Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 6262929, 11 pages
https://doi.org/10.1155/2023/6262929

Research Article

Two-Round Selection-Based Bit Flipping Decoding Algorithm for LDPC Codes

Soufian Addi,[1] **Mostafa Belkasmi**,[1] **Ahlam Berkani**,[2] **and Ahmed Azouaoui**[3]

[1]*ICES Team ENSIAS, Mohammed V University in Rabat, Rabat, Morocco*
[2]*Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco*
[3]*Faculty of Sciences, Chouaib Doukkali University, El Jadida, Morocco*

Correspondence should be addressed to Soufian Addi; soufian.addi@gmail.com

Received 27 February 2023; Revised 7 October 2023; Accepted 18 October 2023; Published 28 November 2023

Academic Editor: Xuesong Cai

This paper presents a novel iterative reliability-based bit flipping (BF) algorithm for decoding low-density parity-check codes. The new decoder is a single BF algorithm called two-round selection -based bit flipping. It introduces the idea of a two-round selection of the flipped bit, based successively on hard and soft received channel values. In the first stage, a set of unreliable bits is identified, and then a second selection is used, to pick out among them the bit to flip. In the second round of selection, the initial belief about received signals, contributes efficiently to selecting the best candidate bit. We demonstrate through simulations over the binary-input additive white Gaussian noise channel and the Rayleigh fading channel that the proposed algorithm exhibits better decoding performance when compared with some well-known soft decision BF algorithms. A complexity analysis of the proposal and a comparison to other BF decoders are also presented.

1. Introduction

Error-correcting codes (ECC) are used to control errors during data transmission. The fundamental principle of ECC is to add redundant bits to the transmitted data in the emission, while in the reception we use a decoding algorithm to detect and correct errors occurring over noisy communication channels. The low-density parity-check (LDPC) codes [1] are currently considered one of the best next generation ECC that allow data transmission to reach Shannon's limit [2]. These codes were first introduced by Gallager [1, 3] in his pioneering Ph.D. thesis in 1962. In 1996, the LDPC codes were rediscovered by MacKay and Neal [4] who brought them back into prominence. When the sum-product algorithm (SPA) [5] is used for decoding, it has shown near-Shannon's limit-capacity performance. Many state-of-the-art communication systems adopt the LDPC codes in their standards, such as 5G network systems, second-generation satellite broadcasting systems (DVB-S2), and IEEE 802.11 systems.

LDPC codes can be decoded by many well-known decoding algorithms [6] such as bit flipping (BF) algorithm, and SPA algorithm [7, 8]. The BF algorithm, proposed by Gallager [1, 3], is a hard decision algorithm that flips a set of bits based on the values computed by the flipping function (FF) for each iteration. Even if the BF algorithm is much simpler than the probabilistic SPA algorithm (soft decision), its bit error rate (BER) performance is far from optimal. Therefore, many variants of Gallager's BF algorithms have been proposed to reduce the performance gap, but in some cases with an increase in complexity. In this class of decoders we find the candidate bit-based bit flipping (CBBF) [9], the weighted candidate bit based bit-flipping (WCBBF) [10], and the single bit flipping (SBF) [11]. In CBBF, a reliability of unsatisfied parity-check equations is calculated in addition to the reliability of each bit. For WCBBF decoding, the authors used a weighted reliability of the parity-check equations, and the weights are prefixed integers. The SBF [11] decoder flips a single bit chosen carefully in each iteration. All of these decoders belong to hard decision variants of the original Gallager's BF algorithm.

Nevertheless, for the hard decision decoding algorithms, considerable performance degradation is noticed compared to the soft decision algorithms. That is why BF techniques moved toward the category of simplified soft decision decoding algorithms. The latter use not only hard information but also soft information during the decoding process. The work on this class of decoding algorithm starts with the weighted bit flipping (WBF) decoder [12]. Algorithms in this class allow improvements in performance without a large increase in complexity. Other algorithms, following the approach of WBF, tried to improve the reliability metric and/or the method of selecting the flipped bits (the FB). They achieve different degrees of enhancement from WBF in performance and convergence rate. We quote in this class-modified weighted bit flipping (MWBF) [13], reliability-ratio-based weighted bit-flipping algorithm (RRWBF) [14], gradient descent bit flipping (GDBF) [15, 16], and the dynamic weighted bit-flipping decoding algorithms (DWBF) [17].

This paper introduces a new reliability-bit based bit-flipping algorithm for decoding LDPC codes called two-round selection-based bit flipping (TRSBF). We show hereafter that the proposed algorithm achieves good tradeoffs between BER performance and decoding complexity.

Our decoder is not in the class of variants of the WBF decoder, but it is indeed a soft decoder. At each iteration, a selection in two rounds is used to pick out the bit to flip. More precisely, first-round selection uses only hard decisions to form a candidate pool. The candidate bits selected in this round are those contained in more than some fixed number of unsatisfied parity-check equations. In the second round, the best candidate bit, which is the closest to the received word, is selected from the candidate pool and flipped. Here, the neighborhood is calculated in terms of Euclidean distance from the received word. At each iteration of our decoder, only a single bit will be flipped.

The remainder of this paper is organized as follows: in Section 2, we will present an overview of the BF algorithm and its variants. Section 3 provides details of the proposed soft information BF algorithm. The simulation results and threshold optimization are presented in Section 4. In Section 5, the decoding complexity comparison is discussed. Finally, conclusions are drawn in Section 6.

2. Bit Flipping Algorithm and Its Variants

2.1. Preliminaries. Let C a binary LDPC code of length n. C is defined by the null space of a parity-check matrix $H = [h_{j,\,k}]$ with m rows and n columns. The code C is said to be a regular LDPC code if the matrix H has constant column weight d_c and constant row weight d_r and is said to be irregular otherwise.

We assume that code words $c = (c_0, c_1 \ldots c_{n-1})$ obtained at the output of the encoder are modulated by a binary phase shift keying (BPSK) modulator and transmitted over a binary input AWGN channel with a variance σ^2. The sequence $r = (\mathrm{r}_0, r_1 \ldots \mathrm{r}_{n-1})$ stands for the sequence of soft channel values obtained at the receiver's output. The hard-decision information $z = (\mathrm{z}_0, z_1 \ldots z_{n-1})$ associated with the sequence r is as follows:

$$z_i = \begin{cases} 1, & \text{if } r_i \geq 0 \\ 0, & \text{else} \end{cases}. \tag{1}$$

We introduce the sequence $\hat{z}$ as the bipolar value sequence corresponding to the hard decision sequence z and define it as follows:

$$\hat{z} \triangleq (2z_0 - 1, 2z_1 - 1, \ldots, 2z_{n-1} - 1). \tag{2}$$

The syndrome $\boldsymbol{s}$ defined by $s = \mathrm{z.H^T}$ is calculated at the first stage of the decoder. If the syndrome $s = \mathrm{z.H^T} = 0$, we can say that z is the most likely transmitted codeword. Otherwise, the decoding process begins.

We denote $\mathrm{N}\ (j) = \{k,\ 0 \leq\ \mathrm{k} \leq n{-}1\text{: } \mathrm{h}_{j,k} = 1\}$ the set of code bits that participate in the j^{th} parity-check equation, and $M\ (k) = \{j,\ 0 \leq j \leq m{-}1\text{: } h_{j,k} = 1\}$ the set of checks that contain the k^{th} code bit.

2.2. Bit Flipping Algorithm. A typical BF algorithm (GBF) [1, 3] is a simple hard-decision algorithm that flips the bits involved in a large number of unsatisfied check equations that exceed a threshold (T) for LDPC codes, because they are most probably incorrect bits. The algorithm is terminated once all the parity-check equations are satisfied, which means a valid code word has been found or the maximum number of iterations, $p_{\max}$ is reached. The algorithm, requires only integer operations for decoding and can therefore be easily implemented by an electronic circuit.

The main part of the standard BF algorithm is the calculation of the flipping metric for each bit and each iteration called the FF. The FF values allow for tentative bit decisions and depend on the binary-value checksums and on the bits connected with the check equations. For the BF algorithm, the FF can be equivalently expressed in two ways as the following formulas:

$$e_k^{(1)} = \sum_{j \in M(k)} \left(2s_j - 1\right), \tag{3}$$

$$v_k = \sum_{j=0}^{m-1} < s_j, h_{j,k} >. \tag{4}$$

The quantity v_k is the scalar product of the syndrome and the k^{th} column of H and it represents the number of unsatisfied parity checks containing the k^{th} bit. It gives information about the reliability of the k^{th} received bit. It is easy to prove that:

$$v_k = \sum_{j \in M(k)} s_j. \tag{5}$$

The sequence $v = (v_0, v_1 \ldots v_{n-1})$ is the so-called reliability profile of the received sequence r (or more precisely of the hard version z) [18].

```
Step 0: Initialize the parameters: $p = 0$ (p is the iteration counter) and T (depends on the variant of the algorithm).
Step 1: Compute $s = (s_0, s_1 \ldots s_{m-1}) \leftarrow z.H^T$. If $s = 0$, then stops the algorithm.
Step 2: Compute $e_k^{(1)}$ for all indices k.
Step 3: If max $(e_k^{(1)}) < T$ (or $\max(v_k) < T$), then stops the algorithm.
Step 4: Flip all bits $z_k$, in the sequence z, with $e_k^{(1)} \geq T$, $p \leftarrow p + 1$.
Step 5: If $p > p_{max}$, then stop the algorithm. Else go to Step 1.
```

Algorithm 1: Gallager BF algorithm.

The steps of a standard BF (GBF) algorithm are described in the Algorithm 1 as follows:

2.3. Single Bit Flipping Algorithm. The single bit flipping (SBF) algorithm [11] is a variant of the standard BF that stays within the scope of hard-decision decoding algorithms. In each iteration, SBF flips one elected bit to avoid flipping correct bits, in contrast to the standard BF, which flips many bits each iteration, and thus may need a longer time for convergence.

The SBF algorithm uses Equation (5) for computing FF values. It did not use a threshold but needs to find a maximum of FF values. The steps 3 and 4 of the SBF are as follows:

(i) **Step 3$'$:** Find the index $k_0 = \text{argmax}\ (v_k)$ where k belongs to $\{0,.., n-1\}$

(ii) **Step 4$'$:** Flip the bit z_{k_0}, $p \leftarrow p + 1$.

The SBF results in better performance than the standard BF and converges toward the final solution with a lower number of iterations [11].

2.4. Candidate Bit Based Bit Flipping. The CBBF algorithm [9] uses the correlation data between the column vectors of the parity-check matrix and the syndrome vector to decode. It has minimal decoding complexity and does not require soft information.

The algorithm has a new parameter $\delta > 0$ which is an integer valued threshold for a decision on candidate bits. We calculate $v_k$in the Equation (5), If $v_k > \delta$, then v_k is marked as a candidate bit for BF.

The steps 3–6 of the CBBF are as follows:

(i) **Step 3$''$:** Find $u_{max} = \text{maxi}\ u_i$. If $u_{max} \leq \delta$, then terminate the decoding procedure.

(ii) **Step 4$''$:** Let c_m be the number of candidate bits included in the m-th parity check equation and calculate $w_m = c_m - 1$ for $m = 1, 2, \ldots, M$.

(iii) **Step 5$''$:** For $i = 1, 2, \ldots, M$ with $u_i = u_{max}$, calculate r_i by:

$$r_i = \sum_{m \in M(i)} w_m. \tag{6}$$

(iv) Step 6$''$: Flip all bits vi with ri = mini ri and $u_i = u_{max}$. Let $l \leftarrow l + 1$. If $l >$ lmax, then terminate the decoding procedure. Go to Step 1.

2.5. Weighted Bit-Flipping Algorithm. The WBF algorithm [12] is a variant of the BF algorithm, and it is considered a soft decoder. In the algorithm, throughout the decoding process, the weights of checks are decided by the soft received channel values and remain unchanged. That is because the weights reflect the decoder's belief in the channel's behavior. The FF of the WBF algorithm can be expressed by the following general formula:

$$e_k^{(2)} = \sum_{j \in M(k)} \left(2s_j - 1\right).r_{\min}^j, \tag{7}$$

where $r_{\min}^j = \min|r_i|$ is the minimum absolute of soft values r_i for the bits participating in the j^{th} parity-check equation (minimum for all indices i in the set $N(j)$).

The WBF algorithm combines the checksum values and the reliability of received messages to make decisions, therefore, the algorithm yields better decoding performance when compared with the BF algorithm.

2.6. Gradient Descent Bit-Flipping Algorithm. The GDBF algorithm [15] derives its FF in Equation (7) by computing the gradient of a nonlinear objective function instead of using a weighted checksum-based FF, which is comparable to the log-likelihood function of the bit decisions with checksum constraints.

The FF of the GDBF algorithm can be expressed by the following general formula:

$$e_k^{(3)} = \sum_{j \in M(k)} \left(2s_j - 1\right) - r_k(\widehat{z}_k). \tag{8}$$

2.7. Noisy Gradient Descent Bit-Flipping Algorithm. In an effort to avoid undesirable local maxima, the noisy GDBF (NGDBF) [16] improves efficiency by adding a random perturbation at each iteration.

The FF of the NGDBF algorithm can be expressed by the following general formula:

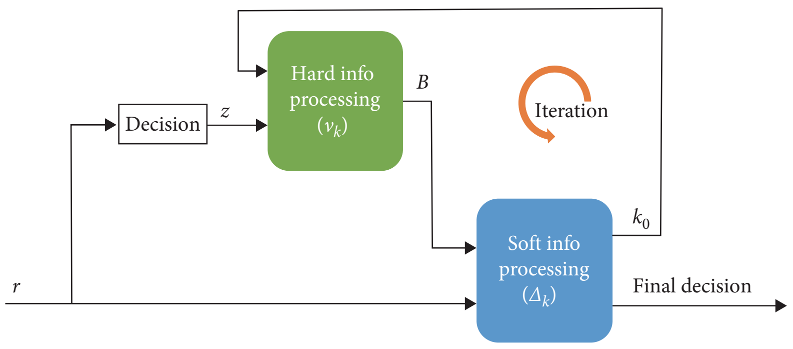
Hard info processing (v_k)
Decision
z
B
Iteration
k_0
Soft info processing (Δ_k)
r
Final decision

Figure 1: Block diagram of the proposed algorithm.

$$e_k^{(4)} = q_k + w. \sum_{j\in M(k)} \left(2s_j - 1\right) - r_k\left(\hat{z}_k\right), \tag{9}$$

where $w \in \mathbb{R}^+$ is a syndrome weight parameter and q_k is a Gaussian distributed random variable with zero mean and variance $\sigma^2 = \eta^2 N_0/2$, where $0 < \eta < 1$. All q_k are independent and identically distributed.

3. Proposed Bit-Flipping Algorithm

3.1. Motivation. All soft BF decoding algorithms combine hard metrics with soft metrics to decide which bit to flip. We believe that the magnitude of a received bit carries additional information on its reliability and can be used separately for decision-making about it. We describe hereafter a new algorithm based on a two-round selection strategy: the TRSBF decoder. The latter is made up of two stages. The first/second round selections are made by the first/second stages, respectively. The proposed algorithm works as follows (see Figure 1).

The first stage, consisting of FF processing, is based only on hard information. This stage can be considered a filter for the next processing step. On the other hand, the second stage of processing is based on soft information. The first stage passes a Set B of unreliable bit positions to the second stage. In the second stage, no FF calculus is done. The processing in the second stage provides a single bit (bit of position k_0) to be flipped. The two processes mentioned constitute a single iteration of the algorithm. In contrast, the known soft BF decoding algorithms combine, hard and soft information into a unique metric for the FF.

3.2. The Proposed TRSBF Algorithm. In the first stage, the TRSBF algorithm calculates the check-based value v_k about regarding the symbol r_k by FF in Equation (4). The value v_k represents the number of unsatisfied parity checks (UPC) containing the bit z_k. Then consider the set of bit positions that satisfy the threshold condition denoted by:

$$B = \{k : v_k \geq T, 0 \leq k < n\}, \tag{10}$$

where B is the set of bits in z that have the largest parity-check failures. Thus, these bits are the less reliable ones.

The identification of Set B is the goal of the first stage of our algorithm. Set B can be seen as a pool of good candidate bits for a second selection.

A first step in the second stage consists of determining a certain number of tentative decision sequences $z^{(k)}$. For every k belonging to the Set B, a sequence $z^{(k)}$ is obtained from the sequence z as follows:

$$z_i^{(k)} = \begin{cases} z_i + 1, \mathit{if} \quad i = k \\ z_i \text{ else} \end{cases}. \tag{11}$$

Let $\hat{z}^{(k)}$ be the bipolar sequence corresponding to $z^{(k)}$.

Then the TRSBF algorithm calculates the squared Euclidean distance between the received soft sequence r and the sequence $\hat{z}^{(k)}$, as shown in Equation (12).

$$d_e^2\left(r, \hat{z}^{(k)}\right) = \sum_{i=0}^{n-1} \left(r_i - \hat{z}_i^{(k)}\right)^2 \text{for each index } k \in B. \tag{12}$$

But minimizing the squared Euclidean distance in Equation (12) is the same as minimizing the set of Δ_k values, with Δ_k defined as follows:

$$\Delta_k = \sum_{i\in B} \left(r_i - \hat{z}_i^{(k)}\right)^2 \text{for each index } k \in B. \tag{13}$$

A final step of the second stage of the algorithm consists of finding the index k_0 of the nearest sequence $z^{(k)}$ from the received sequence r as shown by Equation (14):

$$\mathrm{k}_0 = \operatorname{argmin}(\Delta_k), k \in B. \tag{14}$$

The position k_0 is the selected bit position to be flipped in the current iteration.

The steps of the TRSBF algorithm are described in the Algorithm 2 as follows:

The first stage of the decoding algorithm consists of Steps 1 and 2 and the second stage consists of Steps 3 and 4. The threshold T is to be optimized for each code (see Section 4.1).

One of the strengths of the proposed decoding algorithm is that it can be used for regular LDPC codes as well as irregular ones.

```
Step 0: Initialize the parameters: $p = 0$ and T.
Step 1: Compute $s = (s_0, s_1 \ldots s_{m-1}) \leftarrow z.H^T$. If $s = 0$, then stops the algorithm.
Step 2: Identify the set $B$. If $B$ is empty then stops the algorithm.
Step 3: Compute $\Delta_k$ for each index $k \in B$.
Step 4: Find the index $k_0$ and flip the bit $z_{k_0}$ in z, $p \leftarrow p + 1$.
Step 5: If $p > p_{max}$, then stop the algorithm. Else go to Step 1.
```

Algorithm 2: TRSBF algorithm.

Table 1: Parameters of used LDPC codes.

Code	Length (n)	Dimension k'	m	Rate	d_c
LDPC1	73	45	73	0.616	9
LDPC2	1,057	813	1,057	0.77	33
LDPC3	1,057	813	244	0.77	3
LDPC4	273	191	273	0.69	17

Table 2: Simulation parameters.

Simulation parameters	Value
Min number of transmitted blocks	1,000
Min number of residual bit errors	200
Max number of iterations (p_{max})	45

4. Simulation Results

In order to illustrate the decoding performance of the proposed decoding algorithm, four regular LDPC codes (see Table 1) are considered and used for the simulation: the LDPC1, LDPC2, and LDPC4 are difference-set codes (DSC) family [12], and the LDPC3 code is a pseudorandom or Gallager code and was selected from MacKay's online encyclopedia [19].

We carried out extensive simulations using a communication chain implemented in language C. The communication chains contain an AWGN/Rayleigh channel and a BPSK modulation/demodulator. The Monte Carlo method was used for simulations.

The performance output is given in terms of bit error rate (BER) and block error rate (BLER) as a function of signal-to-noise ratio (SNR), with the default simulation's parameters outlined in Table 2.

4.1. Optimization of the Threshold. The threshold T of the proposed decoding algorithm is optimized using simulation results for the three chosen codes. The criterion for optimization is the BER performance at several SNRs. The communication chains contain an AWGN channel and a BPSK modulation/demodulator. The Monte Carlo method was used for simulations.

These values of SNR depend on the selected code. The interval where T is located depends on the d_c parameters of the codes, but we always have $T \epsilon [1, d_c]$.

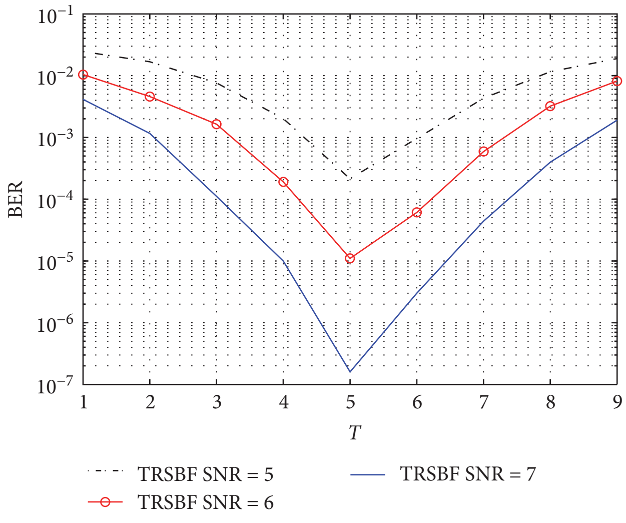

Figure 2: Optimization of T for LDPC1.

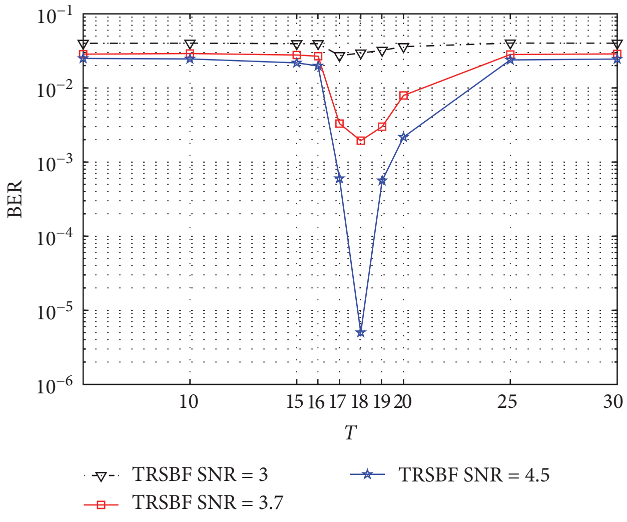

Figure 3: Optimization of T for LDPC2.

Simulation results illustrate the behavior of the parameters T for the proposed algorithm (see Figures 2–4) for three codes.

By observing these figures, the best T for each code is chosen. See Table 3 for the optimal values of the threshold parameter for the three codes. These values are used for the rest of this study.

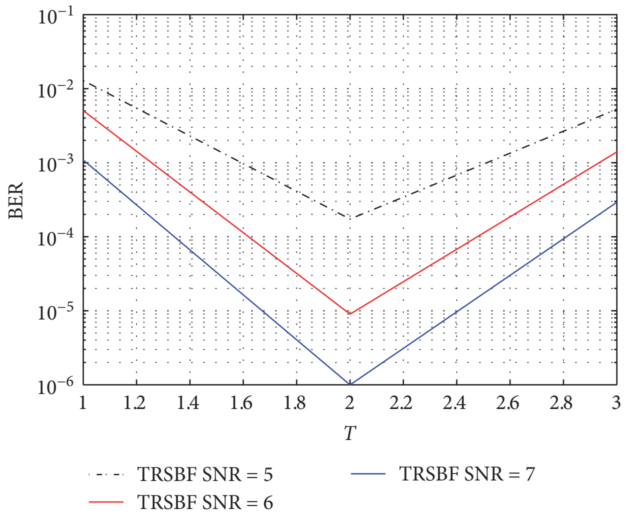

Figure 4: Optimization of T for LDPC3.

Table 3: Optimized threshold T.

Code	d_c	T
LDPC1	9	$5=\lceil\frac{d_c}{2}\rceil$
LDPC2	33	$18\approx\lceil\frac{d_c}{2}\rceil$
LDPC3	3	$2=\lceil\frac{d_c}{2}\rceil$
LDPC4	17	$9=\lceil\frac{d_c}{2}\rceil$

From this observation, the best value of T is determined by:

$$T \approx \left\lceil \frac{d_c}{2} \right\rceil. \tag{15}$$

Equation (15) comes from the following facts:

First, note that d_c represents the maximum number of participations of a bit in the parity-check equations and v_k represents the participation of a given bit in the parity-check equations that are not satisfied.

Then the rule to put an index k of a bit z_k every time $v_k \geq T$ (when $T \geq \lceil d_c/2 \rceil$) in the Set B, is simply applying the majority voting rule.

4.2. Performance Results for AWGN Channel. Figure 5 benchmarks our decoder against some known BF decoders for the LDPC1 code. As shown in Figure 5 our decoder has a better BER performance than SBF and a slight advantage over GDBF. It presents coding gains of 0.95 and 0.2 dB at BER of 3.10^{-5} compared to the SBF and GDBF, respectively.

Figure 6 shows results for the code LDPC2 where our decoder achieves coding gains of 0.5 and 0.7 dB compared to the GDBF and SBF algorithms, respectively at BER of 2.10^{-5}.

Furthermore, our decoder presents a 0.1-dB coding gain compared to the NGDBF for these codes, unlike for the LDPC1 code, where we lose 0.6 dB of performance at BER 10^{-5}. This observation may imply that longer codes will gain a larger improvement over NGDBF.

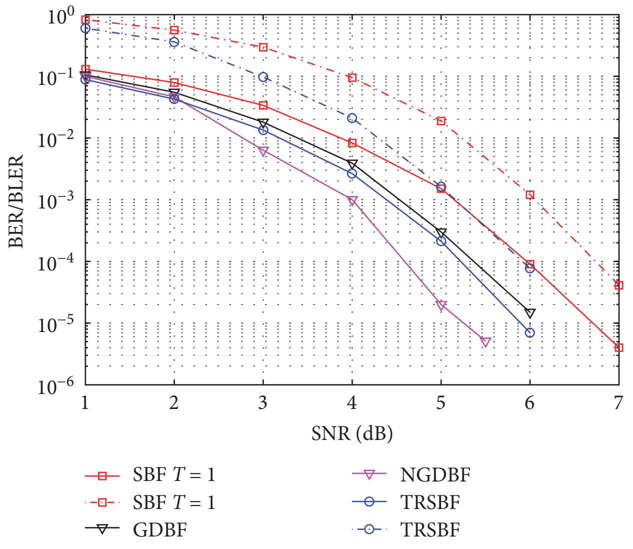

Figure 5: BER (solid) and BLER (dashed) performance comparison of several decoders for LDPC1 code.

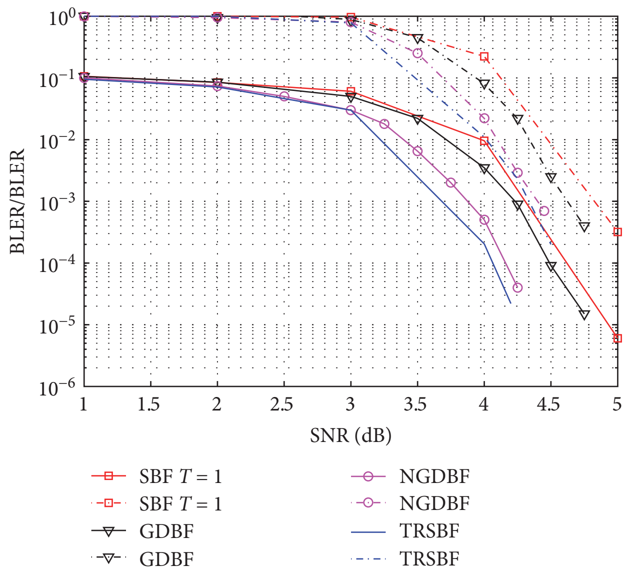

Figure 6: BER (solid) and BLER (dashed) performance comparison of several decoders for LDPC2 code.

Figure 7 compares our decoder to some known BF decoders for the LDPC3 code. The figure shows that our decoder outperforms other algorithms in terms of BER performance. As TRSBF achieves, respectively, coding gains of 2.5, 1.9, 1, and 1 dB compared to the BF, SBF, WBF, and CBBF at BER 2.10^{-5}.

Figure 8 benchmarks our decoder against some known BF decoders for the LDPC4 code. As shown in Figure 8 our decoder has a better BER performance than SBF and GDBF. It presents coding gains of 1.4 and 0.65 dB at BER 10^{-5} compared to the SBF and GDBF respectively, and 0.2 dB of performance loss against NGDBF.

The results highlight that it is important to save the reliability values of received signals during the decoding

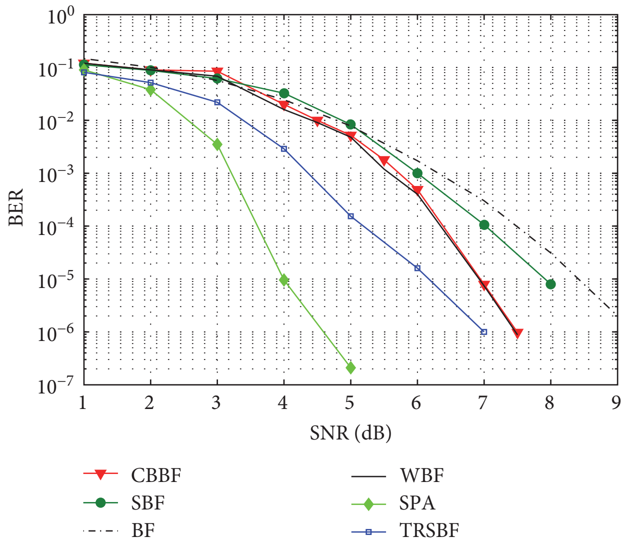

Figure 7: BER performance comparison of several decoders for LDPC3 code.

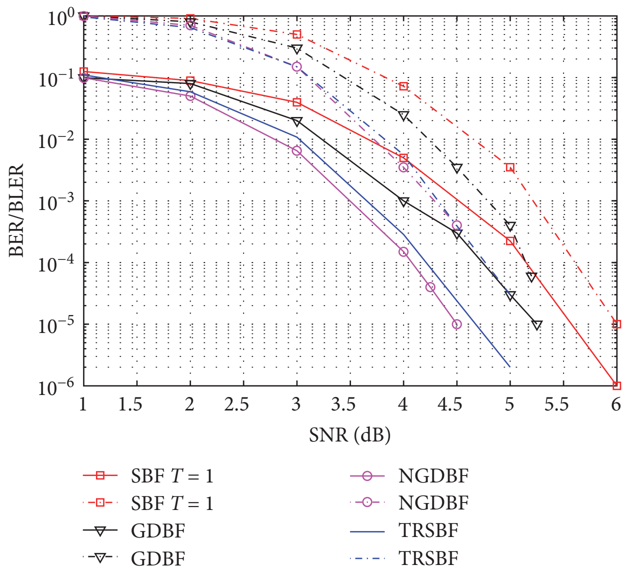

Figure 8: BER (solid) and BLER (dashed) performance comparison of several decoders for LDPC4 code.

process since they are the initial belief of the channel on received signal reliability. In addition, the obtained results show a correlation between the code length and the performance of the proposed decoder.

4.3. Performance Results for Rayleigh Fading Channel. In order to evaluate our new decoder, we have simulated its performance in the Rayleigh fading channel and compared it with the performances of GBF and SBF using the fourth codes in Table 1.

The curves plotted in Figure 9 show that the performance of our TRSBF decoder is better than the GBF and SBF ones.

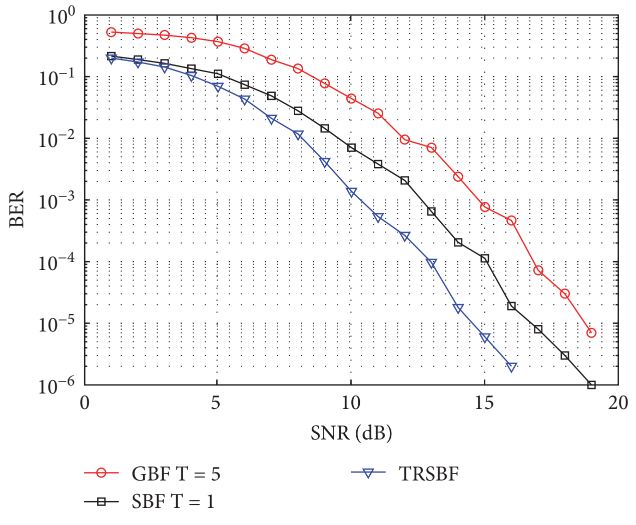

Figure 9: BER performance comparison of several decoders for LDPC1 code over the rayleigh channel.

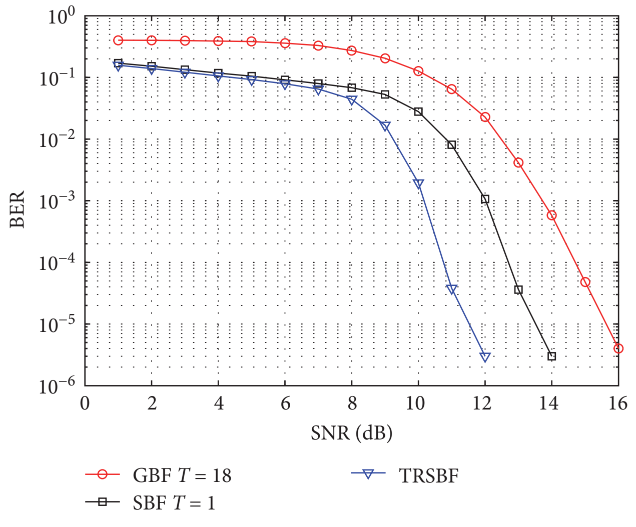

Figure 10: BER performance comparison of several decoders for LDPC2 code over the rayleigh channel.

It presents coding gains of 4.3 and 2.4 dB at BER 10^{-5} compared to GBF and SBF, respectively, at BER 10^{-5}.

Figure 10 benchmarks our decoder for the LDPC2 code. As shown in Figure 10, our decoder has a better BER performance than GBF and SBF. It presents coding gains of 4 and 2 dB at BER 10^{-5} compared to SBF and GDBF, respectively.

Figure 11 shows that our decoder outperforms other algorithms in terms of BER performance for the LDPC3 code. As TRSBF achieves, respectively, a huge coding gain of 1 and 8.5 dB compared to GBF and SBF at BER 4.10^{-5}.

Figure 12 shows results for the LDPC4 code, where our decoder achieves coding gains of 3 and 6 dB compared to the GDBF and the SBF algorithms, respectively, at BER 10^{-5}.

So, we have in the case of Rayleigh fading channel a better performance gain behavior than the case of AWGN.

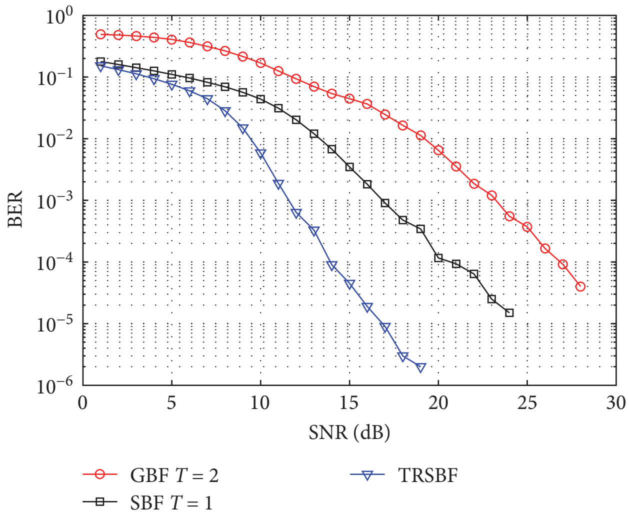

FIGURE 11: BER performance comparison of several decoders for LDPC3 code over the rayleigh channel.

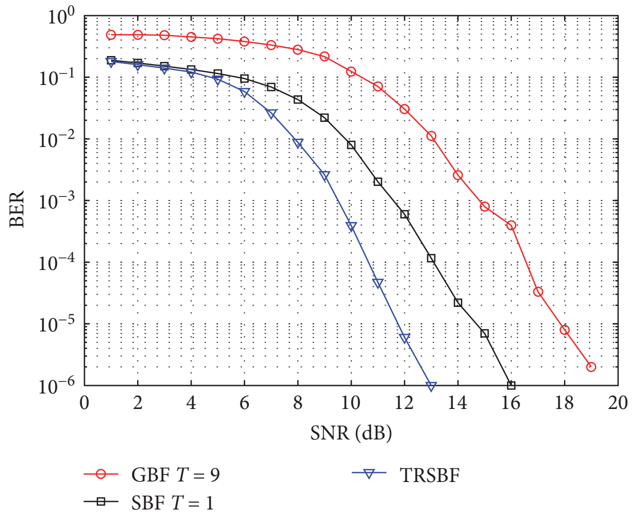

FIGURE 12: BER performance comparison of several decoders for LDPC4 code over the rayleigh channel.

TABLE 4: The complexity of the BF algorithms.

Algorithms	GBF, SBF	WBF	GDBF, NGDBF	TRSBF
Complexity	O (I_{tr}. m.n)	O (I_{tr}. m.n)	O (I_{tr}. m.n^2)	O (I_{tr}. m.n^2)

5. Complexity Study

5.1. Analytic Complexity. Let C denoted by (n, k) (d_r, d_c) be a regular binary LDPC code over GF (2). C is the null space of an $m \times n$ parity-check matrix $H = (H_{m,n})$ which has d_c 1's in each column and d_r 1's in each row with $m = (n–k)$, and I_{tr} the average number of iterations.

Table 4 illustrates the complexity analysis of the BF algorithms (FF). The table shows that the complexity is polynomial in n, m, and I_{tr} for all algorithms.

The soft algorithms (except the WBF) are more complex compared to the hard BF variants. On the other hand, the complexity of the proposed algorithm TRSBF and the GDBF and NGDBF are identic; since m and n are fixed parameters, the study of the average number of iterations (I_{tr}) will determine which algorithm is more complex.

5.2. Average Number of Iterations. We analyze the average number of iterations with respect to the SNR in order to perform a numerical convergence analysis of the proposed decoding scheme and compare it to the known BF variants [20]. We consider the number of simulated transmitted blocks in which at least 200 erroneous decoded words are observed for each SNR to be N and the total number of iterations used for decoding all the N blocks to be P_{all} with $P_{max} = 50$ for each block processed in this study.

The average number of iterations P_{avg} is obtained by the following ratio formula:

$$P_{avg} = \frac{P_{all}}{N}. \quad (16)$$

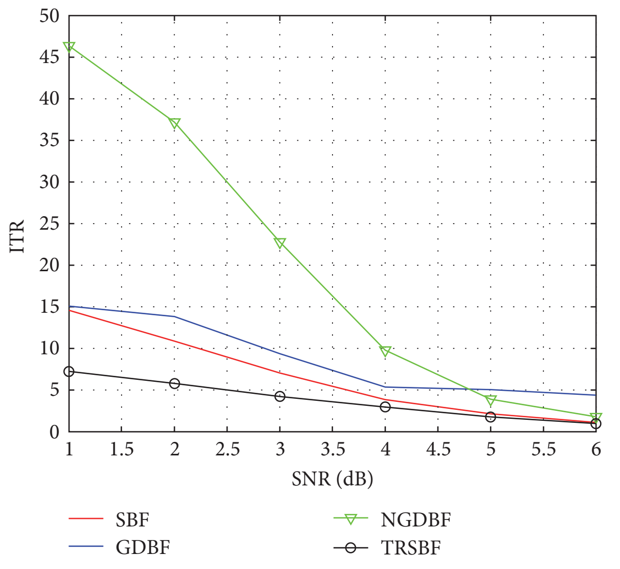

FIGURE 13: Comparison of the average number of iterations of various BF algorithms for LDPC1 code.

The curves corresponding to the average number of iterations for the LDPC1, LDPC2, and LDPC4 codes listed in Table 1 with respect to the different SNRs are shown in Figures 13–15, respectively. In Figure 13 the TRSBF decoder presents a lower complexity in terms of the average number of iterations than the SBF, GDBF, and NGDBF decoders. This decoder also has an advantage in terms of BER performance when compared with the SBF and GDBF decoders.

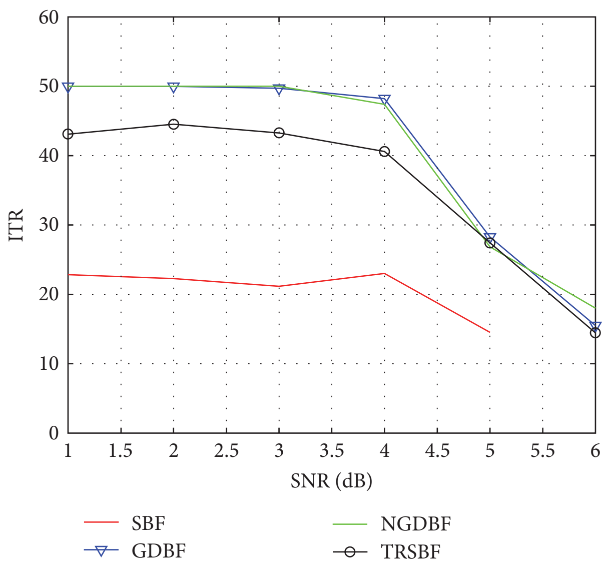

Figure 14: Comparison of the average number of iterations of various BF algorithms for LDPC2 code.

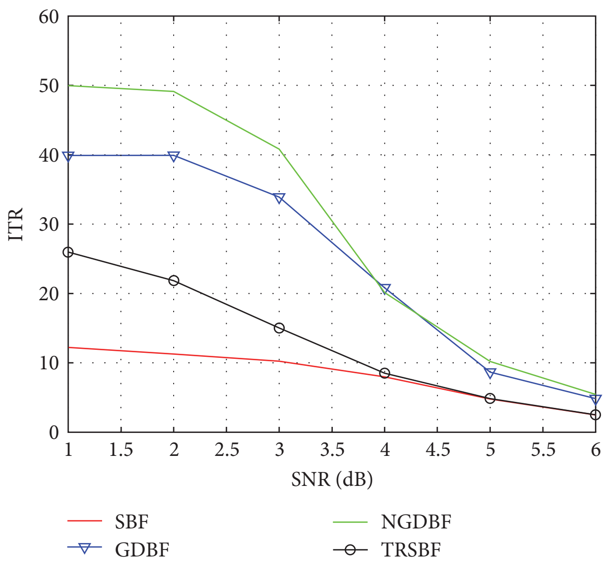

Figure 15: Comparison of the average number of iterations of various BF algorithms for LDPC4 code.

In addition to the gain in BER performance of our TRSBF decoder over the SBF, GDBF, and NGDBF decoders, we can see clearly in Figure 14 that our TRSBF decoder presents a modest advantage in terms of the average number of iterations over the compared decoders except for the SBF.

In Figure 15, we can see that in the entire range of SNRs, the TRSBF decoder needs fewer iterations to achieve convergence than the GDBF and NGDBF decoders.

We can clearly see that the proposed decoding algorithm requires fewer decoding iterations than other soft variants of BF decoders; consequently, the proposed algorithm is less complex than the GDBF and NGDBF when the analytic complexity seen in Table 4 is taken into account; thus, the TRSBF achieves fast convergence.

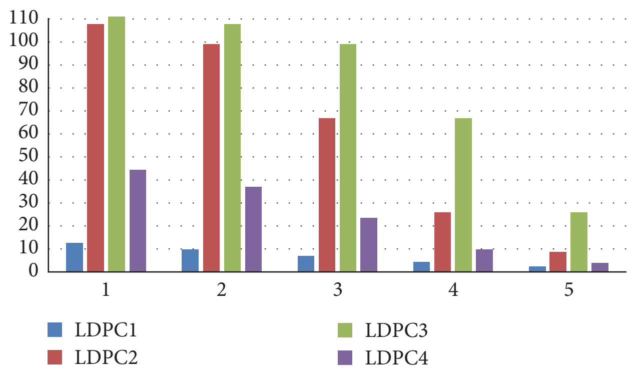

Figure 16: Average size of the Set B.

5.3. Cardinality of the Set B. We explore the average size of the Set B with respect to the SNR for different codes (Table 1) in order to investigate the complexity of the second-stage processing of our decoder.

Figure 16 shows the average size of the Set B for each code across the SNRs. We can see clearly that the average size of Set B decreases as the SNR increases. Also, we can observe for the two codes (with the same code length and rate), LDPC2 and LDPC3 that the average size of the Set B decreased in a fast way for the LDPC2 code compared to the LDPC3 code, and this observation explains the good BER performance given by the LDPC2 code compared to the LDPC3 code and confirms the power of the DSC codes class over the Gallager codes class.

5.4. Computational Complexity. To evaluate the computational complexity of the proposed decoding algorithm for one iteration, we will denote $\delta = m.d_r = n.d_c$ the number of 1-entries in the parity-check matrix H and β the size of the Set B (see Equation (10)).

Table 5 shows a comparison of the computational complexities of several decoding algorithms for LDPC codes.

To compute the syndrome s of a received vector in Step 1 of the TRSBF algorithm? we need $m.(d_r\text{-}1) = \delta\text{-}m$ binary operations (BO). Then to identify the size of the Set B in Step 2 we need n integer comparison (IC). In Step 3 of the algorithm, the computation of Δ_k requires β real addition (RA) and β real multiplication (RM). Thereafter, in Step 4, finding the k_0 requires β real comparison (RC) and flipping the bit z_{k_0} requires one binary operation (BO).

In Table 4, we can see that our decoder has two parts of complexity. The first part, like the hard decoding algorithms (BO, IA, and IC operations). The second part, like soft decoding algorithms (RA, RC, and RM operations), Therefore, to evaluate the complexity of our decoder, we need to study the range of values for the β parameter (the average size of the Set B).

In Figure 16, we have plotted the average size of the Set B obtained for 1,000 erroneous received sequences versus SNRs. Figure 16 also shows the average number of iterations versus SNRs for the LDPC codes.

By considering the real complexity parts (RA, RC, and RM) in Table 5 and the average size of the Set B shown in Figure 8, we can conclude that the complexity of our decoder

Table 5: Computational complexities per iteration for several LDPC decoding algorithms.

Decoding algorithms	Computation cost per iteration						
	BO	IA	IC	RA	RC	RM	Log
GBF	δ-m+b	δ-n	n				
SBF	δ-m + 1	δ-n	n				
WBF	δ-m + 1			δ	n	δ	
SPA	δ-m					6δ	N
TRSBF	δ-m + 1	δ-n	n	β	β	β	

BO: binary operation; IA: integer addition; IC: integer comparison; RA: real addition; RC: real comparison; RM: real multiplication; log: logarithm.

is lower than that of WBF and its variants since $\delta = n.$ $d_c >> n > > \beta$, while our decoder provides a better performance in terms of BER or BLER.

6. Conclusion

This paper proposes a new BF algorithm based on the reliability of the received signal: TRSBF. The proposed algorithm yielded better decoding performance than some known BF algorithms for the studied LDPC codes.

The proposed algorithm uses a two-round selection approach to get the bit to be flipped, by separating the hard decision information from the soft one. In the first round, only hard information is used, and its solutions are refined by the second round, which is based on the soft information. An advantage of our decoder is that it can be applied to both regular and irregular LDPC codes.

The proposed algorithm achieves effective tradeoffs between performance and decoding complexity.

The study of the complexity has proved that our algorithm has a low complexity and fast convergence rate compared to the other soft or hard decoders, either in terms of iterations or computational complexity.

We believe this research's findings invite more investigations on the performance of the proposed algorithm for irregular the codes or codes with large blocklengths.

Data Availability

The data supporting the results of the study can be found with the authors upon request for free.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. Gallager, "Low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[2] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[3] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, MA, 1963.

[4] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645-1646, 1996.

[5] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[6] W. Ullah and A. Yahya, "Comprehensive algorithmic review and analysis of LDPC codes," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 1, pp. 111–130, 2015.

[7] U. Waheed, F. Yang, and Y. Abid, "QC LDPC codes for MIMO and cooperative networks using two way normalized min-sum decoding," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 7, pp. 5448–5457, 2014.

[8] W. Ullah, T. Jiang, F. Yang, and S. M. Aziz, "Two-way normalization of min-sum decoding algorithm for medium and short length low density parity check codes," in *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–5, IEEE, 2011.

[9] G. Dong, Y. Li, N. Xie, T. Zhang, and H. Liu, "Candidate bit based bit-flipping decoding algorithm for LDPC codes," in *2009 IEEE International Symposium on Information Theory*, vol. 2166, pp. 2166–2168, IEEE, Seoul, 2009.

[10] Q. Zhu and L. Wu, "Weighted candidate bit based bit-flipping decoding algorithms for LDPC codes," in *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, pp. 731–734, IEEE, Xianning, China, 2013.

[11] S. Addi, A. Berkani, A. Azouaoui, and M. Belkasmi, "New hard decision decoder of LDPC codes using single Bit Flipping algorithm," in *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–5, IEEE, Rabat, Morocco, 2017.

[12] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711–2736, 2001.

[13] J. Zhang and M. P. C. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *IEEE Communications Letters*, vol. 8, no. 3, pp. 165–167, 2004.

[14] F. Guo and L. Hanzo, "Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes," *Electronics Letters*, vol. 40, no. 21, pp. 1356–1358, 2004.

[15] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding LDPC codes," in *2008 International Symposium on Information Theory and Its Applications*, pp. 1–6, IEEE, Auckland, New Zealand, 2008.

[16] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for LDPC codes," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3385–3400, 2014.

[17] T. C.-Y. Chang and Y. T. Su, "Dynamic weighted bit-flipping decoding algorithms for LDPC codes," *IEEE Transactions on Communications*, vol. 63, no. 11, pp. 3950–3963, 2015.

[18] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*, Cambridge University Press, 2009.

[19] D. J. C. MacKay, "Encyclopedia of sparse graph codes," 2005.

[20] A. Yatribi, M. Belkasmi, and F. Ayoub, *Gradient-Descent Decoding of One-step Majority-Logic Decodable Codes*, Physical Communication, 2020.