IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Low Complexity Bit Reliability and Predication Based Symbol Value Selection Decoding Algorithms for Non-Binary LDPC Codes

WAHEED ULLAH[1], (Member, IEEE), LING CHENG[2], (Senior Member, IEEE) and FAMBIRAI TAKAWIRA.[3],(Member, IEEE)

[1,2,3]School of Electrical and Information Engineering, University of the Witwatersrand (e-mail: {waheed.ullah, ling.cheng, fambirai.takawira}@wits.ac.za)

Corresponding author: Waheed Ullah (e-mail: waheed.ullah@wits.ac.za).

ABSTRACT The main challenge for hardware implementation of non-binary LDPC decoding is the high computational complexity and large memory requirement. To address this challenge, five new low complexity LDPC decoding algorithms are proposed in this paper. The proposed algorithms are developed specifically towards the low complexity, yet effective, decoding of the NB LDPC codes. The proposed decoding algorithms update, iteratively, the hard decision received vector to search for the valid codeword in the vector space of Galois field (GF). The selection criterion for least reliable symbol positions is based on the information from the failed checks and the reliability information from the Galois field structure as well as from the received channel soft information. To choose the correct value for the candidate symbol, two methods are used. The first method is based on the prediction of the error symbol from the set of Galois field symbols which maximize an objective function. In the second method, individual bits are flipped based on the reliability information obtained from the channel. Algorithms 1 and 2 flip a single symbol per iteration whilst the other three algorithms 3,4 and 5 flip multiple symbols in each iteration. The proposed voting based Algorithms 1,2 and 5 first short list the unreliable positions using a majority voting scheme and then choose the candidate symbol value from the set of the symbols in $GF(q)$ while not violating the field order q. These methods simplify the decoding complexity in terms of computation and memory. Results and analysis of these algorithms show an appealing tradeoff between computational complexity and bit error rate performance for NB LDPC codes.

INDEX TERMS Multiple Vote, Non-binary LDPC, Iterative reliability decoding, Symbol Flipping, Sum Product Algorithm, Low Complexity Decoding

I. INTRODUCTION

RELIABLE and efficient communication depends on the performance of forward error correction (FEC) codes. Among error correction codes, non-binary (NB) LDPC codes have shown an excellent bit error rate (BER) performance, especially in comparison to binary LDPC codes [1]–[3]. An important feature of NB LDPC is that they have good performance for short and medium length codes [4]. The performance of LDPC codes depend on the type of decoders used. One of the good decoders is the $q-$ary sum-product algorithm (QSPA) decoder which passes a vector of q probability messages over each edge of the Tanner graph, representing the probability of all q elements of $GF(q)$. However, the high check node computational complexity of the QSPA decoder is a major obstacle to their finding a place in practical applications. After the invention of the NB LDPC codes [1], most of the research has focused on how to reduce the computational complexity of the check node processing. To lower the complexity of QSPA, a Fast Fourier transform based SPA (FFT-SPA) algorithm was proposed in [5] to reduce the check node computational complexity from order of $O(q^2)$ to $O(q\,log\,q)$ for each check node update. Other low complexity algorithms called extended min-sum (EMS) [6], [7], trellis based EMS [8], bubble check EMS [9] and min-max algorithms [10] were proposed but they have a performance degradation in comparison to QSPA. Further,

the complexity of the computations of all these algorithms is still too high for hardware implementation. On the other hand, reliability based message passing algorithms [11]–[14] and the symbol flipping algorithms [15], [16] are simple and computationally very fast for NB LDPC codes but at a cost of reduced performance.

Reliability based majority logic decoding algorithms offer low complexity as they send only the most reliable field message and in this respect are similar to message passing decoding algorithms. In the iterative majority logic decoding (MLgD) algorithm for non-binary LDPC codes, each symbol is iteratively updated by the extrinsic information-sums (EXIs) with the most reliable field element along each edge of the Tanner graph. The iterative reliability based hard (IHRB) and soft (ISRB) majority logic decoding (MLgD) algorithms [17] have lower complexity but suffer from performance degradation. Some performance improvements to IHRB-MLgD and ISRB-MLgD algorithms have been presented in [11]–[14]. To improve the performance of IHRB-MLgD algorithm, an enhanced (E-IHRB-MLgD) algorithm is presented in [12] by introducing soft reliability at the initialization and re-computing the extrinsic-information. The enhanced IHRB algorithm has the same complexity as that of IHRB except at the initialization but it has an improved performance which is close to ISRB algorithm. The majority logic decoding algorithms are particularly suitable for parity check matrices with high column weight, constructed based on finite fields [18] and finite geometries [19]. These algorithms suffer from performance loss for codes with small column weight.

Better performing, low complexity message passing decoding algorithms were presented in [20]. These algorithms introduce reliability updates in terms of the bit rather than symbols and are termed as bit reliability based (BRB) algorithms. The BRB algorithm require integer and Galois field operations only, which reduces the computational complexity. The performance of the BRB algorithm is improved by multiplying the extrinsic information-sum by Hamming distance based weighted coefficients. This weighted BRB (wBRB) algorithm shows improved performance over ISRB and BRB and does not suffer from error floor for parity check matrices with small column weight. A full bit-reliability based decoder [21] is the binary form of weighted BRB as this algorithm uses the binary representation of the non-zero entries of the parity check matrix to facilitate hardware implementation.

Symbol flipping (SF) decoding algorithms have low complexity as compared to the bit reliability based decoding algorithms [20], [21] and the q-ary belief propagation [1] [5] message passing algorithms but at the cost of reduced bit error rate performance. In the symbol flipping decoding algorithm presented in [22], the least reliable position is determined by majority decision, while the flipped symbol value is computed from the reliability of the received bits. The weighted algorithm B (wt.Algo B) in [15] introduces the binary Hamming distance and plurality logic to improve performance. The parallel symbol flipping decoding (PSFD) algorithm [16] uses majority voting to get better flipping decision but this algorithm performs better only if the parity check matrix has a large column weight. The PSFD algorithm is improved further by using multiple votes in [23] and performs better even for the LDPC parity check matrix with small column weight. The non-binary LDPC decoder based on symbol flipping with multiple votes (MV-SF) [24] performs better than MV-PSFD but at the cost of an increased complexity and memory size especially for the higher order Galois field.

The other recently developed symbol flipping algorithms (D-SFDP and P-SFDP) [25] offer better performance than most of the existing symbol flipping algorithms, but a major disadvantage is the exhaustive and computationally complex prediction mechanism. The complexity of these algorithms [25] depend on the size of the Galois field order q and length n of a codeword. A decision-symbol reliability-based SFD (DRB-SFD) algorithm [26] is aimed to show better performance in applications like data storage based on NAND flash memory. Paper [26] is focused on an algorithm for NAND flash memory and the BER performance has not been compared with existing algorithms like MV-SF, MV-PSFD, and D-SFDP.

The objective of this paper is to propose algorithms with good performance and low complexity to fulfill the requirement of low power and efficient memory usage. The proposed algorithms can be divided into three categories; 1) algorithms using voting and prediction [16] [23] [25]; 2) algorithms using voting with channel bit reliability [16] [23] [27] and 3) multiple symbol flipping decoding algorithms based on prediction as well as bit reliability [25] [27].

Algorithm 1 is based on the category 1. In [16] and its improved version [23], the flipping function is computed for all the received symbols and the majority voting scheme is used in parallel to the flipping function to select the flipping position while in the proposed Algorithm 1, the majority voting scheme is used only for short listing the least reliable variable nodes. The flipping function is then computed only for the short listed variable nodes. Algorithm 2 also belongs to category 1, but simplifies the flipping function by using the divide and conquer rule by using the flipping function for identifying first the symbol positions and then finding the candidate symbol value. Algorithm 3 is from category 3 but differs from the algorithm [27] as the proposed Algorithm 3 uses flipping function based on channel reliability information as well as information from Galois field structure while the flipping function in [27] is based on the syndromes and channel likelihood information. Algorithm 4 is also based on category 3 and flips multiple symbols per iteration but it uses the symbol value prediction instead of the bit reliability used in [27]. Algorithm 5 is based on categories 2 and 3. This algorithm uses majority voting scheme for short listing the least reliable symbols unlike in [16] and [23] where they use voting in parallel with the flipping function. The proposed Algorithm 5 flips the unreliable bit of the selected least

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3013814, IEEE Access

reliable symbol instead of the prediction method adopted in [25]. Algorithm 5 also flips multiple symbols per iteration.

The set of failed parity checks and the infinite loop detection procedure are applied only when multiple symbols are flipped in each iteration. In the multiple symbol flipping decoding algorithms, the set of failed parity checks and infinite loop detection procedures restrict the number of symbols to be flipped in consecutive iterations. On the other hand, the proposed algorithms based on the voting scheme except for Algorithm 5, flip single symbol per iteration and do not require information from the failed checks and infinite loop detection procedure.

The rest of the paper is summarized as follows. Section II summaries the relevant literature on NB LDPC decoding algorithms whilst Section III presents the proposed algorithms. Section IV compares the complexity and memory requirement of various algorithms. Section V shows the results and analysis of the existing and proposed algorithms in detail. Section VI gives a conclusion of the paper.

II. SYMBOL FLIPPING DECODING ALGORITHMS

A. PRELIMINARIES

Let $GF(q)$ be the Galois field with q elements. Consider a NB LDPC code $\mathcal{C}$ of length n with a regular parity check matrix H, defined over the $GF(q)$, with m as number of rows and n as number of columns such that each row of H has a constant weight of d_c and each column of H has a constant weight of d_v. Each element $h_{i,j}(1 \leq i \leq m\ , 1 \leq j \leq n)$ of H is an element over the Galois field $GF(q)$ where $q = 2^r$. The non-zero entries of $H(h_{i,j} \neq 0)$ in the Tanner graph show the i^{th} check node (CN) connected to the j^{th} variable node (VN). For the column weight d_v and the row weight d_c, the Tanner graph edge connections are shown as $M(j) = \{i : 1 \leq i \leq m\ ,\ h_{i,j} \neq 0\}$ and $N(i) = \{j : 1 \leq j \leq n, h_{i,j} \neq 0\}$ and for regular matrix $\frac{m}{n} = \frac{d_v}{d_c}$.

Let $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_j, \ldots, \boldsymbol{c}_n)$ be a codeword in $\mathcal{C}$ and the binary representation of 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 modulated with binary phase shift keying (BPSK), where 1 is modulated as +1 and 0 is modulated as -1. 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 additive white Gaussian noise (AWGN) channel. The received binary sequence $\boldsymbol{y}_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,t}, \ldots, y_{j,n})$ is obtained from the transmitted sequence $\boldsymbol{x}_j$ after adding the additive white Gaussian channel noise $\boldsymbol{n} \rightarrow \mathcal{N}(0, \sigma^2)$ with zero mean and two sided power spectral density $N_0/2$ as $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{n}_j$. 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 given by:

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

The hard decision binary sequence $z_{j,t}$ is mapped to an element in $GF(q)$ to obtain the symbol $\boldsymbol{z}_j$. The i^{th} syndrome of the j^{th} hard decision symbol sequence can be defined for $h_{i,j} \neq 0$:

$$\boldsymbol{s}_i = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j \tag{2}$$

Here it is important to mention that the value of $\boldsymbol{s}_i$ is also an element in $GF(q)$. If $\boldsymbol{s}_i \neq 0$, it means that some of the variable nodes contributing to this check node are incorrect.

For the given hard decision symbol $\boldsymbol{z}_j$, the extrinsic information-sum(EXI) denoted as $\sigma_{i,j}$ that is passed from i^{th} check node (CN) to the j^{th} variable node (VN) is given by:

$$\sigma_{i,j}^{(k)} = h_{i,j}^{-1} \sum_{j' \in N(i) \setminus j} h_{i,j'} \boldsymbol{z}_{j'}^{(k)} \tag{3}$$

for $1 \leq i \leq m, j \in N(i)$.

B. SYMBOL FLIPPING NON-BINARY LDPC DECODING ALGORITHMS

The low complexity hard decision symbol flipping decoding algorithm based on the majority logic decision is known as generalized algorithm B [15]. This algorithm is based on plurality logic which counts the number of occurrences of each symbol. Let τ be a predefined threshold, then if the number of occurrence of one finite field symbol over $GF(q)$ exceeds τ, the algorithm chooses that received symbol as reliable, otherwise it chooses the same symbol. This algorithm is further improved by weighting factor $\theta_{d(\alpha, \boldsymbol{z}_j^{(k)})}$ based on the Hamming distance between the finite field symbol and extrinsic information-sum. This improved algorithm is called weighted algorithm B [15]. In this algorithm, the counts of occurrence of a finite field symbol is multiplied by the weighting coefficients assigned to the Hamming distance. The decision is taken based on the largest product. The optimal set of weights was not determined in the weighted algorithm B and is left as future work.

Let $d(\boldsymbol{z}_j, \sigma_{ij},)$ be the binary Hamming distance between EXI $\sigma_{i,j}$ and the hard decision symbol $\boldsymbol{z}_j$, and $\theta_{d(\boldsymbol{z}_j, \sigma_{i,j})}$ be the corresponding weighting factor. Denoting η_α as the number of occurrence of an element $\alpha \in GF(q)$ based on the plurality logic of each element $\sigma_{i,j}$ at the j^{th} variable node, then the weighted algorithm B (wt.Algo B) decision for an estimated correct symbol $v_j^{(k)}$ at k^{th} iteration is obtained as follow:

$$v_j^{(k)} = \begin{cases} \alpha, & \text{if } \eta_\alpha \theta_{d(\alpha, \boldsymbol{z}_j^{(k)})} \geq \tau \\ \boldsymbol{z}_j^{(k)}, & \text{otherwise.} \end{cases} \tag{4}$$

Here α corresponds to the largest product of $\eta_\alpha \theta_{d(\alpha, \boldsymbol{z}_j^{(k)})}$ and the preset threshold τ is determined through simulations.

The concept of hard reliability to improve the performance of the bit reliability based (BRB) decoding algorithm was introduced in [20]. The binary Hamming distance between hard decision symbol sequence $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$ and the extrinsic information-sums (EXIs) indicate the hard reliability of the EXIs. The author in [20] showed that the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3013814, IEEE Access

Hamming distance $d(\boldsymbol{z}_j, \sigma_{i,j})$ indicates the correct probability of extrinsic information-sum $\sigma_{i,j}$. If p_b is raw bit error probability of variable nodes (VNs) over the Galois field GF($q = 2^r$), then the symbol error probability of the individual variable node is $1-(1-p_b)^r$. The relationship between the EXI and Hamming distance [20] is written as:

$$Pr(\sigma_{i,j}|d(\boldsymbol{z}_j,\sigma_{i,j})=u) = \frac{p_b^u q}{p_b^u q-(1-p_b)^{u-r}+(1-p_b)^{-pr+r+u}} \tag{5}$$

for $1 \leq u \leq r$.

Different from the wt.Algo B which only considers the information before flipping, the recent algorithms (D-SFDP and P-SFDP) [25] use both the information before and after flipping. These algorithms also take into account the hard reliability to evaluate correctly the contribution of the unsatisfied checksums. For the received hard decision symbol sequence $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n), (q-1)$ values for each symbol of the sequence is predicted through an exhaustive search for the symbol value which maximizes an objective function. The flipping function derived in [25] predicts symbols for each position of the received hard decision sequence $\boldsymbol{z}_j^{(k)}$ such that $\boldsymbol{z}_j^{(k)} \neq \ddot{\boldsymbol{z}}_j^{(k)}$ and $\ddot{\boldsymbol{z}}_j^{(k)}$ has all possible symbol values of the $GF(q)$ except for the symbol value of $\boldsymbol{z}_j^{(k)} \in GF(q)$. Therefore, there are $q-1$ possible values for each position of $\ddot{\boldsymbol{z}}_j^{(k)}$ which can be written as:

$$\ddot{\Gamma}_j^{(k)} = \{\ddot{\boldsymbol{z}}_j^{(k)} : \ddot{\boldsymbol{z}}_j^{(k)} \epsilon GF(q), \ddot{\boldsymbol{z}}_j^{(k)} \neq \boldsymbol{z}_j^{(k)}\} \tag{6}$$

Define the binary operator for hard decision symbol $\boldsymbol{z}_j^{(k)}$ and its corresponding channel soft symbol information $\boldsymbol{y}_j$ as:

$$\boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j = \sum_{t=0}^{r-1} (2z_{j,t}^{(k)} - 1)\boldsymbol{y}_{j,t} \tag{7}$$

To include the reliability derived from the structure of the Galois field, a vector of extrinsic weighting coefficients are defined as $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_k, \ldots, \theta_r]$. Here θ_k shows the extrinsic weighting factor corresponding to $d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)}) = k$ for $0 \leq k \leq r$. According to equation (5), $d_{\theta(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \geq 2$ does not carry much useful information and it has very low probability to be corrected [20]. According to this, the weighting coefficients are distributed as $\theta = [\theta_0, \theta_1, \theta_2]$ for the extrinsic information-sum and are optimized through simulation.

The flipping function of the D-SFDP algorithm introduced the hard reliability to predict the most reliable candidate symbol and can be written as follow:

$$E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j + \sum_{i \in M_j} \theta_{d(\ddot{\boldsymbol{z}}_j^{(k)}, \sigma_{i,j}^{(k)})} - \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j - \sum_{i \epsilon M_j} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{8}$$

For plurality logic P-SFDP algorithm, the extrinsic weighting coefficients are defined as $\boldsymbol{\eta} = [\eta_0, \eta_1, \ldots, \eta_l, \ldots \eta_{d_c}]$ where η_l shows the weighting coefficient corresponding to the number of occurrence of $\alpha \in GF(q)$ in the extrinsic information-sum $\sigma_{i,j}^k$ for $0 \leq l \leq d_c$. Now equation (8) can be re-written to include the hard reliability, the number of occurrence of the element $\alpha \in GF(q)$, of the j^{th} symbol as follow:

$$E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j + \boldsymbol{\eta}_l(\ddot{\boldsymbol{z}}^{(k)}) - \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j - \boldsymbol{\eta}_l(\ddot{\boldsymbol{z}}^{(k)}) \tag{9}$$

To predict all the $q-1$ values of $\ddot{\boldsymbol{z}}_j^{(k)}$ for each symbol in $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_j, \ldots, \boldsymbol{z}_n)$, the algorithm becomes computationally very complex and also requires more memory to store the processed values. For large values of $GF(q)$ and length of the code n, the algorithm [25] is very slow to find the candidate symbol value for the position to be flipped. The flipping function $E_j^{(k)}$ of the j^{th} symbol and its relevant flipped value $v_j^{(k)}$ are calculated as follows:

$$E_j^{(k)} = \max_{\ddot{\boldsymbol{z}}_j^{(k)} \in \ddot{\Gamma}_j^{(k)}} E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) \tag{10}$$

$$v_j^{(k)} = \underset{\ddot{\boldsymbol{z}}_j^{(k)} \in \ddot{\Gamma}_j^{(k)}}{\operatorname{argmax}} E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) \tag{11}$$

The flipping function $E_j^{(k)}$ is very complex in computation and secondly a large memory is required to store all the $q-1$ predicted values for n symbols.

C. CANDIDATE SYMBOLS SHORT-LISTING BY VOTING

In the voting scheme [16], [23], 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} iterations.

$$V_j^{(k)} = \sum_{i \in M(j)} V_{i,j}^{(k)} \tag{12}$$

where $V_{i,j}^{(k)} = 1$ if $\boldsymbol{s}_i^{(k)} \neq \boldsymbol{0}$, otherwise $V_{i,j}^{(k)} = 0$. For the accumulated voting $V_j^{(k)}$ equal or greater than a predefined threshold V_{th}, those variable nodes fulfilling the condition $V_j \geq V_{th}$ will be passed to calculate the flipping function $E_j^{(k)}$ through equations (7) and (8). Suppose $\boldsymbol{\delta}$ stores the positions of all the variable nodes for $V_j \geq V_{th}$, then the values of $\boldsymbol{\delta}$ at k^{th} iteration can be calculated as follows:

$$\boldsymbol{\delta}_{j'}^{(k)} = V_j^{(k)}, \quad if \quad V_j \geq V_{th} \tag{13}$$

for $1 \leq j' \leq p$ and p shows the total number of the short listed variable nodes. This method reduces the computational complexity from n number of variable nodes to just few variable nodes p. Here $\delta_{j'}^{(k)} \in \boldsymbol{\delta}^{(k)}$ shows the position of each short listed variable node and $\boldsymbol{\delta}^{(k)}$ contains all those positions of the variable nodes. In other words, $\boldsymbol{\delta}^{(k)}$ have all the positions of variable nodes having less reliable information and must be replaced with reliable symbols from $\boldsymbol{z}_{j'}^{*(k)} \in \Gamma_{j'}^{*(k)}$. The multiple voting method presented in [23]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3013814, IEEE Access

defines two voting levels $\zeta_0 > \zeta_1 > 0$, using the same voting function defined in (12). For $s_j^{(k)} \neq 0$, $V_{ij}^{(k)} = \zeta_0$ with the largest flipping function and $V_{ij}^{(k)} = \zeta_1$ for the VN with the second largest flipping function. In the proposed algorithms, a single level voting scheme is used for short listing of least reliable symbols which reduces the memory consumption and computational complexity.

III. PROPOSED ALGORITHMS

A. VOTING BASED SYMBOL VALUE PREDICTION ALGORITHM

In this proposed algorithm, the least reliable variable nodes are short listed by majority voting and the candidate symbol value is selected by prediction method. The prediction based symbol flipping algorithms (D-SFDP and P-SFDP) [25] offer reasonably better performance than most of the symbol flipping algorithms in literature but at the cost of exhaustive and computationally complex prediction mechanism. The other disadvantage of the algorithms in [25] is the high memory requirement to store the matrix of predicted values for all symbols in the codeword. A voting based approach is used in this proposed algorithm, to lower decoding latency and computational power by short listing the symbols with low reliability. The flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ in equation (8) is calculated $q-1$ times for the j^{th} variable node of the codeword n, including the information before and after the flipping to predict correctly the most reliable symbol during each iteration. After calculating $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ for all the variable nodes, a matrix of size $n(q-1)$ is formed and must be stored for further processing.

The complexity of the flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ can be lowered from n to p by first short-listing the candidate symbols through voting as given in equation (11) and (12) respectively. This method also reduces the memory requirement from $n(q-1)$ to $p(q-1)$. This newly improved algorithm is termed as voting based symbol flipping decoding(V-SFD) algorithm. The new rule to calculate the flipping function and the corresponding flipped value for $j' \in \delta^{(k)}$ is given by:

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

$$-\boldsymbol{z}_{j'}^{(k)} \odot \boldsymbol{y}_{j'} - \sum_{i \epsilon M_{j'}} \theta_{d(\boldsymbol{z}_{j'}^{(k)}, \sigma_{i,j'}^{(k)})} \quad (14)$$

The above flipping metric is calculated for the short listed symbols only, resulting in low complexity and memory. The flipping function $E_j^{(k)}$ gives the reliability measure of the j^{th} symbol to be flipped to the value $v_j^{(k)}$.

$$E_{j'}^{(k)} = \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \ddot{\Gamma}_{j'}^{(k)}} E_{j'}^{(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \boldsymbol{z}_{j'}^{(k)}) \quad (15)$$

This flipping function is similar to the maximum likelihood method and the maximum value of the flipping function for all the possible candidate symbols $\ddot{z}_j^{(k)}$ shows the most reliable variable node update.

$$v_{j'}^{(k)} = arg \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \ddot{\Gamma}_{j'}^{(k)}} E_{j'}^{(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \boldsymbol{z}_{j'}^{(k)}) \quad (16)$$

Here j' shows the short listed least reliable variable nodes and the flipping function is calculated only for those selected variable nodes, thus reducing tremendously the computational complexity and memory requirement. In this paper, the Hamming distance based reliability information are used as in D-SFDP algorithm and the plurality logic based reliability information used in the P-SFDP algorithm is left intentionally as intuitively the later will show the same performance with the proposed schemes.

Proposed Algorithm 1 (V-SFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \boldsymbol{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Determine the un-reliable symbol $\delta_{j'}^{(k)}$ using the voting scheme by (12) and (13).
5. Calculate $E_{j'}^{(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \boldsymbol{z}_{j'}^{(k)},)$ for $\boldsymbol{\delta}$ symbols by (14) and then calculate $E_{j'}^{(k)}$ by (15).
6. Find its flipped value $v_{j'}^{(k)}$ by (16) and update $\boldsymbol{z}^{(k+1)}$ with $v_{j'}^{(k)}$.
7. $k = k + 1$ and go to step 3.

B. VOTING BASED SIMPLIFIED SYMBOL VALUE PREDICTION ALGORITHM

This algorithm further reduces the complexity of the flipping function in Algorithm 1 by applying a divide and conquer strategy. Equation (14) is divided into two parts: first the algorithm finds the unreliable positions to be flipped and then predicts the candidate symbol for that position. The unreliable symbol position is determined by the following equation.

$$E_{j'}^{(k)}(\boldsymbol{z}_j^{(k)}) = \boldsymbol{z}_{j'}^{(k)} \odot \boldsymbol{y}_{j'} + \sum_{i \in M_{j'}} \theta_{d(\boldsymbol{z}_{j'}^{(k)}, \sigma_{i,j'}^{(k)})} \quad (17)$$

$$E_{j'}^{(k)} = \min_{1 \leq j' \leq \delta} (E_j^{(k)}(\boldsymbol{z}_{j'}^{(k)})) \quad (18)$$

Equation (18) will determine the most unreliable position to be flipped. From the set $\ddot{\Gamma}_j^{(k)}$ of $q-1$ predicted symbols, a candidate symbol value is selected as the correct symbol which maximize the flipping function in equation (19). After a symbol value is predicted for each unreliable position, then the hard decision sequence is updated with that symbol at every k^{th} iteration.

$$E_{j'}^{*k}(\ddot{\boldsymbol{z}}_{j'}^{(k)}) = \ddot{\boldsymbol{z}}_{j'}^{(k)} \odot y_{j'} + \sum_{i \in M_{j'}} \theta_{d(\ddot{\boldsymbol{z}}_{j'}^{(k)}, \sigma_{i,j'}^{(k)})} \quad (19)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3013814, IEEE Access

After calculating the flipping function $E_{j'}^{*k}(\ddot{\boldsymbol{z}}_{j'}^{(k)})$, the position for the candidate symbols which maximize $E_{j'}^{*k}$ is given by:

$$E_{j'}^{*(k)} = \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \Gamma_{j'}^{(k)}} \{E_{j'}^{*(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}\} \tag{20}$$

The predicted symbol value $\ddot{v}_{j'}^{(k)}$ from the set of the predicted values $\ddot{\Gamma}_{j'}^{(k)}$ is found through the following equation:

$$\ddot{v}_{j'}^{(k)} = arg \max_{\ddot{\boldsymbol{z}}_{j'}^{(k)} \in \Gamma_{j'}^{(k)}} E_{j'}^{*(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)}) \tag{21}$$

The function $E_{j'}^{*(k)}$ in (20), determines the position of the most unreliable j^{th} symbol and equation (21) selects the flipped value $\ddot{v}_{j'}^{(k)}$.

The voting scheme, used in this paper, helps to further reduce the computational complexity of the decoder. The voting scheme also helps in saving the memory consumption as the reliability information related to the selected least reliable variable nodes, are stored for further processing.

Equations (17), and (18) are computed only for $\boldsymbol{\delta}$ positions of the received codeword sequence instead of all the symbols of codeword having length of n. After selecting the most unreliable symbol position, a reliable symbol value from the set of possible predicted $q-1$ values such as $\boldsymbol{z}_{j'}^{(k)} \neq \ddot{\boldsymbol{z}}_{j'}^{(k)}$, is computed using equations (19), (20) and (21) respectively. Since equations (19), (20) and (21) are computed for a single variable node with possible $q-1$ values, the proposed Algorithm 2 further reduces the decoder computational complexity.

Proposed Algorithm 2 (Simplified V-SFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \boldsymbol{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Determine the unreliable symbol $\delta_{j'}^{(k)}$ using the voting scheme by (12) and (13).
5. For $j' \in \boldsymbol{\delta}$, find the unreliable positions $E_{j'}^{(k)}(\boldsymbol{z}_{j'}^{(k)})$ for $\boldsymbol{\delta}$ symbols by (17) and (18).
6. Calculate the predicted candidate symbol position $E_{j'}^{*(k)}(\ddot{\boldsymbol{z}}_{j'}^{(k)})$ by (19) and (20).
7. Find predicted flipped value $\ddot{v}_{j'}^{(k)}$ by (21) and update $\boldsymbol{z}^{(k+1)}$ with $\ddot{v}_{j'}^{(k)}$.
8. $k = k + 1$ and go to step 3.

C. MULTIPLE SYMBOL FLIPPING BIT RELIABILITY BASED DECODING ALGORITHM

In this section, a multiple symbol flipping decoding algorithm for NB LDPC decoder is proposed which flips the unreliable bit of a symbol by using the channel bit reliability information. Variable node short listing by voting is not used in this algorithm. This algorithm finds positions of symbols with less reliability using the information before flipping and then selects the reliable symbol values for those positions based on the information after flipping. The computational cost of this proposed algorithm is smaller than most of the NB LDPC decoding algorithms in literature as this algorithm compare and flips individual bit of a symbol. This algorithm also gives the advantage of multiple symbol flipping in each decoding iteration. The unreliable multiple symbol positions are determined by the following equations:

$$E_j^{(k)}(\boldsymbol{z}_j^{(k)}) = \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j + \sum_{i \in M_j} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{22}$$

$$E_j^{(k)} = \min_{1 \leq j \leq n} \{E_j^{(k)}(\boldsymbol{z}_j^{(k)})\} \tag{23}$$

Let τ be the threshold, then the number of flipping positions can be determined based on the column weight d_v and the number of variable nodes contributing to the failed checks. If ρ is the maximum number of the symbols to be flipped per iteration then it is determined as follows:

$$\rho = \begin{cases} \rho_1 & \text{if} \quad \tau \geq \varepsilon_1 d_v \\ \rho_2 & \text{else} \end{cases} \tag{24}$$

Where ε_1 is an integer value. Equation (24) can also be used directly with $\rho = \beta$ where β is an integer value. Based on the value of ρ, the unreliable multiple flipping positions are determined by (23).

The number of symbols to be flipped is further restricted in the loop detection procedure (34) and secondly if the flipping positions are not in the set of the failed checks. The i^{th} syndrome of the j^{th} hard decision symbol at k^{th} iteration can be defined as:

$$\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j^{(k)} \tag{25}$$

A valid codeword is received if m-tuple $\boldsymbol{s}_i^{(k)} = \boldsymbol{0}$ for $1 \leq i \leq m$. The cause of the failure of check sum is due to the contribution of the variable nodes in error. All those variable nodes are given as follow:

$$\upsilon = \{i : \boldsymbol{s}_i \neq \boldsymbol{0}, 1 \leq i \leq m, j \in M(i)\} \tag{26}$$

The variable nodes contributing to the successful check-sum are given by:

$$\bar{\upsilon} = \{i' : \boldsymbol{s}_i = \boldsymbol{0}, 1 \leq i \leq m, j \in M(i')\} \tag{27}$$

Suppose $\boldsymbol{\psi}$ is the vector containing all the flipping positions found by (23) and $\bar{\upsilon}$ contains all the correct variable nodes contributing to $\boldsymbol{s}_i^{(k)} = \boldsymbol{0}$, then those variable node contributing to the failed checks to which $\boldsymbol{\psi}$ must belong as a subset, are defined as:

$$T = \{\upsilon(j) \backslash \upsilon \cap \bar{\upsilon}\} \tag{28}$$

Equation (28) identifies and removes those variable nodes which are common in equation (26) and (27). This helps to decide the number of flipping positions in $\boldsymbol{\psi}$ as subset

to T. Once the flipping positions are identified, the symbol values are updated by flipping that individual bit of symbol with the smallest absolute value. The absolute channel soft values are termed as reliability information. The smaller the absolute value of the bit, the less is the reliability of that bit and vice versa. For j^{th} received symbols $\boldsymbol{y}_j^{(k)}$, the bit closer to zero or having the minimum absolute value is expressed for $1 \leq t \leq r$ as:

$$\acute{y}_{j,t}^{(k)} = \min_{1 \leq t \leq r} |\boldsymbol{y}_j^{(k)}| \tag{29}$$

The corresponding bit $z_{j,t}^{(k)}$ in the hard decision symbol $z_j^{(k)}$ is then flipped. The expression for the flipped bit is written as:

$$\acute{z}_{j,t}^{(k)} = \begin{cases} 1, & \text{if} \quad z_{j,t}^{(k)} = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Flipping the bit in the hard-decision symbol can cause oscillating behaviour. Therefore, the channel soft reliability information $\boldsymbol{y}_j^{(k+1)}$ is also updated using equation (31) or (32):

$$y_{j,t}^{(k+1)} = -(y_{j,t}^{(k)}) \tag{31}$$

$$y_{j,t}^{(k+1)} = \begin{cases} -1 - y_{j,t}^{(k)}, & \text{if} \quad z_{j,t}^{(k)} = 0 \\ 1 + y_{j,t}^{(k)}, & \text{if} \quad z_{j,t}^{(k)} = 1. \end{cases} \tag{32}$$

In this paper equation (32) is used. The new hard decision symbol sequence will be updated for the next $(k+1)^{th}$ iteration as:

$$\boldsymbol{z}^{(k+1)} = (\boldsymbol{z}_1^{(k)}, \boldsymbol{z}_2^{(k)}, \ldots, \acute{\boldsymbol{z}}_j^{(k)}, \ldots, \boldsymbol{z}_n^{(k)}) \tag{33}$$

If $\boldsymbol{s}_i^{(k)} \neq \mathbf{0}$, then decoding failure due to some of the variable nodes (VNs) contributing to check nodes (CNs) will have occured. The information of the successful and failed syndromes are stored for further processing. Normally if construction of the parity check matrix H is 4 cycles free, then the variable nodes contributing to the failed checks can easily be identified. The process for isolating the incorrect symbols (variable nodes) is done by identifying only those variable nodes contributing to the failure of check nodes. Unique variable node positions contributing to the failure of check nodes are stored and are used further for comparison with the least reliable symbol positions identified through the flipping function.

The proposed multiple symbol flipping algorithm significantly lowers the complexity by selecting multiple positions and then flips the less reliable bit of each of the selected symbols. This newly proposed algorithm is termed as B-MSFD algorithm. Since the prediction mechanism of the D-SFDP algorithm [25] is complex in computation, this method makes an effective tradeoff between complexity and performance.

Proposed Algorithm 3 (B-MSFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$
2. For iteration $k = 1$ to I_{max}
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword
4. Calculate $E_j^{(k)}(\boldsymbol{z}_j^{(k)})$ by (22) and determine the number of symbol positions ρ to be flipped by (24) and calculate $E_\rho^{(k)}$ by (23).
5. Find $\boldsymbol{\psi}^{(k)}$ positions by (28) and then corresponding flipped values by (29) and (30).
6. Update $\boldsymbol{z}^{(k+1)}$ with $\acute{\boldsymbol{z}}_j^{(k)}$ and the soft channel information $\boldsymbol{y}^{(k+1)}$ with $\acute{\boldsymbol{y}}_j^{(k)}$ by (33) and (32) respectively.
7. $k = k + 1$ and go to step 3.

In symbol flipping decoding algorithms, the flipping position might be the same for successive iterations when the decoder is trapped in an infinite loop. To avoid the decoding being trapped in successive iterations, the symbol positions to be flipped are recorded and are compared with the next iteration to make sure that the same symbol is not flipped in consecutive iterations. Therefore a symbol is flipped in iteration k only if

$$\psi_b^{(k)} \neq \psi_b^{(k-1)} \tag{34}$$

for $1 \leq b \leq \rho, k > 1$. At first iteration $k = 1$ all the selected symbols positions will be flipped based on the preset threshold but in successive iterations $(k > 1)$, the number of symbol positions to be flipped depends on condition (34).

D. MULTIPLE SYMBOL FLIPPING PREDICTION BASED ALGORITHM

Different from Algorithm 2 which flips single symbol per iteration and uses the voting scheme to short list the variable nodes before computing the flipping function, this proposed Algorithm 4 flips multiple symbols per iteration and does not short list the least reliable variable nodes for computing the flipping function. This proposed algorithm for non-binary LDPC decoder is primarily divided into two steps. In the first step, the positions of the least reliable symbols are selected and in the second step, the most reliable symbol value for those position are predicted.

In this algorithm, the successful and failed checks are isolated as shown in equations (26),(27) and (28). The set of the variable nodes that contributed to the failed checks, helps to decide which symbol position should be flipped and how many positions needs to be flipped. The unreliable symbols' positions are determined by the following equations.

$$E_j^{(k)}(\boldsymbol{z}_j^{(k)}) = \boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j + \sum_{i \in M_j} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{35}$$

$$E_j^{(k)} = \min_{1 \leq j \leq n} (E_j^{(k)}(\boldsymbol{z}_j^{(k)})) \tag{36}$$

This proposed algorithm is used to flip multiple symbols in each iteration and the number of flipping symbols depend on a pre-defined threshold ρ.

After finding all the unreliable symbol positions ρ, those candidate symbol values are chosen for which the flipping function is maximized. A symbol value is predicted for each of the unreliable positions and then the hard decision sequence is updated with those symbols at every k^{th} iteration.

$$E_j^{*(k)}(\ddot{\boldsymbol{z}}_j^{(k)}) = \ddot{\boldsymbol{z}}_j^{(k)} \odot y_j + \sum_{i \in M_j} \theta_{d(\ddot{\boldsymbol{z}}_j^{(k)}, \sigma_{i,j}^{(k)})} \tag{37}$$

$$E_j^{*(k)} = \max_{\ddot{\boldsymbol{z}}_j^{(k)} \in \Gamma_j^{(k)}} (E_j^{*(k)}(\ddot{\boldsymbol{z}}_j^{(k)})) \tag{38}$$

$$\ddot{v}_j^{(k)} = arg \max_{\ddot{\boldsymbol{z}}_j^{(k)} \in \Gamma_j^{(k)}} E_j^{*(k)}(\ddot{\boldsymbol{z}}_j^{(k)}) \tag{39}$$

In (37), the function $E_j^{*(k)}$ measures the position of the most unreliable j^{th} symbol to be flipped to the value $\ddot{v}_j^{(k)}$. In this proposed prediction based multiple symbol flipping decoding(P-MSFD) algorithm, symbols are predicted for ρ positions only which makes the algorithm very efficient and less complex in computation at the order of $O(\rho q)$ and is independent of the length of the code n. For understanding, let $\rho = 1$, so only $q-1$ values for the position $\boldsymbol{\psi} = j, \boldsymbol{\psi} \in \upsilon$ at k^{th} iteration will be calculated by (38) and (39).

Equation (37) predicts the most reliable position for symbol which maximize the function and then (38) determines the candidate symbol value $\boldsymbol{z}_j^{*(k)} \neq \boldsymbol{z}_j^{(k)}$ for that position. The new hard decision symbol sequence will be updated for the next $(k+1)^{th}$ iteration as:

$$\boldsymbol{z}^{k+1} = (\boldsymbol{z}_1^{(k)}, \boldsymbol{z}_2^{(k)}, \ldots, \ddot{\boldsymbol{z}}_j^{(k)}, \ldots, \boldsymbol{z}_n^{(k)}) \tag{40}$$

Proposed Algorithm 4 (P-MSFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = \sum_{j \in N(i)} h_{i,j} \boldsymbol{z}_j^{(k)} = \mathbf{0}$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Calculate $E_j^{(k)}(\boldsymbol{z}_j^{(k)})$ by (37) and determine the number of symbol positions ρ to be flipped by (24) and calculate $E_\rho^{(k)}$ by (35) and (36).
5. Calculate $E_\rho^{*(k)}(\boldsymbol{z}_\rho^{*(k)})$ and $E_\rho^{*(k)}$ by (37) and (38) respectively for $\ddot{\boldsymbol{z}}_\rho^{(k)} \in \Gamma_\rho^{*(k)}$.
6. Determine $\boldsymbol{\psi}^{(k)}$ positions by (34) and then its flipped value $\ddot{v}_{\boldsymbol{\psi}}^{(k)}$ by (39). Update $\boldsymbol{z}^{(k+1)}$ with $\ddot{v}_{\boldsymbol{\psi}}^{(k)}$.
7. $k = k+1$ and go to step 3.

E. VOTING BASED B-MSFD ALGORITHM

The bit error rate performance and complexity of B-MSFD algorithm is further improved and simplified by using the voting scheme to flip multiple symbols in each iteration. Once these positions are selected, the bit with less reliability in each symbol is chosen to be flipped based on the channel reliability information. This algorithm is termed as voting based multiple symbol flipping decoding (VB-MSFD). The least reliable symbol positions are short-listed by (12) and (13). The selected p number of unreliable positions $\boldsymbol{\delta}$, are used only to find the flipping values using the bit reliability information and the flipping function is not required to determines the least reliable symbols. This algorithm does not need the loop detection procedure and also does not need to isolate the failed and successful variable nodes as in B-MSFD, resulting in further simplification. Finding the flipped value from the channel reliability information is the same as for the B-MSFD algorithm. The proposed algorithm significantly lowers the decoder computational complexity as well as memory requirement to store the information for further processing.

Proposed Algorithm 5 (VB-MSFD)

1. Initialization: For the received symbol $\boldsymbol{y}_j = \{y_{j,1}, y_{j,2}, \ldots, y_{j,r}\}$, the hard decision symbol $\boldsymbol{z}_j$ is calculated by using (1) for $1 \leq t \leq r, 1 \leq j \leq n$.
2. For iteration $k = 1$ to I_{max}.
3. Syndrome Check-Sum: if $\boldsymbol{s}_i^{(k)} = 0$ or if $k = I_{max}$, stop decoding and output $\boldsymbol{z}^{(k)}$ as codeword.
4. Determine the un-reliable symbol $\delta_{j'}^{(k)}$ using the voting scheme by (20) and (21).
5. Find the flipped values by (29) and (30) for each candidate symbol.
6. Update $\boldsymbol{z}^{(k+1)}$ with $\acute{\boldsymbol{z}}_{j'}^{(k)}$ and the channel soft information $\boldsymbol{y}^{(k+1)}$ with $\acute{\boldsymbol{y}}_{j'}^{(k)}$ by (33) and (32) respectively.
7. $k = k+1$ and go to step 3.

IV. COMPLEXITY ANALYSIS

In this section, we evaluate the decoding computational complexity and memory requirement of the proposed algorithms in comparison with various existing NB LDPC decoding algorithms. The complexity analysis is carried out in terms of numerical operations per decoder iteration. Consider a NB LDPC code $\mathcal{C}$ for $GF(q) = 2^r$ having a parity check matrix H of size $m \times n$. Focusing on the regular LDPC code with constant column weight d_v and constant row weight d_c, the total number of edges of the Tanner graph is $\gamma = d_v n = d_c m$. In a single iteration, computational complexity of the proposed algorithms are mainly involved in the equations (14),(17),(22) and (35).

To get the hard decision symbol sequence $\boldsymbol{z}$ at the decoder initialization, the proposed algorithms require nr real comparisons. To calculate the syndrome check-sum for valid codeword, md_c Galois field multiplication and $m(d_c - 1)$ Galois field additions are required. To get the extrinsic information-sums(EXIs), the decoder requires md_c Galois field multiplications and md_c Galois field additions.

The major complexity of the algorithm in [25] involves computation of the binary Hamming distance $d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3013814, IEEE Access

for each edge, the binary function $\boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j$ for n number of symbols and the second binary function $\ddot{\boldsymbol{z}}_j^{(k)} \odot y_j$ for nq predicted values.

Therefore the total complexity of the algorithms in [25] are in order of $O(nq)$ when all $q-1$ values are computed to predict the candidate symbol value while the proposed Algorithms $1, 2$ lower this complexity to order of qp where p shows the number of unreliable variable nodes and is much less than n ($p << n$). The other two proposed Algorithms 3 and 5 further reduces the complexity as they do not require any prediction but simply flip a bit within each selected symbol.

In Algorithm 1, to compute the flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$, the binary functions $\boldsymbol{z}_j^{(k)} \odot \boldsymbol{y}_j$ for the symbol values before flipping and $\ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j$ for the predicted values of $\ddot{\boldsymbol{z}}_j^{(k)} \in \Gamma$ are calculated. Ignoring multiplication of the plus or minus one, r real additions are required for the binary function before flipping and $r(q-1)$ addition are required to compute the $\ddot{\boldsymbol{z}}_j^{(k)} \odot \boldsymbol{y}_j$ for all the predicted symbols. In step 1 of the algorithms in [25], $n(3+d_v)r$ additions are required for all the symbols while $p(3+d_v)r$ additions for the proposed Algorithm 1 and only δr additions for proposed Algorithm 5 as it does not require information after flipping.

For the decoding Algorithm 1, let $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)}) = \mathcal{F}_1 - \mathcal{F}_1$ where $\mathcal{F}_1 = \boldsymbol{z}_j^{(k)} \odot y_j^{(k)} + \sum_{i \in M(j)} \theta_{d(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})}$ and $\mathcal{F}_2 = \ddot{\boldsymbol{z}}_j^{(k)} \odot y_j^{(k)} + \sum_{i \in M(j)} \theta_{d(\ddot{z}_j^{(k)}, \sigma_{i,j}^{(k)})}$. To compute $\mathcal{F}_1$, pd_v real additions are required and to compute $\mathcal{F}_2$, $pd_v(d_v+q)$ real additions are required. Computation of all values of $\mathcal{F}_1 - \mathcal{F}_1$, requires $p(d_v+q)$ real additions. For each symbol in the received sequence, the function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ requires (d_v+q) comparisons. The function $E_j^{(k)}$, for received sequence of length n, $p(d_v+q-1)$ real comparisons are required. Since Algorithms $1, 3$ and 5 also do not need any loop detection procedure, the decoder computational complexity is further reduced.

The proposed algorithms for NB LDPC codes also significantly lower the memory consumption. Suppose r bits are required to store each of the $GF(q)$ element where $q = 2^r$ and b bits are required for storing floating-point values for each of the reliability metric. These algorithms require nrb bits to store the received sequence. The hard decision symbol sequence $\boldsymbol{z}_j^{(k)}$ and the extrinsic information-sums $\sigma_{i,j}^{(k)}$ require nr and nd_vr bits respectively.

For the weighting coefficients θ_d of the binary Hamming distance, d_vb bits are required to store the values. For the received sequence of length n, only p elements need to be stored. qb bits memory is required for each $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$ and pqb bits for all the values. Similarly pb and pr bits of memory are required for storing $E_j^{(k)}$ and $v_j^{(k)}$ respectively.The proposed algorithms reduce the memory requirement from n to p only. The smaller are the number of variable nodes in error, the less is the value of the p in comparison to the length of the received sequence n. The proposed Algorithms 3 and 5 significantly lower the memory consumption as these algorithm does not need to store the set of the predicted values $\ddot{\boldsymbol{z}}_j^{(k)}$.

The Algorithms 3 and 5 require only r bits to store the value for the flipped symbol $v_j^{(k)}$. Algorithm 5 offers significantly lower complexity as it does not calculate the flipping function for δ selected variable nodes. On the other hand, Algorithm 3 stores all the p number of selected variable nodes $\boldsymbol{\delta}$ to be used in finding the least reliable variable node during computation of the flipping function $E_j^{(k)}(\ddot{\boldsymbol{z}}_j^{(k)}, \boldsymbol{z}_j^{(k)})$.

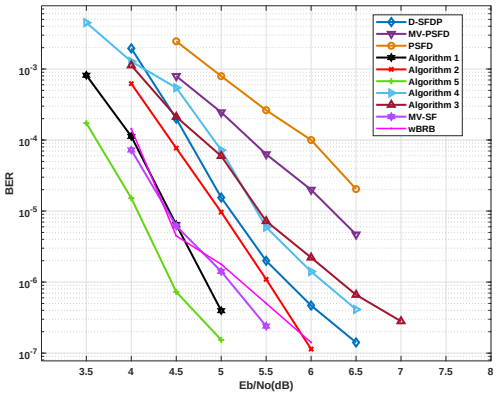

FIGURE 1. BER performance of NB LDPC code (204,102) over GF(16)

V. RESULTS AND DISCUSSION

In this section, the performance and complexity of the proposed algorithms are compared with various algorithms in the literature. An all zeros codeword is transmitted over additive white Gaussian channel(AWGN) using BPSK modulation ($1 \rightarrow +1$ and $0 \rightarrow -1$). In the proposed prediction based algorithms, all the $q-1$ possible values such as $\boldsymbol{z}_j^{*(k)} \neq \boldsymbol{z}(k)_j$ from $GF(q)$ are considered to predict the correct candidate symbol value. The other two Algorithms 3 and 5 do not use the symbol prediction method but only the less reliable bit of erroneous symbol is flipped. In the proposed algorithms, to predict the correct symbol, all possible $q-1$ values of $\boldsymbol{z}_j^{(k)}$ from $\ddot{\Gamma}_j^{(k)}$ are considered. However, only those values from the $GF(q)$ table can be considered whose binary representation is one bit different from the binary representation of $\boldsymbol{z}_j^{(k)}$ as there is higher chances of thier occurrence under AWGN. For example, in the hard decision symbol sequence, if the current value before prediction of the j^{th} symbol $\boldsymbol{z}_j^{(k)}$ is 101 for the code over $GF(8)$, then the predicted values look like $100, 001$, and 010. The maximum number of iterations in the following examples is determined through computer simulation and is set to 15. The Algorithms converge before 15 iterations. For comparison we have included the low complexity bit reliability based majority logic decoding algorithm (wBRB) which shows performance close to q-ary SPA.

Example 1: In order to demonstrate the effective performance of the proposed algorithms, BER performance of NB

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3013814, IEEE Access

TABLE 1 COMPUTATIONAL COMPLEXITY PER ITERATION FOR VARIOUS NON-BINARY LDPC DECODING ALGORITHMS

Algorithms	Number of operations				
	GM	GA	IA/RA	RC/IC	IM/RM
Algorithm1 (V-SFD)	$d_v(2d_c-1)$	$pd_v(d_v+r)+2d_v(d_c-1)$	$p(d_v^2+2d_vr+4r+2d_v)$	$nr+p(d_v-1)+2(p-1)$	
Algorithm5 (VB-MSFD)	md_v	$m(d_v-1)$	nd_v	$rn+p-1$	
D-SFDP	$d_v(2d_c-1)$	$nd_v(d_v+r)+2d_v(d_c-1)$	$n(d_v^2+2d_vr+4r+2d_v)$	$n(d_v+r-1)+2(n-1)$	
P-SFDP	$d_v(2d_c-1)$	$2d_v(d_c-1)$	$n(d_vr+5r+3d_v+1)$	$n(d_v+r-1)+2(n-1)$	
MV-SF	$nd_v(L+2)+2m(L-2)$	$nd_v(L+2)+2m(L-2)$	nd_vq+2nd_vL	$nd_vqlog_2q+n(q-1)+m(\sum_{i=d_c-\xi}^{d_c-1} i)$	
BRB	$2nd_v$	$2nd_v-m$	$2nd_vr$	$nd_v(2r+1)-3m$	
wBRB	$2nd_v$	$2nd_v-m$	$2nd_vr$	$nd_v(2r+1)-3m$	nd_v
PSFD	$2nd_v$	nd_v+2n-m	$2nd_v-2n+m$	$2nd_v-m$	
MV-PSFD	$2nd_v$	nd_v+2n-m	$3nd_v+m-1$	$2nd_v-m-n+nlog_2n$	$2nd_v+1$
EMS	$2nd_vn_m$	$9n_m(nd_v-2m)$	$n_m(20nd_v-18m-12n)$	$n_mlog_2n_m(9nd_v-12m-4n)$	
ISRB	$2nd_v$	$2nd_v-m$	nd_v+n2^r	$2n2^r-2n$	$n2^r$
wt-Algo.B	$2nd_v$	$3nd_v-m$	nd_v	nd_v	nd_v

IM/IC/IA: Integer Multiplication/Comparision/Adittion; p is the total number of variable nodes stored in $\boldsymbol{\delta}$.
RA/RM/RC: Real Addition/Multiplication/Comparision; GA/GM: Galois Field Addtion/Multiplication;
Notations used: $\gamma = nd_v = md_c$, $L = \upsilon^{\xi}$ such that $\xi < d_c$, $n_m < q$

TABLE 2
MEMORY REQUIRED FOR DEOCDING OF NON-BINARY LDPC ALGORITHMS

Algorithms	Required Memory(bits)
Algorithm1 (V-SFD)	$nr(b+1)+b(d_v+q)+p(b+r)$
Algorithm5 (VB-MSFD)	$nr(b+1)+d_vb+r$
D-SFDP	$nrb+(2n+\gamma)r+(n+q+d_v)b+log_2n$
P-SFDP	$nrb+(2n+\gamma)r+(n+q)b+log_2n+q[log_2d_v]$
wBRB	$2nrb+\gamma(r+b-1)$
PSFD	$2nq+3n+\gamma)b+(n+\gamma)r+\gamma+n[log_2d_v]$
MV-PSFD	$2nq+4n+2\gamma)b+(n+\gamma)d+log_2m$
EMS	$2nqb+\gamma n_m(r+b)$
T-EMS	$2nqb+\gamma qb$
wt.Algo.B	$(n+\gamma)r+d_vb+log_2d_v$
MV-SF	$b(\gamma q+nq+d_cq+nr)+r(d_cq+d_cL+n)$

LDPC decoding algorithms under additive white Gaussian noise as shown in Figure 1. The results are for a regular Gallagar code $\mathcal{C}1$ $(204, 102)$ [23] over $GF(16)$ with code rate $1/2$. The code has a constant column weight $d_v = 3$ and constant row weight $d_c = 6$. The extrinsic weighting coefficients $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2]$ of the newly proposed algorithms are set as $\boldsymbol{\theta} = [2, .75, 0.5]$ for $d_{\theta(\boldsymbol{z}_j^{(k)}, \sigma_{i,j}^{(k)})} \geq 2$.

Figure 1 illustrates the bit error rate performance of NB LDPC decoding algorithms. We see that the proposed Algorithms 2 and 4 are close in performance to the D-SFDP algorithm while Algorithms $1, 3$ and 5 perform better than BRB, MV-SFDP, PSFD, and D-SFDP. Algorithm 5 is outperforming all existing symbol flipping algorithms including MV-SF and wBRB. Similarly, the proposed Algorithm 1 performs close to MV-SF at the beginning and after 4.5 dB, it outperforms the MV-SF algorithm. The proposed Algorithms 3, 4 and 5 are multiple symbol flipping algorithms and perform better than BRB, ISRB, PSFD, MV-PSFD. At the BER 10^-6, the proposed Algorithm 5 outperforms by about 1.5 dB and 0.65 dB in comparison to D-SFDP, MV-SF and wBRB algorithms respectively. The MV-PSFD algorithm performs better than PSFD and the performance gap between them is about 0.5 dB but less in performance to the proposed algorithms.

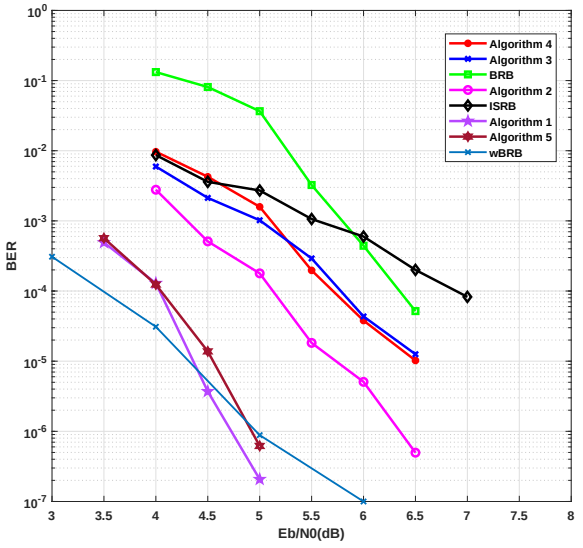

FIGURE 2. BER performance of NB LDPC code (120,60) over GF(128)

Example 2: In this example, the performance of the proposed algorithms is shown for a quasi cyclic (QC) NB LDPC code (120,60) over $GF(128)$. This is a regular parity check matrix with constant column weight $d_v = 4$ and constant row weight $d_c = 8$ with code rate $1/2$. The performance of the proposed algorithms is compared with reliability based algorithms like BRB and ISRB. The extrinsic weighting coefficient are set as $\theta = [2, .75, 0.5]$.

From the curves in the Figure 2, we see that all the proposed algorithms outperform than BRB and ISRB algorithms

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3013814, IEEE Access

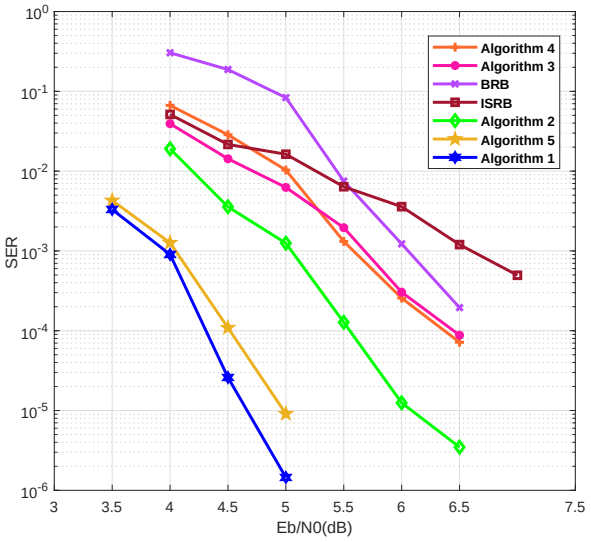

FIGURE 3. SER performance of NB LDPC code (120,60) over GF(128)

and are better than wBRB after 4.5 dB. At the bit error rate performance of 10^{-4}, the proposed Algorithms 1 and 5 achieves performance gain of about 2.2 and 3 dB over BRB and ISRB respectively. The performance gap between the proposed Algorithm 2 and BRB algorithm at BER 10^{-4} is around 1.3 dB and from the ISRB algorithm, the gap is around 1.8 dB. The Algorithms 3 and 4 have similar performance trend and show significant performance gain over all the algorithms in Figure 2. It is also observed that the decoding Algorithm 1 performs better than Algorithm 5. The reason for this is the higher order Galois field ($q = 2^r = 128, r = 7$) where each symbolS_j is composed of 7 bits ($e.g. S_j = b_0b_1b_2b_3b_4b_5b_6b_7$). The proposed Algorithm 5 correct one bit in each symbol per iteration and there are more chances to get more than one bit in error. The more are the number of bits in symbols, the more iterations are required to correct that symbol using Algorithm 5 in comparison to Algorithm 1 which is using the maximum likelihood technique. The Algorithm 1 try one by one each of the symbols $\ddot{\boldsymbol{z}}_j^{(k)} \neq \boldsymbol{z}_j^{(k)} \in GF(128)$ for the selected unreliable position and chooses the best candidate symbol value for the selected position. Due to this fact, Algorithm 1 performs better than Algorithm 5 for higher order Galois field $GF(q)$.

Similarly, Figure 3 shows the symbol error rate performance (SER) of the proposed algorithms in comparison to BRB and ISR algorithms. The performance curves show similar trends as the BER curves in Figure 2. The proposed decoding algorithms SER performance is also better than the BRB and ISRB algorithms in the literature.

Example 3: In this example the average number of iterations versus BER and SNR are plotted to show the complexity and convergence of the proposed NB LDPC decoding algorithms in comparison to some other algorithms in the literature. The code for Example 2 with the extrinsic weighting coefficient set as $\boldsymbol{\theta} = [2, .75, 0.5]$ is used here as well. From the curves in the graphs shown in Figures 4 and 5, we see that the proposed Algorithm 5 converges slowly compared to the other proposed algorithms as well as to BRB and ISRB algorithms. BRB algorithm is the fast converging algorithm followed by ISRB while the proposed algorithms convergence speed is slower than BRB and ISRB but faster than D-SFDP.

TABLE 3
COMPUTATIONAL COMPLEXITY PER ITERATION OF VARIOUS ALGORITHMS AT THE BER 10^{-4} FOR CODE(120,60) OVER GF(128)

Algorithms	Number of operations				
	GM	GA	IA/RA	RC/IC	Avg. Itr
Algorithm5 (VB-MSFD)	60	216	136	8	8.4
Algorithm1 (V-SFD)	60	1112	216	20	10.89
D-SFDP	60	5336	12960	1438	13.91
BRB	960	900	6720	7020	2.1
ISRB	960	900	15840	30480	4.5

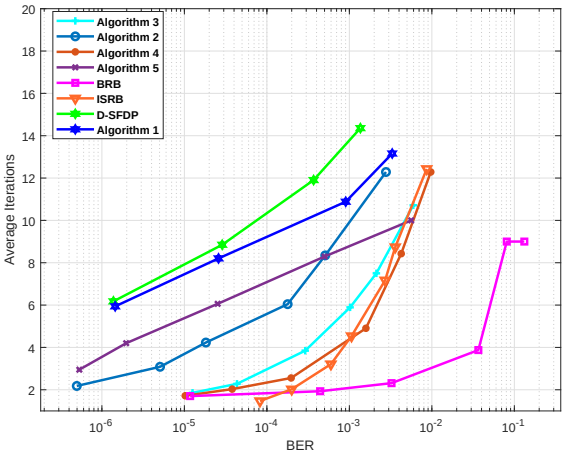

FIGURE 4. BER versus average number of iterations for NB LDPC code (120,60) over GF(128)

Since the proposed Algorithms 1 and 2, flip a single symbol per iteration, the average number of iterations shown in Figures 4 and 5 are more than the proposed Algorithm 3 and 4 which flip multiple symbols in each iteration. Algorithm 5 flips a single bit in each of the multiple selected symbols and that is why it takes more iterations for higher order Galois field. At the BER 10^{-4}, for the given average number of iterations, the computational complexity of BRB, ISRB and the proposed Algorithms 3 and 4 is lower than the D-SFDP and the Algorithms 2 and 5. As shown in Figure 5, we see that the Algorithm 5 performs better than other algorithms in terms of the average number of iterations, especially at the lower SNR. At higher SNR, the average number of iterations, except for Algorithm 5, exhibits almost the same trends.

Table 3 summarizes the complexity per iteration of the

various algorithms in comparison to the proposed Algorithms 1 and 5. At BER 10^{-4}, the typical value for the p is around 6 to 15 at the first iteration and decreases after every iteration if the symbol flipped at the previous iteration is correct.

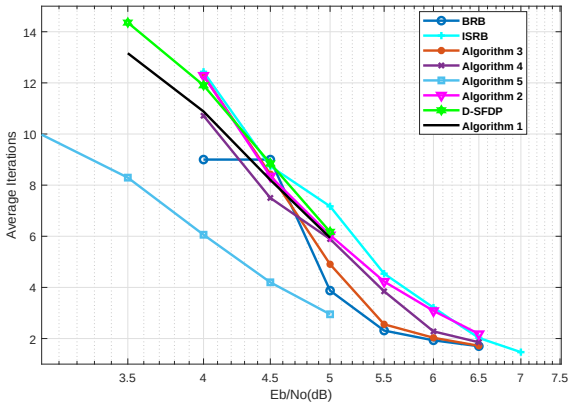

FIGURE 5. SNR versus average number of iterations for NB LDPC code (120,60) over GF(128)

From the analysis of the performance curves in graphs, the proposed algorithms show an appealing trade-off between complexity and performance and therefore can be recommended as good candidates for applications like data storage and communication systems.

VI. CONCLUSION

In this paper, low complexity symbol flipping decoding algorithms have been proposed. Incorporating the voting scheme to short list the least reliable symbols improves BER performance and reduces decoding latency. The voting scheme also helps in reducing memory requirements for storing the reliability values and extrinsic information for processing. Also, multiple symbol flipping decoding algorithms are proposed which helps in reducing the overall computational complexity of the NB LDPC decoder. Simulation results illustrate that the voting based proposed algorithms outperform all the existing symbol flipping decoding algorithms except Algorithm 2 whose performance is less than [24]. However, the Algorithm 2 is much superior when compared to the decoder complexity of [24]. The other proposed Algorithms 3 and 4, achieve the BER performance close to the algorithm [25] with significantly reduced complexity. From the results and complexity analysis, the proposed Algorithms 1 and 5 have shown an appealing trade-off between BER performance and computational complexity and are more suitable for practicable applications.

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] M. C. Davey and D. MacKay, "Low-density parity check codes over GF(q)," IEEE Communications Letters, vol. 2, no. 6, pp. 165–167, 1998.
[2] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF (q)," IEEE Trans. Communications, vol. 55, no. 4, pp. 633–643, 2007.
[3] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, dc)-LDPC codes over GF (q) using their binary images," IEEE Transactions on Communications, 2008.
[4] B.-Y. Chang, D. Divsalar, and L. Dolecek, "Non-binary protograph-based LDPC codes for short block-lengths," in 2012 IEEE Information Theory Workshop. IEEE, 2012, pp. 282–286.
[5] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(q)," in Information Theory Workshop, 2003. Proceedings. 2003 IEEE. IEEE, 2003, pp. 70–73.
[6] D. Declercq and M. Fossorier, "Extended minsum algorithm for decoding LDPC codes over GF (q)," in Proceedings. International Symposium on Information Theory, 2005. ISIT 2005. IEEE, 2005, pp. 464–468.
[7] Voicila, Adrian and Declercq, David and Verdier, François and Fossorier, Marc and Urard, Pascal, "Low-complexity decoding for non-binary LDPC codes in high order fields," IEEE transactions on communications, 2010.
[8] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary ldpc codes and its hardware structure," IEEE Transactions on Communications, vol. 61, no. 7, pp. 2600–2611, 2013.
[9] E. Boutillon and L. Conde-Canencia, "Bubble check: A simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," Electronics Letters, vol. 46, no. 9, pp. 633–634, 2010.
[10] V. Savin, "Min-Max decoding for non binary LDPC codes," in 2008 IEEE International Symposium on Information Theory. IEEE, 2008, pp. 960–964.
[11] C. Xiong and Z. Yan, "Improved iterative soft-reliability-based majority-logic decoding algorithm for non-binary low-density parity-check codes," in Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on. IEEE, 2011, pp. 894–898.
[12] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 11, pp. 1938–1950, 2012.
[13] C. Xiong and Z. Yan, "Improved iterative hard-and soft-reliability based majority-logic decoding algorithms for non-binary low-density parity-check codes," IEEE Transactions on Signal Processing, vol. 62, no. 20, pp. 5449–5457, 2014.
[14] S. Yeo and I.-C. Park, "Improved Hard-Reliability Based Majority-Logic Decoding for Non-Binary LDPC Codes," IEEE Communications Letters, vol. 21, no. 2, pp. 230–233, 2017.
[15] 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, 2011, pp. 1–7.
[16] Huang, Chao Cheng and Wu, Chi Jen and Chen, Chao-Yu and Chao, Chi chao, "Parallel symbol-flipping decoding for non-binary LDPC codes," IEEE communications letters, vol. 17, no. 6, pp. 1228–1231, 2013.
[17] 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 Transactions on Communications, vol. 58, no. 11, pp. 3140–3147, 2010.
[18] Jiang, Xueqin and Lee, Moon Ho, "Large girth non-binary LDPC codes based on finite fields and euclidean geometries," IEEE Signal Processing Letters, 2009.
[19] Zeng, Lingqi and Lan, Lan and Tai, Ying Yu and Zhou, Bo and Lin, Shu and K.Abdel-Ghaffar, "Construction of nonbinary cyclic, quasi-cyclic and regular LDPC codes: A finite geometry approach," IEEE transactions on Communications, 2008.
[20] Q. Huang, M. Zhang, Z. Wang, and L. Wang, "Bit-reliability based low-complexity decoding algorithms for non-binary LDPC codes," IEEE Transactions on Communications, vol. 62, no. 12, pp. 4230–4240, 2014.
[21] Q. Huang and S. Yuan, "Bit reliability-based decoders for non-binary LDPC codes," IEEE Transactions on Communications, vol. 64, no. 1, pp. 38–48, 2016.
[22] B. Liu, J. Gao, G. Dou, and W. Tao, "Majority decision based weighted symbol-flipping decoding for nonbinary LDPC codes," in Future Computer and Communication (ICFCC), 2010 2nd International Conference on, vol. 3. IEEE, 2010, pp. V3–6.

[23] 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 Code." IEEE Communications Letters, vol. 19, no. 6, pp. 905–908, 2015.

[24] F. Garcia-Herrero, D. Declercq, and J. Valls, "Non-binary LDPC decoder based on symbol flipping with multiple votes," IEEE Communications Letters, vol. 18, no. 5, pp. 749–752, may 2014.

[25] S. Wang, Q. Huang, and Z. Wang, "Symbol flipping decoding algorithms based on prediction for non-binary LDPC codes," IEEE Transactions on Communications, vol. 65, no. 5, pp. 1913–1924, 2017.

[26] Oh, Jieun and Han, Seokju and Ha, Jeongseok, "An Improved Symbol-Flipping Algorithm for Nonbinary LDPC Codes and its Application to NAND Flash Memory,," IEEE Transactions on Magnetics, 2019.

[27] L. F. dos Santos, J. Portugheis, and C. de Almeida, "Symbol-flipping decoding of nonbinary LDPC codes over BI-AWGN channels," in 2014 IEEE Latin-America Conference on Communications (LATINCOM). IEEE, nov 2014.