

https://dspace.org/

Adaptive Group Shuffled Symbol Flipping Decoding Algorithm

Waheed Ullah; Ling Cheng; Fambirai Takawira

https://hdl.handle.net/10539/41952
WOS

Downloaded from a DSpace repository. https://dspace.org/

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 February 2024, accepted 3 March 2024, date of publication 10 April 2024, date of current version 19 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3387278

RESEARCH ARTICLE

Adaptive Group Shuffled Symbol Flipping Decoding Algorithm

WAHEED ULLAH, (Senior Member, IEEE), LING CHENG, (Senior Member, IEEE), AND FAMBIRAI TAKAWIRA, (Member, IEEE)

School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg 2017, South Africa

Corresponding author: Waheed Ullah (waheed@ieee.org)

ABSTRACT The paper introduces two decoding techniques based on grouping for symbol flipping non-binary LDPC codes. Firstly, a new technique of adaptive grouping of variable nodes in each iteration using bit reliability and majority voting of each received symbol is presented. Grouping of variable nodes and the subsequent decoding are based on individual symbol reliability and majority voting. To form groups adaptively, the cumulative density of variable nodes is used where a high priority group is considered to contain the most unreliable variable nodes and is decoded while the second priority group contains variable nodes having a second level of reliability and so on. Secondly, a fixed grouping technique is applied to symbol flipping decoding of non-binary LDPC codes. In the fixed grouping decoding, each group contains an equal number of variable nodes, and selected symbols in each group are flipped according to pre-defined flipping criteria. Numerical results and analysis show that the proposed group-based hard decision symbol flipping decoding algorithms have the advantage of reduced computational complexity and show better performance. Therefore, the proposed algorithms can be considered for applications like data storage and mobile communication.

INDEX TERMS Adaptive group shuffled decoding, non binary LDPC, symbol flipping decoding, sum product algorithms, layered decoding, channel coding.

I. INTRODUCTION

The current focus of research on LDPC codes is aimed at reducing decoding computational complexity and improving performance, particularly for non-binary LDPC (NB-LDPC) codes. While NB-LDPC codes have higher computational complexity compared to binary LDPC codes, [1], [2], they offer the advantage of being directly mappable to the order of modulation, as demonstrated in several works such as [3], [4], and [5]. The NB-LDPC message passing algorithm in [3] over $GF(q)$ has a computational complexity in an order of $O(q^2)$ during check node processing and is the main hurdle in implementation. The check node complexity is reduced by using an efficient log-domain based fast Fourier transform (FFT) in [6]. The FFT based sum-product algorithm in [6] reduces the check node processing complexity from $O(q^2)$ to $O(qlogq)$. To further reduce the computational and implementation complexity, an Extended Min-Sum (EMS) algorithm in [7] is presented which has greatly reduced the check node computational complexity and has the implementation advantage over the other algorithms. For hardware implementation, authors in [8] and [9] present a Bubble Check algorithm to reduce the number of comparisons during check node processing without any performance loss. Reliability based majority logic decoding algorithm is similar to message passing decoding but it sends only the most reliable field message and is considered as the low computational algorithm. In the iterative majority logic decoding (MLgD) algorithm for non-binary LDPC codes, each symbol is iteratively updated by extrinsic information-sums (EXIs) with the most reliable field element along each edge of the Tanner graph. The iterative reliability based hard

The associate editor coordinating the review of this manuscript and approving it for publication was Petros Nicopolitidis.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

(IHRB) and soft (ISRB) MLgD algorithms in [10] have low complexity but show better performance only for parity check matrices with high column weights. These two algorithms in [10] are further improved by introducing soft reliability information at the initialization to achieve better trade-off between complexity and bit error performance by proposing several modifications in [11] and [12]. To overcome this drawback, an improvement to these algorithms is presented in [13] by introducing reliability updates in terms of bit rather than symbols. The bit reliability based decoding algorithm for NB-LDPC is comparatively more efficient and is termed as weighted bit reliability based (wBRB) algorithm [14]. This bit reliability based algorithm requires integer and Galois field operations only which helps to reduce the hardware implementation complexity and processing time. To further enhance the wBRB decoding algorithm, a full bit-reliability based decoding scheme is proposed in [14]. To lower the processing complexity, this algorithm uses the binary representation of non-zero entries of the parity check matrix.

However, the hardware implementation of the NB-LDPC codes is still a challenge. To achieve hardware friendly architecture, a reduced complexity decoder architecture via layered decoding of LDPC codes is presented in [15]. A more hardware friendly high-throughput quasi-cyclic LDPC codes based on layered decoding scheme is presented in [16] for binary LDPC codes [17], [18]. Later this concept has also been adopted in [19] and [20] for non-binary LDPC decoding algorithms.

In a layered scheme, the LDPC decoding algorithm processes information by dividing check nodes or variable nodes of the Tanner graph. Basically, this scheme divides the Tanner graph into subgraphs which are termed as layers [20], [21], [22]. Interestingly, both the vertical and horizontal processing of the layers achieve the same goal of fast convergence and better performance. Fully serialization of the LDPC codes can restrict the throughput and, thus, partial serial architecture, in which each layer contains more than one column or row, for VLSI implementation is presented in [21]. Standard parallel message passing LDPC decoder [23] normally takes much more iterations for decoding process to converge which causes high decoding delay. Secondly, desired LDPC code can have large codeword length which can be difficult for hardware implementation in a fully parallel manner. The aim of the layered LDPC decoder [21] is to increase the convergence speed and facilitate the hardware implementation.

The standard sum-product algorithm (SPA) updates the check node at k^{th} iteration from the information values calculated at the previous iteration, $(k-1)^{th}$, while in the shuffle decoding technique, certain values could already be computed based on a partial computation of values at variable nodes. These partial computed information can be used for the check node update at the same k^{th} iteration instead of previous values calculated at $(k-1)^{th}$ iteration. This method provides the shuffling of the check and variable nodes and is called a group shuffle decoding [19], [24].

Other than the message-passing decoding, a simplified method to correct an error in the received sequence is the symbol flipping (SF) decoding algorithm. In the SF decoding, an unreliable symbol position is detected and a correct symbol value is chosen for that position. A majority logic decision based algorithm is presented in [25], where a symbol position to be flipped is determined by majority decision while the flipped value is obtained from channel output by flipping individual bits of a symbol with low reliability. Another algorithm termed as weighted algorithm B (wt.Algo B) presented in [26] introduces the binary Hamming distance and plurality logic for performance improvement. The parallel symbol flipping decoding (PSFD) algorithm in [27] flips multiple symbols per iteration and has introduced the concept of voting for each variable node. The maximum value of the flipping function is used in conjunction with the voting value of the corresponding variable nodes to identify a symbol to be flipped. The PSFD has shown good performance only for parity check matrix of large column weight. A multiple voting based PSFD (MV-PSFD) algorithm is proposed in [28] to improve bit error rate performance of the PSFD algorithm. The MV-PSFD algorithm has implied a method of multiple voting levels from each failed checksum to the corresponding variable nodes. The performance of a symbol-reliability based message-passing decoding (SRBMP) decoding algorithm [29] has been improved by multiple voting symbol flipping (MV-SF) decoding algorithm [30]. The MV-SF only passes the most reliable Galois field elements from each variable node to the corresponding check node. Therefore, a list of test vectors is required to compute the flipping function for each check node. MV-SF has shown an improved performance but due to the test vectors, its complexity increase with an increase in the size of the Galois field. Recently, low complexity voting-based symbol flipping decoding algorithms are presented in [31] and [32] which can flip single as well as multiple symbols per iteration and have shown better bit error rate performance.

The method of adaptive layering [19] based on the most unreliable VNs, can be explored for possible utilization in a symbol flipping NB-LDPC decoding algorithm to improve performance as well as convergence speed. To address the low complexity implementation of the SF decoding algorithms, an adaptive group shuffled and a fixed group-based symbol flipping decoding algorithm is proposed. The proposed methods have two main advantages over other NB-LDPC algorithms. One advantage is the hardware-friendly simple architecture and second is the fast convergence due to multiple symbols flipping per iteration which reduces the overall decoding computation. The followings are the main contributions of this paper:

1) The article introduces a new method for adaptively grouping variable nodes by utilizing the cumulative

distribution function of the received bit sequence, which has been corrupted by Gaussian random noise. This approach combines the reliability of variable nodes based on majority voting. It differs from the layered decoding method presented in [21] and the shuffled decoding technique presented in [19]. The proposed method adaptively selects variable nodes in each group based on their reliability. Groups with lower reliability are decoded first, and then, three levels of voting are defined to determine whether to select the next group for processing or terminate and move on to the next iteration. Voting and variable node reliability have not been used in the literature to form and process groups until now. In [19], the non-binary sum-product algorithm decodes the least reliable variable nodes in a group, and variable nodes that meet certain conditions are excluded from the group.

2) The main challenge in the conventional symbol flipping decoding algorithm is the computation of the flipping function, which leads to high computational complexity. To address this issue, an adaptive group-based symbol flipping algorithm is proposed. In this approach, the flipping function is only computed for the selected group, which reduces processing latency. This differs from the traditional SF decoding presented in [31] and [32]. A group is selected for decoding based on certain conditions, especially after the first iteration. Therefore, this technique helps to reduce overall computation as not all groups are selected for decoding. Additionally, this adaptive grouping scheme aims to improve the bit error rate (BER) performance, especially at low signal to noise ratio (SNR) regions.

3) To reduce computational complexity in the symbol flipping decoding, a fixed grouping technique is employed. Variable nodes are evenly divided among a predetermined number of groups. Unlike the fixed grouping technique presented in [21], this proposed technique only processes a group if it contains variable nodes that are erroneous.

The rest of the paper is divided into sections as follows. Section II briefly explains the existing layered and grouping schemes in literature. The proposed group based symbol flipping decoding algorithms are presented in section III. Section IV is based on the results and discussion of the existing and proposed methods. Finally, Section V gives the conclusion of the paper.

II. BACKGROUND

A. SYSTEM MODEL

Consider a regular parity check matrix, denoted as H with dimension $m \times n$ defined over the Galois field (GF) size $q > 2$. Each element $h_{i,j}(1 \leq i \leq m, 1 \leq j \leq n)$ of H belongs to $GF(q)$ where $q = 2^r$ and r represents the number of bits in each symbol. Now, let's think of a specific type of NB-LDPC code $\mathcal{C}(N, \gamma, \rho)$ of length N which is designed for a regular parity check matrix H. This code has a length of N for a parity check matrix of H, each column has a constant weight of γ and each row has a constant weight of ρ. In the Tanner graph representation of this LDPC code with parity check matrix H, the non-zero entries ($h_{i,j} \neq 0$) show the i^{th} check node (CN) connected to the j^{th} variable node (VN) and the Tanner graph edge connection is given by $M(j) = \{i : 1 \leq i \leq m, h_{i,j} \neq 0\}$ and $N(i) = \{j : 1 \leq j \leq n, h_{i,j} \neq 0\}$.

Now, consider a codeword $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_j, \ldots, \boldsymbol{c}_n)$ in $\mathcal{C}$. The binary representation of the j^{th} symbol in $\boldsymbol{c}$ is $\boldsymbol{c}_j = (c_{j,1}, c_{j,2}, \ldots, c_{j,t}, \ldots, c_{j,r}), 1 \leq t \leq r$. This binary sequence is subjected to binary phase shift keying (BPSK), where 1 is modulated as $+1$ and 0 is modulated as -1. Consequently, the BPSK modulated j^{th} symbol $\boldsymbol{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,t}, \ldots, x_{j,r})$ is transmitted over an additive white Gaussian noise (AWGN) channel. At receiver, the j^{th} symbol $\boldsymbol{y}_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,t}, \ldots, y_{j,r})$ is obtained from the transmitted sequence $\boldsymbol{x}_j$ by adding the additive white Gaussian channel noise sequence $\boldsymbol{n}_j$, with $\mathcal{N}(0, \sigma^2)$ distribution for its samples $\boldsymbol{y}_j$, represented as $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j$. Finally, the hard decision (HD) binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is determined as:

$$z_{j,t} = \begin{cases} 0, & \text{if } y_{j,t} < 0 \\ 1, & \text{if } y_{j,t} \geq 0 \end{cases} \tag{1}$$

B. GROUP FORMATION

In literature, Group-Shuffled [21], [24] sum-product decoding algorithms divide either check nodes or variable nodes of the relevant bipartite graph into small sub-groups called layers. Also each main iteration is broken into multiple sub-iterations. The group might have one or more than one rows and columns. In some methods, a quasi cyclic parity check matrix is divided horizontally into groups in such a manner that each group has exactly one non-zero entry in each column. This is the most common method used for sum-product LDPC decoder hardware implementation. For the variable nodes $V_1, V_2, \ldots, V_j, \ldots, V_n$, let G_τ be a group, then a set of groups G is given by:

$$G = \{G_1, G_2, \ldots, G_\tau, \ldots, G_\alpha\}, \ 1 \leq \tau \leq \alpha \tag{2}$$

The n number of variable nodes are divided into α groups and each group is comprised of $n/\alpha = \beta$ variable nodes for $1 \leq j' \leq \beta$ where j' is the variable node index within the group. For $\alpha = 1$, the group based decoding becomes the standard flooded schedule SPA.

It is observed that in one iteration the SPA algorithm can fully process in parallel while the shuffle SPA decoding process message passing in serial. This parallel shuffle scheme for LDPC code is called group shuffle decoding. In this algorithm, the code length is divided into specified groups where in each group the updating of messages is completed in parallel while inside the group, information is processed serially. The group shuffle decoding is summarised as:

- Within the same group, message passing is processed in parallel.
- Between the groups, message passing is processed in serial.

The variable nodes can be divided into groups of different sizes based on the local girth and edges of Tanner graph [24] which increases the bit error performance and convergence speed. The local girth of a variable node is used for the appropriate partition.

A dynamically re-grouping of VNs in each iteration based on simple binary and integer decision is presented in [19]. The following metric decides the selection of VNs from the index set variable nodes with least reliable decision and high probability to be corrected during updates:

$$E_j = \Omega_j(\sum_{i \in M(j)} \boldsymbol{s}_i) \tag{3}$$

where $\Omega_j(x) = \frac{x}{\gamma(j)}\gamma_{max}$ and $\gamma_{max} = \max_j[\gamma(j)]$. For $\mathcal{C}$ to be an irregular code, then $\gamma(j)$ is the number of 1's in j^{th} column and γ_{max} is the maximum number of 1's in the codeword $\mathcal{C}$. This equation (3) can be associated with the flipping function of the Gallager algorithm B and it gives the number of failed check-sum. The flipping metric is defined as:

$$F_j = \sum_{i \in N(j)} q_{(i,j)}\boldsymbol{s}_i \tag{4}$$

where

$$q_{(i,j)} = \begin{cases} 1, & \text{if } \max\limits_{j' \in N(i)} E_{j'} = E_j \text{ and } E_j \geq \hbar \\ 0, & \text{else} \end{cases} \tag{5}$$

Here, F_j counts the number of unsatisfied check-sums and $\hbar$ is an integer optimized numerically. This method of identifying the most unreliable VNs can be explored for possible utilization in the NB-LDPC to improve performance as well as convergence speed. In literature, Group-Shuffled [21], [24] sum-product decoding algorithms divide either check nodes or variable nodes of the relevant bipartite graph into small sub-groups called layers. Also each main iteration is broken into multiple sub-iterations. The group might have one or more than one rows and columns. Also the quasi cyclic parity check matrix is divided horizontally into groups in such a manner that each group has exactly one non-zero entry in each column.

In layered decoder, extrinsic information is exchanged more sufficiently due to division of main iteration into sub-iteration. Therefore, this scheme results in more accurate decision and fast convergence as compared to non-layered decoding techniques. In the group decoding algorithm, groups are processed sequentially and VNs belonging to the same group are updated in parallel. These grouping techniques are developed for the message passing sum-product algorithm and have not yet applied to the symbol flipping decoding technique.

III. PROPOSED ADAPTIVE GROUP SHUFFLED SF DECODING

In this section, a new dynamically re-grouping of variable nodes in each iteration using bit reliability and majority voting of individual received symbol is presented for the class of symbol flipping decoding of NB-LDPC codes. The VNs achieving least reliability are grouped together and then variable nodes with second level of reliability are grouped and so on with the last group comparatively being more reliable. This grouping is accomplished by taking cumulative density function (CDF) of the received sequence which is being contaminated by Gaussian random noise. In this algorithm, the number of variable nodes in a group are selected according to the reliability value of each variable node. Secondly, a fixed grouping scheme is applied to the symbol flipping decoding algorithm in which the variable nodes are divided equally among predefined number of groups. During the sub-iterations, a group is selected for decoding if it contains a variable node connected to a failed check.

A. ADAPTIVE GROUP SHUFFLING

For the given parity check matrix H defined over $GF(q)$, the received sequence is divided into groups such that each group contains variable nodes based on the bit reliability of each symbol.

Variable nodes are grouped based on the reliability of the channel received soft and hard decision information. A symbol is considered to be less reliable if it has a smaller reliability value and vice versa. The following equation is used to compute reliability of individual symbol.

$$R_j^{(k)}(\boldsymbol{z}_j^{(k)}, \boldsymbol{y}_j^{(k)}) = \sum_{t=1}^{r}(2z_{j,t}^{(k)} - 1)y_{j,t}^{(k)} \tag{6}$$

Here, $R_j^{(k)}$ calculates the reliability information of a j^{th} symbol at k^{th} iteration for the hard decision symbol sequence $\boldsymbol{z}_j^{(k)}$ and the channel soft information $\boldsymbol{y}_j^{(k)}$. A symbol is considered to be less reliable if the numerical value of $R_j^{(k)}(\boldsymbol{z}_j^{(k)}, \boldsymbol{y}_j^{(k)})$ is smaller. As the received bit sequence is Gaussian random, the individual bit reliability is combined as a symbol reliability using equation (6).

It is important to mention that each group can have different number of variable nodes. The group containing the variable nodes with maximum votes is considered to be the most unreliable. After decoding of the first group, the next group is decoded under the defined criterion.

In this paper, a group priority scheme is used to speed up the convergence. A priority is given to a group which contains variable nodes with most probable erroneous tentative decision and are more likely to be corrected. In (2), the reliability of a group increases from G_1 to G_α. Based on preset threshold, the variable nodes with minimum reliability are grouped as G_1, and variable nodes with the second reliability are grouped in G_2 and so on. A group can be chosen randomly in the case that there is a tie among groups.

To group variable nodes, assume that the cumulative density function of R_j is known and let this be F_{R_j} and the probability density function (PDF) be f_j.

Definition: Use inverse of F_{R_j} for grouping and let α be the total number of groups. Then the number of variable nodes are allocated to the group τ if and only if the following condition holds:

$$F_{R_j}\left(1-\frac{\tau}{\alpha}\right) < R_j \leq F_{R_j}\left(1-\frac{\tau-1}{\alpha}\right) \tag{7}$$

In this section, majority voting scheme presented in [32] and [31] is explained where each unsatisfied check node gives one vote to the relevant variable node. The j^{th} variable node then collects all the votes, say $V_j^{(k)}$, from the failed check nodes at k^{th} iteration.

$$V_j^{(k)} = \sum_{i \in M(j)} V_{i,j}^{(k)} \tag{8}$$

where $V_{i,j}^{(k)} = 1$ if $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, otherwise $V_{i,j}^{(k)} = 0$. Those variable nodes fulfilling the condition $V_j \geq V_{th}$ will be passed to calculate the flipping function. This reduces the computational complexity from n number of variable nodes to just few variable nodes, say $\boldsymbol{\delta}^{(k)}$. Here $\boldsymbol{\delta}^{(k)}$ stores all the positions of those variable nodes having votes greater than the predefined threshold V_{th} and $\delta_{j'}^{(k)}$ is the position of the j^{th}variable node. In other words, $\boldsymbol{\delta}^{(k)}$ stores the positions of all the variable nodes having less reliable information and must be replaced with reliable symbols from $GF(q)$.

After decoding of the first least reliable group, the decoder has to take decision whether to terminate the sub-iteration and go to main iteration or continue to decode next group. To make this decision, average voting of the variable nodes in that group is computed again. The average voting before and after flipping of the group are compared in order to decide whether to move to next group or terminate sub-iteration and shift to next main iteration. If the average voting of the group after flipping is less than before flipping, then next group is selected for decoding based on the defined criterion. Therefore, three types of average voting of the variable nodes are required.

Average voting of each group before flipping are stored until the group processing is terminated for next iteration. However, average voting of each group is re-computed only after symbol flipping in that particular group. Average voting of a group remains the same if no symbol is flipped in a group. At each decoding iteration, three types of voting values are computed as follow:

$\bar{V}_\tau$: Average votes of each group. This voting is fixed after grouping the VNs till next iteration.

$\hat{V}_\tau$: Average votes for group G_τ after decoding.

$\ddot{V}_\tau$: Average votes for group G_τ after decoding of group $\tau - 1$.

Let $\beta^{(k,\tau)}$ be the number of variable nodes in a group τ. Then, the voting based reliability of each group τ at k^{th} iteration is determined by the following method:

$$\bar{V}_\tau^{(k)} = \frac{V_\tau^{(k)}}{\beta^{(k,\tau)}} \tag{9}$$

where $V_\tau^{(k)}$ is the sum of votes for a group τ.

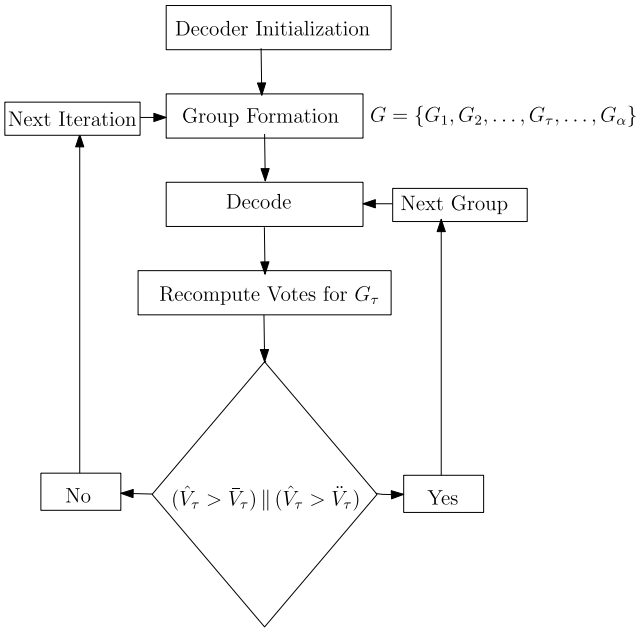


FIGURE 1. Flow chart of group decoding.

Let τ be a group to be decoded, then a next group $G_{\tau+1}$ is selected for decoding based on voting using the following criterion:

$$\tau = \begin{cases} G_{\tau+1}, & \text{if } (\hat{V}_\tau > \bar{V}_\tau) \parallel (\hat{V}_\tau > \ddot{V}_\tau) \\ \text{Terminate sub-iteration}, & \text{otherwise.} \end{cases} \tag{10}$$

Here, $\|$ is used for logical OR operator. The flow chart for equation (10) is shown in Figure 1. Algorithm 1 outlines the procedure for implementing the adaptive group shuffling technique.

Algorithm 1 Adaptive Group Shuffling

1. Set τ=1 to α.
2. Compute average voting $\bar{V}_\tau$ for each group using equation (9).
3. Decode group τ.
4. Compute votes $\hat{V}_\tau$ after flipping.
5. If the condition in equation (10) is false then terminate sub-iterations.
6. Set $\tau = \tau + 1$. If $\tau = \alpha$, terminate sub-iteration, else go to 2.

B. BIT RELIABILITY BASED SF DECODING

In this section, each group is decoded using bit reliability based symbol flipping decoding algorithm (B-MSFD) [31]. The flipping function is computed only for the variable nodes in the selected group τ which results in reduced computational complexity.

The symbol position that needs to be inverted should be part of the failed checks' set. If the flipping position doesn't fall within the set of erroneous variable nodes $\mathbf{T}$.

Then, a variable node is chosen at random from the set of erroneous VNs. This random selection ensures that the decoder's flipping operation alters function values to prevent infinite loops [4], [31], [32]. The i^{th} syndrome for the j^{th} hard decision symbol at the k^{th} iteration can be expressed as:

$$\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j^{(k)}. \tag{11}$$

A received codeword is considered valid when $\boldsymbol{s}_i^{(k)}$ equals $\mathbf{0}$ for the i^{th} check node. The reason for the check sum failure, denoted as $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, can be attributed to the involvement of variable nodes with errors. These check sums that have failed are stored in a variable υ. For $1 \leq i \leq m$, this can be expressed as:

$$\upsilon = \{i | \, \boldsymbol{s}_i \neq \mathbf{0}\}. \tag{12}$$

The variable nodes contributing to the successful check sum are stored in a variable $\bar{\upsilon}$. For $1 \leq i \leq m$, this can be represented as follows:

$$\bar{\upsilon} = \{i | \, \boldsymbol{s}_i = \mathbf{0}\}. \tag{13}$$

Assume that ψ is a vector representing the positions to be flipped; it must be a subset of $\mathbf{T}$ as specified in equation (14). Since $\bar{\upsilon}$ comprises all the correct variable nodes contributing to $\boldsymbol{s}_i^{(k)} = \mathbf{0}$, the variable nodes responsible for the failed checks, to which ψ must belong as a subset, are defined for $j \in N(i)$ in the following manner:

$$\psi \in \mathbf{T} = N(i) \backslash (\upsilon \cap \bar{\upsilon}). \tag{14}$$

Equation (14) contains all the variable nodes contributing to the failed checks.

To incorporate the dependability stemming from the Galois field's structure, we define a vector of extrinsic weighting coefficients denoted as $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_k, \ldots, \theta_r]$. In this context, θ_k represents the extrinsic weighting factor corresponding to $d(\boldsymbol{z}^{(k)} j, \sigma^{(k)} i, j) = k$ for values of k within the range of $0 \leq k \leq r$. When $d_{\theta(\boldsymbol{z}^{(k)} j, \sigma^{(k)} i, j)} \geq 2$, it contains limited valuable information and carries a notably low probability of being corrected [31]. Consequently, the distribution of the weighting coefficients is adjusted to $\theta = [\theta_0, \theta_1, \theta_2]$ for the summation of extrinsic information. These coefficients are subsequently optimized through simulation.

Let j' be the symbol in a group τ. Then, the flipping function $E_{j'}^{(k,\tau)}(\boldsymbol{z}_{j'}^{(k,\tau)})$ to find unreliable variable node position $P_{j'}^{(k,\tau)}$ is computed for the group τ as follows:

$$E_{j'}^{(k,\tau)}(\boldsymbol{z}_{j'}^{(k,\tau)}) = \boldsymbol{z}_{j'}^{(k,\tau)} \odot \boldsymbol{y}_{j'}^{(k,\tau)} + \sum_{i \in M_{j'}} \theta_{d(\boldsymbol{z}_{j'}^{(k,\tau)}, \sigma_{i,j'}^{(k,\tau)})} \tag{15}$$

Here, $\odot$ is the binary operator. Let $\beta^{(k,\tau)}$ be the size of group τ which shows the total number of variable nodes in that group.

$$P_{j'}^{(k,\tau)} = \underset{1 \leq j' \leq \beta^{(k,\tau)}}{\arg\min} \{E_{j'}^{(k,\tau)}(\boldsymbol{z}_{j'}^{(k,\tau)})\} \tag{16}$$

The flipping position $P_{j'}^{(k)}$ must be a member of $\mathbf{T}$ as specified in equation (14); otherwise, a variable node is randomly selected from the set of erroneous variable nodes mentioned in (14).

Once the least reliable flipping position is chosen, the symbol value at that position is updated by flipping the erroneous bit in the symbol. The absolute values of the channel's soft information are referred to as reliability information, where a bit with the smallest absolute value within a symbol is considered to be erroneous, as explained in [25]. The smaller the absolute value of a bit, the lower its reliability, and conversely. The formula for calculating the absolute value of a bit in the j'^{th} received symbol is provided as follows:

$$\acute{y}_{j',t}^{(k,\tau)} = \underset{1 \leq t \leq r}{\arg\min} |y_{j',t}^{(k,\tau)}| \tag{17}$$

The corresponding bit $\acute{z}_{j',t}^{(k,\tau)}$ in the hard decision symbol $z_{j'}^{(k,\tau)}$ is then flipped. The expression for the flipped bit is written as:

$$\acute{z}_{j',t}^{(k,\tau)} = \begin{cases} 1, & \text{if } z_{j',t}^{(k,\tau)} = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Corresponding to the hard decision bit flipping in a symbol, the channel soft reliability information $\boldsymbol{y}_{j'}^{(k,\tau)}$ is also updated using equation:

$$y_{j',t}^{(k,\tau+1)} = \begin{cases} -1 - y_{j',t}^{(k,\tau)}, & \text{if } z_{j',t}^{(k,\tau)} = 0 \\ 1 + y_{j',t}^{(k,\tau)}, & \text{if } z_{j',t}^{(k,\tau)} = 1. \end{cases} \tag{19}$$

The new hard decision symbol sequence is updated for the next $(\tau+1)^{th}$ sub-iteration as:

$$\boldsymbol{z}^{(k,\tau+1)} = (\boldsymbol{z}_1^{(k,\tau)}, \boldsymbol{z}_2^{(k,\tau)}, \ldots, \acute{\boldsymbol{z}}_{j'}^{(k,\tau)}, \ldots, \boldsymbol{z}_{\beta^{(k,\tau)}}^{(k,\tau)}) \tag{20}$$

Algorithm 2 presents the steps for computing the proposed A-BMFSD technique.

C. FIXED GROUP SF DECODING

In this section, the concept used in [15], has applied to the symbol flipping decoding algorithm. For the given parity check matrix H defined over $GF(q)$, variable nodes are divided into groups such that each group contains more than one variable node and the sizes of all the groups are equal. For the variable nodes $v_1, v_2, \ldots, v_j, \ldots, v_n$, let G_τ be a group of H as shown in Figure 2. Then, a set of the groups G is given by:

$$G = \{G_1, G_2, \ldots, G_\tau, \ldots, G_\alpha\} \tag{21}$$

for $1 \leq \tau \leq \alpha$ and α shows the total number of groups. It should be noted that in the proposed fixed group decoding, only those groups are chosen for decoding which have erroneous variable nodes and result in failure of check-sum.

This method helps to decrease overall decoding complexity as all the groups are not computed specially for iteration $k > 1$. The group containing all the reliable variable nodes are not included in the computation of flipping function and symbol value selection. The proposed grouping scheme

Algorithm 2 Proposed Adaptive Group Shuffled SF (A-BMSFD)

1. Initialization: For $1 \leq j \leq n$, $1 \leq t \leq r$ and $1 \leq i \leq m$. The hard decision binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is obtained by (1).
2. Create CDF values.
3. For $k = 1$ to I_{max}.
4. Check if $\boldsymbol{s}^{(k)} = \boldsymbol{0}$ or if $k = I_{max}$, declare $\boldsymbol{z}^{(k)}$ as codeword and stop decoding.
5. Map CDF values to the reliability values R as in (7) to form groups from low to high reliability.
6. Compute average voting $\bar{V}_\tau$ for each group using equation (9).
7. For $\tau = 1$ to α.
8. Find those VNs contributing to failed checks-sum in G_τ by using (12), (13) and (14).
9. Select candidate symbol value by using (15),(11) and (17).
10. Update HD sequence and the soft information by using (18), (19) and (20) respectively.
11. Calculate votes $\hat{V}_\tau$ after flipping.
12. If the condition in equation (10) is false, then go to 14.
13. Set $\tau = \tau + 1$. If $\tau = \alpha$, terminate sub-iteration, else go to 8.
14. Set $k = k + 1$. Go to 4.

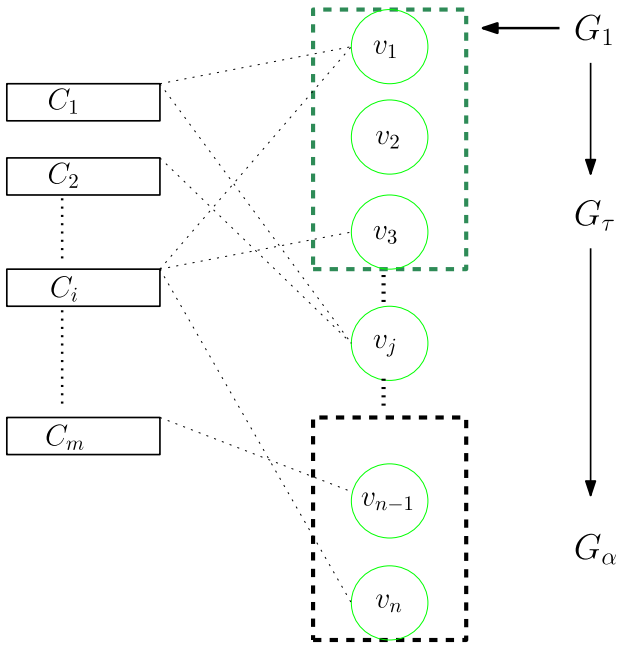


FIGURE 2. Vertical group decoding.

results in reducing the decoder computational burden. Based on the following condition, a new set of groups $\bar{G}$ is formed as follows:

$$\bar{G} = \{G_\tau : G_\tau \cap T \neq \phi, 1 \leq \tau \leq \alpha\}, \tag{22}$$

where ϕ is an empty set. Let $c^{(k)}$ is the number of groups such that $c^{(k)} \leq \alpha$. Then the new set of groups $\bar{G}$ contains $c^{(k)}$ number of groups at k^{th} iteration which might be less than or equal to α and each group size is fixed as $\beta_F = \frac{n}{\alpha}$. Figure 2 shows the vertical processing order of the proposed scheme.

To decide the number of symbols to be flipped per group, let Υ be the preset threshold, then it can be determined in combination to the column weight γ and the number of the variable nodes contributing to the failed checks [31]. If ξ is the total number of symbols to be flipped per iteration then it can be given as follows:

$$\xi = \begin{cases} \xi_1, & \text{if} \quad \Upsilon \geq \varepsilon_1 \rho \\ \xi_2, & \text{else} \end{cases} \tag{23}$$

where ε_1 is an integer value. Based on the value of ξ, the number of flipping positions is determined. ξ_2 is normally kept as 1 to avoid decoder oscillation. The proposed decoding algorithm for fixed grouping is given as Algorithm 3.

Algorithm 3 Fixed Group SF (F-BMSFD)

1. Initialization: For $1 \leq j \leq n$, $1 \leq t \leq r$ and $1 \leq i \leq m$. The hard decision binary sequence $\boldsymbol{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,t}, \ldots, z_{j,r})$ for the j^{th} received symbol $\boldsymbol{y}_j$ is obtained by (1).
2. Divide the parity check matrix H into groups using (2).
3. For $k = 1$ to I_{max}.
4. Check if $\boldsymbol{s}^{(k)} = \boldsymbol{0}$ or if $k = I_{max}$, declare $\boldsymbol{z}^{(k)}$ as codeword and stop decoding.
5. Find erroneous VNs by using (12), (13) and (14).
6. Select groups to decode by using (22).
7. For $\tau = 1$ to $c^{(k)}$ where $c^{(k)}$ is the number of groups having at least one erroneous VN.
8. Select candidate symbol value by using (15),(11) and (17).
9. Update HD sequence and the soft information by using (18), (19) and (20).
10. Set $\tau = \tau + 1$. If $\tau = c^{(k)}$, then terminate sub-iteration, else go to 8.
11. Set $k = k + 1$. Go to 4.

IV. COMPLEXITY ANALYSIS

The decoding computational complexity and memory requirement of the proposed algorithms in comparison with existing B-MSFD decoding algorithm are evaluated in this section. The computational complexity of the algorithms are explained in terms of numerical computation per iteration. The flipping function for the proposed adaptive group decoding is computed for the selected group only. Let α be the total number of groups and $\beta^{(k,\tau)}$ is the size of a group τ at k^{th} iteration, where $\beta^{(k,\tau)}$ varies each time with k and τ. Let $\mathcal{C}$ be a NB-LDPC code over $GF(q) = 2^r$, for a given regular parity check matrix H of size $m \times n$ where each column of H has constant weight of γ and each row of H has constant weight of ρ. The total number of edges of the Tanner graph

is $\delta = n\gamma$. The main complexity of the algorithm is due to the computation of the flipping function in the equations (15) and (11). To convert the received sequence into hard decision binary sequence and then to symbol sequnce $\boldsymbol{z}$ at the decoder initialization, the algorithms require nr real comparisons.

Let b and $c^{(k)}$ be the converging values such that $1 \leq k \leq b$ and $1 \leq \tau \leq c^{(k)}$, then to compute the syndrome check-sum for valid codeword, $n\gamma$ Galois field multiplications and $n\gamma - m$ Galois field additions are required. For each transmitted code-word, the numerical value of the variables k and τ changes. b is the convergence value for a particular transmitted code-word of length n. Similarly, the numerical value of vairable $c^{(k)}$ changes during each iteration and might not obtain the maximum value α. To get the extrinsic information-sums (EXIs) for the n symbols, the decoder requires $n\gamma$ Galois field multiplications and $n\gamma$ Galois field additions. But in the case of the layered decoder, the EXIs are computed for $\beta^{(k,\tau)}$ number of variable nodes in a group τ at the k^{th} iteration. Then, $\beta^{(k,\tau)}\gamma$ Galois field additions and multiplications are required. To compute the reliability of j^{th} symbol, r real comparisons are required. Voting for each group is obtained after syndrome check-sum computation.

To get votes for a particular symbol j, $\gamma - 1$ additions are required. For the code-word of symbols n, a total of $n\gamma - m$ additions are required. This is the additional computation in the proposed adaptive layered decoding (A-BMSFD) algorithm in comparison to F-BMSFD and B-MSFD algorithms. The complexity of the proposed A-BMSFD algorithm involves computation of the binary Hamming distance $d(\boldsymbol{z}_j^{(k,\tau)}, \sigma_{i,j}^{(k,\tau)})$ for each edge, the binary function $\boldsymbol{z}_j^{(k,\tau)} \odot \boldsymbol{y}_j^{(k,\tau)}$ for $\beta^{(k,\tau)}$ number of symbols. To compute reliability $R_j^{(k)}$, r real additions are required. To form predefined number of groups α based on the reliability, n comparisons are required. Similarly, to compute the votes for each group, $\beta^{(k,\tau)}(\gamma - 1)$ integer additions are required. To compute the Hamming distance $d(\boldsymbol{z}_{j'}^{(k,\tau)}, \sigma_{i,j'}^{(k,\tau)})$ of a symbol j' in a group τ, $\beta^{(k,\tau)}\gamma$ integer comparisons are required and to compute summation of $\theta^{(k,\tau)}_{d(\boldsymbol{z}_{j'}^{(k,\tau)}, \sigma_{i,j'}^{(k,\tau)})}$, $\beta^{(k,\tau)}(\gamma - 1)$ additions are required. To find the position of least reliable symbol $P_{j'}^{(k,\tau)}$, $\beta^{(k,\tau)}$ comparisons are required and $\xi^{(k,\tau)}r$ comparisons are required to flip a bit in $\xi^{(k,\tau)}$ number of symbols. After symbol flipping, to move to next group or exit to main iteration $k+1$, $\alpha - 1$ comparisons are required. Since the proposed A-BMSFD algorithm does not need any loop detection procedure as in B-MSFD algorithm, the decoder computational complexity is reduced. For the F-BMSFD algorithm, the size of each group β_F is fixed and also the grouping is not adaptive, therefore, it does not require voting and reliability information. In comparisons to B-MSFD algorithm, the group based decoding reduces the computation of the flipping function as it computed for the number of variable nodes in a group instead of whole code-word n.

The proposed algorithms for NB-LDPC codes also have an advantage of reduced memory consumption for storing the values of the flipping function as well as flipped values due to the fact that computation is done for the selected group only. However, the values for the CDF are required to store for the allocation of VNs in the next iteration which increases the memory size. Let r bits be required to store each of the symbol over $GF(q)$ and b-bit be required for storing floating-point values for each of the reliability metric, then these algorithms require nrb bits to store the received sequence. Similarly, the hard decision symbol sequence $\boldsymbol{z}_{j'}^{(k,\tau)}$ requires nr bits and the extrinsic information-sums $\sigma_{i,j'}^{(k,\tau)}$ requires $\beta^{(k,\tau)}\rho r$ bits.

To store the weighting coefficients $\theta^{(k,\tau)}_{d(\boldsymbol{z}_{j'}^{(k,\tau)}, \sigma_{i,j'}^{(k,\tau)})}$ of the binary Hamming distance, ρb bits are required to store the values. To keep each value of the flipping function in (15), a b bits memory is required and $\beta^{(k,\tau)}rb$ bits for all the values. Similarly, $\beta^{(k,\tau)}b$ and $\xi^{(k,\tau)}r$ bits of memory are required for storing $P_{j'}^{(k,\tau)}$ and $\hat{y}_{j',t}^{(k,\tau)}$ respectively. The proposed algorithms reduce the memory requirement from n to $\beta^{(k,\tau)}$. The A-BMSFD algorithm requires only r bits to store the value for the flipped symbol.

Table 1 shows the complexity of the proposed adaptive and fixed group based decoding algorithms in comparison to B-MSFD algorithm. From the table, it can be observed that algorithms A-BMSFD, F-BMSFD and B-MSFD algorithms have similar complexity in the case all the groups are decoded. Since the overall complexity of an algorithm depends on the converging value b, the effect of number of iterations k has been shown in computing numerical values of various decoding operations. The converging value of k varies for each code-word transmission and also for each value of SNR.

Also in case of adaptive group shuffled decoding, the values of $c^{(k)}$ and $\beta^{(k,\tau)}$ are not fixed and can be as small as 1, while in the fixed layered decoding algorithm, $c^{(k)}$ can be less than α as those groups having no erroneous variable nodes are not included in decoding.

V. RESULTS AND PERFORMANCE ANALYSIS

In this section, the proposed adaptive group shuffled SF algorithm is compared with the flooding schedule SF algorithms. The algorithms in the following examples for the NB-LDPC code (n, γ, ρ) with γ and ρ as the column and row weights respectively, have been simulated using BPSK as the modulation technique. In the following examples, the total number of iterations are kept as $I_{max} = 15$. The performance and processing complexity of the proposed groups based algorithms are compared with flooding schedule SF algorithm in the literatures [31] and [32]. An all-zeros codeword is transmitted over additive white Gaussian channel using binary phase shift keying modulation. 1 is modulated as $+1$ and 0 is modulated as -1. The CDF values are generated once for each value of SNR. Both $y - axis$ and $x - axis$

TABLE 1. Computational complexity for various decoding algorithms.

Algorithms	Number of operations			
	GA	GM	IA/RA	RC/IC
A-BMSFD	$\sum_{k=1}^{b}(n\gamma - m) + \sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta^{(k,\tau)}\gamma)$	$\sum_{k=1}^{b}(n\gamma) + \sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta^{(k,\tau)}\gamma)$	$\sum_{k=1}^{b}(n\gamma - m) + \sum_{k=1}^{b}\sum_{\tau=1}^{c(k)}(r+\beta^{(k,\tau)}(\gamma-1)+\xi^{(k,\tau)}r)$	$nr + \sum_{k=1}^{b}((n) + \sum_{\tau=1}^{c^{(k)}}(\beta^{(k,\tau)}r + (\alpha - 1) + \beta^{(k,\tau)} + \xi^{(k,\tau)}r)$
F-BMSFD	$\sum_{k=1}^{b}(n\gamma - m) + \sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta_F\gamma)$	$\sum_{k=1}^{b}(n\gamma) + \sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta_F\gamma)$	$\sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta_F(\gamma-1) + r + \xi_{\tau}^{(k)}r)$	$nr + \sum_{k=1}^{b}\sum_{\tau=1}^{c^{(k)}}(\beta_F(r + 1) + \alpha\beta_F + \xi_{\tau}^{(k)}r)$
B-MSFD	$\sum_{k=1}^{b}(2nr - m)$	$\sum_{k=1}^{b}(2nr)$	$\sum_{k=1}^{b}(nr + r + \xi^{(k)}r)$	$nr + \sum_{k=1}^{b}(n(r+1) + \xi^{(k)}r)$

IC/IA: Comparision/Adittion;
RA/RC: Real Addition/Comparision; GA/GM: Galois Field Addtion/Multiplication;
Notations used: $\delta = n\gamma = m\rho$.

values are rounded to two digits for simplicity and for easy mapping. The weighting coefficients $\theta = [\theta_0, \theta_1 \ldots, \theta_r]$ of the proposed A-BMSFD as well as B-MSFD algorithm [31] are set as $\theta = [2, 0.75, 0.5]$ for $d_{\theta(\mathbf{z}_j^{(k,\tau)}, \sigma_{i,j}^{(k,\tau)})} \geq 2$ as it has very low probability to be corrected and does not carry much useful information [13]. Also, the values of θ should be in a decreasing order of reliability information. The higher value means more reliability and vice versa. It should be noted that in the following examples, the proposed algorithms flip single symbol per group. The number of symbol per group for the B-BMSD algorithm is also set as 1. In the following examples, the maximum group size of the proposed decoding algorithms are kept as 6.

Example 1: In this example, the proposed decoding algorithms are simulated using a half rate quasi-cyclic (QC) NB-LDPC code (120,3,6) over $GF(128)$, whose column and row weights are 3 and 6, respectively. The performance of the proposed algorithms are also compared with B-MSFD algorithm in the literature. The BER performance of the proposed A-BMSFD and F-BMSFD algorithms show better BER performance than B-BMSFD algorithm as illustrated by performance curves in Figure 3.

The performance gap between non-layered B-MSFD and the proposed layered algorithms is around $0.5dB$ at the BER of 10^{-3}. BER curves show that the proposed adaptive group shuffled based NB-LDPC decoding algorithm has an improved performance than the proposed fixed group decoding algorithm.

Example 2: In this example, the NB-LDPC code (105,2,5) over $GF(8)$ is used to demonstrate the BER performance of the proposed decoding algorithms in comparison to B-MSFD. Here, the parity check matrix H with constant row weight $\gamma = 2$ is based on the construction method presented in [33] and the alist format is available on the web site [34]. In this example, the code-word length is 105 and this cannot be divided equally into groups. This is a special case of F-BMSFD algorithm and in this case, the even three groups have one more variable node and the group size is 18, while the odd three groups contain 17 variable nodes. Thus, the first odd layer contains 17 variable nodes and the second even layer contains 18 variable nodes and so on.

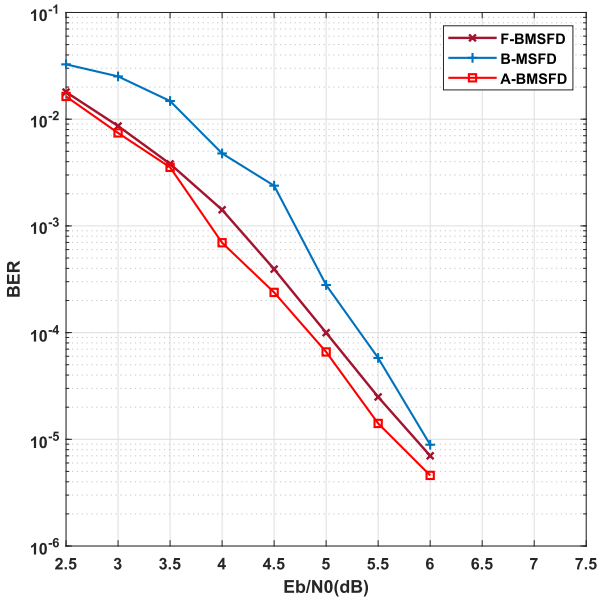


FIGURE 3. BER performance of NB-LDPC code (120, 3, 6) over $GF(128)$.

From the BER curves shown in Figure 4, it is observed that the proposed adaptive group shuffled decoding algorithm performs better than flooded schedule B-MSFD algorithm at higher SNR. From the curves, it is also clear that these algorithms have less performance for ultra low column parity check matrices. The reason is that these algorithms use the technique to isolate the variable nodes contributing to failed checks and for the ultra low column weight matrices, it becomes difficult to isolate the variable nodes in error by comparing the set of VNs contributing to failed and successful check nodes. In this example, as shown in Figure 4, the proposed A-BMFSD algorithm has shown better BER performance in comparison to F-MSFD and B-MSFD algorithms.

Example 3: In this example, Figure 5 shows the symbol error rate performance (SER) of the proposed algorithms in comparison to non-layered symbol flipping decoding algorithm. The non-binary LDPC decoding algorithms are simulated for a regular QC NB-LDPC code (120,4,8) over

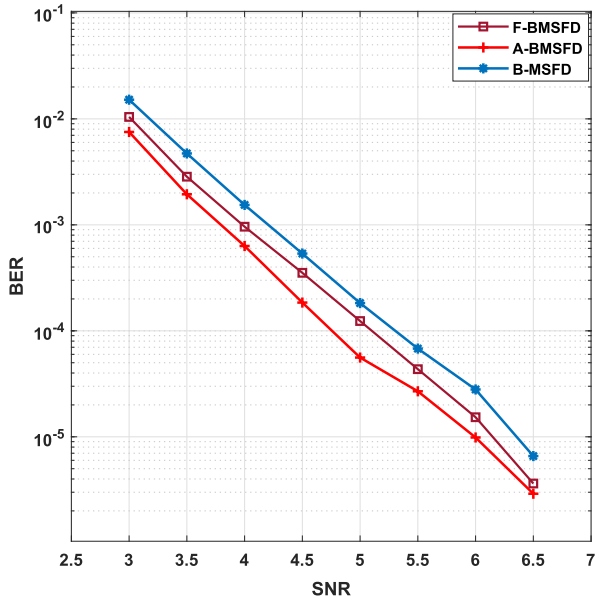


FIGURE 4. BER performance of NB-LDPC code (105, 42) over $GF(8)$.

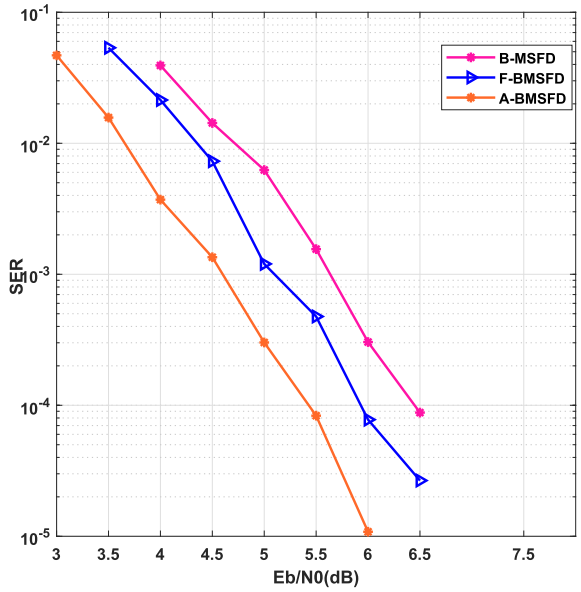


FIGURE 5. SER performance of NB-LDPC code(120, 4, 8) over $GF(128)$.

$GF(128)$ to get the SER performance curves. This code has a parity check matrix with constant column weight $\gamma = 4$ and constant row weight $\rho = 8$. The SER performance curves show similar trends as the BER performance curves.

From the symbol error rate performance curves in the Figure 5, it is clear that the proposed decoding algorithms SER performance is also better than that of the B-MSFD algorithm. From the Figure 5, the proposed F-BMSFD algorithm SER performance at $\approx 10^{-3}$ is better than B-MSFD algorithm by approximately 0.5dB. Similarly, at SER=$\approx 10^{-3}$, the coding gain for A-BMSFD algorithm compared to F-MSFD algorithm is about $0.5dB$. The proposed A-BMSFD algorithm has shown better SER performance in comparison to F-BMSFD as well as B-MSFD algorithms. In this example, it can be noted that the proposed grouping based decoding performance is better for the relatively high column weigh of the NB-LDPC codes. In the symbol flipping decoding algorithm, the complexity is, mainly, given by the computation of the flipping function. The proposed A-BMSFD algorithm takes advantage of the adaptive grouping and computes the flipping function for the variable nodes contained in the selected group τ only.

The reason for the better BER performance of the A-BMSFD algorithm is the reliability and voting based additional information to identify the position of least reliable variable node. In the adaptive scheme, the most unreliable variable nodes are grouped together. This benefits the flipping function to identify the flipping position more accurately as compared to the B-MSFD algorithm.

Example 4: A half rate QC NB-LDPC code (120, 3, 6) over $GF(128)$ is used whose column and row weights are 3 and 6,

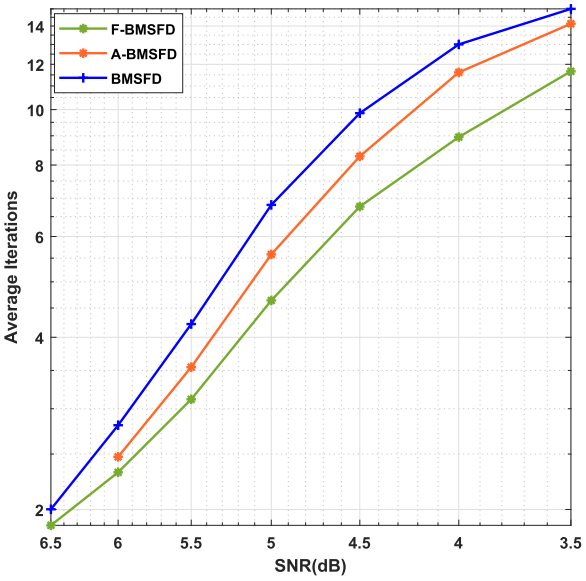


FIGURE 6. SNR versus average number of iterations for NB-LDPC code (120, 3, 6) over $GF(128)$.

respectively. The average iteration (b) versus SNR is given in Figure 6 to illustrate the complexity and convergence of the proposed NB-LDPC decoding algorithm in comparison to B-MSFD algorithm in the literature.

For the average number of iteration at 5.5dB, the numerical computation complexity of the considered algorithms are listed in Table 2. In the B-MSFD and F-BMSFD algorithms, as the number of symbols to be flipped in each iteration and group, are not fixed and can vary each time. Therefore, to compute numerical complexity, the number of symbol flipped per group is fixed as 1 for all these algorithms.

TABLE 2. Numerical complexity of various algorithms at 5.5dB for code(120, 3, 6) over *GF*(128).

Algorithms	Number of operations			
	GA	GM	RA/IA	IC/RC
A-BMSFD	1478	1704	2252	1460
F-BMSFD	1339	1534	2210	3488
B-MSFD	6807	7059	3588	4903

Remember that B-MSFD algorithm is using the code-word as single group. Looking at curves in Figure 6, the trend line of the convergence of B-MSFD algorithm is slower than adaptive layered and fixed layered algorithms. At 5.5dB, the average number of iteration of the adaptive layered decoder is getting close to B-MSFD algorithm. Since at higher SNR, there are less number of errors in the received sequence, therefore the convergence of the three algorithms are getting closer at 6.5dB. The primary reason for fast convergence of the fixed layered decoding algorithm is due to the identification of erroneous variable nodes in each group. In a F-BMSFD algorithm, a group is processed only if it contains variable nodes which is useful in identifying a least reliable symbol, but this in turn increase the number of integer comparisons as it is obvious from the IC/RC section of Table 2. This feature of the F-BMFSD algorithm increases the complexity by 40% from A-BMSFD algorithm, but 16% less than B-MSFD algorithm.

In Table 1, computational complexity of the algorithms are listed while in the Table 2, the computations are given as empirical values which are computed at 5.5dB. In the Table 2, the computation are carried out for NB-LDPC code (120, 3, 6) over $GF(128)$ and the average computations are obtained by repeating the code-word transmission for 1000 times. From the Table 2, it is clear that complexity of the newly proposed algorithms is much lower than the existing symbol flipping B-MSFD algorithm in literature. The total number of operations for the proposed NB-LDPC algorithms as well as for the B-MSFD algorithm are given in Table 1.

The empirical data in Table 2 is obtained by computing the statistical average of the numerical values of $c^{(k)}$ and $\beta^{(k,\tau)}$.

The numerical result in Table 2 are dependant on the variables listed in Table 1. In case of A-BMSFD algorithm, the numerical values depend on b which is the maximum value attained at which the algorithm converges, $c^{(k)}$, $\beta^{(k,\tau)}$ and $\xi^{(k,\tau)}$. The convergence of the proposed decoding algorithms depends on b which is the main iteration loop while $c^{(k)}$ varies at every k^{th} iteration and shows the complexity of the sub-iteration loop. Table 2 clearly demonstrates that the proposed A-BMSFD algorithm is numerically less complex compared to the other algorithms listed. The size of the specific group τ, denoted as $\beta^{(k,\tau)}$, affects both the decoding latency and numerical complexity. This observation is further supported by the curves depicted in Figure 6. From the complexity curves in Figure 6, the average value of b for B-MSFD algorithm is around 4.2. The total number of variable nodes for B-MSFD algorithm is 120. By replacing the variables in Table 1 by numerical values to compute GA, the number of operations are computed as 6804 which are approximately the same as the empirical values in Table 2.

Similarly, for the F-BMSFD algorithm, the value of b is 3.2 as illustrated in Figure 6. The value of $c^{(k)}$ is obtained as 2.69 by comparing the values in Table 1 and Table 2 for $b = 3.2$ and $\beta_F = 20$. These values are in close approximation to get the empirical result in Table 2. The average number of iterations for A-BMFSD algorithm is $b = 4.12$ as shown in Figure 6, while the empirical value of group size is $\beta^{(k,\tau)} = 14.4$. By replacing these values in the Table 1 and comparing with Table 2, the value of $c^{(k)}$ is computed as 1.35. From the Table 2, it is observed that the proposed adaptive layered decoder has almost 64% less computational complexity than B-MSFD algorithm in terms of GF additions and multiplications, but approximately 4% more than fixed layered decoding algorithm. As the extrinsic information sum is computed for the selected number of variable nodes in a group, i.e β_F and $\beta^{(k,\tau)}$ for F-BMSFD and A-BMSFD algorithms respectively, the number of computation for the proposed algorithms are much less than B-MSFD algorithm. Secondly, the proposed algorithms converge faster than B-MSFD algorithm and, thus, the numerical complexity is computed for less number of iterations.

The proposed adaptive group A-BMSFD algorithm has extra integer additions to compute voting of each symbol in the received hard decision sequence. Therefore, it has an increased complexity in comparison to the fixed group F-BMSFD decoding algorithmic as shown in Table 2. From Table 2, it can be observed clearly that the complexity due to IA/RA of the adaptive layered decoder is around 22% less than that of B-MSFD algorithm and almost similar to the F-BMSFD algorithm.

The methods were evaluated for their time consumption using a Dell Optiplex 7780 AIO PC equipped with an Intel Core i7-10700 CPU and 16.0 GB of RAM in order to demonstrate the algorithm's complexity. MATLAB 2021 and Windows 10 were used as simulation software. In the case of the A-BMSFD algorithm, the time required for each iteration to compute from equation 15 to equation 20 was measured at 1.82 seconds. In contrast, the BMSFD algorithm took 2.07 seconds per iteration for an NB-LDPC code (120, 3, 6) over GF(128). Similarly, for the same code, the F-BMSFD algorithm took 1.87 seconds per iteration.

The average number of iterations of the layered decoding algorithms are less than non-layered symbol flipping decoding and it has an improved BER performance. Therefore, the proposed schemes provide an appealing trade-off between BER and complexity.

VI. CONCLUSION

The paper introduces two layered decoding algorithms for symbol flipping NB-LDPC codes, which employ grouping techniques to lower the computational complexity of the flipping function for the received symbols. These techniques help to reduce the decoder processing latency when selecting

new candidate symbol values to correct errors in the message. The CDF based adaptive grouping scheme applied to the SF decoding algorithm helps to reduce numerical computation and also results in an improved BER performance. The proposed A-BMSFD algorithm uses the reliability of the received bit sequence and also takes into account the voting based reliability of each variable node. The voting based reliability informations are used to process the next group or terminate the sub-iterations. In the fixed decoding algorithm, each group contains a fixed number of variable nodes and the group is processed only if it has some erroneous variable nodes. The proposed two grouping methods can be applied to quasi-cyclic as well as non-quasi-cyclic NB-LDPC codes. The numerical results show that the proposed grouping schemes have efficiently reduced the decoder computation and also achieved better bit error rate performance. Compared to non-layered symbol flipping decoding, the average number of iterations in the layered decoding algorithm is lower and has a better BER performance. Therefore, the proposed method provides an attractive trade-off between BER and complexity.

VII. ACKNOWLEDGMENT

The authors would like to thank Telkom South Africa and Sentech SOC Ltd., for sponsoring this research. They are also thankful to the anonymous reviewers and the editor for their useful suggestions to improve this manuscript.

REFERENCES

[1] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[2] W. Ullah and A. Yahya, "Comprehensive algorithmic review and analysis of LDPC codes," *TELKOMNIKA Indonesian J. Electr. Eng.*, vol. 16, no. 1, p. 111, Oct. 2015.

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

[4] W. Ullah, L. Cheng, and F. Takawira, "Predictive syndrome based low complexity joint iterative detection-decoding algorithm for non-binary LDPC codes," *IEEE Access*, vol. 9, pp. 33464–33477, 2021.

[5] W. Ullah, D. N. K. Jayakody, J. Li, and Y. Chursin, "Iterative joint detection-decoding algorithms using Euclidean distance-based feedback," in *Proc. 10th Int. Conf. Inf. Autom. Sustainability (ICIAfS)*, Aug. 2021, pp. 19–24.

[6] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proc. IEEE Inf. Theory Workshop*, Apr. 2003, pp. 70–73.

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

[8] E. Boutillon and L. Conde-Canencia, "Bubble check: A simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *Electron. Lett.*, vol. 46, no. 9, p. 633, 2010.

[9] C. Marchand, E. Boutillon, H. Harb, L. Conde-Canencia, and A. A. Ghouwayel, "Extended-forward architecture for simplified check node processing in NB-LDPC decoders," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2017, pp. 1–6.

[10] C.-Y. Chen, Q. Huang, C.-C. Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3140–3147, Nov. 2010.

[11] C. Xiong and Z. Yan, "Improved iterative soft-reliability-based majority-logic decoding algorithm for non-binary low-density parity-check codes," in *Proc. Conf. Rec. 45th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2011, pp. 894–898.

[12] S. Yeo and I.-C. Park, "Improved hard-reliability based majority-logic decoding for non-binary LDPC codes," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 230–233, Feb. 2017.

[13] Q. Huang, M. Zhang, Z. Wang, and L. Wang, "Bit-reliability based low-complexity decoding algorithms for non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 62, no. 12, pp. 4230–4240, Dec. 2014.

[14] Q. Huang and S. Yuan, "Bit reliability-based decoders for non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 64, no. 1, pp. 38–48, Jan. 2016.

[15] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop onSignal Process. Syst.*, Oct. 2004, pp. 107–112.

[16] K. Zhang, X. Huang, and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 985–994, Aug. 2009.

[17] M. Sarvaghad-Moghaddam, W. Ullah, D. N. K. Jayakody, and S. Affes, "A new construction of high performance LDPC matrices for mobile networks," *Sensors*, vol. 20, no. 8, p. 2300, Apr. 2020.

[18] W. Ullah, T. Jiang, F. Yang, and S. M. Aziz, "Two-way normalization of min-sum decoding algorithm for medium and short length low density parity check codes," in *Proc. 7th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Sep. 2011, pp. 1–5.

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

[20] Y.-L. Ueng, C.-Y. Leong, C.-J. Yang, C.-C. Cheng, K.-H. Liao, and S.-W. Chen, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 2, pp. 385–398, Feb. 2012.

[21] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[22] Y.-L. Ueng, K.-H. Liao, H.-C. Chou, and C.-J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using max-log-QSPA," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2940–2951, Jun. 2013.

[23] O. Abassi, L. Conde-Canencia, A. Al Ghouwayel, and E. Boutillon, "A novel architecture for elementary-check-node processing in nonbinary LDPC decoders," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 2, pp. 136–140, Feb. 2017.

[24] A. Manada, K. Yoshida, H. Morita, and R. Tatasukawa, "A grouping based on local girths for the group shuffled belief propagation decoding," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2133–2136, Nov. 2016.

[25] B. Liu, J. Gao, G. Dou, and W. Tao, "Weighted symbol-flipping decoding for nonbinary LDPC codes," in *Proc. 2nd Int. Conf. Netw. Secur., Wireless Commun. Trusted Comput.*, vol. 1, Apr. 2010, pp. 223–226.

[26] K. Jagiello and W. E. Ryan, "Iterative plurality-logic and generalized algorithm B decoding of q-ary LDPC codes," in *Proc. IEEE Inf. Theory App. Workshop*, Feb. 2011, pp. 1–7.

[27] C.-C. Huang, C.-J. Wu, C.-Y. Chen, and C.-c. Chao, "Parallel symbol-flipping decoding for non-binary LDPC codes," *IEEE Commun. Lett.*, vol. 17, no. 6, pp. 1228–1231, Jun. 2013.

[28] N.-Q. Nhan, T. M. N. Ngatched, O. A. Dobre, P. Rostaing, K. Amis, and E. Radoi, "Multiple-votes parallel symbol-flipping decoding algorithm for non-binary LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 6, pp. 905–908, Jun. 2015.

[29] C. Chen, B. Bai, X. Ma, and X. Wang, "A symbol-reliability based message-passing decoding algorithm for nonbinary LDPC codes over finite fields," in *Proc. 6th Int. Symp. Turbo Codes Iterative Inf. Process.*, Sep. 2010, pp. 251–255.

[30] F. Garcia-Herrero, D. Declercq, and J. Valls, "Non-binary LDPC decoder based on symbol flipping with multiple votes," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 749–752, May 2014.

[31] W. Ullah, L. Cheng, and F. Takawira, "Low complexity bit reliability and predication based symbol value selection decoding algorithms for non-binary LDPC codes," *IEEE Access*, vol. 8, pp. 142691–142703, 2020.

[32] W. Ullah, L. Cheng, and F. Takawira, "Prediction and voting based symbol flipping non-binary LDPC decoding algorithms," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–6.

[33] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2,dDesign of regular (2,d_c)-LDPC codes over GF(q) using their binary images c)-LDPC codes over GF(q) using their binary images," *IEEE Trans. Commun.*, vol. 56, no. 10, pp. 1626–1635, Oct. 2008.

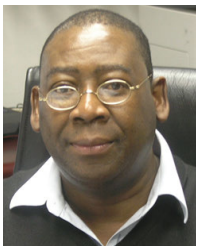
[34] [Online]. Available: https://perso-etis.ensea.fr/~declercq/graphs.php



WAHEED ULLAH (Senior Member, IEEE) received the B.Sc. degree in electrical and electronics engineering from the University of Engineering and Technology Peshawar, Pakistan, the master's degree in communication and information systems from Nanjing University of Aeronautics and Astronautics, China, in 2012, and the Ph.D. degree from the University of the Witwatersrand, South Africa, in 2022. He has published several papers in high quality journals and conferences of well reputations. His research interests include wireless communication systems and networks, LDPC decoding algorithms, SWIPT, coding-based secure communication, NOMA, massive MIMO, and OTFS modulation. He is the Chair of the Industrial Relation Coordination, IEEE Islamabad Section.



LING CHENG (Senior Member, IEEE) received the B.Eng. degree from the Huazhong University of Science and Technology, in 1995, the M.Ing. degree, in 2005, and the D.Ing. degree from the University of Johannesburg, in 2011. He is currently a Full Professor with the University of the Witwatersrand (Wits), Johannesburg, South Africa. He has published over 150 papers in journals and conference proceedings. His research interests include telecommunications and artificial intelligence. He is the Vice-Chair of the IEEE South African Information Theory Chapter.



FAMBIRAI TAKAWIRA (Member, IEEE) received the Ph.D. degree from the University of Cambridge, U.K., in 1986. From 2012 to 2016, he was the Head of the School of Electrical and Information Engineering, University of the Witwatersrand, South Africa. Prior to that, he was a Professor and the Dean of the Faculty of Engineering, University of KwaZulu-Natal, Durban, South Africa. He is currently a Professor with the University of the Witwatersrand. His research interests include wireless communication systems and networks. From 2012 to 2013, he was a member of the COMSOC Board of Governors. From 2012 to 2017, he was a member of the COMSOC Nominations and Elections Committee. From 2013 to 2017, he was a member of the COMSOC GIMS Committee. He was a member of the technical committee at several IEEE AFRICON conferences. He has also served as the General Co-Chair for IEEE-AFRICON 2015 and the Executive Co-Chair for the International Conference on Communications 2010 (ICC 2010), Cape Town, South Africa. He has been an active volunteer in IEEE and ComSoc. He was the Director of Europe Middle East and Africa Region, from 2012 to 2013. He was an Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, from 2008 to 2010. He was a ComSoc Treasurer, from 2018 to 2021.