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

Soufian Addi\textsuperscript{a1}, Mostafa Belkasmi\textsuperscript{a2}, Ahlam Berkani\textsuperscript{a}, Ahmed Azouaoui\textsuperscript{b}

\textsuperscript{a}ICES Team ENSIAS, Mohammed V University in Rabat, Morocco.
\textsuperscript{b}FS - El Jadida, Chouaib Doukkali University, Morocco.

ABSTRACT

This paper presents a novel iterative reliability-based Bit Flipping (BF) algorithm for decoding Low-Density Parity-Check (LDPC) codes. The new decoder is a single bit flipping algorithm called Two-Round Selection based Bit Flipping (TRSBF). 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 selection, the initial belief about received signals, contributes efficiently to select the best candidate bit. We demonstrate through simulations over the binary-input additive white Gaussian noise (AWGN) channel and the Rayleigh fading channel that the proposed algorithm exhibits better decoding performance when compared with some well-known soft decision BF algorithms. The performance for the Rayleigh fading channel will be given also for reference. Complexity analysis of the proposal and a comparison to other BF decoders are then presented.

Keywords: Bit-flipping decoding, Low-density parity-check codes (LDPC), Soft decision decoding, Iterative decoding, Hard decision decoding, Reliability, Rayleigh fading channel.

1 INTRODUCTION

Error correcting codes (ECC) are used for controlling 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 occurred over noisy communication channels. The LDPC codes \cite{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 in his pioneer PhD thesis in 1962 [1][3]. In 1996, the LDPC codes were rediscovered by MacKay [4] who brought them back into prominence. When the Sum-Product Algorithm (SPA) [5] is used for decoding, they have been 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 such as Bit Flipping (BF) algorithm, and SPA algorithm. 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)[6], the Weighted Candidate Bit based Bit-Flipping (WCBBF)[7] and Single Bit Flipping (SBF) [8]. 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 [8] 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’s why BF techniques moved toward the category of simplified soft decision decoding algorithms. The latter do not use hard information only but soft information also during the decoding process. The work on this class of decoding algorithm starts with the Weighted Bit Flipping (WBF) decoder [9]. Algorithms in this class allow improvement in performance with no 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 enhancements from WBF in performance and convergence rate. We quote in this class Modified Weighted Bit Flipping (MWBF)[10], Reliability-Ratio based Weighted Bit-Flipping algorithm (RRWBF)[11], Gradient Descent Bit Flipping (GDBF) [12-13], and Dynamic Weighted Bit-Flipping decoding algorithms (DWBF)[14].

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 trade-offs 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 bits pool. The candidate bits selected in this round are those contained in more than some
Two-Round Selection based Bit Flipping Decoding Algorithm for LDPC Codes

fixed number of unsatisfied parity-check equations. In the second round, the best candidate bit, which is the nearest 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 rest of this paper is organized as follows. In Section 2, we will present an overview of 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, 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 Low Density Parity Check (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 regular LDPC code if the matrix H has constant column weight $d_c$ and constant row weight $d_r$ and is said irregular otherwise.

We assume that code words $c=(c_0, c_1, ..., c_{n-1})$ 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 variance $\sigma^2$. The sequence $r=(r_0, r_1, ..., r_{n-1})$ stands for the sequence of soft channel values obtained at the receiver’s output. The hard-decisions information $z=(z_0, z_1, ..., z_{n-1})$ associated to the sequence $r$ is got as follows:

$$z_i = \begin{cases} 1, & \text{if } r_i \geq 0 \\ 0, & \text{else} \end{cases}$$  \hspace{1cm} (1)

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

$$\hat{z} = (2z_0 - 1, 2z_1 - 1, ..., 2z_{n-1} - 1)$$  \hspace{1cm} (2)

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

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

2.2 Bit Flipping algorithm

A typical Bit-Flipping algorithm (GBF) [1][3] is a simple hard-decision algorithm that flips the bits involved in a large number of unsatisfied check equations that exceeds a threshold ($T$) for LDPC codes, because it’s most probably incorrect bits. The algorithm is terminated once all the parity-check equations are satisfied, which means a valid codeword has been found or the maximum number of iterations $p_{\text{max}}$ is reached. The
algorithm requires only integer operations for decoding and can therefore be easily implemented by an electronic circuit.

The main part in the standard BF algorithm is the calculation of Flipping metric for each bit and each iteration called Flipping Function (FF). The FF values allow the 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 by two ways as the following formulas:

\[ e^{(1)}_k = \sum_{j \in M(k)} (2s_j - 1) \quad (3) \]

\[ v_k = \sum_{j=0}^{m-1} < s_j, h_{j,k} > \quad (4) \]

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 proof that:

\[ v_k = \sum_{j \in M(k)} s_j \quad (5) \]

The sequence \( v = (v_0, v_1... v_{n-1}) \) is the so-called reliability profile of the received sequence \( r \) (or more precisely of the hard version \( z \)) [15].

The steps of a Standard BF (GBF) algorithm are as follows:

**Algorithm 1:** Gallager BF algorithm

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

### 2.3 Single Bit Flipping algorithm

The Single Bit Flipping (SBF) [8] 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 formula (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:

- **Step 3’**: Find the index $k_0 = \text{argmax} (v_k)$ where $k$ belongs to $\{0,\ldots, n-1\}$
- **Step 4’**: Flip the bit $z_{k_0}$, $p \leftarrow p + 1$.

The SBF results better performance than the standard BF and converges toward the final solution with lower number of iterations [8].

### 2.4 Weighted Bit-Flipping algorithm

The WBF algorithm [9] is a variant of the BF algorithm and it’s 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 on the channel behavior.

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

$$e_k^{(2)} = \sum_{j \in M(k)} (2s_j - 1) \cdot r_{\min}^j$$

Where $r_{\min}^j = \text{min} |r_i|$ 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 checksums values and the reliability of received messages to make decisions, therefore, the algorithm yields better decoding performance when compared with the BF algorithm.

### 2.5 Modified Weighted Bit-Flipping algorithm

The Modified Weighted Bit-Flipping algorithm (MWBF)[10] improves on the WBF decoding algorithm by using the reliability of the related bit itself in the FF values of the WBF(6).

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

$$e_k^{(3)} = \sum_{j \in M(k)} (2s_j - 1) \cdot r_{\min}^j - \alpha |r_k|$$

Where $\alpha$ is the weighting factor which is a real positive number.

### 2.6 Reliability-Ratio Based Weighted Bit Flipping algorithm

In an iterative process, the Reliability-Ratio based Weighted Bit Flipping algorithm (RRWBF) [11] tries to find the most reliable message nodes associated with message bits, instead of finding the unreliable message nodes for the WBF.

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

$$e_k^{(4)} = \sum_{j \in M(k)} (2s_j - 1) \cdot (R_{j,k})^{-1}$$

Where $R_{j,k} = \frac{|r_k|}{|r_{\max}^j|}$, $r_{\max}^j = \text{max} |r_i|$ is the maximum absolute of soft values $r_i$ for
the bits participating in the $j^{th}$ parity check equation (maximum for all indices $i$ in the set $N(j)$) and $\varphi$ is a normalization factor to keep $\sum_{k \in N(j)}(R_{j,k})^{-1} = 1$.

2.7 Gradient Descent Bit-Flipping algorithm

The GDBF algorithm [12] derives its FF (9) 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:

$$e_k^{(5)} = \sum_{j \in M(k)} (2s_j - 1) - r_k(\hat{z}_k)$$

(9)

3 PROPOSED BIT FLIPPING ALGORITHM

3.1 Motivation

All soft bit flipping 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 Two-Round Selection based Bit Flipping (TRSBF) decoder. The latter is made up of two stages. The first/second round selection is made by the first/second stage respectively. The proposed algorithm works as follows (see Fig.1):

![Fig. 1. Block diagram of the 3 proposed algorithm](image)

The first stage, consisting of FF processing, is based only on hard information. This stage can be considered as a filter for the next processing. On the other hand the second
stage 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, in 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 (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:

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

(10)

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

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

A first step in the second stage, consists in 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:

$$z_i^{(k)} = \begin{cases} z_i + 1, & \text{if } i = k \\ z_i, & \text{else} \end{cases}$$

(11)

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

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

$$d^2_e (r, \hat{z}^{(k)}) = \sum_{i=0}^{n-1} (r_i - \hat{z}_i^{(k)})^2 \quad \text{for each index } k \in B$$

(12)

But minimizing squared Euclidean distance in equation (12) is the same as minimizing the set of $\Delta_k$ values, with $\Delta_k$ defined as follows:

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

(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 (14):

$$k_0 = \text{argmin}(\Delta_k), \quad k \in B$$

(14)

The position $k_0$ is the selected bit position to be flipped in the current iteration.
The steps of the TRSBF algorithm are as follows:

**Algorithm 2**: TRSBF algorithm

- **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 $p > p_{\text{max}}$, then stop the algorithm. Else go to Step 1.

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 the 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.

## 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 [9] and the LDPC3 code is a pseudorandom or Gallager code and was selected from MacKay’s online encyclopedia [16].

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

### Table 1. Parameters of LDPC Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Length $n$</th>
<th>Dimension $k'$</th>
<th>$m$</th>
<th>Rate</th>
<th>dc</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDPC1</td>
<td>73</td>
<td>45</td>
<td>73</td>
<td>0.616</td>
<td>9</td>
</tr>
<tr>
<td>LDPC2</td>
<td>1057</td>
<td>813</td>
<td>1057</td>
<td>0.77</td>
<td>33</td>
</tr>
<tr>
<td>LDPC3</td>
<td>1057</td>
<td>813</td>
<td>244</td>
<td>0.77</td>
<td>3</td>
</tr>
<tr>
<td>LDPC4</td>
<td>273</td>
<td>191</td>
<td>273</td>
<td>0.69</td>
<td>17</td>
</tr>
</tbody>
</table>
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 default simulation’s parameters outlined in Table 2.

**Table 2. Simulation Parameters**

<table>
<thead>
<tr>
<th>Simulation parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min number of transmitted blocks</td>
<td>1000</td>
</tr>
<tr>
<td>Min number of residual bit errors</td>
<td>200</td>
</tr>
<tr>
<td>Max number of iterations ($p_{\text{max}}$)</td>
<td>45</td>
</tr>
</tbody>
</table>

### 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.

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

Simulation results illustrate the behavior of the parameters $T$ for the proposed algorithm (see Fig.2, Fig.3 and Fig.4) for three codes.

![Fig. 2. Optimization of T for LDPC1](image)

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

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

(15)

The equality (15) comes from the following facts:

Firstly 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 $T \geq \left\lceil \frac{d_c}{2} \right\rceil$) in the set $B$, is simply applying the majority voting rule.

**Table 3. Optimized Threshold $T$**

<table>
<thead>
<tr>
<th>Code</th>
<th>$T$</th>
<th>$d_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDPC1</td>
<td>$5=\left\lceil \frac{d_c}{2} \right\rceil$</td>
<td>9</td>
</tr>
<tr>
<td>LDPC2</td>
<td>$18=\left\lceil \frac{d_c}{2} \right\rceil$</td>
<td>33</td>
</tr>
<tr>
<td>LDPC3</td>
<td>$2=\left\lceil \frac{d_c}{2} \right\rceil$</td>
<td>3</td>
</tr>
<tr>
<td>LDPC4</td>
<td>$9=\left\lceil \frac{d_c}{2} \right\rceil$</td>
<td>17</td>
</tr>
</tbody>
</table>

**4.2 Performance results for AWGN channel**

The figure 5 benchmarks our decoder against some known Bit Flipping decoders for the LDPC1 code. As shown by figure 5 our decoder has a better BER performance than SBF and a slight advantage over GDBF. It presents coding gains of 0.95 dB and 0.2 dB at BER of $3.10^{-5}$ compared to SBF and GDBF Respectively.

![Fig. 5. BER (solid) and BLER (dashed) performance comparison of several decoders for LDPC1 code.](image)

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

The figure 7 compares our decoder to some known Bit flipping decoders for the LDPC3 code. The figure shows that our decoder outperforms other algorithms in terms of BER performance. As TRSBF achieves, respectively, coding gain of 2.5 dB, 1.9 dB, 1 dB and 1 dB compared to BF, SBF, WBF and CBBF at BER $2 \cdot 10^{-5}$.

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

Fig. 7. BER performance comparison of several decoders for LDPC3 code.
The figure 8 benchmarks our decoder against some known Bit Flipping decoder for the LDPC4 code. As shown by figure 8 our decoder has a better BER performance than SBF and GDBF. It presents coding gains of 1.4 dB and 0.65 dB at BER $10^{-5}$ compared to 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 process since they are the initial belief of the channel on received signals reliability. In addition, the obtained results show a correlation between the code length and the performance for 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 we compared them with the performances of GBF and SBF using respectively 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 one. It presents coding gains of 4.3 dB and 2.4 dB at BER $10^{-5}$ compared to GBF and SBF Respectively at BER $10^{-5}$.
Fig. 9. BER performance comparison of several decoders for LDPC1 code over the Rayleigh channel.

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

Fig. 10. BER performance comparison of several decoders for LDPC2 code over the Rayleigh channel.

The 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 13 dB and 8.5 dB compared to GBF and SBF at BER $4.10^{-5}$.

Figure 11 shows results for the LDPC3 code where our decoder achieves coding gains of 3dB and 6dB compared to the GDBF and the SBF algorithms respectively at BER $10^{-5}$.

Figure 12 shows results for the LDPC4 code where our decoder achieves coding gains of 3dB and 6dB compared to the GDBF and the SBF algorithms respectively at BER $10^{-5}$.
Therefore, we have in the case of Rayleigh fading channel a better performance gain behavior than in the case of AWGN.

5 COMPLEXITY STUDY

5.1 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 [17]. We consider the number of simulated transmitted blocks that at least 200 erroneous decoded words are observed for each SNR is \( N \) and the total number of iterations used for decoding all the \( N \) blocks is \( P_{\text{all}} \) with \( P_{\text{max}}=50 \) for each block processed in this study.

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

\[
P_{\text{avg}} = \frac{P_{\text{all}}}{N}
\]  

(16)

Fig. 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 different SNRs are shown in figures 13, 14 and 15 respectively.

In the figure 13 the TRSBF decoder presents a lower complexity in term of the average number of iterations than the SBF, GDBF and NGDBF decoders. This decoder has also an advantage gained in the term of BER performance when compared with SBF and GDBF decoders.
Two-Round Selection based Bit Flipping Decoding Algorithm for LDPC Codes

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

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

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

We can see clearly that the proposed decoding algorithm requires a lower number of decoding iterations compared to other variants of BF decoders; consequently, the proposed algorithm achieves a fast convergence.

Fig. 15. Comparison of the average number of iterations of various BF algorithms for LDPC4 code.
5.2 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. Average size of the set B](image)

The 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 the 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 fast way for the LDPC2 code compared to the LDPC3 code and this observation explain the good BER performance given by the LDPC2 code compared to LDPC3 code and confirm the powerful of the DSC codes class over the Gallager codes class.

5.3 Computational Complexity

To evaluate the computational complexity of the proposed decoding algorithm for one iteration, we will denote \( m.d_r = n.d_c \) the number of 1-entries in the parity-check matrix H and \( \tau \) the size of the set B (see (10)).

<table>
<thead>
<tr>
<th>Decoding algorithms</th>
<th>BO</th>
<th>IA</th>
<th>IC</th>
<th>RA</th>
<th>RC</th>
<th>RM</th>
<th>log</th>
</tr>
</thead>
<tbody>
<tr>
<td>GBF</td>
<td>m-b</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBF</td>
<td>m-1</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WBF</td>
<td>m-1</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPA</td>
<td>m</td>
<td></td>
<td></td>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRSBF</td>
<td>m-1</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BO: Binary Operation; IA: Integer Addition; IC: Integer Comparison; RA: Real Addition; RC: Real Comparison; RM: Real Multiplication; Log: Logarithm.
The table 4 shows a comparison of computational complexities of several decoding algorithms for LDPC codes.

To compute the syndrome \( s \) of a received vector at step 1 of the TRSBF algorithm we need \( m.(d_r - 1) = \\frac{1}{m} \) binary operations (BO). Then to identify the size of the set \( B \) at step 2 we need \( n \) integer comparison (IC). At the step 3 of the algorithm, the computation of \( \Delta_k \) require \( \frac{n}{m} \) real addition (RA) and \( \frac{n}{m} \) real multiplication (RM). Thereafter, at the step 4, finding the \( k_0 \) require \( \frac{n}{m} \) real comparison (RC) and flipping the bit \( z_{k_0} \) need one binary operation (BO).

In table 4 we can see that our decoder has two complexity parts. The first part, like the hard decoding algorithms (BO, I\( A \) 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 \( \frac{n}{m} \) parameter (the average size of the set \( B \)).

In Figure 8 we have plotted the average size of the set \( B \) obtained for 1000 erroneous received sequences versus SNRs. Figure 8 also shows the average number of iterations versus SNRs for the LDPC2 code.

By considering the real complexity part (RA, RC and RM) in the table 4 and the average size of the set \( B \) shown in the figure 8, we can conclude that the complexity of our decoder is lower than that of WBF and its variants since \( \frac{n}{m} = \frac{n.d_r}{m} \gg n \gg \frac{1}{n} \), while our decoder provides a better performance in terms of BER or BLER.

6 CONCLUSION

This paper proposed a new BF algorithm based on the reliability of received signal: the Two-Round Selection based Bit Flipping (TRSBF). The proposed algorithm yielded better decoding performance than some known BF algorithms for 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 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 trade-offs between performance and decoding complexity.

The study of the complexity has proved that our algorithm has a low complex and fast convergence rate compared to 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 codes or codes with large block-lengths.
REFERENCES


Communications 63.11, pp. 3950-3963.


