Hindawi
Mobile Information Systems
Volume 2021, Article ID 5548091, 16 pages
https://doi.org/10.1155/2021/5548091

Research Article

Node-Subset Scheduling with a Subset-Reset Mechanism for the Decoding Algorithm of Nonbinary LDPC Codes

Wentao Fu,[1] **Xilun Luo**,[2] **Yuanfa Ji**,[1,2] **and Xiyan Sun**[1]

[1]*National & Local Joint Engineering Research Centre of Satellite Navigation and Location Service, Guilin University of Electronic Technology, Guilin 541004, China*
[2]*College of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China*

Correspondence should be addressed to Yuanfa Ji; jiyuanfa@163.com

Received 9 January 2021; Revised 24 March 2021; Accepted 22 April 2021; Published 10 May 2021

Academic Editor: Quanzhong Li

For the conventional extended min-sum (EMS) algorithm, all check nodes update their check-to-variable (C2V) messages in every iteration. Selected scheduling, which reduces the number of check nodes for message updating in one iteration, can effectively reduce the complexity of the decoding algorithm, but it also lead to some performance degradation. With the introduction of a metric based on node stability, we propose stability-based node-subset scheduling (SNS) for the EMS algorithm, which can improve the performance of node-subset scheduling (NS). Second, to further improve the decoding performance of SNS while maintaining low complexity, we propose the SNS-EMS algorithm with a subset-reset mechanism (RSNS-EMS) based on the abnormal stability found in the processing node subset, which will cause the estimated codeword to fail to converge. The RSNS-EMS algorithm enhances performance through a sliding window detection and reset mechanism, and it resets the elements in the processing node subset to force all check nodes to update new messages when abnormal stability is detected. The simulation results show that the proposed algorithm can reduce complexity by approximately 25% with negligible performance degradation.

1. Introduction

Low-density parity-check (LDPC) codes [1] have been shown to achieve performance close to the Shannon limit when using the iterative belief-propagation (BP) algorithm for very long code lengths [2, 3]. LDPC codes have attracted much attention, and novel types of LDPC codes are constantly being proposed, such as root-photograph LDPC codes [4], which were proposed for block-fading channels and were also used to design bit-interleaved coded modulation (BICM) with iterative demapping and decoding under the constraint of limited bandwidth for wireless communication [5]. Moreover, for medium and short code lengths, nonbinary LDPC (NB-LDPC) codes have shown better decoding performance than their binary counterparts because they are able to avoid error floor problems [6]. NB-LDPC codes were originally proposed by Davey and Mackay [7], who considered codes defined in the finite field and correspondingly proposed an implementation of the BP algorithm for these codes, usually referred to as the Q-ary sum-product algorithm (QSPA). The computational complexity of the direct implementation of the QSPA is too high, which makes NB-LDPC codes difficult to apply in practice. To reduce the complexity of the decoding algorithm for NB-LDPC codes, the extended min-sum (EMS) algorithm [8] and its variant version [9] have been proposed. In these algorithms, the calculation amount of check nodes is reduced by using the truncation rule, which can reduce the length of the message vector from field size q to a specific number less than q.

In addition, for an iterative algorithm like the BP algorithm, its convergence rate is crucial for its implementation. Recently, different scheduling strategies have been proposed for BP decoders of binary LDPC codes. The conventional BP algorithm has been implemented with a standard flooding scheduling strategy [1], in which all variable nodes and, subsequently, all check nodes update the message and propagate new messages to their neighbors in

one iteration. Shuffled [10–12] and layered [13–15] scheduling are two commonly used sequential scheduling strategies to accelerate the convergence speed of flooding scheduling.

However, the order of message updating is fixed in these algorithms, which limits the acceleration of convergence. To further speed up convergence and improve the performance of LDPC codes, many informed dynamic strategies (IDSs) have been proposed. Residual belief-propagation decoding [16] is the represented IDS; it prioritizes message updates dynamically based on a metric called a residual. Although the RBP algorithm can achieve faster convergence speed, it is shown to be a greedy algorithm in which some edges and nodes will not be updated throughout the entire iteration process. Such greediness will prevent the RBP algorithm from converging to a low error rate. To solve this problem, different algorithms have recently been proposed, such as variable node-based dynamic scheduling [17], dynamic scheduling based on tabu search [18], and residual-decaying-based residual belief propagation [19].

Another subset of scheduling strategies is aimed at reducing the iteration cost of the LDPC decoder. The lazy scheduling strategy [20] was proposed to decide whether a check node should be updated at a given iteration based on its reliability and its update history. The selective-update decoding algorithm [21] uses the stability of variable nodes, in which the outgoing messages are similar or not similar for the last several iterations, to determine which check nodes do not need to be updated. When all variable nodes connected to a given check node are stable and reliable, it is not useful to update them again. Check node reliability-based scheduling [15] led to the creation of a concept called check node reliability, which is associated with the a posteriori probability of the existence of neighboring variable nodes, to determine whether the check node should be updated in the next iteration. Check node reliability-based scheduling also employed a random variable to overcome the high false reliability of the check nodes. Node-subset scheduling for the EMS (NS-EMS) decoding algorithm [22] uses the largest a posteriori probability among all tentatively estimated symbols as the metric to determine whether a variable node is reliable. Then, the check node is divided into two subsets—the processing subset and nonprocessing subset—according to the number of unreliable variable nodes connected to the check nodes. This method can effectively reduce computational complexity.

In conclusion, among these scheduling strategies, the most important issue that has a significant impact on the performance and computational complexity of the decoding algorithm is to decide which check nodes should be updated. Therefore, we have carried out research on this issue, and the main contributions of this paper are twofold. First, a scheme called stability-based node-subset scheduling (SNS) based on node-subset (NS) scheduling for the EMS algorithm is proposed to improve the error-correcting performance of the NS-EMS algorithm. In this algorithm, a metric based on node stability is introduced to determine whether a variable node is reliable, and the threshold is heuristically defined to match the iteration and communication channels. Second, to further improve performance while still maintaining low computational complexity, we studied the behavior of the processing subset of error frames in the SNS-EMS algorithm and found abnormal stability in the processing subset of error frames, which would lead to the stagnation of the information transfer between check nodes and variable nodes. When the information transfer between these two nodes gradually stagnates, iterative decoding will no longer play its own role; thus, the errors in the estimated codeword will not be able to be corrected. Then, according to this property of error frames, stability-based node-subset scheduling with a subset-reset mechanism (RSNS) is proposed. The kernel of the RSNS-EMS algorithm is a form of sliding window detection for the processing subset. When it detects that there is such abnormal stability in the decoding process, the elements in the processing subset are reset so that all check nodes will update messages in the next iteration. Hence, abnormal stability will be alleviated, and performance will be improved. The simulation results demonstrate that the proposed algorithm can achieve better decoding performance at lower computational complexity compared to its counterpart.

The rest of this paper is organized as follows. Section 2 gives a brief introduction of the extended min-sum (EMS) algorithm and node-subset scheduling. Section 3 introduces the proposed algorithms. The performance evaluation of various algorithms is demonstrated and discussed in Section 4. Finally, Section 5 concludes the paper.

2. Preliminary Works

2.1. Normal Graph of Nonbinary LDPC Codes. Let F_q be the finite field with $q\,(q = 2^{\ell})$ elements. A nonbinary LDPC code $C_q = [n, k]$ over F_q can be defined as the zero space of its parity check matrix $H = [h_{i,j}]_{m\times n}$ with $h_{i,j} \in F_q$. Message vector $v = (v_0, v_1, \ldots, v_{n-1})$ is a legal codeword of this NB-LDPC code if and only if it satisfies $Hv^T = 0$. We define the two following index sets:

$$\mathcal{N}_i = j\colon 0 \le j \le n-1, \quad h_{i,j} \neq 0, \tag{1}$$

for each row i of H, and

$$\mathcal{M}_j = i\colon 0 \le i \le m-1, \quad h_{i,j} \neq 0, \tag{2}$$

for each column j of H.

For a given parity check matrix, we can use the normal graph shown in Figure 1 to describe the decoding progress of NB-LDPC nodes. In a normal graph, edges represent variables, and vertices represent constraints. There are three kinds of vertices in a normal graph: (1) the left vertex is called the variable node, which represents a column of H; (2) the right vertex is called the check node, which represents a row of H; and (3) the middle vertex is called the intermediate node, which represents the nonzero elements $h_{i,j} \neq 0$ in H, which bridge the check node and variable node; moreover, (4) all edges connected to the $j-$th variable node must take identical values, and all edges connected to the $i-$th check node must add up to zero.

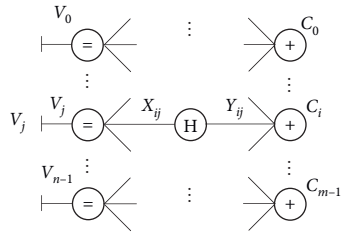

Figure 1: Normal graph for nonbinary LDPC codes.

2.2. M-EMS Algorithm. In this subsection, we will describe the specific steps of the M-EMS algorithm [5] combined with a normal graph, including initialization, message truncation, and the transfer and processing of information among variable nodes, check nodes and intermediate nodes.

2.2.1. Initialization. Consider an NB-LDPC code, $C_q = [n, k]$, defined over F_q, and $v = (v_0, v_1, \ldots, v_{n-1}) \in F_q^n$ is a codeword, where any of the codeword symbols v_i can be represented by a ℓ bit binary vector. For BPSK modulation, the codeword can be mapped to a bipolar sequence $x = (x_0^{(0)}, x_0^{(1)}, \ldots, x_0^{(\ell-1)}, \ldots, x_{n-1}^{(0)}, x_{n-1}^{(1)}, \ldots, x_{n-1}^{(\ell-1)})$, where $x_i^{(j)} = 1 - 2v_i^{(j)}$ for $0 \le i \le n-1, 0 \le j \le l-1$. Then, the sequence received over the AWGN channel is as follows:

$$y = \left(y_0^{(0)}, y_0^{(1)}, \ldots, y_0^{(\ell-1)}, \ldots, y_{n-1}^{(0)}, y_{n-1}^{(1)}, \ldots, y_{n-1}^{(\ell-1)}\right). \quad (3)$$

For a given received sequence y, the channel initialization vector can be calculated in the following manner. First, we calculate the real log-domain message as follows:

$$R_{V_j}(s) = \frac{1}{\ell} \sum_{k=0}^{\ell-1} y_i^{(k)} \left(1 - 2s^{(k)}\right), \quad s \in F_q, \quad (4)$$

where $s^{(k)}$ represents the $k-$th bit of a binary vector of finite field symbol s.

Let $\Delta > 0$ and $b > 1$ be the two parameters to be designed. Using these two parameters, we can clip and quantize the real log-domain message to an integer message according to the following equation:

$$L_{V_j}^{(|\longrightarrow V_j)}(s) = \begin{cases} -\left(2^b - 1\right), & \text{if } \dfrac{R_{V_j}(s)}{\Delta} \le -\left(2^b - 1\right), \\ \left[\dfrac{R_{V_j}(s)}{\Delta}\right], & \text{if } \left|\dfrac{R_{V_j}(s)}{\Delta}\right| < \left(2^b - 1\right), \\ \left(2^b - 1\right), & \text{if } \dfrac{R_{V_j}(s)}{\Delta} \ge \left(2^b - 1\right), \end{cases} \quad (5)$$

where $s \in F_q$ and $[\cdot]$ denotes a rounding operation, that is, taking the nearest integer to it.

2.2.2. Variable Node Update. In the iterative decoding process, variable node V_j receives message $L_{X_{ij}}^{(C_i \longrightarrow V_j)}, s \in F_q$ passed by check node C_i connected thereto and updates the a posteriori message according to

$$L_{V_j}(s) = L_{V_j}^{(|\longrightarrow V_j)}(s) + \sum_{i \in M_j} L_{X_{ij}}^{(C_i \longrightarrow V_j)}(s), \quad s \in F_q. \quad (6)$$

The variable-to-check (V2C) message that needs to be passed to the check nodes is calculated as follows:

$$L_{X_{ij}}^{(V_j \longrightarrow C_i)}(s) = L_{V_j}(s) - L_{X_{ij}}^{(C_i \longrightarrow V_j)}(s), \quad s \in F_q. \quad (7)$$

2.2.3. Check Node Update. We can use the forward-backward algorithm [23] based on the trellis diagram [9] to represent message processing at check nodes.

Let $\alpha_t = (\alpha_t(0), \alpha_t(1), \ldots, \alpha_t(q-1))$ and $\beta_t = (\beta_t(0), \beta_t(1), \ldots, \beta_t(q-1))$ denote the forward and backward recursion vectors, respectively.

Set $\alpha_0 = (0, -\infty, \ldots, -\infty)$, and let d_r denote the degree of the $i-$th check node. Then, for $0 \le t < d_r$ and $\forall s \in F_q$, compute the forward recursion vector and its truncated version:

$$\alpha_{t+1}(s) = \max_{z \in F_q} \left\{ \alpha_t(s - z) + L_{X_{ij}}^{(V_j \longrightarrow C_i)}(z) \right\}, \quad s \in F_q. \quad (8)$$

Set $\beta_{d_r} = (0, -\infty, \ldots, -\infty)$, and then, for $d_r \ge t > 1$ and $\forall s \in F_q$, recursively compute the backward recursion vectors and its truncated version:

$$\beta_{t-1}(s) = \max_{z \in F_q} \left\{ \beta_t(s - z) + L_{X_{ij}}^{(V_j \longrightarrow C_i)}(z) \right\}, \quad s \in F_q. \quad (9)$$

Next, for $0 \le t \le d_r - 1$ and $\forall s \in F_q$, compute the extrinsic message:

$$L_{Y_{ij}}^{(C_i \longrightarrow V_j)}(s) = \max_{a \in F_q} \{\alpha_t(a) + \beta_{t+1}(s + a)\}, \quad s \in F_q. \quad (10)$$

Finally, for $0 \le t \le d_r - 1$, postprocess the extrinsic message:

$$L_{Y_{ij}}^{(C_i \longrightarrow H_{ij})}(s) = \begin{cases} \left[\xi L_{Y_{ij}}^{(C_i \longrightarrow V_j)}(s)\right], & \text{if } L_{Y_{ij}}^{(C_i \longrightarrow H_{ij})}(s) \ne -\infty, \\ 0, & \text{others}, \end{cases} \quad (11)$$

where ξ is a scaling factor and $[\cdot]$ denotes a rounding operation.

2.2.4. Node-Subset Scheduling. Let $\left\{\widehat{v} | \widehat{v}_j^{(l)} \in F_q, j \in N_i\right\}$ be the set of estimated symbols of variable nodes which is connected to the $i-$th check node at the $l-$th iteration. The syndrome of this check node can be computed by

$$c_i^{(l)} = \sum_{j \in N_i} h_{i,j} \cdot \hat{v}_j^{(l)}. \tag{12}$$

For a check node, if its syndrome is nonzero, then it means that the tentatively estimated symbols of variable nodes connected to the check node has not converged to the correct symbols. Therefore, these check nodes need to be updated in the next iteration. When the syndrome of a check node is zero, it does not always mean that the variable nodes connected to it have converged, it may be caused by the existence of two or more error symbols.

To distinguish which kind of situation occurs when the syndrome is zero, node-subset (NS) scheduling [22] is based on the reliability of the variable node. If the reliability of the variable node is large enough, then the current decoding symbol is considered reliable. When the number of unreliable variable nodes connected to a check node with zero syndrome is less than two, the check node will be divided into a nonprocessing subset. These check nodes in the nonprocessing subset will not update their C2V messages in the next iteration.

NS scheduling can be described as follows. Let $f_j^{(l)}$ denote the mark of the $j-$th variable node at the $l-$th iteration. If the reliability of the current estimated symbol exceeds predefined threshold T_c, then this variable node is considered reliable and marked as 0; otherwise, it is marked as 1:

$$f_j^{(l)} = \begin{cases} 0, & \max\left\{L_{V_j}^{(l)}(z)\right\} \geq T_c \\ 1, & \max\left\{L_{V_j}^{(l)}(z)\right\} < T_c \end{cases}, \quad z \in F_q, \tag{13}$$

where $0 \leq j \leq n-1$.

Let $M^{(0)}$ be the set of all check nodes and let $M^{(l)} \subseteq M^{(0)}$ be the processing subset during the $l-$th decoding iteration. The check nodes contained in $M^{(l)}$ can be determined by the following equation:

$$M^{(l)} = \left\{C_i \middle| c_i^{(l)} \neq 0\right\} \cup \left\{C_i \middle| \left(\sum_{j \in N_i} f_j^{(l)} \geq 2 | c_i^{(l)} = 0\right)\right\}, \tag{14}$$

where $0 \leq i \leq m-1$ and $c_i^{(l)}$denotes the syndrome of the $i-$th check node at the $l-$th iteration.

3. Proposed Schemes

3.1. Metric for the Stability of Variable Nodes. In this subsection, we propose a metric based on the stability of the variable node to determine whether this node is reliable. First, we define stability metric $d(j)$ as follows:

$$d(j) = \max\left\{L_{V_j}(s)\right\} - \max^*\left\{L_{V_j}(s)\right\}, \quad s \in F_q, \tag{15}$$

where $\max^*$ denotes the second-largest value of the log-domain message of $L_{V_j}(s)$.

In the M-EMS algorithm [9], the value distribution of $L_{V_j}(s)$ is affected by decoding iteration and the channel environment. The current decoding symbol of a variable node will often change between the first few finite field symbols with the highest reliability value. In other words, in the$L_{V_j}(s)$ of a variable node, there will be several symbols with similar reliability values.

As shown in Figure 2, these two variable nodes have similar values of$\max\left\{L_{V_j}(s)\right\}$. However, there are several symbols with close values of $L_{V_j}(s)$ in variable node (b), so the estimated symbol of this node is more likely to change compared to variable node (a). However, when using $\max\left\{L_{V_j}(s)\right\}$ as the metric of reliability, both variable nodes should be considered reliable.

With the progress of decoding iteration, the variable nodes gradually converge to the correct symbol and no longer change, which means that they become stable. According to this property, when a variable node is not stable, then we consider it unreliable and in need of being updated in the next iteration.

Therefore, we use $d(j)$ as the metric of the reliability of a variable node. For a variable node, the larger its $d(j)$ value is, the greater the difference in $L_{V_j}(s)$ between the estimated decoding symbol and the second reliable symbol is; thus, it is less likely for the estimated symbol to change, which means that the variable node will become more stable.

The method proposed in this paper for determining which variable nodes are stable is described as follows. Let $m_j^{(l)}$ represent the mark of the $j-$th variable node at the $l-$th decoding iteration. If the $d(j)$ value of this variable node exceeds a designed threshold, T_d, then it is considered sufficiently reliable and marked as 0. Otherwise, the variable node is considered unreliable and marked as 1.

$$m_j^{(l)} = \begin{cases} 0, & d(j) \geq T_d \\ 1, & d(j) < T_d \end{cases}, \quad 0 \leq j \leq n-1. \tag{16}$$

For threshold T_d, we present a method that can adaptively calculate the threshold based on decoding iteration and channel variation:

$$T_d = \delta \cdot \frac{1}{n} \sum_{t=0}^{n-1} d(t), \tag{17}$$

where δ is a revision factor to be designed.

T_d depends on the $d(j)$ value of all variable nodes in one iteration, so it does not need to be redetermined due to the variation in the parity check matrix. Moreover, as the iterative decoding process proceeds, an increasing number of variable nodes will converge to the correct symbol and become stable. Therefore, we use the statistic mean of $d(j)$ for all variable nodes to determine this threshold, which can reflect the average stability level of the variable nodes in this iteration. The variable nodes with $d(j)$ values under threshold T_d will be considered unreliable. The improved stability-based node-subset schedule for the EMS algorithm (SNS-EMS) can be described by pseudocode, as shown in Algorithm 1.

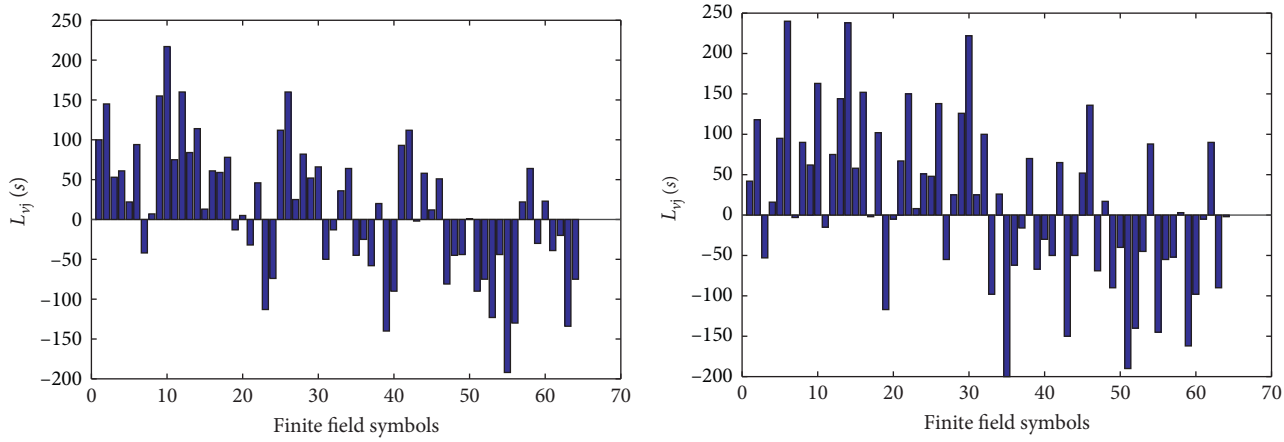

Figure 2: Distribution of $L_{V_j}(s)$ of two variable nodes in the same frame decoding process at one iteration.

(1) for the given relevant parameters ξ, b, Δ, initialize all $L_{V_j}^{(|\longrightarrow V_j)}(s)$, and set $L_{X_{ij}}^{(C_i\longrightarrow V_j)}(s) = 0$
(2) set $\text{iter}_{\max}$, and put all check nodes into $M^{(l)}$
(3) **for** $i = 0$ to $\text{iter}_{\max}$ **do**
(4) update the check nodes in $M^{(l)}$, compute $L_{X_{ij}}^{(C_i\longrightarrow V_j)}(s)$
(5) compute $L_{V_j}(s)$, and update V2C message $L_{X_{ij}}^{(V_j\longrightarrow C_i)}(s)$ for all variable nodes
(6) **for** $j = 0$ to $j = n - 1$, **do**
(7) compute the estimated symbol
$\widehat{v}_j = \text{argmax}_{s\in F_q}\left\{L_{V_j}(s)\right\}$
(8) **end for**
(9) **for** all variable nodes, **do**
(10) mark $m_j^{(l)}$ according to equation (15)
(11) **end for**
(12) divide check nodes into two node subsets according to equation (17)
(13) **if** $H\widehat{v}^T = 0$ is not satisfied, **then**
(14) go back to line 3
(15) **else** return estimated codeword
(16) **end if**
(17) **end for**
(18) return estimated codeword

Algorithm 1: SNS-EMS.

3.2. SNS Scheduling Combined with a Subset-Reset Mechanism. In the previous section, we described the proposed SNS-EMS algorithm. According to the simulation results, the SNS-EMS algorithm can make a good tradeoff between decoding performance and computational complexity. However, compared with the original M-EMS algorithm, its error correcting performance has a certain degree of degradation.

Since SNS scheduling mainly affects which check node will update its C2V message in iterative decoding, we take the processing subset as the point of penetration and study the variation in the processing subset in the iterative decoding process of the SNS-EMS algorithm. Note that, in all experiments in this subsection, the $C_{64}(88, 44)$ NB-LDPC code of rate 0.5 is used, and the maximum number of iterations is set to 50.

In the first experiment, we track the variation in the number of processing nodes for both converged frames and error frames at SNRs of 1.0 dB, 1.4 dB, 1.6 dB, 1.8 dB, 2.2 dB, and 2.4 dB, and their collected error frames total 50, 30, 30, 20, 10, and 10, respectively. Note that when a frame converges, we set the number of processing nodes in the subsequent iterations to 0.

Figure 3 shows the average number (over frames) of processing nodes in the iterative process of converged frames and error frames of the SNS-EMS algorithm under different SNRs. We can see that there is a clear difference in the variation curves between converged frames and error frames. The number of processing nodes of converged frames decreases greatly and always shows a downward trend until dropping to a very low level, which means that this frame has converged. Moreover, as the SNR continues to increase, the

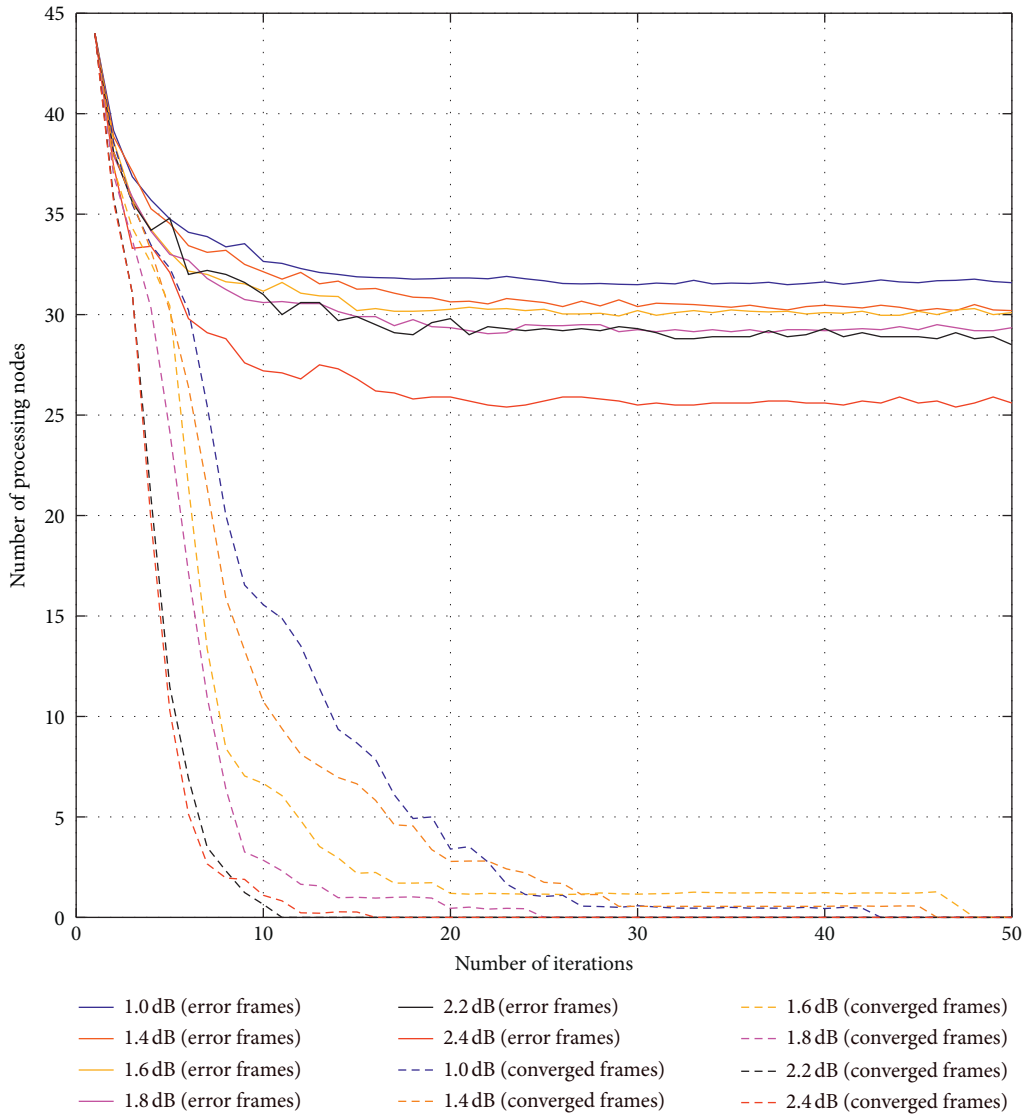

Figure 3: Average number of processing nodes of error frames and converged frames in the SNS-EMS algorithm with iterations under different SNRs.

convergence speed also increases. When the SNR is greater than 1.8 dB, the number of processing nodes can be reduced to a very low level after the first several iterations.

Conversely, the number of processing nodes in error frames exhibits a gentle downward trend. The number of processing nodes will no longer continue to decline after approximately 10 iterations but will remain at a high value. At this time, the number of processing nodes shows a stable state, where only a small fluctuation occurs.

Based on the above experiment, we further studied the changes in elements in the processing subset of error frames. Let S_c and S_l denote the processing subset of the current iteration and that of the next iteration, respectively, and let S_i denote the intersection of S_c and S_l, which contains the same elements in these two sets. Consequently, the numbers of elements in S_i and S_l are u_i and u_l, respectively. We use the ratio of u_i to u_l as the metric, which is referred to as repeatability ratio ρ_R, to intuitively reflect the changes in the elements in the processing node subset. When ρ_R is close to 1, it means the elements in the processing subset are almost unchanged in two successive iterations.

$$\rho_R = \frac{u_i}{u_l}. \tag{18}$$

In this experiment, we track the variation in the repeatability ratio at SNRs of 1.0 dB, 1.4 dB, 1.6 dB, 1.8 dB, 2.2 dB, and 2.4 dB, and their collected error frames total 50, 30, 30, 20, 10, and 10, respectively.

As shown in Figure 4, all the ρ_R values of the processing subset of error frames under different SNRs are at a very high level, and as the iteration continues, the ratio gradually increases. After 20 iterations, ρ_R under all SNRs is higher than 0.98, which means that the elements in the processing subset will hardly change in the subsequent iterations. These results combined with those in Figure 3 show that when the

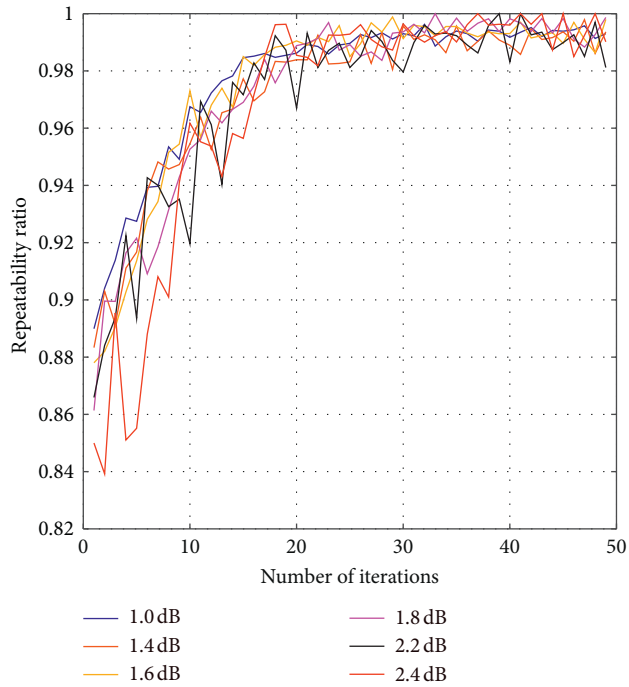

Figure 4: Average variation in the ρ_R values of error frames in SNS-EMS algorithm at different SNRs.

number of processing nodes of the error frames tends to be stable, almost the same check nodes update their C2V messages. This situation continues until the decoder reaches the maximum number of iterations and stops decoding, but the estimated code has not converged.

According to the above experiments, we speculate that, after dividing the node subset, some check nodes stop updating their C2V messages. Then, some of the V2C messages of variable nodes cannot be updated correspondingly, according to equations (6) and (7). As a result, $L_{V_j}(s)$ will no longer be updated, which is used to tentatively decide the estimated symbols for variable nodes. Therefore, even though there are many check nodes for C2V message updating in the processing node subset, the errors in the estimated codeword still cannot be corrected.

To verify the correctness of the above speculations, we conduct research and analysis on the vector $L_{V_j}(s)$ of variable nodes during the iteration process. Since the calculation of these vectors is related to both C2V and V2C messages, the changes in this message can reflect the changes in these two messages. We start by defining the vector $H_{V_j}^{(l)}(s)$ as follows:

$$H_{V_j}^{(l)}(s) = \left| L_{V_j}^{(l)}(s) - L_{V_j}^{(l-1)}(s) \right|, \quad s \in F_q, \tag{19}$$

where j and l are the $j-$th variable node and the $l-$th iteration, respectively, and $|\cdot|$ is the absolute value. The elements in $H_{V_j}^{(l)}(s)$ denote the magnitude of log-domain message $L_{V_j}^{(l)}(s)$ for each finite field symbol between two consecutive iterations.

Then, for each variable node, a metric, $e_{v_j}^{(l)}$, is stored to trace how many finite-field symbols in $H_{V_j}^{(l)}(s)$ vary more than variation threshold T_{var}. $e_{v_j}^{(l)}$ can approximately reflect the change in vector $L_{V_j}^{(l)}(s)$ in the iteration process: if the value of $e_{v_j}^{(l)}$ is larger, then there are a greater number of symbols whose log-domain messages have changed in $L_{V_j}^{(l)}(s)$.

We take the arithmetic mean of the $e_{v_j}^{(l)}$ values of all the variable nodes in the $l-$th iteration as $\varphi^{(l)}$ and use it to show the overall situation of $L_{V_j}^{(l)}(s)$ for all variable nodes as decoding iterations progress. It is worth noting that there are two situations where $L_{V_j}^{(l)}(s)$ will not change: (1) when the estimated codeword converges and the decoder stops decoding, the calculation of $L_{V_j}^{(l)}(s)$ will stop accordingly, and (2) when C2V messages stop being updated, the $L_{V_j}^{(l)}(s)$ value will not change. To distinguish these two cases, we set the $L_{V_j}^{(l)}(s)$ values of converged frames to q after they have converged, and q denotes the order of finite field F_q.

The experimental results are shown in Figure 5, where $T_{\text{var}} = 3$. It is worth noting that the $\varphi^{(l)}$ value of error frames gradually decreases as the number of iterations increases, especially after 20 iterations, and the value of $\varphi^{(l)}$ remains at a very low level under all SNRs. This finding means that, during these iterations, $L_{V_j}(s)$ hardly changes. Combined with the above experiments, we know that after "abnormal" stability appears in the processing subset, although there are still a large number of check nodes updating their C2V messages in the subsequent iterative decoding, it does not help with error correction. Since the information transfer between check nodes and variable nodes no longer generates new correction information, $L_{V_j}(s)$, which is used for deciding the estimated symbol of variable nodes, hardly changes. Therefore, the errors in the estimated codeword cannot be corrected, and thus, it will fail to converge to the correct codeword.

The proposed stability-based node-subset scheduling with a subset-reset mechanism for EMS (RSNS-EMS)

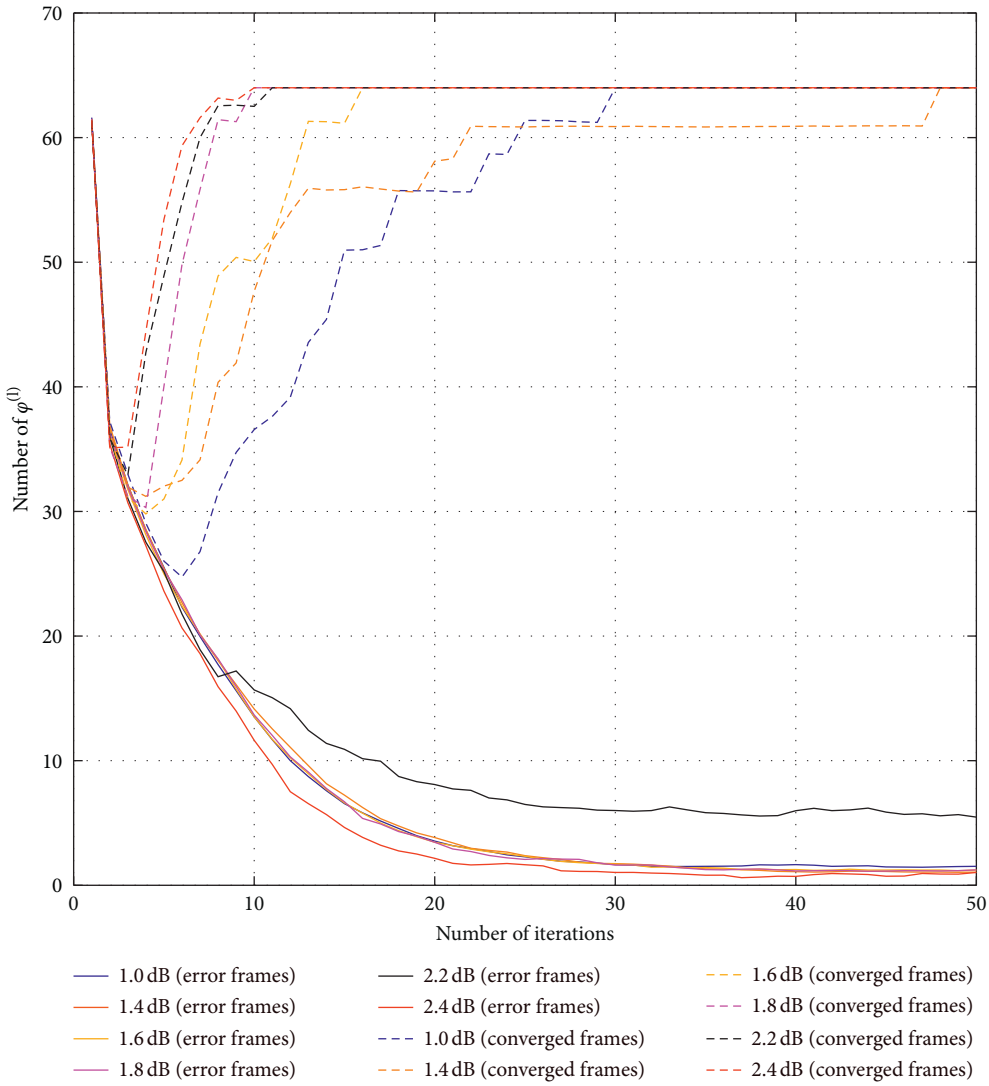

FIGURE 5: Variation in the $\varphi^{(l)}$ values of error frames and converged frames in the SNS-EMS algorithm with iteration processing at different SNRs.

(1) for the given relevant parameters ξ, b, Δ, initialize all $L_{V_j}^{(|\longrightarrow V_j)}(s)$, and set $L_{X_{ij}}^{(C_i \longrightarrow V_j)}(s) = 0$
(2) set $\text{iter}_{\max}$, and put all check nodes into $M^{(l)}$
(3) **for** $i = 0$ to $\text{iter}_{\max}$ **do**
(4) update the check nodes in $M^{(l)}$, and compute $L_{X_{ij}}^{(C_i \longrightarrow V_j)}(s)$
(5) compute the $L_{V_j}(s)$, and update V2C message $L_{X_{ij}}^{(V_j \longrightarrow C_i)}(s)$
(6) **for** $j = 0$ to $j = n - 1$, **do**
(7) compute the estimated symbol
$$\widehat{v}_j = \text{argmax}_{s \in F_q}\left\{L_{V_j}(s)\right\}$$
(8) **end for**
(9) **for** all variable nodes, **do**
(10) mark $m_j^{(l)}$ according to equation (15)
(11) **end for**
(12) divide check nodes into two node subsets according to equation (15)
(13) collect the n_i values of the last ω iterations
(14) **if** current iteration $T_c > T_{\text{iter}}$, **then**
(15) compute the d_ω and D

ALGORITHM 2: Continued.

(16) **if** $D < \mu$, **then**
(17) reset the processing subset, and put all check nodes into $M^{(l)}$
(18) **end if**
(19) **end if**
(20) **if** $H\hat{v}^T = 0$ is not satisfied, **then**
(21) go back to line 4
(22) **else** return estimated codeword
(23) **end if**
(24) **end for**
(25) return estimated codeword

Algorithm 2: RSNS-EMS.

Table 1: Analysis of computational complexity per iteration for various algorithms.

	SU-EMS [21]	NS-EMS [22]	SNS-EMS (ours)	RSNS-EMS (ours)	Reduced operations
FA	0	0	0	0	$3d_r\sigma q^2$
RA	$d_rM + Nq$	d_rM	$d_rM + 2N$	$d_rM + 2N + 2\omega$	$3d_r\sigma q^2$
RM	0	0	2	3	0
CM	$M + N(q+1)$	$M + N$	$M + N$	$M + N$	0
Max	0	0	N	N	$3d_r\sigma q^2$

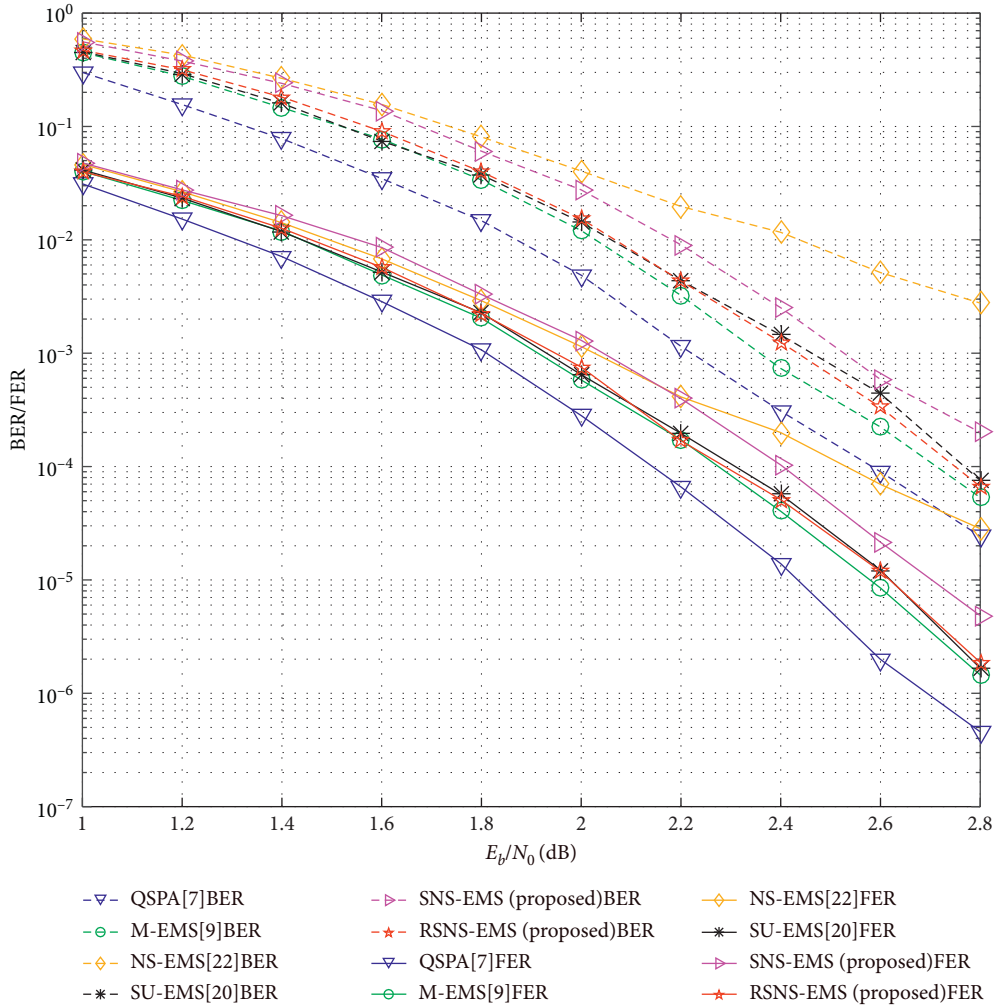

Figure 6: Decoding performance of BER and FER versus E_b/N_0 of various algorithms for NB-LDPC code $C_{64}(88, 44)$.

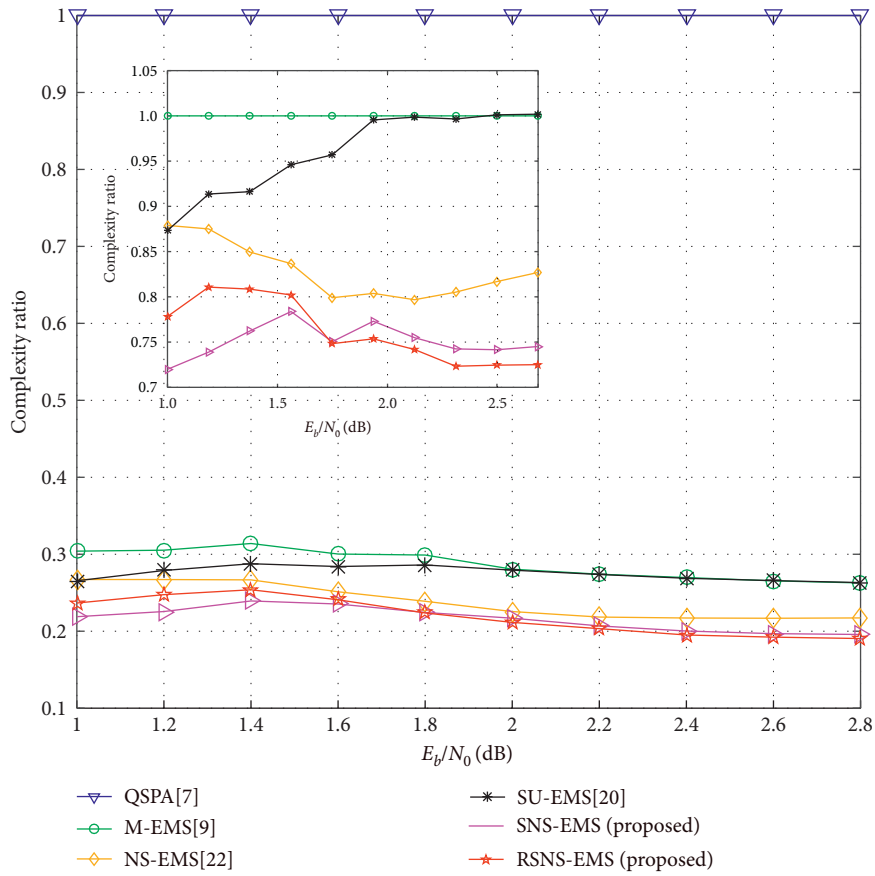

Figure 7: Complexity ratio of various algorithms compared to the QSPA versus E_b/N_0 for NB-LDPC code $C_{64}(88, 44)$.

algorithm has been developed based on this "abnormal" stability of the processing subset. The main idea of this method is to distinguish the frames with this property during the decoding process. Then, the elements in the processing subset are reset so that all check nodes will update their C2V messages in the next iteration. When new C2V messages are updated, it is expected that the abnormal stability of the processing subset will be alleviated and that the message transfer between the two nodes will be accelerated. Therefore, the estimated codeword can converge, thereby mitigating the degradation of the error-correcting performance of the SNS-EMS algorithm. Note that since only a small number of error frames will be reset and the FER will gradually decrease as the number of SNRs increases, we can expect that this method will not significantly increase overall computational complexity.

To implement the previously described method, we design a sliding window detection mechanism. First, because in the first few iterations the number of processing nodes for both error frames and converged frames exhibits a downward trend, we set a threshold of iteration T_{iter}, and the sliding window detection mechanism will only be activated when the number of iterations is greater than T_{iter}. To trace the abnormal stability behavior of the decoding frame, a sliding window mechanism of width ω can be defined as follows: let n_i denote the number of processing nodes in each iteration, where $i = 1, 2, \ldots,$ max iter. We collect n_i for the last ω iterations and define vector d_ω as follows:

$$d_\omega = \left(n_{c-\omega} - n_{c-\omega-1}, n_{c-\omega+1} - n_{c-\omega}, \ldots, n_c - n_{c-1}\right), \tag{20}$$

where n_c denotes the number of processing nodes for the current iteration. The elements in d_ω represent the difference in n_i between two consecutive iterations.

Finally, we calculate the arithmetic mean D of the value in d_ω, which denotes the average value of the difference in the number of processing check nodes within ω successive iterations:

$$D = \frac{1}{\omega} \sum_{k=0}^{\omega} \left(n_{c-k} - n_{c-k-1}\right). \tag{21}$$

We use it as the measurement to determine whether the processing subset has abnormal stability. When the value of D is less than the designed variation tolerance μ, the processing subset of this frame has abnormal stability. Then, the elements of the processing subset are reset, and all check nodes update their C2V messages in the next iteration. It should be noted that the interval within the resetting of these two subsets needs to be greater than window width ω. The proposed RSNS-EMS algorithm can be described by pseudocode, as shown in Algorithm 2.

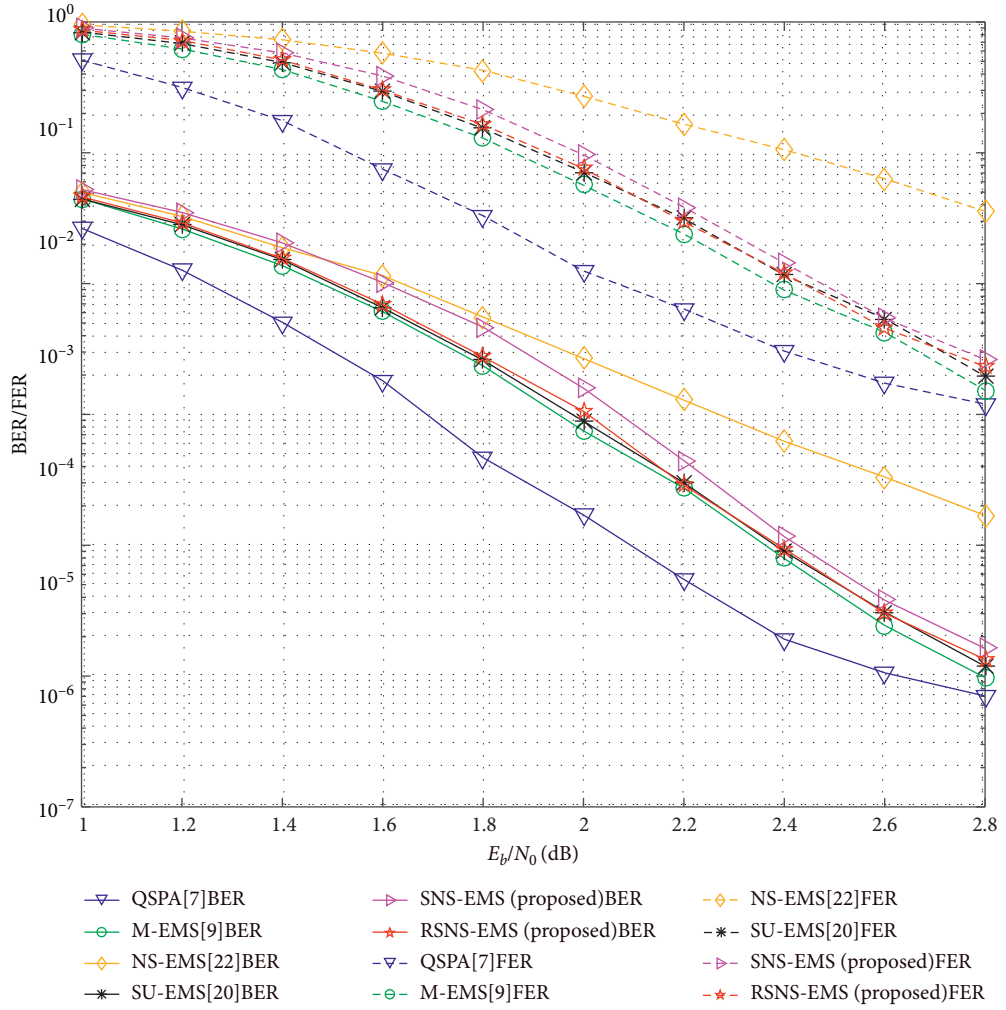

FIGURE 8: Decoding performance of BER and FER versus E_b/N_0 of various algorithms for NB-LDPC code $C_{32}(200, 100)$.

4. Performance Evaluation

In this section, we will evaluate the performances of the proposed SNS-EMS and RSNS-EMS algorithms in terms of computational complexity and error-correcting performance through a large number of simulations and compare them with other algorithms, including the QSPA [7], M-EMS [7], SU-EMS [21], and NS-EMS [22] algorithms. In this evaluation, to verify the reliability of the proposed method, we select three NB-LDPC codes with different finite fields and construction methods.

The first code, $C_{64}(88, 44)$, is a regular NB-LDPC code over F_{64} with a rate of 0.5, which has been designed using optimized rows and optimized short loops according to [24, 25]. The second code, $C_{32}(200, 100)$, is a regular NB-LDPC code over F_{32} and it has been constructed by using the progress-edge-grow (PEG) algorithm [26]. The third code, $C_{256}(72, 36)$, is a regular NB-LDPC code over F_{256} with a rate of 0.5, which has been constructed using the same method as the first code. All the simulations are performed at BPSK modulation over the AWGN channel. The maximum number of iterations is set to 30.

4.1. Complexity Analysis. In this subsection, we will analyze the complexity of the proposed algorithms. The computational complexity of the decoding algorithm of NB-LDPC codes is mainly concentrated on the part of check node update. The check node update can use forward-backward algorithms [23] on the trellis, so computational complexity is determined by the total number of live branches in the trellis. Thus, we will count the total number of live branches of different algorithms on the trellis. Conversely, different decoding algorithms may have different convergence speeds. Therefore, we will find the statistical mean (averaging over frames) of the total number of live branches involved in all the iterations for the decoding of one frame. Because other algorithms are used to reduce the computational complexity of the QSPA, we take the following complexity ratio as the measurement of complexity:

$$\rho = \frac{\text{total number of live branches of a given algorithm}}{\text{total number of live branches of the QSPA}}, \tag{22}$$

which can approximately reflect the reduction of complexity compared to the QSPA.

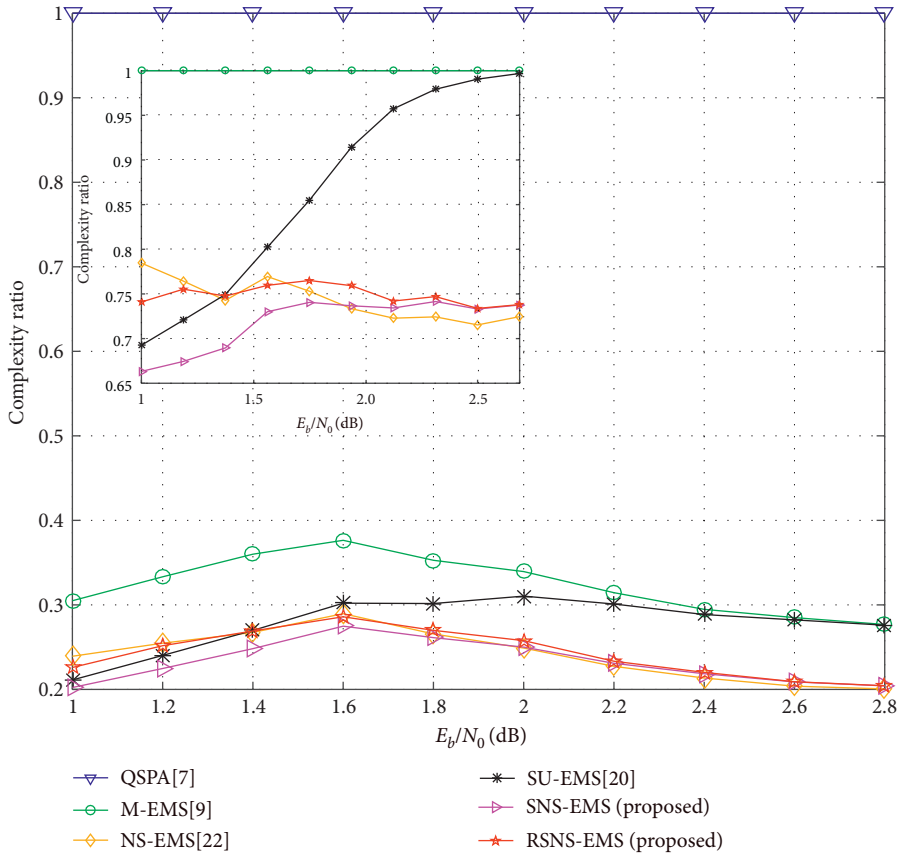

Figure 9: Complexity ratio of various algorithms compared to the QSPA versus E_b/N_0 for NB-LDPC code $C_{32}(200, 100)$.

In addition, the abovementioned complexity ratio reflects the complexity of each algorithm approximately but intuitively. We think that it is necessary to compare the computational complexity of the M-EMS algorithm and the QSPA more accurately, since all selected scheduling strategies are used in the M-EMS algorithm. The computation of messages in check node updates can use the method proposed in [27]. According to this paper, the complexity of check node updates in one iteration varies on the order of $\mathcal{O}(Md_c n_m \log_2(n_m))$ and with $n_m \ll q$, which is a great computational reduction compared to the QSPA of $\mathcal{O}(Md_c q^2)$.

Regarding the introduction of various mechanisms for reducing complexity in various selected scheduling strategies, some extra operations have also been introduced. Thus, we will also analyze these newly introduced operations. According to the relevant steps of these algorithms, we divide the operations into five categories: finite-field addition (FA), real addition (RA), real multiplication (RM), comparison (CM), and max operation (Max).

The extra operations introduced by different algorithms compared to the M-EMS algorithm in one iteration are shown in Table 1. For the NS-EMS algorithm, the newly introduced operation determines whether the variable node is reliable and divides the node subsets. These two steps require $d_r M$ RAs, $M + N$ CMs, and N Maxs. Compared to the NS-EMS algorithm, the SNS-EMS algorithm additionally needs to compute threshold T_d and find the estimated symbol with the second-largest log-domain message in $L_{V_j}(s)$. Thus, the extra operations required by the SNS-EMS algorithm are $d_r M + 2N$ RAs, 2 RMs, $M + N$ CMs, and N Maxs. Due to the introduction of abnormal stability detection in the RSNS-EMS algorithm, the required extra operations are $d_r M + 2N + 2\omega$ RAs, 3 RMs, $M + N$ CMs, and N Maxs. The SU-EMS algorithm needs to find the difference in each finite-field symbol in $L_{V_j}(s)$ of two consecutive iterations and determine whether it is greater than threshold ε. Therefore, the extra operations introduced by the SU-EMS algorithm are $d_r M + Nq$ RAs and $M + N(q + 1)$ CMs.

Since these scheduling strategies can reduce the number of check nodes updated in one iteration, we also list the reduction in complexity in Table 1. Because in each iteration of the QSPA and the M-EMS algorithm the number of processing check nodes is a fixed value M, and the calculation for the C2V messages of each iteration requires $3d_r \sigma q^2$ FAs, $3d_r \sigma q^2$ Ras, and $3d_r \sigma q^2$ Maxs in forward-backward recursion. All algorithms listed in Table 1 reduce the complexity by reducing the number of processing check nodes in one iteration. Therefore, in each iteration, the reduced operations are the corresponding operations of these nonprocessing check nodes. Let σ denote the number of nonprocessing check nodes in one iteration for various algorithms. Then, the reduction in complexity is $3d_r r \sigma q^2$ FAs, RAs, and Maxs.

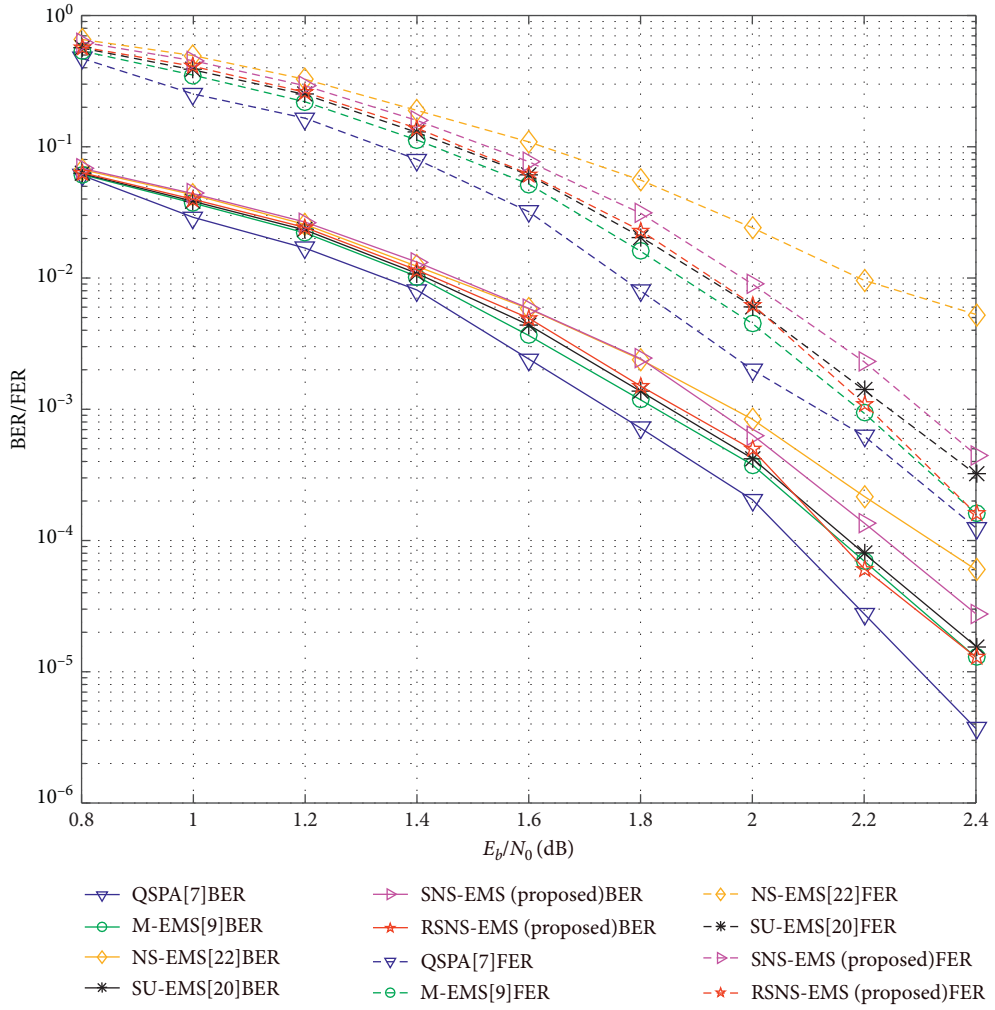

FIGURE 10: Decoding performance of BER and FER versus E_b/N_0 of various algorithms for NB-LDPC code $C_{256}(72, 36)$.

We can notice that the additional complexity is linear and that the reduced complexity is proportional to the square of finite-field order q. With regard to the case of short and medium code lengths, where NB-LDPC codes have more advantages compared to binary LDPC codes, the additional operations caused by the introduction of a new mechanism are much smaller than the reduced computational complexity.

4.2. Simulation Results. Error-correcting performance is measured by the bit error rate (BER) and frame error rate (FER). First, we consider a regular NB-LDPC code $C_{64}(88, 44)$ that has been designed using optimized rows and optimized short loops according to [24, 25]. The parity-check matrix of this code has row weight $d_r = 4$ and column weight $d_c = 2$. The parameters of the implemented algorithms are listed below.

(1) For the M-EMS algorithm, $M = 32$; for the NS-EMS algorithm, $M = 32, T_c = 220$; for the SU-EMS algorithm,$M = 32, \omega = 3, \varepsilon = 5$; for the SNS-EMS algorithm, $M = 32$, $\delta = 1$; and for the RSNS-EMS algorithm, $M = 32, \delta = 1$, $T_{\text{iter}} = 10, \omega = 4, \mu = 3$. (2) For all algorithms, the scaling factor $\xi = 0.9$ and $b = 8, \Delta = 1/128$.

The simulation results are shown in Figures 6 and 7. Figure 6 shows the BER and FER performance of various algorithms, where the dashed lines represent the FER, and the solid lines represent the BER. Figure 7 shows the complexity ratio of various algorithms compared to the QSPA, and the subwindow shows the complexity ratio compared to the M-EMS algorithm.

The optimal QSPA has the best error-correcting performance, and the EMS algorithm has the second-best performance. We can see that the computational complexity of the other algorithms is much lower than that of the QSPA, almost reducing the complexity by more than 70%. Due to the introduction of the truncation rule, the M-EMS algorithm reduces complexity and brings about some performance degradation. For the proposed algorithms, we can see that there is negligible (nearly no) performance

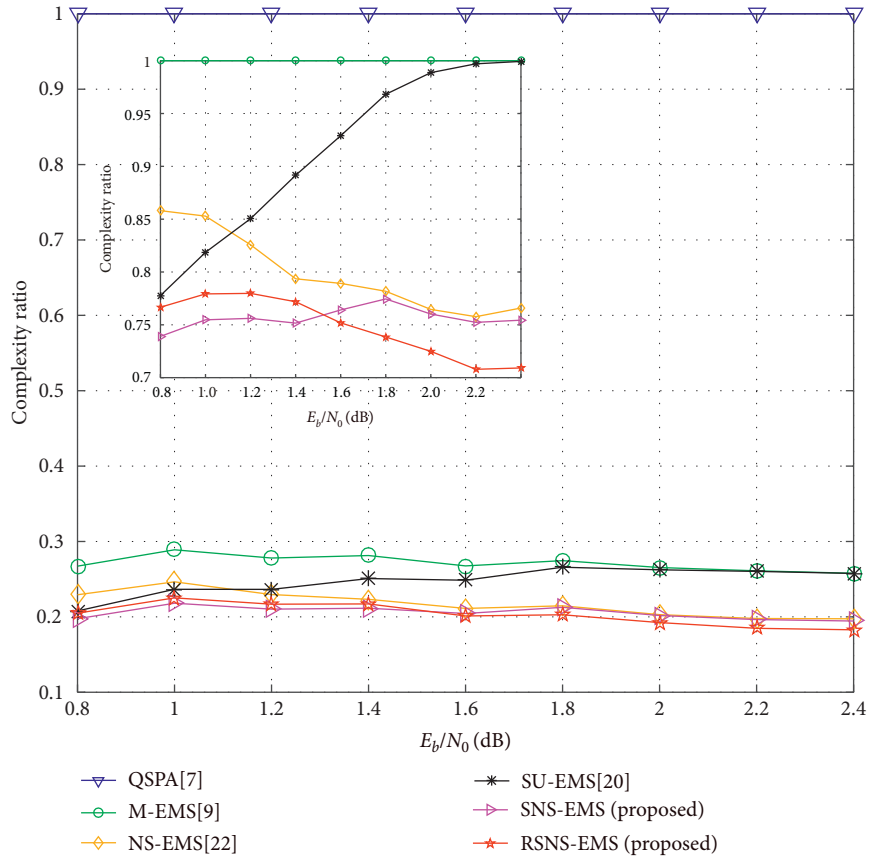

Figure 11: Complexity ratio of various algorithms compared to the QSPA versus E_b/N_0 for NB-LDPC code $C_{256}(72, 36)$.

degradation between the proposed RSNS-EMS algorithm and the original M-EMS algorithm. Moreover, it is obvious that the proposed RSNS-EMS algorithm can outperform the NS-EMS and SNS-EMS algorithms. When the SNR is less than 1.8 dB, the complexity of the RSNS-EMS algorithm is slightly higher than that of the SNS-EMS algorithm, and when the SNR is greater than 1.8 dB, the complexity of the RSNS-EMS algorithm is even lower than that of the SNS-EMS algorithm. Compared to the SU-EMS algorithm, the error-correcting performance of the proposed RSNS-EMS algorithm is basically the same, and the complexity of the RSNS-EMS algorithm is much lower than that of the SU-EMS algorithm, which increases with increasing SNR and is basically the same as that of the M-EMS algorithm when the SNR is greater than 2.0 dB.

Second, we consider a regular NB-LDPC code, $C_{32}(200, 100)$. The parity-check matrix of this code has row weight $d_r = 4$ and column weight $d_c = 2$. The parameters of the implemented algorithms are listed below.

(1) For the M-EMS algorithm,$M = 16$; for the NS-EMS algorithm, $M = 16, T_c = 230$; for the SU-EMS algorithm, $M = 16, \omega = 3, \varepsilon = 5$; for the SNS-EMS algorithm, $M = 16, \delta = 1$; and for the RSNS-EMS algorithm, $M = 16, \delta = 1, T_{\text{iter}} = 10, \omega = 4, \mu = 3$. (2) For all algorithms, the scaling factor $\xi = 0.9$ and $b = 8, \Delta = 1/128$.

The simulation results are shown in Figures 8 and 9. It can be observed that the QSPA has the best error-correcting performance among all algorithms, but it also has the highest computational complexity. The performance of the EMS algorithm is second-best, and it can reduce complexity by nearly 70% with approximately 0.2 dB degradation. Although the proposed SNS-EMS algorithm has the overall lowest complexity, its performance has a certain degree of degradation compared to the M-EMS algorithm. The proposed RSNS-EMS algorithm can outperform the NS-EMS and NS-EMS algorithms; it has negligible (nearly no BER) degradation with an approximately 25% reduction in complexity compared to the M-EMS algorithm. It is worth noting that the complexity ratio of the SU-EMS algorithm starts from 0.7 and increases to be almost equal to that of the M-EMS algorithm with increasing SNR. Therefore, compared to the SU-EMS algorithm, the RSNS scheduling algorithm proposed in this paper has almost the same error-correcting performance but has lower computational complexity.

Finally, we consider a regular NB-LDPC code, $C_{256}(72, 36)$. The parity-check matrix of this code has row weight $d_r = 4$ and column weight $d_c = 2$. The parameters of the implemented algorithms are listed below.

(1) For the M-EMS algorithm, $M = 128$; for the NS-EMS algorithm, $M = 128, T_c = 400$; for the SU-EMS algorithm, $M = 128, \omega = 2, \varepsilon = 20$; for the SNS-EMS algorithm, $M = 128, \delta = 1$; and for the RSNS-EMS algorithm, $M = 128, \delta = 0.9, T_{\text{iter}} = 10, \omega = 5, \mu = 4$. (2) For all algorithms, the scaling factor $\xi = 0.9$ and $b = 8, \Delta = 1/256$.

The simulation results are shown in Figures 10 and 11. It can be observed that, in these three simulations, the complexity ratio of SU-EMS algorithm increases with increasing SNR, and it is almost the same as the M-EMS algorithm at high SNR. We believe the reason is that the method used by the SU-EMS algorithm to decide whether a variable node is reliable is tracing the message evolution over iterations, where the variable node will be considered reliable only if its outgoing message is stable in ω iterations. The convergence speed of decoder is accelerated under high SNR, and it is possible that all variable nodes have already converged before being considered reliable. Therefore, the SU-EMS algorithm and M-EMS algorithm are basically the same in this situation, so there will be similar computational complexity. Besides, compared with Figures 8 and 9, we set T_c = 400 in this simulation, which makes the complexity ratio of NS-EMS algorithm higher than those of the proposed algorithms. In this case, the correcting performance of NS-EMS algorithm is better than that of the SNS-EMS algorithm and it is close to that of the RSNS-EMS algorithm in low-to-moderate SNR, but it still has a certain degradation with the proposed algorithms in moderate-to-high SNR. Conversely, the proposed algorithms, especially the RSNS-EMS algorithm, can reduce the complexity by nearly 25% while keeping the error-correcting performance almost unchanged compared to the original M-EMS algorithm.

5. Conclusions

In this paper, we propose two innovative scheduling strategies for the EMS algorithm—the SNS-EMS and RSNS-EMS algorithms—to reduce the computational complexity of the decoder. The SNS-EMS algorithm improves the error-correcting performance of the NS-EMS algorithm by using a new metric to judge whether or not a variable node is reliable. More importantly, to further improve the performance of the SNS-EMS algorithm while not significantly increasing its computational complexity, we investigate the variation in the processing node subset of both error frames and converged frames in the SNS-EMS algorithm. According to the experimental results, the RSNS-EMS algorithm is proposed, and it can alleviate performance degradation by employing a subset-reset mechanism. The simulation results show that the RSNS-EMS algorithm can outperform the NS-EMS and SNS-EMS algorithms with similar computational complexity. It not only effectively alleviates the performance degradation compared to the M-EMS algorithm but also reduces the computational complexity by approximately 25%. In conclusion, the proposed method can provide a good tradeoff between BER/FER and decoding complexity compared to other methods.

However, this paper only discovers that such abnormal stability in the processing subset will affect decoding performance, as well as why this characteristic occurs has not been deeply discussed. We believe that continuing to study the reasons for this abnormal stability can provide a theoretical basis for the design of this algorithm. Furthermore, when resetting the processing subset, the method in this paper chooses to put all check nodes into subsets. If we continue to study which check nodes will have a greater effect on alleviating abnormal stability and use this to reset the processing subset, then it is expected that the computational complexity of the algorithm will continue to be reduced. Finally, since there are less researches for informed dynamic schedule and selected schedule on the implementation of decoder, combining proposed scheduling with reduced (hardware) complexity decoding [28, 29] is also an interesting topic for the implementation of NB-LDPC codes. We hope that this work will motivate more researches in these areas.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Key R&D Program of China (2018YFB0505103), the National Natural Science Foundation of China (61561016 and 61861008), and the funding of Science and Technology Major Project of Guangxi (AC16380014, AA17202048, and AA17202033).

References

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

[2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457-458, 1997.

[3] 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.

[4] Y. Fang, P. Chen, G. Cai, F. C. M. Lau, S. C. Liew, and G. Han, "Outage-limit-approaching channel coding for future wireless communications: root-protograph low-density parity-check codes," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 85–93, 2019.

[5] Y. Fang, G. Zhang, G. Cai, F. C. M. Lau, P. Chen, and G. Han, "Root-protograph-based BICM-ID: a reliable and efficient transmission solution for block-fading channels," *IEEE Transactions on Communications*, vol. 67, no. 9, pp. 5921–5939, 2019.

[6] B. Amiri, J. Kliewer, and L. Dolecek, "Analysis and enumeration of absorbing sets for non-binary graph-based codes," *IEEE Transactions on Communications*, vol. 62, no. 2, pp. 398–409, 2014.

[7] M. C. Davey and D. MacKay, "Low-density parity check codes over GF (q)," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, 1998.

[8] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, 2007.

[9] X. Ma, K. Zhang, H. Chen, and B. Bai, "Low complexity X-EMS algorithms for nonbinary LDPC codes," *IEEE Transactions on Communications*, vol. 60, no. 1, pp. 9–13, 2012.

[10] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, 2005.

[11] T. C.-Y. Chang and Y. T. Su, "Adaptive group shuffled decoding for LDPC codes," *IEEE Communications Letters*, vol. 21, no. 10, pp. 2118–2121, 2017.

[12] Z. Yang, Y. Fang, G. Cai, G. Zhang, and P. Chen, "Design and optimization of tail-biting spatially coupled protograph LDPC codes under shuffled belief-propagation decoding," *IEEE Communications Letters*, vol. 24, no. 7, pp. 1378–1382, 2020.

[13] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proceedings of the IEEE Workshop on Signal Processing Systems, SIPS 2004*, pp. 107–112, Austin, TX, USA, October 2004.

[14] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient serial message-passing schedules for LDPC decoding," *IEEE Transactions on Information Theory*, vol. 53, no. 11, pp. 4076–4091, 2007.

[15] G. Han, Y. L. Guan, and X. Huang, "Check node reliability-based scheduling for BP decoding of non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 61, no. 3, pp. 877–885, 2013.

[16] I. V. Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proceedings of the 2007 IEEE International Conference on Communications*, pp. 932–937, Glasgow, Scotland, May 2007.

[17] X. Liu, Y. Zhang, and R. Cui, "Variable-node-based dynamic scheduling strategy for belief-propagation decoding of LDPC codes," *IEEE Communications Letters*, vol. 19, no. 2, pp. 147–150, 2015.

[18] X. Liu, C. Fan, and X. Chen, "Dynamic scheduling decoding of LDPC codes based on tabu search," *IEEE Transactions on Communications*, vol. 65, no. 11, pp. 4612–4621, 2017.

[19] H. Zhang and S. Chen, "Residual-decaying-based informed dynamic scheduling for belief-propagation decoding of LDPC codes," *IEEE Access*, vol. 7, pp. 23656–23666, 2019.

[20] D. Levin, E. Sharon, and S. Li, "Lazy scheduling forLDPC decoding," *IEEE Communications Letters*, vol. 11, no. 1, pp. 70–72, 2007.

[21] S. El Hassani, M. Hamon, and P. Pénard, "Selective-update decoding of non-binary LDPC codes," in *Proceedings of the IEEE 71st Vehicular Technology Conference*, pp. 1–5, Taipei, Taiwan, May 2010.

[22] Y. Sun, H. Chen, X. Li, H. Wan, and T. Qin, "Decoding algorithm for non-binary LDPC codes based on node-subset and k-order message truncation," *ACTA ELECTRONICA SINICA*, pp. 128–133, 2017.

[23] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[24] S. Scholl, "Entwurf and decodierung nicht-binärer low-density-parity-check-codes," Diploma thesis, University of Kaiserslautern, Kaiserslautern, Germany, 2010.

[25] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, d/sub c/)-LDPC codes over GF (q) using their binary images," *IEEE Transactions on Communications*, vol. 56, no. 10, pp. 1626–1635, 2008.

[26] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, 2005.

[27] J. O. Lacruz, F. García-Herrero, J. Valls, and D. Declercq, "One minimum only trellis decoder for non-binary low-density parity-check codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 177–184, 2015.

[28] L. Song, Q. Huang, Z. Wang, M. Zhang, and S. Wang, "Two enhanced reliability-based decoding algorithms for nonbinary LDPC codes," *IEEE Transactions on Communications*, vol. 64, no. 2, pp. 479–489, 2016.

[29] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity, low-memory EMS algorithm for non-binary LDPC codes," in *Proceedings of the 2007 IEEE International Conference on Communications*, pp. 671–676, Glasgow, UK, June 2007.