sensors

Article

Sum of the Magnitude for Hard Decision Decoding Algorithm Based on Loop Update Detection

Jiahui Meng *, Danfeng Zhao, Hai Tian and Liang Zhang

College of Information & Communication Engineering, Harbin Engineering University, Harbin 150001, China; zdfeng@hrbeu.edu.cn (D.Z.); tianhai123@hrbeu.edu.cn (H.T.); zhangliang@hrbeu.edu.cn (L.Z.)
* Correspondence: mengjiahui@hrbeu.edu.cn; Tel.: +86-188-4601-1013

Received: 8 November 2017; Accepted: 14 January 2018; Published: 15 January 2018

Abstract: In order to improve the performance of non-binary low-density parity check codes (LDPC) hard decision decoding algorithm and to reduce the complexity of decoding, a sum of the magnitude for hard decision decoding algorithm based on loop update detection is proposed. This will also ensure the reliability, stability and high transmission rate of 5G mobile communication. The algorithm is based on the hard decision decoding algorithm (HDA) and uses the soft information from the channel to calculate the reliability, while the sum of the variable nodes' (VN) magnitude is excluded for computing the reliability of the parity checks. At the same time, the reliability information of the variable node is considered and the loop update detection algorithm is introduced. The bit corresponding to the error code word is flipped multiple times, before this is searched in the order of most likely error probability to finally find the correct code word. Simulation results show that the performance of one of the improved schemes is better than the weighted symbol flipping (WSF) algorithm under different hexadecimal numbers by about 2.2 dB and 2.35 dB at the bit error rate (BER) of 10^{-5} over an additive white Gaussian noise (AWGN) channel, respectively. Furthermore, the average number of decoding iterations is significantly reduced.

Keywords: 5G; non-binary LDPC; hard decision decoding; reliability; loop detection; sum of magnitude

1. Introduction

On 14 October 2016, the 3rd generation partnership project radio layer 1 (3GPP RAN1) conference determined the long-code block coding scheme for 5G communication using the low density parity check node (LDPC) code as the data information of the mobile bandwidth enhance mobile broadband (eMBB) service [1,2]. The 5G includes both people-centric and machine-centric communications [3,4]. These two types of scenarios have different needs. People-centric communications pursue high performance and high-speed communications. Corresponding end-user data rates should reach 10 Gbps and base station data rates should reach 1 Tbps. Machine-centric communication pursues low power consumption with the corresponding sensor rate of only 10–100 Bps, while industrial control applications require a particularly high delay of 10^{-4} s [5].

Gallager proposed the LDPC code [6]. Furthermore, he created the bit flip hard decision decoding algorithm based on the check and statistics as well as proposing the soft decision decoding algorithm based on a posterior probability (APP). The decoding algorithms of multiple LDPC codes are divided into two categories according to the different decision methods [7]: Soft decision algorithm (SDA) and hard decision algorithm (HDA). The representative algorithm of the SDA is the belief propagation (BP) decoding algorithm [8,9]. The error correction performance of the BP algorithm is excellent, but the computational complexity of the BP algorithm is high with a large amount of computation and a large amount of cached data that must be stored in memory during decoding iteration, which is not conducive to engineering implementation [10]. Therefore, the SDA is chosen mainly from the perspective of reducing the decoding complexity of the study [11], with the improved algorithm applied to 5G mobile communications in the human-centric high-speed communications. The representative algorithm in HDA is a bit-flipping (BF) algorithm [12], which has low complexity and computational complexity. The algorithm is simple and easy to implement in hardware, but the decoding performance is not as good as the SDA. Therefore, the HDA is mainly studied from the perspective of improving the decoding performance, with the improved algorithm applied to a machine-centric low-rate, low-latency system in 5G mobile communications.

In the future, the 5G mobile communication system in 2020 will provide higher data transmission rate, more service connections and better user experience while ensuring low cost, transmission security, reliability and stability [13]. Channel coding in 5G mobile communication involves non-binary LDPC codes [14]. Although non-binary LDPC codes have excellent error correction performance, the complexity of high decoding is the main reason that restricts its practical application. Therefore, this paper optimizes the performance of the HDA with low complexity, and optimizes it using the symbol flipping (SF) algorithm. For the hard decision decoding algorithm, the weighting symbol flipping (WSF) algorithm proposed in literature [15] constructs a new flip function by using the minimum value of the variable node amplitude adjacent to the check node (CN) as the reliability of the check equation. On this basis, the reliability information of VN is used as the input of the flip function by the weighting factor and thus, the performance of the modified weighted symbol flipping (MWSF) algorithm will be improved [16]. Based on the belief propagation theory, the decoding algorithm proposed by literature [17] achieves a certain degree of coding gain. In this paper, this is called the improved MWSF (IMWSF) algorithm. Since then, there have been proposals of an average probability and stopping criterion weighted symbol flipping (APSCWSF) algorithm [18,19], multiple-vote symbol flipping (MV-SF) algorithm [20–22] and so on [23,24].

Current communication systems need to have faster and more efficient decoders than previous ones. On this basis, many scholars have studied the fast convergence and low complexity decoding algorithm with minimum decoding loss [25–28]. For the WSF algorithm proposed in literature [19,25], the destruction of the parity relation is attributed to the lowest reliability symbol in the check equation. However, if one check equation is not established, all the symbols involved in the equation may be wrong, but the symbols with lower reliability are more likely to be wrong. Therefore, a new calculation method of reliability is redefined in this paper. When calculating the reliability of variable nodes from each check equation, the reliability information of each variable node participating in the check is taken into account. Furthermore, the sum of the amplitudes of the variable nodes adjacent to the check node is used as the reliability of the check equation. At the same time, when calculating the reliability of participating check symbols, excluding the information carried by the symbol itself, a weighted symbol flipping decoding algorithm based on the magnitude sum (SMWSF) is proposed to obtain more effective soft information for reliability. On this basis, the calculation method of flip function is improved, because the variable node's own reliability information plays an important role in the construction of the flip function. The weighted value of the symbol itself is added by a weighting factor and a modified weighted symbol flipping decoding algorithm based on the magnitude sum (MSMWSF) is proposed to improve the efficiency of symbol flipping. This is intended to improve the decoding performance and speed up the convergence. As there is a problem of repeatedly flipping the wrong symbols when the symbol is flipped, a new detection algorithm of infinite loops is proposed in this paper. Combined with the MSMWSF algorithm, this paper proposes a sum of the magnitude for weighted symbol flipping decoding algorithm based on loop update detection (LUDMSMWSF) to further speed up the convergence rate and reduce the decoding complexity.

2. Basic Definitions

In the non-binary (N, K) LDPC codes, N is the code word length and K is the information bits length. The non-binary LDPC codes are determined by the sparse non-binary parity check matrix $\boldsymbol{H}$, while the check matrix $\boldsymbol{H}$ has $M \geq N - K$ rows and N columns. Each row represents a check equation and each column represents a code word. For any hard decision vector $\mathbf{y}$, the check equation is $\mathbf{s}_h^{(k)} = \mathbf{y}^{(k)} H^T$, where the real numbers involved are carried out under the Galois field. h represents the elements in the checksum matrix. k represents the number of iterations. The number of non-zero elements in each row and each column in the check matrix is equal, which is called the rule code, while the others are called the irregular codes. The row weight of the check matrix $\boldsymbol{H}$ is d_c and the column weight is d_v. A non-"0" element on the nth column of the Galois field is represented by $M(n)$ in the check matrix $\boldsymbol{H}$. A non-"0" element on the mth row of the Galois field is represented by $N(m)$ in the check matrix $\boldsymbol{H}$. $M(n) := \{m : h_{mn} \neq 0\}$, $N(m) := \{n : h_{mn} \neq 0\}$.

We selected any code word $\mathbf{c} = [c_0, c_1, \cdots, c_{N-1}] \in \{\mathrm{GF}(q)\}^N$, where $q = 2^b$. Before sending, the code word is mapped to the bipolar vector $\mathbf{t} = [t_0, t_1, \cdots, t_{N-1}]$ according to the transmitted signal constellation Ω, where $t_n = \left[t_{nb}, t_{nb+1}, \cdots, t_{(n+1)b-1}\right], 0 \leq n < N$. Using Binary Phase Shift Keying (BPSK) modulation, $\phi : \mathrm{GF}(q) \to \Omega^b$, where $\Omega = \{-1, 1\}$. The specific mapping rule is $\phi(c_n) = t_n = \left[t_{nb}, \cdots, t_{(n+1)b-1}\right]$, where $t_i \in \Omega, 0 \leq i < Nb$. After the transmission of the Additive White Gaussian Noise (AWGN), the output is:

$$r = t + n \tag{1}$$

where n is the independent white noise real vector with noise power $\sigma^2 = (2R_c E_b/N_0)^{-1}$; E_b/N_0 is the signal-to-noise ratio of each information bit; and $R_c = K/N$ is the bit rate.

3. Algorithm Description

3.1. Sum of the Magnitude of Weighted Symbol Flipping Decoding Algorithm

The WSF decoding algorithm uses the information derived from the channel to calculate the reliability [15,20]. The core idea is to compute the measure of each symbol according to the reliability of the check equation. According to the size of the measured value, the most unreliable symbol is selected to be flipped. It is the simplest SF decoding algorithm based on the hard decision. The flip function $E_{n,\alpha}^{(k)}$ of the MWSF algorithm and the IMWSF algorithm takes into account the information carried by the symbol itself, but still neglects the reliability information of a large number of variable nodes when calculating the reliability of the check equation $w_{n,m,\alpha}$ [19,25]. However, the variable node involved in each check equation is a set and the reliability calculation of the check equation should also contain the reliability information for each variable node in the set. Therefore, in this paper, the reliability and flip function are redefined. Based on the hard decision decoding, some soft information from the channel is used to calculate the reliability. The sum of the amplitude of the variable nodes connected to the check node is used as the reliability of the check equation, the influence of all symbols is added into the reliability of the check equation and the weighting factor is introduced. Using $|L_{n,\alpha}|$ as part of the flip function, we obtained the WSF algorithm based on the sum of the magnitude (SMWSF). The specific process of SMWSF decoding algorithm is as follows.

- Step 1: Initialization parameters

  1. Set initial iterations $k = 0$
  2. Calculate the initial hard decision output sequence, before using this sequence as the output of the decoding iteration in $k = 0$, which is denoted as $y^{(0)}$

Firstly, the hard decision sequence $\mathbf{x} = [x_0, x_1, \cdots, x_{Nb-1}]$ is obtained by the hard decision of the binary sequence r of channel output. The rules of the decision are as follows: If $r_i \geq 0$, the x_i is determined to be 1; if $r_i < 0$, the x_i is determined to be 0 $(0 \leq i < Nb - 1)$. As the hard decision is a binary sequence, so according to the binary and q-ary conversion principle, converted to N length, the sequence becomes $\mathbf{y} = [y_0, y_1, \cdots, y_{N-1}]$. The sequence of $\mathbf{y} = [y_0, y_1, \cdots, y_{N-1}]$ is the output sequence.

3. Calculate the probability vector $L_n(\alpha)$

After the hard decision, it is necessary to calculate the initial likelihood probability of the symbol and to prepare the average likelihood probability of the check equation adjacent to all variable nodes in the subsequent iterative decoding process. The definition $\mathrm{GF}_0(q) = \{\alpha_1, \cdots, \alpha_{q-1}\}$ to remove the Galois field of zero element, while $\mathrm{GF}(q)$ does not contain zero elements. Gives the reliability vector α on the nth symbol:

$$\mathbf{L}_n = \left[L_{n,\alpha_1}, \cdots, L_{n,\alpha}, \cdots, L_{n,\alpha_{q-1}}\right] \tag{2}$$

where the symbol probability log likelihood is shown as follows:

$$L_{n,\alpha} = \ln \frac{P(a = \alpha_i)}{P(a = 0)} = \frac{2}{\delta^2} \sum_{j:\phi(\alpha)_j = +1} r_{nb+j} \tag{3}$$

Obviously, the log likelihood ratio of symbol probability is directly proportional to $\sum_{j:\phi(\alpha)_j=+1} r_{nb+j}$. In this present study, we ignored the coefficient $\frac{\delta^2}{2}$ related to channel noise [29] and used $\sum_{j:\phi(\alpha)_j=+1} r_{nb+j}$ approximation as the symbol probability log likelihood ratio, without any effect on performance.

4. Compute the reliability $w_{n,m,\alpha}$ of the external information of the received symbol y_n

We defined $L_{n,\alpha} = \sum_{j:\phi(\alpha)_j=+1} r_{nb+j} (0 \le j \le b-1)$ and used Equation (3) to calculate probability matrix $\mathbf{L}_n = [L_{n,\alpha}], 0 \le n < N, \alpha_1 \le \alpha < \alpha_{q-1}$. In this, $w_{n,m,\alpha}$ represents the average probability information of the symbol α with all the variable nodes adjacent to the mth check equation. As shown in Equation (4), the formula of $w_{n,m,\alpha}$ is:

$$w_{n,m,\alpha} = \sum_{i \in N(m)n} |L_{i,\alpha}|, 0 \le n < N, 0 \le m < M, \alpha \in \mathrm{GF}_0(q) \tag{4}$$

- Step 2: Calculating check equation $s_h^{(k)}$

We assume that the symbol vector is $\mathbf{y}^{(k-1)}$ before the $k (k \ge 1)$ iteration, while the corresponding calibration equation is $\mathbf{s}_h^{(k-1)} = \mathbf{y}^{(k-1)} H^T \ne 0$. Then, the check equation $\mathbf{s}_h^{(k)}$ is calculated, $\mathbf{s}_h^{(k)} = \mathbf{y}^{(k)} H^T$. If the check equation $\mathbf{s}_h^{(k)} = 0$, the decoding iteration is stopped and the decoding is shown to be successful.

- Step 3: $k \leftarrow k+1$, if the $k > k_{\max}$ ($k_{\max}$ is the maximum number of iterations set for the user), the decoding is declared to fail and stop.

The completion of a symbol of the flip need to determine the two parameters: (1) The position of the flip; and (2) the value or amplitude of the flip.

- Step 4: Determining the position of the flip symbol

We calculated the measured value $E_n^{(k)}, 0 \le n < N$ for each variable node at the k$^{\text{th}}$ iteration. The core of the WSF algorithm involves flipping out the symbols that do not satisfy the check equation. To calculate $E_n^{(k)}$, we first need to calculate the reliability of variable node n, respectively $\{\alpha_1, \cdots, \alpha_{q-1}\}$.

$$E_{n,\alpha}^{(k)} = \sum_{m \in M(n)} \left(2 s_{h,m}^{(k)} - 1\right) w_{n,m,\alpha} - \beta |L_{n,\alpha}| \tag{5}$$

In Equation (5), $\beta (\beta \ge 0)$ is the weighting factor. At that time, $\beta = 0$, which allows us to integrate the MSMWSF algorithm into SMWSF algorithm. For the given non-binary LDPC codes, the optimal value of the weighting factor can be obtained by Monte Carlo simulation under the specified number of iterations [30].

After this, we update the check equation to get the adjacency value of the hard decision:

$$s_{h,m}^{(k)} = \begin{cases} 0, s_h^{(k)} = 0 \\ 1, s_h^{(k)} \neq 0 \end{cases} \tag{6}$$

Among them, m is the CN. If $\mathbf{s}_h^{(k)} = 0$, a valid code word is obtained and the search is stopped. If $\mathbf{s}_h^{(k)} \neq 0$, the flip function $E_n^{(k+1)}, 0 \leq n < N$ is calculated and looped accordingly.

In order to estimate the reliability measure of a symbol, the measure of each symbol must be calculated:

$$E_n^{(k)} = \sum_{\alpha \in \mathrm{GF}_0(q)} E_{n,\alpha}^{(k)} \tag{7}$$

We select the location where the symbol is to be flipped, which is $n^{(k)}, 0 \leq n < N$.

The summation operation in Equation (7) is a weighted method to measure the location possibility of flipping symbols. The sum of the weights of 1 can enhance the measure of the symbol position and provide a more accurate judgment standard. In this way, the symbol corresponding to the maximum value of $E_n^{(k)}$ is the position of the flip symbol, such as the Equation (8).

$$n^{(k)} = \mathrm{argmax} E_n^{(k)}, n \in [1, N], n \notin A \tag{8}$$

- Step 5: Determining the value or magnitude of the flip

For the chosen symbol $y_n^{(k)}$, flip the bit corresponding to the minimum value of $|r_{ni}|$, where $0 \leq i < b$, and update $\mathbf{y}^{(k-1)}$.

- Step 6: Decoding, according to the results of the flip to get a new hard decision decoding sequence.

Assign the value of $\mathbf{y}^{(k-1)}$ to $\mathbf{y}^{(k)}$, $\mathbf{y}^{(k)} \leftarrow \mathbf{y}^{(k-1)}$, and return to the second step.

3.2. Loop Update Detection Algorithm

Due to the possibility of an infinite loop in the process of symbol flipping, the symbol after the current flipping is still an error symbol and cannot be flipped to correct the symbol. Therefore, this paper proposes a loop update detection algorithm to further improve the decoding performance and speed up the convergence.

Firstly, the output symbol sequence matrix and the infinite loop detection matrix are defined, which involves the sequential traversal of each symbol. After this, sorting them from smallest to largest, we determine the location of the flip symbol. From the second smallest symbol, we re-flip the same error symbol, which is converted to the bit value. The wrong bits are sorted from smallest to largest. According to the definition of small to large index value, the bit is flipped and then converted into a symbol value to replace the previous decoding symbol. The symbol position corresponding to the maximum value is placed in the exclusion sequence. Essentially, after the symbol position corresponding to the maximum value is excluded, the maximum value error symbol is searched again. This process is repeated for decoding the iteration. The specific decoding process is as follows.

- Step 1: Initialization

The excluded sequence $\boldsymbol{A}$ is initialized to an empty set, which is used to store the position corresponding to the symbol that does not satisfy the flipping function. The bit flipping identifier $\boldsymbol{F}$ is defined and initialized to 1, which is used as the counter. The maximum value is $\boldsymbol{b}$ $(1bq)$ and the value of $\boldsymbol{F}$ determines the number of bits to be flipped in the flip symbol.

- Step 2: Determining the magnitude of symbol flipping

The bit position to be flipped in the symbol position $n^{(k)}$ is determined by the binary sequence $\mathbf{r} = [r_0, r_1, \cdots, r_{Nb-1}]$, which is transmitted after the AWGN channel is transmitted. The symbol at position $n^{(k)}$ of the symbol in the to-be-flipped state can be converted into b bits, which correspond to $\left[r_{nb}, r_{nb+1}, \cdots, r_{(n+1)b-1}\right]$ in $\mathbf{r}$. We sort $|r_i|(nb \le i \le (n+1)b-1)$ from largest to smallest, with a smaller $|r_i|$ indicating lower reliability of the corresponding bit. Therefore, the $\boldsymbol{F}$ bit position with the smallest absolute value in $\left[r_{nb}, r_{nb+1}, \cdots, r_{(n+1)b-1}\right]$ is selected according to the bit flipping identifier $\boldsymbol{F}$, while the F bits in the corresponding position are reversed to obtain a new symbol sequence $\mathbf{y}^{(k)} = [y_0, y_1, \cdots, y_{N-1}]$.

- Step 3: Detects whether there is an infinite loop

The loop update detection algorithm (LUD) is as follows. First, the output symbol sequence matrix $Y_{(k+1)\times N}$ is defined, as shown in Equation (9).

$$\mathbf{Y} = \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(k)} \end{bmatrix} \tag{9}$$

Following this, an infinite loop detection matrix $\mathbf{E}_{k\times N}$ is defined, as shown in Equation (10).

$$\mathbf{E} = \begin{bmatrix} y^{(k)} - y^{(0)} \\ y^{(k)} - y^{(1)} \\ \vdots \\ y^{(k)} - y^{(k-1)} \end{bmatrix} \tag{10}$$

As long as all the elements of one row in an infinite loop matrix are 0, an infinite loop is detected. Otherwise, an infinite loop is not detected.

(1) If an infinite loop is detected and F does not reach the maximum $\boldsymbol{b}$, we increase the value of $\boldsymbol{F}$ by 1. After this, we return to step 2 to re-determine the position of the specific bit to be flipped by the symbol, before flipping the $\boldsymbol{F}$ bits of the corresponding position.
(2) If an infinite loop is detected but $\boldsymbol{F}$ has reached the maximum $\boldsymbol{b}$, the currently selected flip symbol position is stored in the exclusion symbol sequence $\boldsymbol{A}$. $\boldsymbol{F}$ is set to 1 and the flip symbol position is rediscovered.
(3) If no infinite loop is detected, the exclusion symbol sequence $\boldsymbol{A}$ is set as an empty set, the bit flipping identifier $\boldsymbol{F}$ is 1 and the calibration equation is recalculated.

Combining the loop update detection algorithm with the weighted symbol flipping algorithm based on sum of magnitude, a modified sum of the magnitude for the weighted symbol flipping decoding algorithm based on loop update detection (LUDMSMWSF) is proposed. On the basis of the MSMWSF algorithm, the algorithm repeatedly flips the bits corresponding to the error code word and looks for them in the order of the most probable error probability. Finally, this algorithm finds the correct code word. If this error traverses all the symbols, there is still an infinite loop, which indicates that the current symbol is not an error symbol. Following this, the algorithm adds the current symbol to the excluded symbol set to ensure that the next current position will not be continuously found. The specific decoding algorithm flow chart is shown in Figure 1.

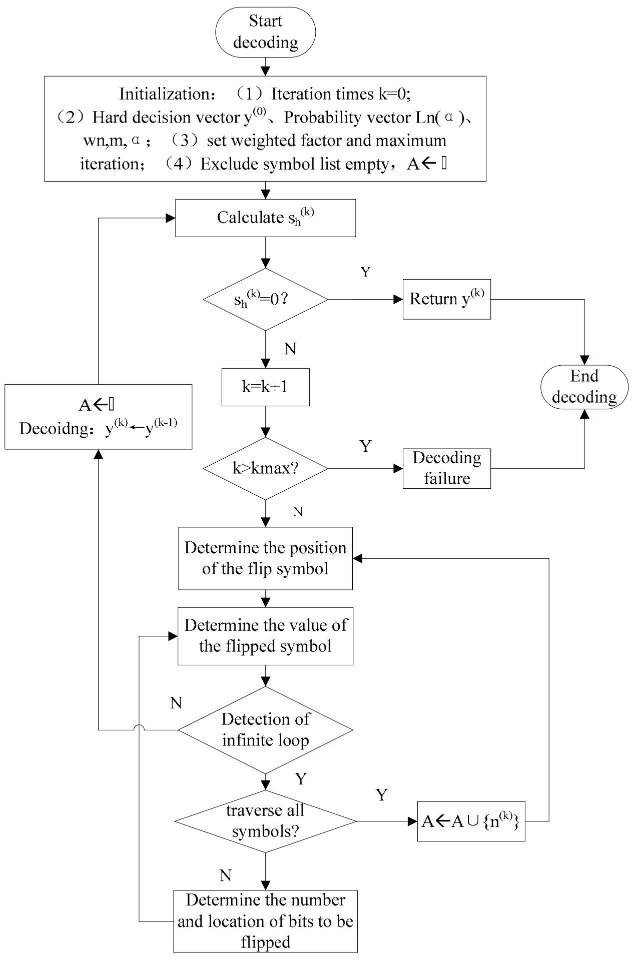
Start decoding
Initialization: (1) Iteration times k=0; (2) Hard decision vector y(0)、Probability vector Ln(α)、wn,m, α; (3) set weighted factor and maximum iteration; (4) Exclude symbol list empty, A←∅
Calculate sh(k)
sh(k)=0?
Y
Return y(k)
N
k=k+1
End decoding
A←∅
Decoidng: y(k)←y(k-1)
k>kmax?
Y
Decoding failure
N
Determine the position of the flip symbol
Determine the value of the flipped symbol
N
Detection of infinite loop
Y
traverse all symbols?
Y
A←A∪{n(k)}
N
Determine the number and location of bits to be flipped

Figure 1. Flow chart of LUDMSMWSF algorithm.

4. Complexity Analysis

In this section, the conditions used to analyze the complexity of the decoding algorithm are: (1) Ignoring a small number of binary calculation and multiplication operations in the algorithm, with the assumption that the comparison operation is equivalent to the addition operation; and (2) taking the regular non-binary LDPC codes as an example to compare the average number of real addition operations in each algorithm. In the past, the computational complexity of the initial stage before the first iteration is often neglected when analyzing the computational complexity [31]. However, the simulation analysis of the decoding algorithm in the next section shows that the highest signal-to-noise ratio (SNR) in the frame occurs through iterative decoding less frequently after convergence. Thus, there is strong complexity of the initialization phase in the decoding process of the high proportion, if the neglect leads to a significant error. Therefore, the computational complexity involved in the initialization stage before the first iteration is taken into account. At the same time, the principle of second sections shows that all WSF algorithms perform symbol flipping in the final stage of each iteration, which is a serial decoding algorithm. After the symbol flipping is completed, the check equation is updated and the flip function is updated. In each iteration, there are three steps involved: (1) Calculating the adjoining vector $\mathbf{s}_h^{(k)}$; (2) updating the flip function E_n^{k+1}; and (3) searching the flipped bit. Taking the standard WSF algorithm as an example, if the variable node n is flipped during the previous iteration, the adjoining vectors $\mathbf{s}_h^{(k)}$ corresponding to the d_v check nodes connected to them need to be updated. After the update of the vector $s_h^{(k)}$ is completed, the flip function E_n^{k+1} of d_c variable nodes connected to the check node m is calculated. Therefore, for both steps, the total number of operations required is $d_c \cdot d_v$. In Step (3), it is assumed that the node position of the variable corresponding to the maximum flip function E_n^{k+1} is found in n variable nodes, while further comparison operations are performed for $N-1$ times. Therefore, the calculation amount of an iterative process in a standard WSF algorithm is $N-1+d_c \cdot d_v$. The specific update process is as follows. First, we calculate $s_{h,m}^{k+1} = 1 - s_{h,m}^{k}, m \in M(\theta)$ and update it according to the updated formula of flip function, which is $E_n^{k+1} = E_n^k + 2 \sum\limits_{m \in M(\theta)} w_{n,m,\alpha}\left(2s_{h,m}^{k+1} - 1\right), n \in N(m), m \in M(\theta)$. The average number of iterations of the five decoding algorithms is AI_1–AI_6, while d_c represents line weight and d_v represents column weight. Table 1 gives the calculation methods of the total number of real operations of each decoding algorithm. From Table 1, it can be concluded that the complexity of the LUDWSF, LUDSMWSF and LUDMSMWSF algorithms is lower than the WSF, SMWSF and MSMWSF algorithms. Taking one algorithm as an example, the computation amount of WSF algorithm and LUDWSF algorithm is the same for each iteration with the difference of the average iteration times. However, the average iterations of LUDWSF algorithm are much less than that of WSF algorithm. From Table 1, the LUDMSMWSF algorithm only requires real additions, complexity is $Mq(2d_c-1)+Nqd_v+(N-1)+(N-1+d_cd_v)(AI_5-1)$. However, complexity of the Fast Fourier Transform-based belief propagation decoding algorithm (FFT-BP) [32,33] includes real additions, multiplications and divisions, the complexities of which are $AI_6[2Nd_vq\log_2 q + 2Nd_v(q-1) + M(d_c-1)]$, $AI_6[Nd_vq(d_c+2d_v-1)+Md_c]$ and $AI_6[Nd_v(q+2)]$, respectively. Real multiplications and divisions are more consumable units than real additions. Although the proposed WSF algorithm with flipping pattern requires more iterations for decoding than FFT-BP, it needs less real additions than FFT-BP and requires no multiplying the iterations with computational requirements in each iteration, the total computational requirement of WSF algorithm is still lower than FFT-BP. Therefore, the computational requirement of WSF algorithm is much lower than FFT-BP.

Table 1. The average total number of real operations in each algorithm.

SF Algorithm	Addition Operations	Multiplication Operations	Division Operations
WSF	$Mq(d_c-1)+Nq(d_v-1)+(N-1)$ $+(N-1+d_cd_v)(AI_1-1)$	**0**	**0**
SMWSF	$Mq(2d_c-1)+Nq(d_v-1)+(N-1)$ $+(N-1+d_cd_v)(AI_2-1)$	**0**	**0**
MSMWSF	$Mq(2d_c-1)+Nqd_v+(N-1)$ $+(N-1+d_cd_v)(AI_3-1)$	**0**	**0**
LUDSMWSF	$Mq(2d_c-1)+Nq(d_v-1)+(N-1)$ $+(N-1+d_cd_v)(AI_4-1)$	0	0
LUDMSMWSF	$Mq(2d_c-1)+Nqd_v+(N-1)$ $+(N-1+d_cd_v)(AI_5-1)$	0	0
FFT-BP	$AI_6\left[\begin{array}{c} 2Nd_vq\log_2 q \\ +2Nd_v(q-1)+M(d_c-1) \end{array}\right]$	$AI_6\left[\begin{array}{c} Nd_vq(d_c+2d_v-1) \\ +Md_c \end{array}\right]$	$AI_6[Nd_v(q+2)]$

5. Simulation Results and Statistical Analysis

The simulation parameters used in this section are as follows: 384,192 LDPC codes with a code rate of 0.5 and a column weight of three. The matrix is generated by the progressive edge growth (PEG) algorithm [34,35], divided into 4-ary (Code 1) and 16-ary (Code 2) simulation, which has a maximum number of iterations of 100. Under the AWGN channel conditions and using BPSK modulation, at least 1000 error bits are collected at each SNR. The link level simulation block diagram is shown in Figure 2.

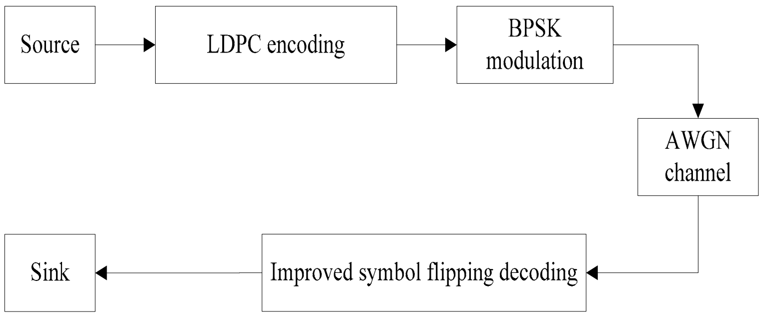

Figure 2. The link level simulation block diagram.

5.1. Weighted Factor Test

For non-binary LDPC codes with given column weights, the decoding performance is different under the same signal-to-noise ratio with different weighting factors. When choosing a constant and optimal weighting factor, the performance loss is negligible and the complexity of the implementation is decreased [36,37]. The optimal value of the weighting factor is generally related to the non-binary LDPC codes and the specific code structure. Figures 3 and 4 show the bit error rate performance of Code 1 and Code 2 with different weighting factors when using MSMWSF algorithm under different SNR conditions. Based on the definition of the optimal value of weighted factor, it can be seen from Figure 3 that the influence of weighting factor on bit error rate (BER) is not obvious at lower SNR as the optimal value of weighting factor varies little with an increase in SNR. As shown in Figures 3 and 4, the optimal value of the weighting factor of the MSMWSF algorithm in Code 1 in this paper is 1.8, while the optimal value of the weighting factor of the MSMWSF algorithm in Code 2 is one.

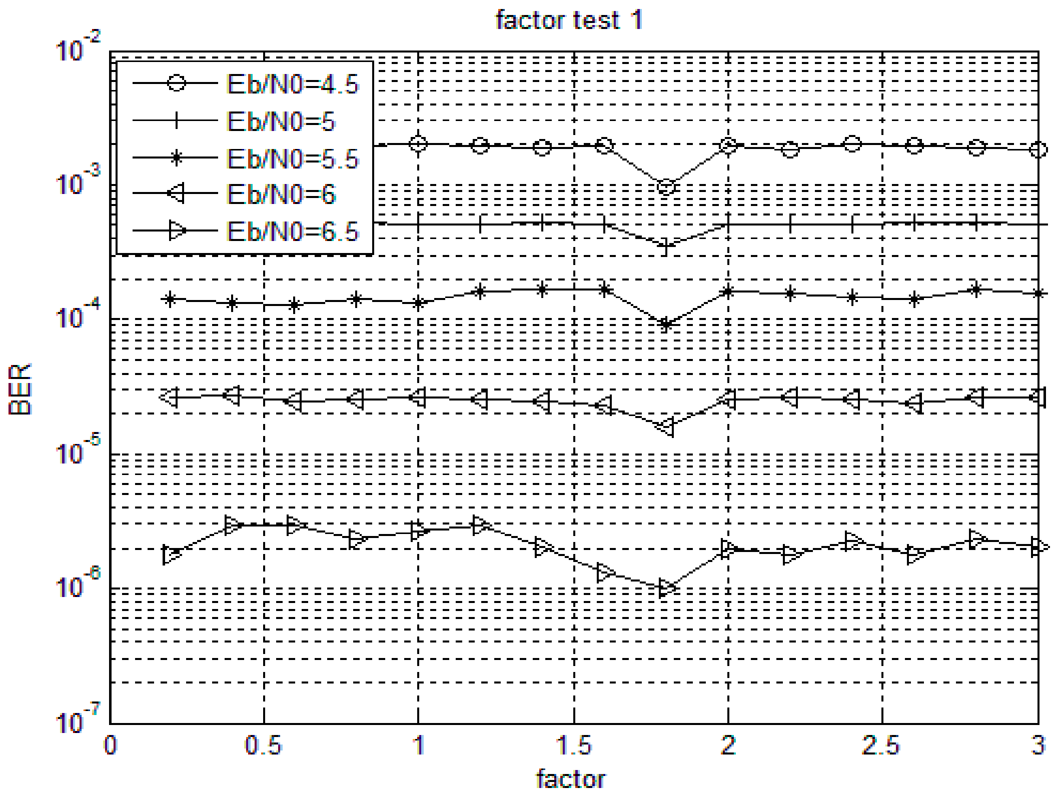

Figure 3. Code 1 weighted factor test.

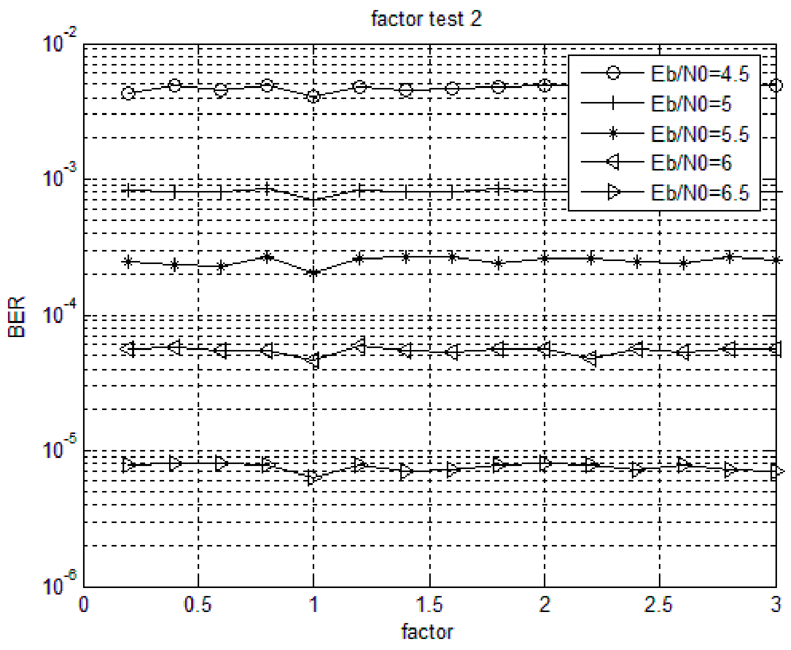

Figure 4. Code 2 weighted factor test.

5.2. Comparison of Algorithm Performance and Average Iteration Numbers

The performance comparison between Code 1 and Code 2 in five different decoding algorithms under the optimal parameters is shown in Figures 5 and 6, respectively. With an increase in SNR, the coding gain of MSMWSF algorithm is gradually increasing compared with other existing algorithms. At a low SNR, the performance of MSMWSF algorithm is almost the same as other algorithms. According to the simulation diagram of reference [34–36], a higher number of binary numbers indicates a poorer performance of the WSF algorithm. From Figures 5 and 6, we can see that the performance of Code 1 is better than that of Code 2, which proves the effectiveness of the algorithm proposed in this paper.

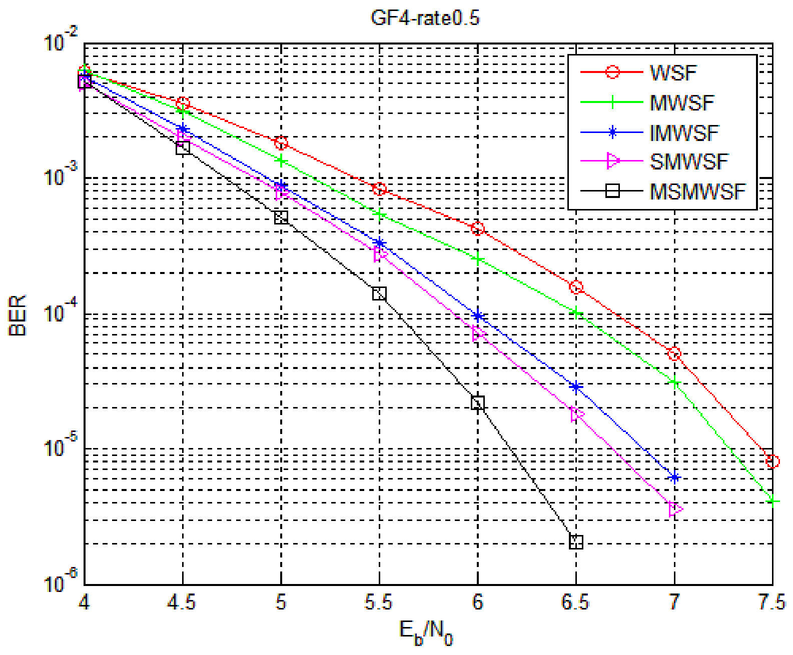

Figure 5. Comparison of five algorithms under Code 1.

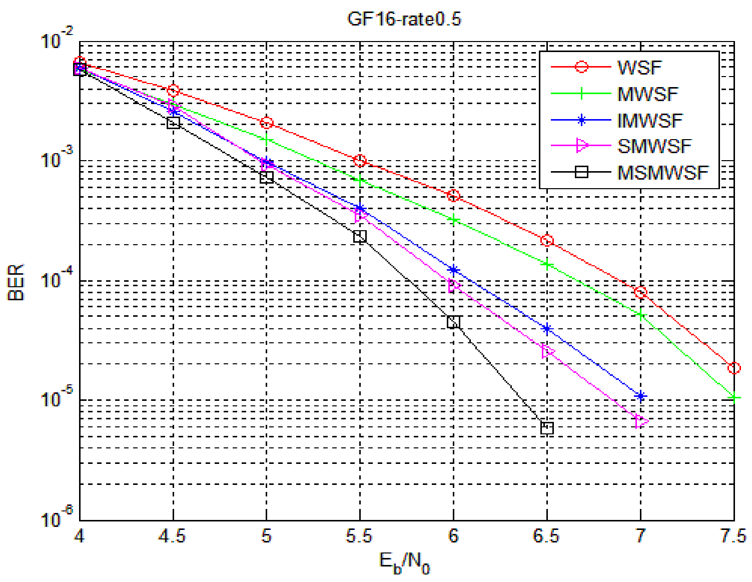

Figure 6. Comparison of five algorithms under Code 2.

Figures 7 and 8 show the performance of Code 1 and Code 2 under the SMWSF algorithm and MSMWSF algorithm, respectively, with the optimal parameters of the weighting factors after introducing the loop update detection algorithm. As seen from Figures 7 and 8, the LUDMSMWSF algorithm improves the decoding performance compared to the MSMWSF algorithm, accelerating the convergence speed. Compared with the WSF algorithm, a coding gain of 2.2 dB is obtained in the case of Code 1 and a coding gain of 2.35 dB in the case of Code 2. Under the same decoding algorithm, the coding gain of about 0.8–1.1 dB can be obtained after introducing the proposed loop update detection algorithm. At the same time, the LUDMSMWSF algorithm is about 0.85–1.05 dB away from FFT-BP decoding at BER of 10^{-5} with a tremendous reduction of computational requirement. In view of the results on Figures 7 and 8, we argue that the proposed symbol flipping algorithm offer the tradeoff points between performance and computational cost.

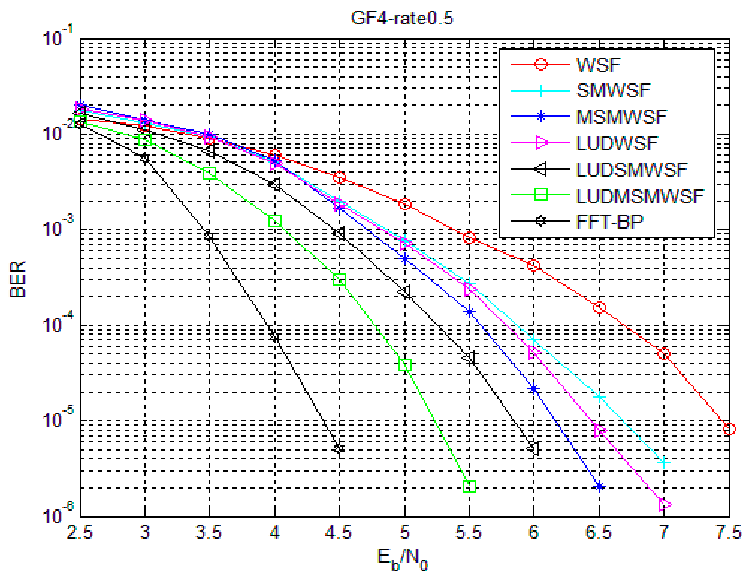

Figure 7. Comparison of improved algorithms under Code 1.

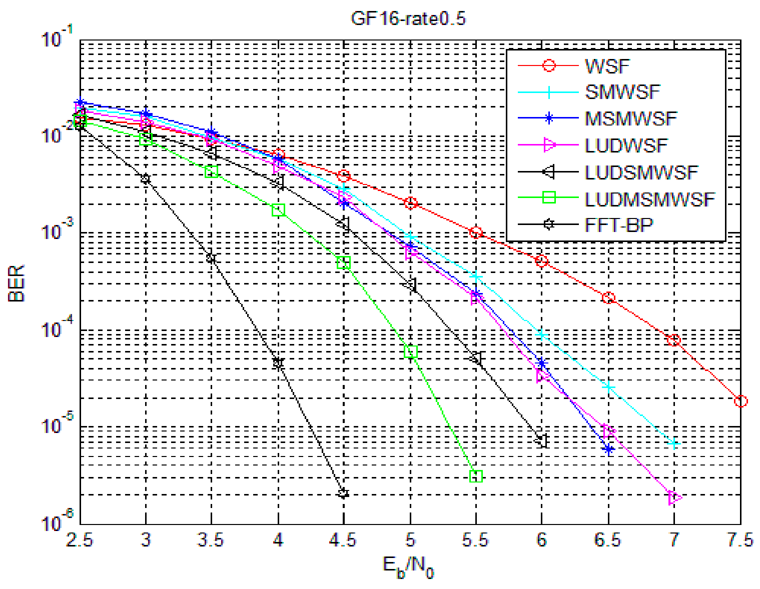

Figure 8. Comparison of improved algorithms under Code 2.

Figures 9 and 10 gives the average number of iterations and average number of addition operations of Code 1 under five decoding algorithms. As shown in Figures 9 and 10, the two algorithms proposed in this paper are significantly less than the average number of iterations and the average number of addition operations in the traditional algorithm. Because the algorithm proposed in this paper has more efficient and accurate symbol flipping function, it improves the performance and reduces the complexity of the algorithm to a certain extent. We see that MSMWSF algorithm achieves fast convergence and low complexity.

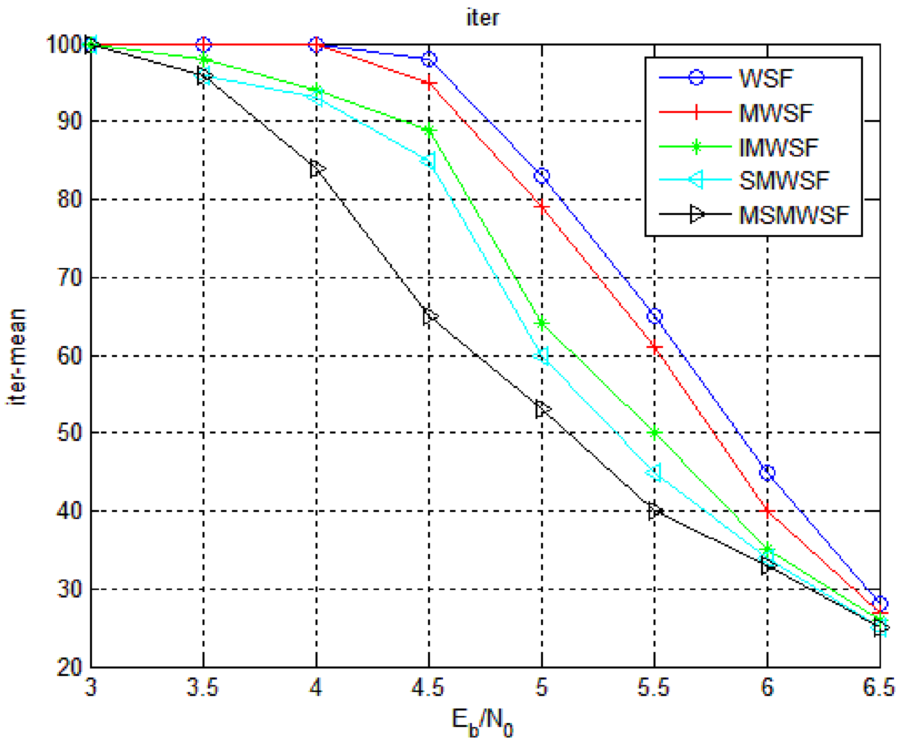

Figure 9. The average number of iterations of five algorithms under Code 1.

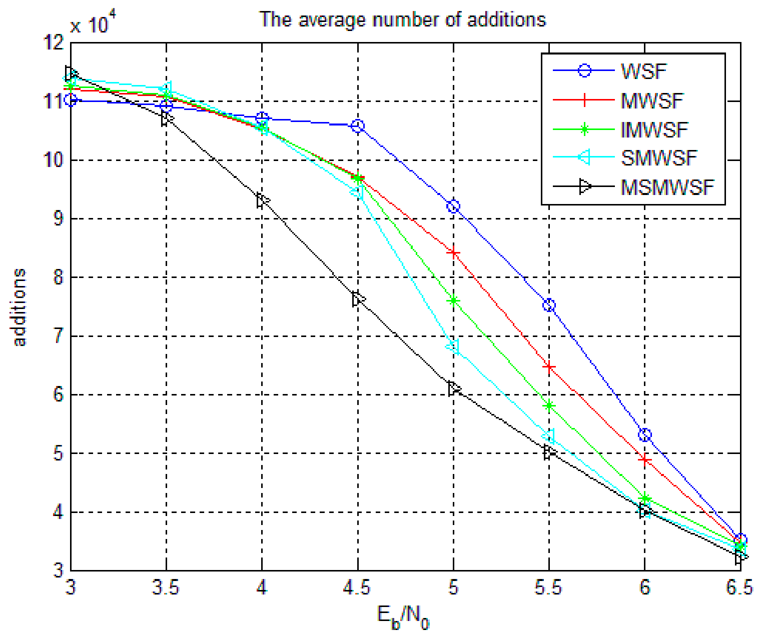

Figure 10. The average number of addition operations of five algorithms under Code 1.

We set the SNR to be 3.5 dB with the simulation of 10,000 frames under the condition of Code 1. We then combined the statistical five algorithms for decoding failure frames, as shown in Table 2. Table 2 shows that the WSF algorithm has nearly 99% frame failure. Two decoding algorithms are proposed in this paper. The failure of the frame is about 62% and 42%, which is less than traditional algorithms. It can also be seen that the algorithm proposed in this paper does not increase the complexity of the implementation, but has been reduced to a certain extent.

Table 2. Five algorithms for decoding failure frames.

SF Algorithm	Decoding Failure Frames
WSF algorithm	9915
MWSF algorithm	9840
IMWSF algorithm	6544
SMWSF algorithm	6200
MSMWSF algorithm	4184

The total computation required for decoding this LDPC codewith three various algorithms at 4.5 dB is shown in Table 3. From Table 3, the FFT-BP algorithm is nearly six times the computational requirement of the MSMWSF algorithm only in real addition. Therefore, we can prove that the computational requirement of MSMWSF algorithm is much lower than FFT-BP with no real multiplication and division. The MSMWSF algorithm has the lowest complexity and does not need to consume hardware resources and multiplication and division operations in software overhead.

Table 3. Under the condition of Code 2, Eb/N0 = 4.5 dB, decoding complexity of three algorithms.

SF Algorithm	Addition Operations	Multiplication Operations	Division Operations
WSF algorithm	205644	0	0
MSMWSF algorithm	92710	0	0
FFT-BP algorihtm	567723	744870	66267

6. Conclusions

This paper proposes a sum of the magnitude for hard decision decoding algorithm based on loop update detection. The algorithm combines the magnitude of the sum information of the variable nodes adjacent to the check node and uses it as the reliability information. At the same time, the reliability information of the variable node itself is taken into account and a more effective flip function is obtained, which improves the flip efficiency of the symbol and improves the decoding performance of the algorithm. The loop update detection algorithm is introduced to improve the accuracy of symbol flipping and to further accelerate the convergence rate of decoding. Simulation results show that compared with the WSF algorithm, the proposed LUDMSMWSF algorithm gains about 1.3 dB and 1.8 dB respectively and the decoding complexity is greatly reduced. Therefore, the algorithm proposed in this paper can be better applied to the 5G mobile communication system and meet the requirements of the decoding algorithm. It is a good candidate decoding algorithm for high speed communication devices.

Acknowledgments: This research work was supported by the ZTE Cooperation Forum Project (Grant No. KY10800160020), the National Natural Science Foundation of China (Grant No. 61371099), the Fundamental Research Funds for the Central Universities of China (Grant No.HEUCF150814/150812/150810), the International Exchange Program of Harbin Engineering University for Innovation-Oriented Talents Cultivation.

Author Contributions: Danfeng Zhao and Hai Tian conceived the ideas and concept. Jiahui Meng implemented the software and carried out the experiments and wrote the manuscript. Liang Zhang carried out simulation experiments and further verified the experimental results. Danfeng Zhao critically reviewed the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Naga, B.; Li, J.Y.; Durga, P.M. Network densification: The dominant theme for wireless evolution into 5G. *IEEE Commun. Mag.* **2014**, *52*, 82–89.
2. Guo, L.; Ning, Z.L.; Song, Q.Y. Joint encoding and grouping multiple node pairs for physical-layer network coding with low-complexity algorithm. *IEEE Trans. Veh. Technol.* **2017**, *66*, 9275–9286. [CrossRef]
3. Wang, L.S.; Wang, Y.M.; Ding, Z.Z. Cell selection game for densely-deployed sensors and mobile devices in 5G networks integrating heterogeneous cells and internet of things. *Sensors* **2015**, *15*, 24230–24256. [CrossRef] [PubMed]
4. Fortuna, C.; Bekan, A.; Javornik, T. Software interfaces for control, optimization and update of 5G type communication networks. *Comput. Netw.* **2017**, *129*, 373–383. [CrossRef]
5. Gao, X.; Ove, E.; Fredrik, R. Massive MIMO performance evaluation based on measured propagation data. *IEEE Trans. Wirel. Commun.* **2015**, *14*, 3899–3911. [CrossRef]
6. Cheng, L.; Zhu, H.; Li, G. LDPC encoder design and FPGA implementation in deep space communication. In Proceedings of the International Conference on Logistics, Engineering, Management and Computer Science, Shenyang, China, 24–26 May 2014; pp. 343–346.
7. Wang, C.L.; Chen, X.H.; Li, Z.W. A simplified Min-Sum decoding algorithm for non-binary LDPC codes. *IEEE Trans. Commun.* **2015**, *61*, 24–32. [CrossRef]
8. Marc, P.C.F.; Miodrag, J.M.; Hideki, I. Reduced complexity iterative decoding of low-density parity check node based on belief propagation. *IEEE Trans. Commun.* **1999**, *47*, 673–680.
9. Aslam, C.A.; Guan, Y.L.; Cai, K. Edge-based dynamic scheduling for belief-propagation decoding of LDPC and R S codes. *IEEE Trans. Commun.* **2017**, *65*, 525–535. [CrossRef]
10. Namrata, P.B.; Brijesh, V. Design of hard and soft decision decoding algorithms of LDPC. *Int. J. Comput. Appl.* **2014**, *90*, 10–15.
11. Evdal, A.; Najeeb, U.H.; Michael, L. Challenges and some new directions in channel coding. *J. Commun. Netw.* **2015**, *17*, 328–338.
12. Balasuriya, N.; Wavegedara, C.B. Improved symbol value selection for symbol flipping-based non-binary LDPC decoding. *EURASIP J. Wirel. Commun. Netw.* **2017**, *2017*, 105. [CrossRef]
13. Chen, M.; Hao, Y.X.; Qiu, M.K. Mobility-Aware caching and computation off loading in 5G ultra-dense cellular network. *Sensors* **2016**, *16*, 974. [CrossRef] [PubMed]

14. Swapnil, M. *High-Throughput FPGA QC-LDPC Decoder Architecture for 5G Wireless*; The State University of New Jersey: Rutgers, NJ, USA, 2015.
15. Kou, Y.; Lin, S.; Fossorier, M. Low-density parity-check codes based on finite geometries: A rediscovery and new results. *IEEE Trans. Inf. Theory* **2001**, *47*, 2711–2736. [CrossRef]
16. Zhang, J.; Fossorier, M. A modified weighted bit-flipping decoding of low-density parity-check codes. *IEEE Commun. Lett.* **2004**, *8*, 165–167. [CrossRef]
17. Jiang, M.; Zhao, C.M.; Shi, Z. An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes. *IEEE Commun. Lett.* **2005**, *9*, 814–816. [CrossRef]
18. Guo, R.; Liu, C.; Wang, M. Weighted symbol-flipping decoding for non-binary LDPC codes based on average probability and stopping criterion. *J. Commun.* **2016**, *37*, 43–52.
19. Zhang, G.Y.; Zhou, L.; Su, W.W. Average magnitude based weighted bit-flipping decoding algorithm for LDPC codes: Average magnitude based weighted bit-flipping decoding algorithm for LDPC codes. *J. Electron. Inf. Technol.* **2014**, *35*, 2572–2578. [CrossRef]
20. Garcia-Herrero, F.; Li, E.B.; Declercq, D. Multiple-Vote Symbol-Flipping Decoder for Nonbinary LDPC Codes. *IEEE Trans. Very Large Scale Integr. Syst.* **2014**, *11*, 2256–2267. [CrossRef]
21. Garcia-Herrero, F.; Declercq, D.; Valls, J. Non-binary LDPC decoder based on symbol flipping with multiple votes. *IEEE Commun. Lett.* **2014**, *18*, 749–752. [CrossRef]
22. Nhan, N.Q.; Ngatched, T.M.N.; Dobre, O.A. Multiple-votes parallel symbol-flipping decoding algorithm for non-binary LDPC codes. *IEEE Commun. Lett.* **2015**, *19*, 905–908. [CrossRef]
23. Ueng, Y.L.; Wang, C.Y.; Li, M.R. An efficient combined bit-flipping and stochastic LDPC decoder using improved probability traces. *IEEE Trans. Signal Process.* **2017**, *65*, 5368–5380. [CrossRef]
24. Le, K.; Ghaffari, F.; Declercq, D. Efficient hardware implementation of probabilistic gradient descent bit-flipping. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 906–917. [CrossRef]
25. Lu, M. *Research on Construction and Decoding Algorithm of Nonbinary LDPC Codes*; Beijing Jiaotong University: Beijing, China, 2013.
26. Mackay, D.J.; Wilson, S.T.; Davey, M.C. Comparison of constructions of irregular Gallager codes. *IEEE Trans. Commun.* **1999**, *47*, 1449–1454. [CrossRef]
27. Garcia-Herrero, F.; Canet, M.J.; Valls, J. Nonbinary LDPC Decoder Based on Simplified Enhanced Generalized Bit-Flipping Algorithm. *IEEE Trans. Very Large Scale Integr. Syst.* **2014**, *22*, 1455–1459. [CrossRef]
28. Thi, H.P. Two-Extra-Column Trellis Min-max Decoder Architecture for Nonbinary LDPC Codes. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 1781–1791. [CrossRef]
29. Guo, F.; Hanzo, L. Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes. *Electron. Lett.* **2004**, *40*, 1356–1358. [CrossRef]
30. Zhang, J.Y. Simplified symbol flipping algorithms for nonbinary low-density parity-check codes. *IEEE Trans. Commun.* **2017**, *65*, 4128–4137. [CrossRef]
31. Liu, B.; Tao, W.; Dou, G.Q. Weighted symbol-flipping decoding for nonbinary LDPC codes based on a new stopping criterion. *J. Electron. Inf. Technol.* **2011**, *33*, 309–314. [CrossRef]
32. Sulek, W. Non-binary LDPC decoders design for maximizing throughput of an FPGA implementation. *Circuits Syst. Signal Process.* **2016**, *35*, 4060–4080. [CrossRef]
33. Kang, J.Y.; Huang, Q.; Zhang, L. Quasi-cyclic LDPC codes: An algebraic construction. *IEEE Trans. Commun.* **2010**, *58*, 1383–1396. [CrossRef]
34. Jiang, X.Q.; Hai, H.; Wang, H.M. Constructing large girth QC protograph LDPC codes based on PSD-PEG algorithm. *IEEE Access* **2017**, *5*, 13489–13500. [CrossRef]
35. Bai, Z.L.; Wang, X.Y.; Yang, S.S. High-efficiency Gaussian key reconciliation in continuous variable quantum key distribution. *Sci. China Phys. Mech. Astron.* **2016**, *59*, 614201. [CrossRef]
36. Chang, T.C.Y.; Su, Y.T. Dynamic weighted bit-flipping decoding algorithm for LDPC codes. *IEEE Trans. Commun.* **2015**, *63*, 3950–3963. [CrossRef]
37. Bazzi, L.; Audah, H. Impact of redundant checks on the LP decoding thresholds of LDPC codes. *IEEE Trans. Inf. Theory* **2015**, *61*, 2240–2255. [CrossRef]