Hindawi
Modelling and Simulation in Engineering
Volume 2022, Article ID 1407788, 12 pages
https://doi.org/10.1155/2022/1407788

Research Article

Low-Complexity LDPC Decoding Algorithm Based on Layered Vicinal Variable Node Scheduling

Mhammed Benhayoun,[1] **Mouhcine Razi**,[1] **Anas Mansouri**,[2] **and Ali Ahaitouf**[1]

[1]*Faculty of Sciences and Technology, Laboratory of Intelligent Systems, Georesources and Renewable Energies, University Sidi Mohammed Ben Abdellah, P.O. Box 2202, Fez, Morocco*
[2]*National School of Applied Sciences, Laboratory of Intelligent Systems, Georesources and Renewable Energies, University Sidi Mohammed Ben Abdellah, Fez, Morocco*

Correspondence should be addressed to Ali Ahaitouf; ali.ahaitouf@usmba.ac.ma

Received 11 October 2021; Accepted 27 December 2021; Published 19 January 2022

Academic Editor: Noé López Perrusquia

The informed dynamic scheduling (IDS) strategies for the low-density parity check (LDPC) decoding have shown superior performance in error correction and convergence speed, particularly those based on reliability measures and residual belief propagation (RBP). However, the search for the most unreliable variable nodes and the residual precomputation required for each iteration of the IDS-LDPC increases the complexity of the decoding process which becomes more sequential, making it hard to exploit the parallelism of signal processing algorithms available in multicore platforms. To overcome this problem, a new, low-complexity scheduling system, called layered vicinal variable nodes scheduling (LWNS) is presented in this paper. With this LWNS, each variable node is updated by exchanging intrinsic information with all its associated control and variable nodes before moving to the next variable node updating. The proposed scheduling strategy is fixed by a preprocessing step of the parity control matrix instead of calculation of the residuals values and by computation of the most influential variable node instead the most unreliable metric. It also allows the parallel processing of independent Tanner graph subbranches identified and grouped in layers. Our simulation results show that the LWNS BP have an attractive convergence rate and better error correction performance with low complexity when compared to previous IDS decoders under the white Gaussian noise channel (AWGN).

1. Introduction

LDPC codes were introduced by Gallager [1] since 1962 and rediscovered by Mckay and Neal [2] in 1996. LDPC codes have attracted considerable attention because of their superior error correction capability in belief propagation decoding (BP) [2]. BP decoding is conventionally performed by the repetition of the flood schedule [3], where successively all variable-to-check (V2C) and all check-to-variable (C2V) messages are updated in parallel.

However, the convergence process is slowed down because the latest updated information is not available until the next iteration. To accelerate the convergence and improve error correction performance, sequential scheduling methods were presented, with both predetermined and fixed sequence of updates. This sequential scheduling strategy is different from flooding in a way that the latest information is available in the current iteration. The shuffled [4–6] and layered [7, 8] schedules are two variants of this strategy and allow to speed up the a convergence of the decoding twice as fast as the flooding schedule. In order to achieve faster convergence performance informed dynamic scheduling (IDS) strategies have been introduced, among which residual belief propagation (RBP) decoding [9, 10]. The RBP was used for LDPC decoding by Casado et al. [9] after being introduced by Elidan et al. [11] in 2006. It consists of a dynamically adjusted order of message updates, based on the residual value defined as the difference between the current and old message values.

Compared to the sequential and flooding algorithms, the convergence speed of the RBP algorithm is faster, but it was reported that the error correction capability after the first iterations was unsatisfactory [9]; this is because the main computing resources are allocated or focused on a few nodes and edges with the maximum C2V residual; the RBP is then qualified as a greedy algorithm [9]. In order to reduce this greediness effect in of RBP, the node wise RBP (NW RBP) algorithm has been introduced [10], in which all C2V messages of the C2V edges that are connected to the selected control node with the maximum C2V residue are simultaneously propagated.

Based on different message selection and update strategies, several dynamic decoding algorithms have been reported. The variable to check- (VC-) RBP [12] where the priority of message updates is given to the V2C message with the highest residue; The VC-RBP has the advantage of having a lower computation but the error correction performance was not as good as that RBP, because some silent variable nodes with a minimal V2C residual never contribute in the decoding process. Then, a silent variable node-free (SVNF) RBP [13] has been proposed where each variable node had a chance to contribute its intrinsic messages throughout the edge with the maximum C2V residual. The SVNF has the advantage of having a faster convergence speed than RBP but higher computation complexity. The dynamic silent variable node-free scheduling (D-SVNFS) algorithm proposed in the same paper [13] refers to SVNF scheduling but allows a dynamic participation sequence of the variable nodes, which further improves the convergence speed and requires more computation to determine the next variable node to update. The tabu search-based dynamic scheduling (TSDS) [14] algorithm remember that some variable nodes are temporarily stored in a tabu list according to the residual; the variable nodes thus stored can not be updated until they are shifted out of their list after a few number of iterations. This procedure of updating order results in a noticeable reduction of the greediness effect but it still exists keeping a poor error correction capability after the first iterations. Also, in order to tackle this greediness limitation, the residual decaying-based residual belief propagation (RD RBP) [15] has been proposed. In this algorithm, the value of C2V residuals will be gradually decayed according to the update occurrence of the corresponding C2V messages. Consequently, each edge had a chance to being selected in the decoding process, thus mitigating the IDS greediness.

2. Related Work

To approach or even outperform the performances of RBP algorithms, some dynamic decoding algorithms based on the residual message and other selection criteria have also been proposed [16–18]. In the (RM RBP) [16], a reliability metric is used to define the priority of message updates. Firstly, the number of unsatisfied parity check equations for each variable node is calculated; then, all variable nodes are divided into two sets based on whether the number of unsatisfied parity check equations is equal to its maximum value or not. The set including the variable node with the maximum value is a set of all most unreliable variable nodes;

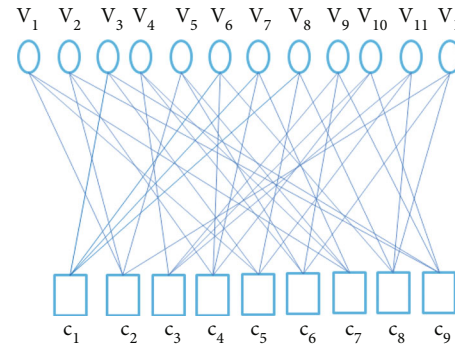


Figure 1: Example of Tanner graph with $N = 12$ variable nodes v_j and $M = 9$ control nodes c_i.

the most reliable variable nodes are involved in the second set. In the IVC RBP algorithm [17], the priority of message update is given to the most unstable variable node, the variable node is qualified as unstable if its sign before and after an update is reversed, and the stability metric calculation increase the computation. In the OV RBP [18], the stability metric based on the number of unsatisfied parity check equations of each variable node is used to select the scheduling order; the most unstable variable node with the high number of unsatisfied parity check equations is updated first.

These new dynamic decoding algorithms showed a significant performance than the RBP prior arts when considering the BER convergence rate. However two major drawbacks still exist. First, the greediness persists even if its influences are reduced, because the variable nodes are selected according to their metric or residual value, and consequently, all the variable nodes do not have the same chance of being selected. Second, the dynamic search for the most unstable or/and unreliable variable nodes, and the residual computation required for each iteration of the IDS LDPC decoding increase the complexity of decoding and then make the decoding process more sequential.

In order to reduce the decoding complexity, to increase the convergence speed and to allow a parallel processing of decoding, we suggest in this work the layered vicinal variable node scheduling (LWNS) algorithm, in which every variable node is forced to contribute with its intrinsic information after each update without the residual value calculations, nor the most unreliable variable node searching computations.

The decoding schedule is fixed and determined by a first preprocessing step, executed outside the decoding process, of the parity control matrix. This preprocessing phase is aimed at identifying for each variable node the subgraph to which it is attached and the influential metric in term of the number of connection. This step is then followed by classifying the parallel subgraph according to the influential metric and finally grouping the parallel subgraph in layers allowing parallelization of the decoding process. The proposed algorithm is tested on the quasicyclic irregular LDPC codes (576, 288), (576,144), (1152, 576), and (2304, 1152) constructed based on the IEEE 802.16e standard, under the white Gaussian noise channel (AWGN), and our simulation results show that the proposed LWNS achieves better

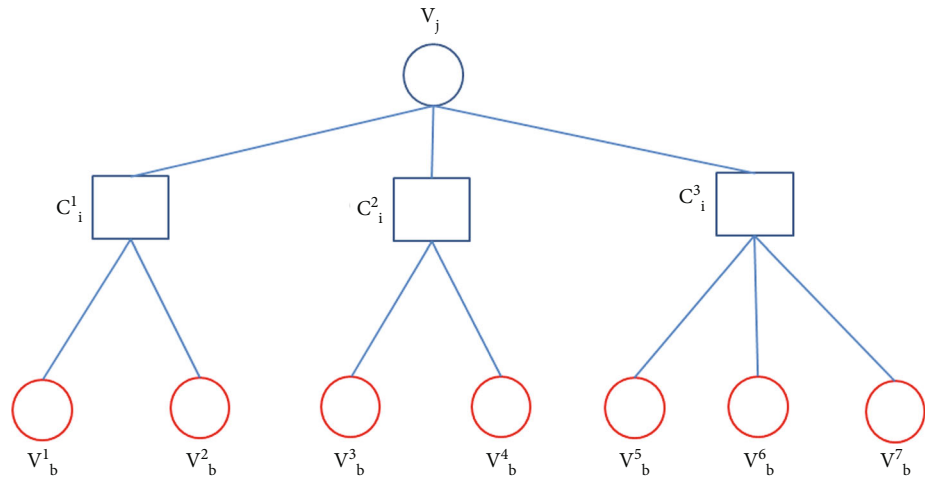


FIGURE 2: Subgraph of head variable node v_j. The variable nodes $v^k{}_b$ are the vicinal variable nodes of v_j and the $c^k{}_i$ are the check nodes connected to v_j.

convergence rate speed and error correcting performance than IVC RBP and OV RBP.

The rest of the paper is organized as follows. Section 2 reviews the LLR BP, the RBP, the IVC RBP, and the OV RBP algorithms. Section 3 introduces the proposed LWNS BP decoding algorithms. Section 4 compares the error correcting performance, the convergence rate, and the complexity of the two proposed algorithms with those of the previous works. Finally, Section 5 concludes the paper.

3. Basic Decoding LDPC Algorithms

3.1. LLR BP Decoding for LDPC Codes. A binary LDPC code can be described using a Tanner graph of N variable and M control nodes as illustrated in Figure 1, where v_j denotes the jth variable node and c_i denotes the ith control node. When the BP LLR decoding algorithm is used for the LDPC decoding, the C2V messages $m_{ci\longrightarrow vj}$ that propagate from c_i to v_j are initialized at zero and the V2C messages $m_{vj\longrightarrow ci}$ that propagates from $\mathrm{v_j}$ to $\mathrm{c_i}$ are initialized as follows:

$$m(0)_{vj\longrightarrow ci} = L(0)_{v_j} = \log\left(\frac{p\left(y_j/v_j=0\right)}{p\left(y_j/v_j=1\right)}\right), \tag{1}$$

where y_j denotes the channel information of the jth variable node and v_j denotes the jth code bit.

After each initialization, the updated C2V messages $m_{ci\longrightarrow vj}$ are generated according to

$$m_{ci\longrightarrow vj} = 2\tan h^{-1}\left(\prod_{vb\in N(ci)\backslash vj}\tanh\left(\frac{m_{vb\longrightarrow ci}}{2}\right)\right), \tag{2}$$

where $N(c_i)\setminus v_j$ denotes the neighboring variable nodes that are connected to check node c_i, excluding variable node v_j.

The updated of V2C messages $m_{\mathrm{v}j\longrightarrow ci}$ are calculated as

$$m_{vj\longrightarrow ci} = C_{v_j} + \sum_{ca\in N(vj)\backslash ci} m_{ca\longrightarrow vj}, \tag{3}$$

where $N(v_j)\setminus c_i$ denotes the neighboring check nodes that are connected to variable node v_j, except for check node c_i.

After the generation and propagation of all the updated V2C and C2V messages, a hard decision on the variable node v_j is taken based on the new LLR value $L(v_j)$ calculated as

$$L\left(\mathrm{v}_j\right) = C_{\mathrm{v}_j} + \sum_{ca\in N(\mathrm{v}j)} m_{ca\longrightarrow \mathrm{v}j}, \tag{4}$$

where $N(v_j)$ denotes all the neighboring check nodes that are connected to variable node v_j.

The message propagation in the decoding process will not stop until all the check equations are satisfied or the predefined maximum number of iterations is reached.

3.2. Scheduling Strategies for LDPC Decoding. Flooding is the simplest scheduling LLR BP algorithm, which first updates all C2V messages using Equation (2) and then updates all V2C messages using Equation (3). Therefore, the newly generated information in the current iteration can only be used in the next iteration. This is slightly different from the sequential scheduling algorithm where the new information can be generated and immediately used in the same iteration.

In these two cases, the messages updating follow a predefined order.

In opposite, the IDS dynamically adjust the order of message updates. The first RBP IDS algorithm determines the order of message updating based on the C2V residual. The C2V residual is defined as the magnitude of the difference between the C2V messages before and after an update.

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}
c_1	0	0	1	0	0	1	1	1	0	0	0	0
c_2	1	1	0	0	1	0	0	0	0	0	0	1
c_3	0	0	0	1	0	0	0	0	1	1	1	0
c_4	0	1	0	0	0	1	1	0	0	1	0	0
c_5	1	0	1	0	0	0	0	1	0	0	1	0
c_6	0	0	0	1	1	0	0	0	1	0	0	1
c_7	1	0	0	1	1	0	1	0	0	0	0	0
c_8	0	0	0	0	0	1	0	1	0	0	1	1
c_9	0	1	1	0	0	0	0	0	1	1	0	0

(a)

$c_1 : v_3 \oplus v_6 \oplus v_7 \oplus v_8 = 0$

$c_2 : v_1 \oplus v_2 \oplus v_5 \oplus v_{12} = 0$

$c_3 : v_4 \oplus v_9 \oplus v_{10} \oplus v_{11} = 0$

$c_4 : v_2 \oplus v_6 \oplus v_7 \oplus v_{10} = 0$

$c_5 : v_1 \oplus v_3 \oplus v_8 \oplus v_{11} = 0$

$c_6 : v_4 \oplus v_5 \oplus v_9 \oplus v_{12} = 0$

$c_7 : v_1 \oplus v_4 \oplus v_5 \oplus v_7 = 0$

$c_8 : v_6 \oplus v_8 \oplus v_{11} \oplus v_{12} = 0$

$c_9 : v_2 \oplus v_3 \oplus v_9 \oplus v_{10} = 0$

(b)

Figure 3: (a) Example of parity check matrix. (b) Its corresponding parity check equations, where $\oplus$ denotes a modulo 2 addition (i.e., XOR); the variable node v_1 appear in c_2, c_5, and c_7; the variable nodes $\{v_2, v_5, v_{12}, v_3, v_8, v_{11}, v_4, v_7\}$ are the vicinal of v_1.

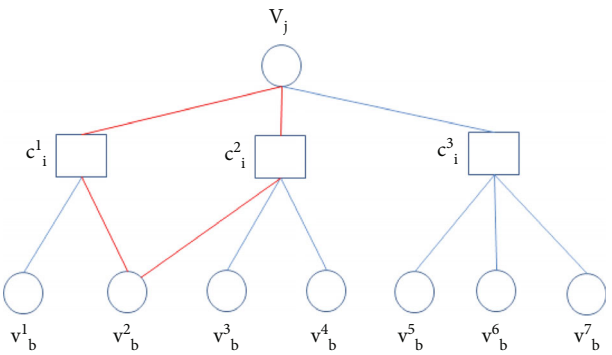


Figure 4: Cycle of length 4 between v_j and their vicinal variable nodes.

The C2V residual can be defined [8] as

$$r_{ci \longrightarrow vj} = \left\| m^{\text{pre}}_{ci \longrightarrow vj} - m_{ci \longrightarrow vj} \right\|, \tag{5}$$

where $m^{\text{pre}}_{ci \longrightarrow vj}$ is the precomputation of the next updated C2V message. As mentioned above, this algorithm only updates the C2V edge with the maximum C2V message residual.

Hence, at the beginning of the decoding process, all C2V messages are calculated in order to identify the C2V edge candidate for the update.

First, the precomputed C2V message of the maximum C2V residual is propagated to variable nodes; then, the V2C messages are updated and propagated to its neighbor check node. The check nodes receiving the new V2C messages value actualize their precomputed C2V messages and their residual value. Finally, the residual of the C2V message being currently updated is set at 0, and the C2V message with the largest residual is selected for the next update.

In the IVC RBP algorithm [17], variable nodes are divided into two groups formed by stable or an unstable node.

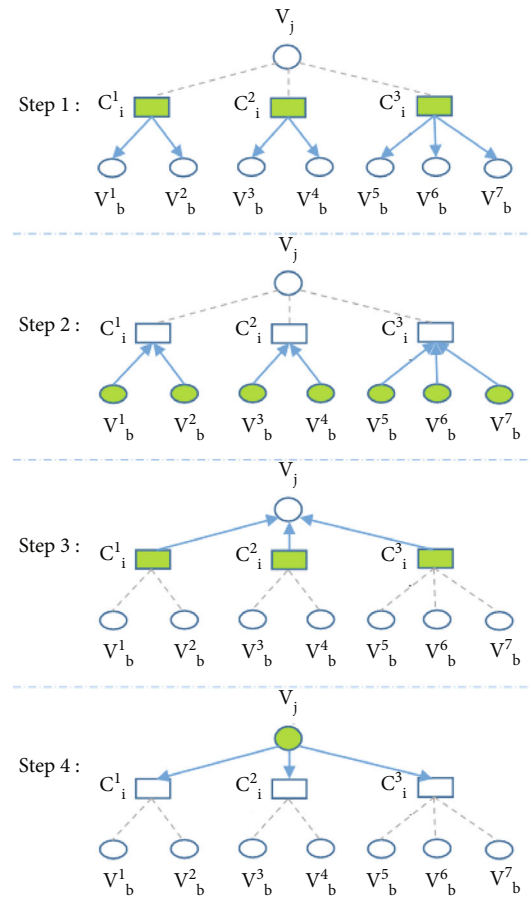


Figure 5: Decoding step of the LWNS algorithm for each variable node.

After identifying the most unstable variable nodes, the update of the selected messages is performed based on the RBP procedure.

In addition, to this instability metric, the OV RBP algorithm [18] scheduling strategy looks at the number of unsatisfied parity check equations, defined for each variable node as

1: Initialize all $m_{ci\longrightarrow vj} = 0$, $m_{vj\longrightarrow ci} = L(0)_{v_j}$ and Itermax
2: for $i = 1$ to Itermax do
3: for every v_j ordered based on the influence metric do
4: for every $c_i \in N(v_j)$ do
5: for every $v_b \in N(c_i) \setminus v_j$ do
5: Generate and propagate $m_{ci\longrightarrow vb}$
6: end for line 5
7: end for line 4
8: for every $v_b \in N(c_i) \setminus v_j$ do
7: Calculate $L(v_b)$
8: Generate and propagate $m_{vb\longrightarrow ci}$
9: enf for line 8
10: Calculate $L(v_j)$
11: end for line 3
12: if stopping rule is not satisfied then
13: Go back to line 3
14: end if
15: end for line 2

Algorithm 1: LWNS scheduling.

$$U(\mathrm{v}j) = \sum_{ci\in N(\mathrm{v}j)} T(c_i), \tag{6}$$

where $T(c_i)$ denotes the result of parity check node c_i. The more this check equation is not verified by a given variable node, the more it is considered an unreliable variable node. Hence, each set of unstable or stable variable nodes is divided into two others sets based on whether $U(v)$ is equal to its maximum value or not.

After selecting the most unstable variable node with the maximum number of unsatisfied parity check equations, the update of the selected messages is performed based as for the RBP.

The main computing resources are allocated specifically for nodes and edges with the maximum value of metric and residual; therefore, the greediness influence still occurs.

All judgment criteria seen above and the residual precomputation performed in each decoding iteration generate additional calculations and consequently increase considerably the complexity of the decoding.

4. Proposed LWNS Algorithms

In order to bring down the greediness influences and reduce the decoding complexity, with a better BER convergence performance, we introduce the layered vicinal variable node scheduling (LWNS) to be applied to classical LLR BP algorithm.

In contrast to different dynamic RBP scheduling algorithms, in the LWNS, all the variable nodes will be updated with the same chances, but the processing order is defined by a parity check matrix preprocessing performed outside the decoding sequence in order to define a fixed sequential schedule.

4.1. Subgraph Vicinal Variable Nodes. In the LWNS, each variable node v_j is updated closely by exchanging intrinsic information with all variable nodes $v^k{}_b$ connected to variable node v_j through all check node $c^k{}_i$ connected to v_j, except the variable node v_j. This set of variable nodes is said the vicinal variable nodes of v_j and noted $v(v_j)$.

Figure 2 shows the subgraph in which each variable node v_j update is operated. The purpose is to process each variable node v_j, closely in its subgraph; in this way, the latest updated information directly related to v_j is available during it update, thus allowing a fast convergence of decoding.

Figure 3 presents an example of the parity check matrix (a) and its related check equations (b); the variable node v_1 is updated by exchanging intrinsic information closely within the check equations c_2, c_5, and c_7 and the vicinal variable nodes $v(v_1) = \{v_{2,}\, v_{5,}\, v_{12,}\, v_{3,}\, v_{8,}\, v_{11,}\, v_{4,}\, v_7\}$.

4.2. The Most Influential Variable Node Metrics. Each variable node generates its own subgraph; consequently, the number of subgraphs is equal to the number N of variable nodes.

To define the subgraph update order, we use the most influential variable nodes metric concept instead of the unstable variable nodes metric.

Given that the more neighbor's a variable node has, the more influenced it becomes.

For each variable node v_j, we calculate the influence metrics $D(v_j)$ as the number of edge between the v_j and their vicinal set:

$$D(v_j) = dv(v_j) + \sum_{i=1}^{dv} (dc(c_i) - 1), \tag{7}$$

where $dv(v_j)$ denotes the total number of check nodes connected with the variable node v_j and $dc(c_i)$ denotes the total number of variables nodes connected with the check nodes c_i. This metrics is calculated outside the decoding processing.

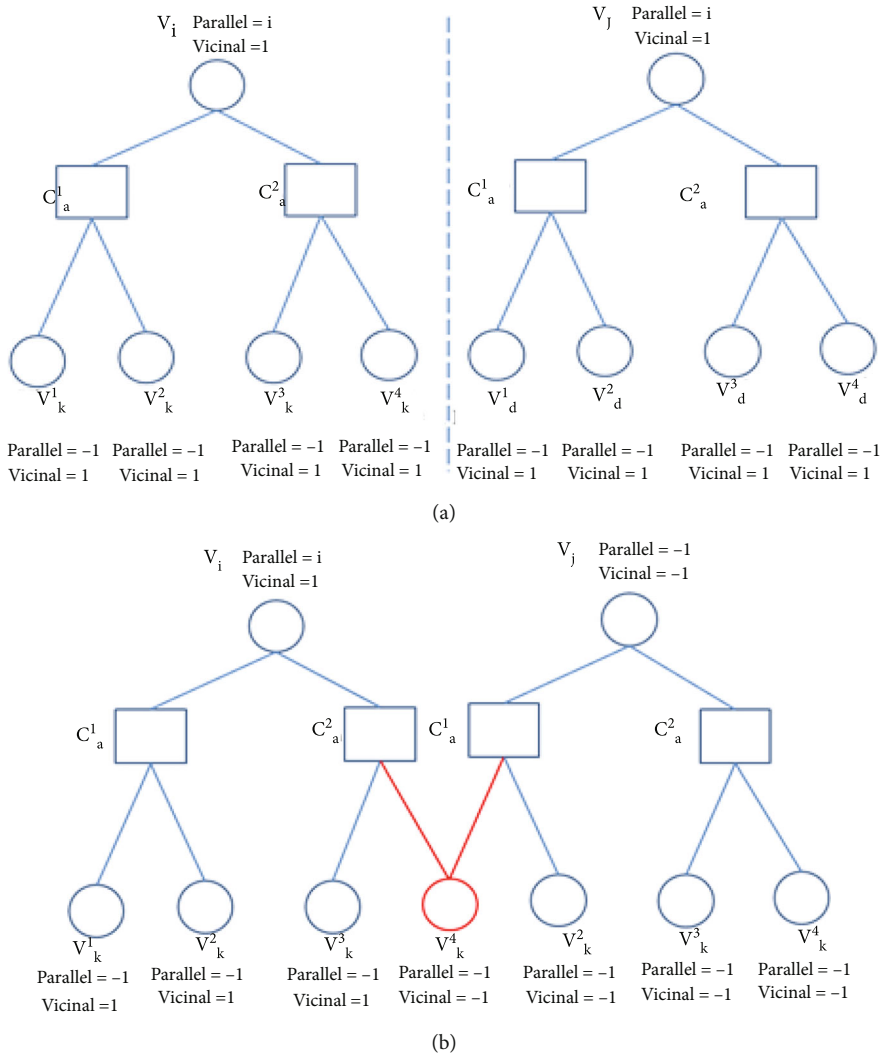
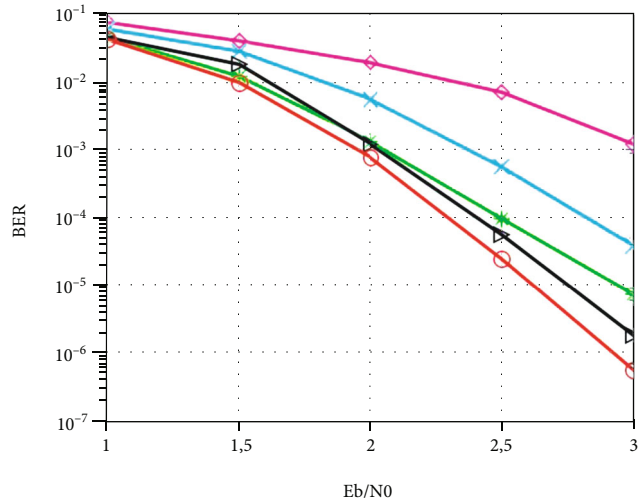


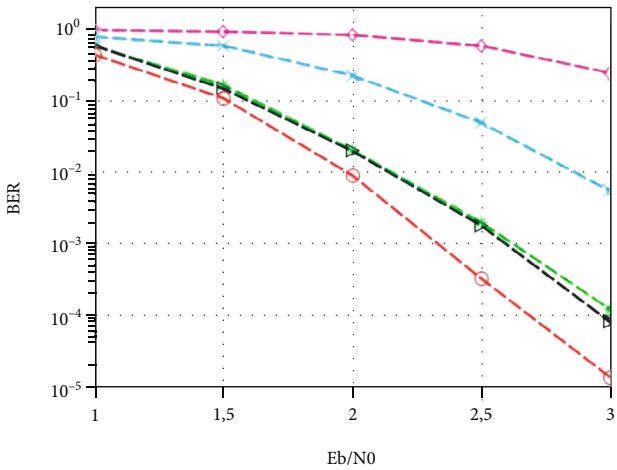
Figure 6: Parallel layered scheduling. (a) Variable node v_j and v_i are parallel $v(v_i) \bigcap v(v_j) = \varnothing$. (b) Variable nodes v_j and v_i are dependent $v(v_i) \bigcap v(v_j) = v_k^4$.

1: for every v_j from 0 to $N-1$ do
2: Initialize all parallel(v_i) = −1, vicinal(v_k) = −1
3: end for line 1
4: for every v_i from 0 to $N-1$ do
5: if parallel(v_i) = −1
6: parallel(v_i) = i, vicinal(v_i) = 1
7: Search $v(v_i)$ and assigned to each v_k de $v(v_i)$vicinal(v_k) = 1
8: for each v_j from 0 to $N-1$ do
9: if parallel(v_i) = −1
10: Search $v(v_j)$
11: if there is no v_d de $v(v_j)$ that vicinal(v_d) = 1
12: parallel(v_j) = i
13: end if line 11
13: end if line 9
16: end for line 8
13: end if line 5
17: end for line 4

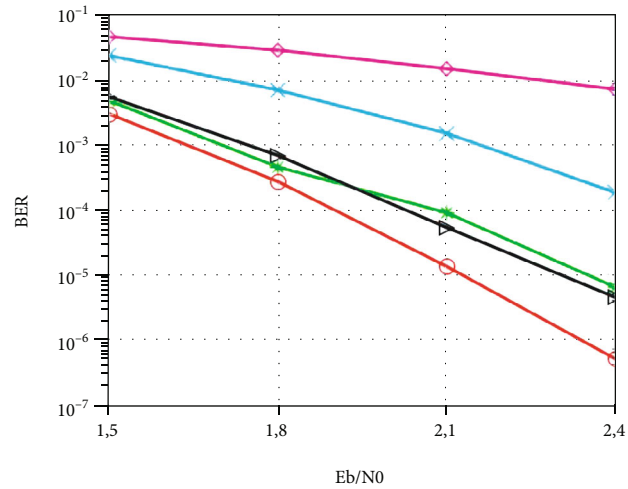
Algorithm 2: Parallel layer variable nodes.



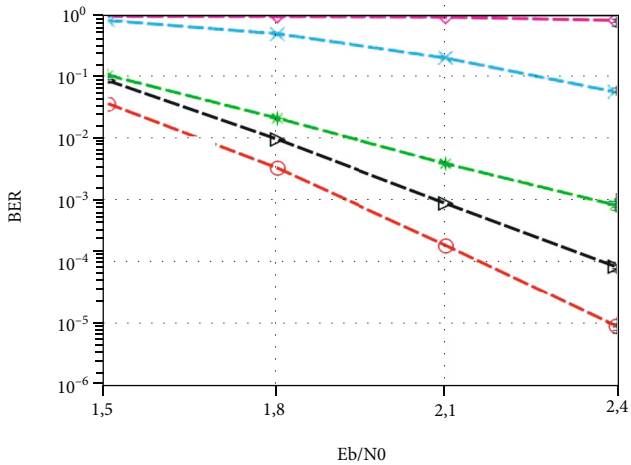
(a) BER LDPC code (576, 288)



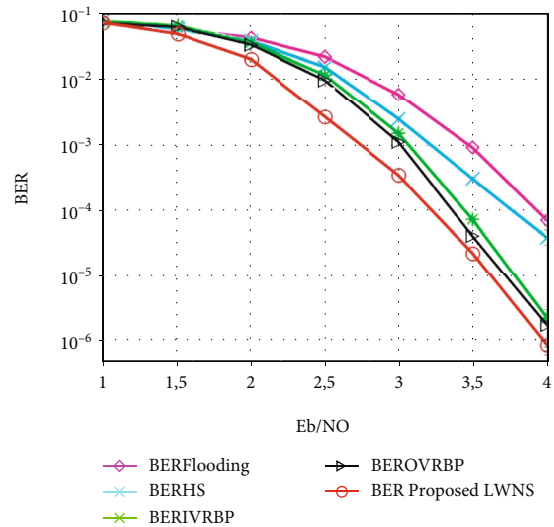
(b) FER LDPC code (576, 288)



(c) BER LDPC code (1152, 576)

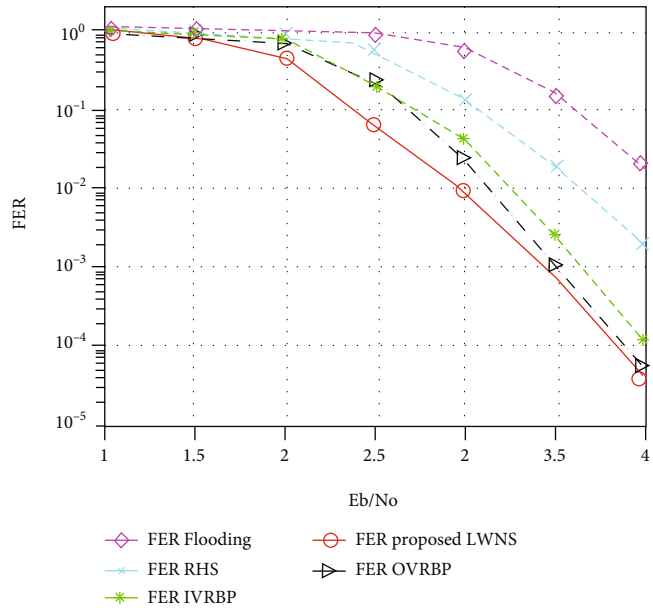


(d) FER LDPC code (1152, 576)



(e) BER LDPC code (576, 144)

Figure 7: Continued.



(f) FER LDPC code (576, 144)

Figure 7: BER and FER vs. E_b/N_0 of the decoding algorithms: flooding, HS, IVRBP, OVRBP, and LWNS for irregular LDPC (576, 288), (1152, 576), and (576, 144) codes.

Table 1: Comparison of performance of OVRBP [18] and proposed LWNS algorithms.

Algorithms	Code length	Rate	SNR (dB)	I_{avg}	FER	BER
OVRBP [18]	576	0.5	3	1.3	8×10^{-5}	1.810^{-6}
Proposed LWNS	576	0.5	3	1.3	1.3×10^{-5}	5.6×10^{-7}
OVRBP [18]	576	0.75	4	1	5.2×10^{-5}	1.7×10^{-6}
Proposed LWNS	576	0.75	4	1	4.1×10^{-5}	8.5×10^{-7}
OVRBP [18]	1152	0.5	2.4	2	5.5×10^{-5}	4.3×10^{-6}
Proposed LWNS	1152	0.5	2.4	1.9	9.1×10^{-6}	4.8×10^{-7}

When a variable node v_j is connected to a vicinal node $v^2{}_b$ by two separate control nodes $c^1{}_i$ and $c^2{}_i$, the value of $D(v_j)$ increases, but the variable node does not have so much influence because the messages oscillate in a closed cycle.

This is typically the case corresponding to the cycle of length 4 in the matrix H, as shown in Figure 4. The length of the cycle is defined as the number of crossed edges to quit and return to a node without passing through the same edges. This case is moved by eliminating the edges causing cycle 4 [19].

The decoding process is performed in the descending order of the influence metric value.

Figure 5 shows the decoding process followed in the LWNS algorithm adopted for each variable node v_j, classified in the descending order of the influence metric. In the first step, using Equation (2), the messages $m_{ci\longrightarrow vb}$ from check nodes c_i where $c_i \in N(vj)$ to all vicinal variable nodes $v_b \in v(v_j)$ are generated by taking in consideration the variable nodes v_j and propagated to all vicinal variable nodes v_b. In the second step, using Equation (4), the LLR values of the vicinal variables node v_b are calculated, and using Equation (3), all V2C messages are updated from the vicinal variable nodes v_b to all check nodes c_i neighbor's, where $c_i \in N(v_j)$. In the two first steps, the vicinal variable nodes are updated taking in consideration the LLR value of the variable node v_j.

Then, in step 3, using Equation (2), the messages $m_{ci\longrightarrow vj}$ from check nodes c_i where $c_i \in N(v_j)$ to variable nodes v_j are generated by taking in consideration all vicinal variable nodes $v(v_j)$ and propagated to variable nodes v_j. At the last, the LLR values of the variables node v_j are calculated using Equation (4). In this way, the variable node v_j is updated taking on consideration the LLR value of the all vicinal variable node $v(v_j)$.

The detailed decoding process for the LWNS algorithm with pseudocode is described in Algorithm 1.

5. The Proposed Layered Parallel Architecture

In parallel architectures, one has to avoid data dependencies during the decoding process for a parallel memory accesses.

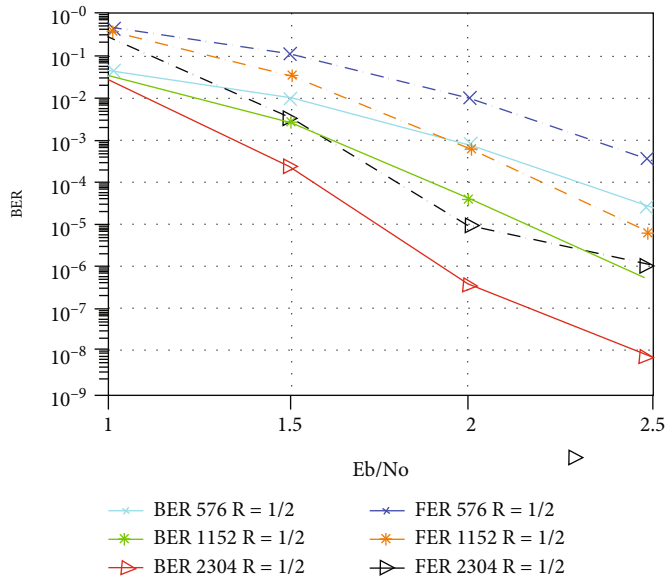


Figure 8: BER vs. E_b/N_0 of the proposed LWNS decoding algorithm for irregular LDPC (576, 288), (1152, 576), and (2304, 1152) codes.

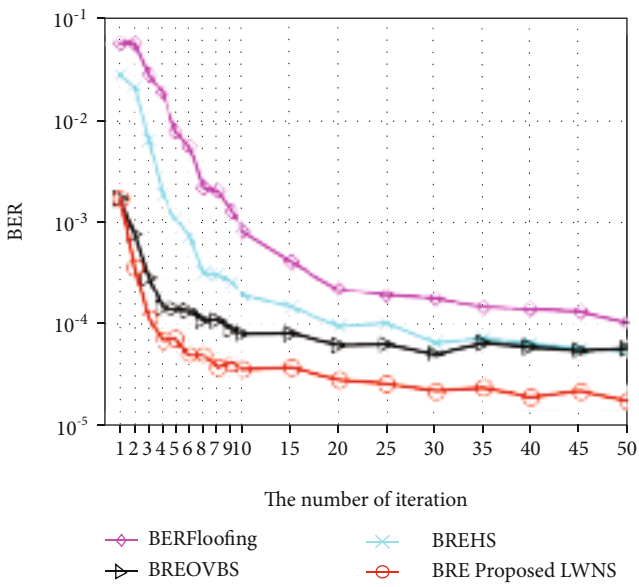


Figure 9: BER vs. the maximum number of iteration of decoding algorithms: flooding, HS, OVRBP, and LWNS for irregular LDPC code (576, 288) at $E_b/N_0 = 2.5$ dB.

The dependency of the variable nodes comes from the parity check matrix H and from the scheduling strategy.

We define the parallel LWNS scheduling strategy as the LWNS strategy where instead of updating only the variable node with the largest influence metrics, k variable nodes are updated at the same time.

Therefore, the parallel processing is allowed between k variable nodes if their vicinal set does not share any variable nodes; otherwise,

$$\bigcap_{k\,\text{element}} v(v_{\text{element}}) = \varnothing \tag{8}$$

For all variable nodes, we define two variables: "parallel" and "vicinal." The variable nodes satisfying Equation (8) are parallel and have the same "parallel" value. The "vicinal" value is set to one to mark that this variable node is already belongs to the vicinal of another variable node.

Figure 6 shows the value of these variables whether the variables nodes are linked through their vicinal nodes or not. In Figure 6(a), the variable nodes v_i and v_j are parallel, so the "parallel" variable of these two variable nodes is set at the value of column i, and their "vicinal" variable is set at 1 to indicate that this variable node has already been covered. In opposite, in Figure 6(b), the variable nodes v_i and v_j are linked by their vicinal variable node $v^4{}_k$; in this case, v_i and v_j are dependent, so the "parallel" and "vicinal" variables will not be changed.

Hence, all variable nodes are divided into parallel subgraphs; the detailed parallel variable node matrix preprocessing with pseudocode is described in Algorithm 2.

Parallel variable nodes will be updated at the same time. In order to synchronize the parallel processes and to avoid waiting times, the parallel variable nodes must have the same computational complexity to ensure that the parallel processing end at the same time and to move on to the next series. The k variable nodes that have the same influential metrics are updated simultaneously.

6. Performance Evaluation

In order to evaluate the decoding performances of the proposed LWNS algorithm, a comparison with other sequential algorithms is performed, including flooding [3], horizontal shuffle [5], IDS IVC-RBP [17], and VO-RBP [18] algorithms.

A large number of simulations are used to obtain the decoding performance of the proposed LWNS algorithm, with millions of frames lunched in the CNRST/HPC-

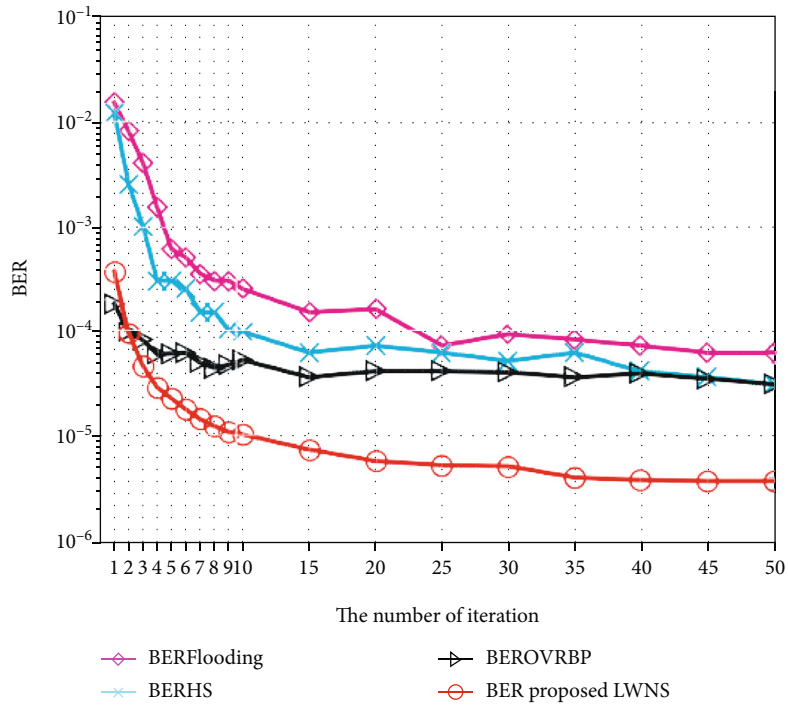


Figure 10: BER vs. the maximum number of iteration of decoding algorithms: flooding, HS, OVRBP, and LWNS for irregular LDPC code (576, 144) at $E_b/N_0 = 3.5$ dB.

Table 2: Equivalent complexity of the proposed algorithm.

Step	Search computations	V2C computation	C2V computation	$r(v)$
Step 1	0	0	$dv.(dc-1)$	0
Step 2	0	$dv.(dc-1)$	0	0
Step 3	0	0	dv	0
Step 4	0	dv	0	0

MARWAN platform [20]. The computation complexity of these algorithms is evaluated according to the number of computations required in each decoding step. All simulations are performed on the AWGN channel with the BPSK modulation. Codes (N, M) designates the codes used in the simulation, where N is the block length and M denotes the number of parity check bits. We used the codes (576, 288), (576,144), (1152, 576), and (2304, 1152) which are the quasicyclic irregular LDPC codes constructed based on the IEEE 802.16e standard [21].

The efficient memory parity check matrix optimization proposed in [22, 23] is used to reduce the memory requirement.

6.1. Error Correction Performance. Figures 7(a)–7(d) shows the bit error ratio (BER) and the frame error ratio (FER) performances of the decoding algorithms for LDPC codes of coding rate $R = 1/2$ and, respectively, block lengths 576 and 1152.

It is clear that the error correcting performance of the proposed algorithm is better than the others. As can be observed, the performance advantage is preserved as the signal to noise ratio increase to reaches 5.6×10^{-7} when $E_b/N_0 = 3$ dB for block lengths 576 and 4.8×10^{-7} when $E_b/N_0 = 2.4$ dB for block lengths 1152.

Figures 7(e) and 7(f) deals, respectively, with the BER and FER performances of the irregular quasicyclic 576 LDPC code with the coding rate $R = 3/4$; the proposed LWNS algorithm shows the best BER compared with the OVRBP algorithm. Between $E_b/N_0 = 2$ dB and 3.5 dB, the gain is more than 0.25 dB, and in $E_b/N_0 = 4$ dB the BER is very low reaching 8.10^{-7}.

In Table 1, a comparison between the OVRBP [18] algorithms in terms of BER, FER, and the average of decoding iteration number I_{avg} is presented. It can be seen that the BER and FER performances in the proposed LWNS algorithms are improved with the same decoding iteration number.

To evaluate the proposed LWNS algorithm performances versus the block length code, the BER and FER performances for block length 576, 1152, and 2304 versus the same range of signal to nose ratio (E_b/N_0) are simulated and reported in Figure 8,

It can clearly observe that the LWNS decoding algorithm performances are getting more attractive as the length of block codes increases, the BER and FER are reaching 6.9×10^{-9} and $10^{-6,}$ respectively, when $E_b/N_0 = 2.4$ dB; in other words, for 138 Mbits $(1/6, 910^{-9}$ bits), only one bit will be wrong.

In order to demonstrate that the proposed algorithm converges more quickly, Figures 9 and 10 show the BER performance versus the number of iterations for (576, 288) LDPC codes at $E_b/N_0 = 2.5$ dB, and the (576, 144) LDPC

Table 3: Equivalent complexity of different decoding algorithms.

Schedules	Search computations	V2C computations	C2V computations
Flooding	0	E	E
HS	0	$E.dv$	E
IVRBP	$M(M.dc-1)$	$E[(dv-1)+(dc-1)]$	$E(dv-1)(dc-1)$
OVRBP	$N(N-1)$	E	$E.dc$
Proposed LWNS	0	$E.dc$	$E.dc$

codes at $E_b/N_0 = 3.5$ dB. The simulations for both of them demonstrate that the proposed LWSN algorithm converge faster than any others algorithms within about 15 iterations. In other words, the LWSN not only has better error correction performance but also has faster convergence speed.

The greediness influences of the OVRBP algorithm is showed in Figures 9 and 10, and the error correction performance after the first five iterations was reported to be low. In contrast, the BER of the proposed LWNS algorithm grow with increasing the number of iterations, to reach a stable value after 15 iterations, 3.10^{-6} at $E_b/N_0 = 3.5$ dB, instead of 3.10^{-5} for the OVRBP algorithm for instance, the most rapid of the used algorithms for comparison.

6.2. Complexity Analysis. In this subsection, we evaluated the decoding complexity of the proposed algorithm in terms of the number of C2V and V2C updates required in each decoding iteration.

For some decoding algorithms, the residual calculation is necessary for updating. The precomputational complexity of this part is added to the C2V computation. In addition, if an algorithm needs to find the most unstable variable nodes among all variable nodes, a certain amount of searching computations is required.

This complexity is inherent to each iteration of LDPC decoding, where one iteration means the process of selecting and updating all edges in the Tanner graph. The total number of edges in the entire Tanner graph is $E = dc.M = dv.N$, where dc and dv denote the average degree of check nodes and variable nodes, respectively.

Table 2 presents the updating complexity calculation of the proposed algorithm, which consists of the number of V2C updating, C2V updating by step shown in Figure 5.

For each variable node v_j, in step 1, there are $dv.(dc-1)$ C2V messages calculated to update the vicinal variable nodes, and then in step 2, these vicinal variable nodes calculate $dv.(dc-1)$ V2C messages to update the check nodes connected to them. In step 3, the updated check node generate and propagate dv C2V to be used to update the variable node v_j; in step 4, finally, the LLR value of the variable node v_j is updated and dv V2C messages are propagated to the check nodes connected to v_j.

Hence, for each LWSN iteration, the required number of V2C, C2V updates is $\{N.[dv + dv.(dc-1)]\} = E.dc$.

In Table 3, the updating complexity for different decoding algorithms is presented. The LWSN algorithm replace the computations of the residual values and the computations for updating the most unreliable variable by a preprocessing step not included in decoding processing.

Because of the search computations required for determining the most unstable variable node in each iteration, the complexity of the IVRBP and OVRBP algorithms can be expressed as $O(M^2)$ and $O(N^2)$, respectively, while the complexity of the LWNS algorithm is about $O(E)$.

The proposed LWSN algorithm clearly reduces complexity compared to the IDS IVRBP and OVRBP algorithms.

In addition, the proposed LWNS algorithm can operate in parallel fashion, by updating the variable nodes by layer. Each set of variable nodes that are labeled by the same variable "parallel" is updated as shown in the Algorithm 1 in a parallel way.

Therefore, the complexity can highly be reduced depending on the multicore platform and the level of the adopted parallelism.

7. Conclusion

In this paper, we present an innovative layered algorithm, LWNS, in order to decrease the decoding complexity with the same performance obtained by the IDS algorithms. In the LWNS algorithm, all variable nodes participate with the same opportunity to the update processing to overcome the greediness of IDS. The search of the unreliable messages and the residual computation are replaced by the matrix H preprocessing. The proposed algorithm achieves much better BER, FER performance and convergence speed; the BER and FER are reaching 6.9×10^{-9} and 10^{-6}, respectively, when $E_b/N_0 = 2.4$ dB and the complexity is reduced to $O(E)$. In addition, the complexity of the LWSN algorithm can be significantly decreased in the platforms where parallelization is possible. Our simulation results and analysis show that the proposed algorithms behaves much better than the other reported algorithms in the literature when considering the BER, convergence rate, and decoding complexity as a whole.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported through computational resources of HPC-MARWAN (http://www.marwan.ma/hpc) provided by the National Center for Scientific and Technical Research (CNRST), Rabat, Morocco.

References

[1] R. G. Gallager, "Low-density parity-check codes," *IRE 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. 32, no. 18, pp. 1645-1646, 1996.

[3] F. R. Kschischang, B. J. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

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

[5] C. A. Aslam, Y. L. Guan, and K. Cai, "Improving the belief-propagation convergence of irregular LDPC codes using column weight based scheduling," *IEEE Communications Letters*, vol. 19, no. 8, pp. 1283–1286, 2015.

[6] M. Razi, A. Mansouri, A. Aït Madi, and A. Ahaitouf, "Horizontal-shuffled scheduling for the low density parity check codes decoding for WiMAX application," in *Proceedings of the 1st International Conference on Electronic Engineering and Renewable Energy*, Saidia, Morocco, 2018.

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

[8] M. Kim, D. Kim, and Y. H. Lee, "Serial scheduling algorithm of LDPC decoding for multimedia transmission," in *IEEE international Symposium on Broadband Multimedia Systems and Broadcasting*, pp. 1–4, Seoul, Korea, 2012.

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

[10] A. I. V. Casado, M. Griot, and R. D. Wesel, "LDPC decoders with informed dynamic scheduling," *IEEE Transactions on Communications*, vol. 58, no. 12, pp. 3470–3479, 2010.

[11] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: informed scheduling for asynchronous message passing," in *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 165–173, Cambridge, MA, USA, 2006.

[12] J. Kim, M. Nam, and H. Song, "Variable-to-check residual belief propagation for LDPC codes," *IET Electronics Letters*, vol. 45, no. 2, pp. 117-118, 2009.

[13] C. Adnan Aslam, Y. Guan, K. Cai, and G. Han, "Low-complexity belief-propagation decoding via dynamic silent-variable-node-free scheduling," *IEEE Communications Letters*, vol. 21, no. 1, pp. 28–31, 2017.

[14] 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.

[15] 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.

[16] X. Liu, L.'e. Zi, D. Yang, and Z. Wang, "Improved decoding algorithms of LDPC codes based on reliability metrics of variable nodes," *IEEE Access*, vol. 7, pp. 35769–35778, 2019.

[17] Y. Gong, X. Liu, W. Ye, and G. Han, "Effective informed dynamic scheduling for belief propagation decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 59, no. 10, pp. 2683–2691, 2011.

[18] 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.

[19] N. Farheen, *Detection and removal of cycles in LDPC codes, [M.S. thesis]*, Concordia University, 2018.

[20] http://www.marwan.ma/hpc.

[21] IEEE, "IEEE Std. 802.16e-2005 and IEEE Std.802.16-2004/Cor1-2005," 2006.

[22] M. Benhayoun, M. Razi, A. Mansouri, and A. Ahaitouf, "Efficient memory parity check matrix optimization for low latency quasi cyclic LDPC decoder," in *Proceedings of the 2nd International Conference on Electronic Engineering and Renewable Energy Systems*, Saidia, Morocco, 2020.

[23] M. Benhayoun, M. Razi, A. Mansouri, and A. Ahaitouf, "New memory load optimization approach for software implementation of irregular LDPC encoder/decoder," in *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, pp. 1–6, Fez, Morocco, 2019.