SCL2 and CRC-SCL4 algorithms. The introduction of the OMS algorithm in the AGA-OMS-BPF decoding scheme further reduces computational complexity, effectively balancing performance and complexity.

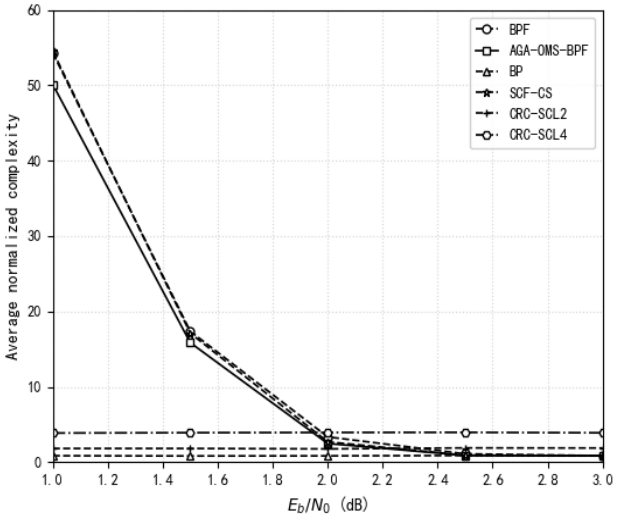

Figure 5 Decoding Computational Complexity of Different Algorithms at N=1024.

2) Decoding Latency Analysis

Figure 6 compares the normalized decoding latency of various algorithms for a code length of N=1024 and code rate R=0.5. The latency is normalized to that of a standard BP decoding attempt, which serves as the baseline unit. Decoding latency reflects the actual time consumption of the algorithm and is closely related to its computational structure and degree of parallelism.

To clarify the source of the latency data, the evaluation principles and main components of each algorithm's latency are given below:

a) Bp*: The latency primarily comes from the sequential execution of message-passing iterations , where I_{BP} is the number of iterations required for decoding. Its latency is modeled as:*

$$L_{BP} = I_{BP} \tag{19}$$

b) Bpf: *In addition to BP iterations, this algorithm introduces extra operations including LLR sorting and flipping trials. Its latency is modeled as:*

$$L_{EPF} = I_{BP} + F \cdot (C_{sort} + C_{flip}) \tag{20}$$

Where F is the average number of flipping trials, $C_{sort} = O(NlogN)$ accounts for the sorting overhead of LLR values, and C_{flip} represents the latency of a single flip attempt.

c) Aga-oms-bpf: The latency of this algorithm is dominated by the adaptive genetic optimization. Let G be the number of generations and P the population size. The latency is estimated as:

$$L_{AGA-OMS-BPF} = G \cdot P \cdot (I_{BP} + C_{OMS}) \tag{21}$$

d) Crc-scl2 / crc-scl4: As SCL aided by CRC, their decoding processes are largely sequential. The normalized latency is primarily determined by the list size L and the code length N:

$$L_{CRC-SCL} = L \cdot N \cdot logN \tag{22}$$

It can be observed from Figure 6 that among all the flipped decoding algorithms compared, the BPF based critical set algorithm exhibits the highest normalized decoding latency.

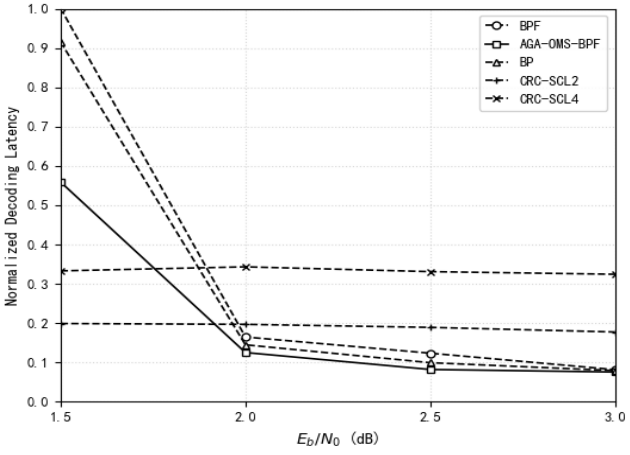

Figure 6 Decoding Latency of Different Algorithms at N=1024.

This is attributed to the fact that when constructing the flip set, the BPF based critical set algorithm requires comparing and sorting the LLR values of all information bits, which thereby increases the decoding latency. For a code length of 1024, in comparison with the BPF algorithm, the AGA-OMS-BPF algorithm demonstrates lower normalized decoding latency in the low SNR range. Furthermore, when compared with the CRC-SCL2 algorithm, the AGA-OMS-BPF algorithm not only achieves lower normalized decoding latency across the high SNR range but also outperforms the CRC-SCL2 algorithm in terms of decoding performance within this high SNR range.

C. Analysis of the Adaptive Behavior of the AGA-OMS-BPF Decoder

To more intuitively demonstrate the evolutionary trend of the optimal solution during the iterations of the adaptive genetic algorithm, the fitness values of the best individual in each generation are normalized. This normalization procedure does not participate in the genetic operations such as selection, crossover, or variational, and is solely used for performance analysis and visualization purposes. The genetic algorithm runs for a total of G generations, and the fitness value of the best individual in generation g (where g=1,2,⋯,G) is denoted as $f^{(g)}$. Let the maximum fitness value over all generations be denoted as $f_{\max} = \max\left\{ f^{(1)}, f^{(2)}, \ldots, f^{(G)} \right\}$. Then, the normalized fitness value for generation g is defined as:

$$f_{\text{norm}}^{(g)} = \frac{f^{(g)}}{f_{\max}} \tag{23}$$

The normalized fitness values $f_{\text{norm}}^{(g)}$, lie within the interval [0,1]. This normalization effectively eliminates the influence of differing original fitness scales caused by varying problem instances or parameter settings, thereby enabling comparability of the evolutionary process under different experimental conditions. It also

highlights the dynamic trend of the optimal solution converging towards the global optimum.

Figure 7 illustrates the fitness evolution trend of the AGA-OMS-BPF decoding algorithm during the iterative process with length N=1024 and R=0.5. By recording the best fitness value of the population in each generation of the genetic algorithm, this figure visually reflects the search trajectory and convergence behavior of the algorithm throughout the optimization process.

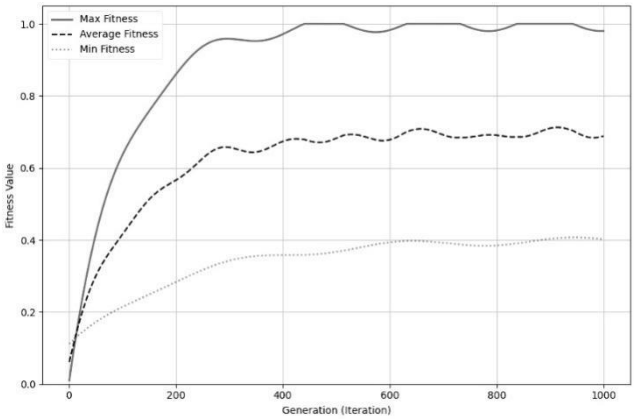

Figure 7 Fitness Evolution Trend in the AGA-OMS-BPF Decoding Algorithm.

As shown in the figure 7, during the early stages of population evolution, the maximum fitness rapidly increases. Superior individuals in the population are quickly improved through selection, crossover, and variational operations, indicating that the algorithm can effectively escape local optima near the initial solution. In this phase, both the average and minimum fitness values exhibit noticeable upward trends, reflecting well-maintained population diversity and effectively preventing premature convergence. In the middle and later stages, the maximum fitness curve gradually flattens and approaches the theoretical optimum, while the average fitness curve steadily rises toward the maximum fitness, demonstrating an overall improvement in population fitness and a stabilization of the optimization effect. The minimum fitness slowly increases with low fluctuation, indicating that the population retains a certain degree of solution diversity, which facilitates further exploration of potentially better solutions and helps avoid getting trapped in local optima.

The adaptive genetic algorithm dynamically adjusts the crossover and variational probabilities: when individual fitness is low, these probabilities increase to enhance population diversity and avoid early stagnation at local minima; when fitness is high, the variational probability is reduced to preserve superior solutions and accelerate convergence. This mechanism effectively guides the decoding process from local optima toward the global optimum, fully leveraging the global search capability of genetic operations and the dynamic adjustment advantage of the adaptive mechanism, thereby improving decoding accuracy and convergence speed.

V. Conclusions

This work presented an optimized belief propagation bit-flipping (BPF) decoding algorithm for polar codes, termed AGA-OMS-BPF, which integrates the advantages of the Offset Min-Sum (OMS) approximation and the adaptive genetic algorithm (AGA). The proposed method employs the OMS approximation to reduce the computational complexity of the BP decoding stage, while the adaptive genetic algorithm is utilized to construct an accurate bit-flip set through evolutionary operations, including selection, crossover, and mutation. The crossover and mutation probabilities are dynamically adjusted according to the fitness distribution, enabling the decoding process to evolve toward more reliable flipping positions. From a computational perspective, this work leverages intelligent optimization and numerical simulation techniques to improve decoding efficiency and robustness. Simulation results demonstrate that, for a polar code with length 1024 and code rate 0.5, the proposed AGA-OMS-BPF algorithm achieves approximately 1.3 dB BER performance gain at a BER of 10^{-5} compared with conventional BPF decoding, while effectively reducing decoding complexity. In summary, the proposed algorithm successfully addresses the trade-off between decoding performance and computational cost in BP-based bit-flipping schemes. Future work will focus on extending the proposed adaptive optimization framework to neural network–aided decoders and exploring hardware implementation for real-time communication systems.

REFERENCES

[1] Arikan E. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels [J]. IEEE Transactions on Information Theory, 2009, 55(7):3051-3073.

[2] Ahmed E, Moustafa E, Sebastian C, et al. Belief propagation list decoding of polar codes [J]. IEEE Communications Letters, 2018, 22(8):1536-1539.

[3] Yu Y, Pan Z, Liu N, et al. Belief Propagation Bit-Flip Decoder for Polar Codes [J]. IEEE Access, 2019,7: 10937-10946.

[4] Shen Y, Song W, Ren Y, et al. Enhanced belief propagation decoder for 5G polar codes with bit-flipping [J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020, 67(5): 901-905.

[5] Zhang M, Li Z, Xing L. An Enhanced Belief Propagation Decoder for Polar Codes [J]. IEEE Communications Letters, 2021, 25(10): 3161-3165.

[6] Mohammed, N. A., Mansoor, A. M, Ahmad, R. B., & Azzuhri, S. R. B.(2022). Deployment of Polar Codes for Mission-Critical Machine-Type Communication over Wireless Networks. Computers," Materials*&Continua, 71(1), 1557-1573.

[7] Niu, K., Zhang, P., Dai, J, Si, Z, & Dong, C. (2023). A Golden Decade of Polar Codes: From Basic Principle to 5G Applications. China Communications, 20(2), 94-121.

[8] Han, T, Fu, M., Zhang, M., & Hu, G. (2023). Puncturing scheme for polar codes based on channel reliability estimation. Telecommunication Systems, 82.499-508.

[9] Z. Aharoni, B. Huleihel, H.D. Pfister, and H.H. Permuter, “Data-Driven Polar Codes for Unknown Channels With and Without Memory,”.in -2023IEEE International Symposium on · Information Theory (ISIT), Taipei, Taiwan: IEEE, Jun. 2023, pp.1890-1895.

[10] P. Mohr, S. A.A. Shah, and G. Bauch, “Implementation-Efficient Finite Alphabet Decoding of Polar Codes,”in GLOBECO 2023--2023 -IEEE Global Communications Conference, Kuala Lumpur, Malaysia: IEEE, Dec. 2023, pp. 5318-5323.

[11] C.-F. Teng, K.-S. Ho, C.-H. Wu, S.-S. Wong, and A.-Y. Wu, “Convolutional neural network-aided bit-flipping for belief propagation decoding of polar codes,” 2019, arXiv: 1911.01704.

[12] Shen Y, Song W, Ren Y, et al. Enhanced Belief Propagation Decoder for 5G Polar Codes with Bit-Flipping [J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020, 67(5):901-905.

[13] Z. Yang and L. Chen, “An Enhanced Belief Propagation Decoding Algorithm With Bit-Flipping for Polar Codes,” IEEE Commun. Lett. vol. 29, no. 2, pp. 348–352, Feb. 2025.

[14] Holland John H. Adaptation in natural and artificial systems [J]. Ann Arbor: University of Michigan Press, 1975.

[15] Xu W, Tan X, Be'ery Y, et al. Deep Learning-Aided Belief Propagation Decoder for Polar Codes [J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2020, 10(2): 189-203.