A PEG Construction of LDPC Codes Based on the Betweenness Centrality Metric

Insah BHURTAH-SEEWOOSUNGKUR[1], Pierre Clarel CATHERINE[2], Krishnaraj Madhavjee Sunjiv SOYJAUDAH[3]
[1]*Electrical and Electronic Engineering Department, University of Mauritius, Réduit, Mauritius*
[2]*Industrial Systems Engineering, School of Innovative Technologies and Engineering, University of Technology, Mauritius*
[3]*Tertiary Education Commission, Réduit, Mauritius*
i.bhurtah@ieee.org, c.catherine@ieee.org, ssoyjaudah@uom.ac.mu

Abstract—**Progressive Edge Growth (PEG) constructions are usually based on optimizing the distance metric by using various methods. In this work however, the distance metric is replaced by a different one, namely the betweenness centrality metric, which was shown to enhance routing performance in wireless mesh networks. A new type of PEG construction for Low-Density Parity-Check (LDPC) codes is introduced based on the betweenness centrality metric borrowed from social networks terminology given that the bipartite graph describing the LDPC is analogous to a network of nodes. The algorithm is very efficient in filling edges on the bipartite graph by adding its connections in an edge-by-edge manner. The smallest graph size the new code could construct surpasses those obtained from a modified PEG algorithm - the RandPEG algorithm. To the best of the authors' knowledge, this paper produces the best regular LDPC column-weight two graphs. In addition, the technique proves to be competitive in terms of error-correcting performance. When compared to MacKay, PEG and other recent modified-PEG codes, the algorithm gives better performance over high SNR due to its particular edge and local graph properties.**

Index Terms—**AWGN channels, block codes, channel coding, error correction codes, parity check codes.**

I. INTRODUCTION

LDPC codes are among the most powerful error-correcting codes for wireless communication of high-speed data. Their concept was developed by Robert G. Gallager in 1960 [1]. However, they were forgotten as technology was not mature enough for their efficient implementation. In 1999, David MacKay rediscovered the codes and found that they enabled data transmission rate close to the Shannon's theoretical limit [2].

LDPC codes are usually identified by a parity-check matrix **H** containing mostly '0's and very few '1's. Such a matrix can efficiently be represented by a bipartite graph which consists of bit and check nodes corresponding to columns and rows in **H**. Decoding is achieved using the belief propagation algorithm also known as the message passing algorithm or the sum-product algorithm (SPA) that passes messages along the edges of the bipartite graph. The performance of the SPA is determined by particular edge connections and local topologies of the graph [3-5]. Short cycles should be avoided since the "bad" nodes of the bipartite graph will convey the wrong information to other nodes to which they are connected. These receiving nodes will increase their false confidence, and they will in turn provide other nodes with wrong messages eventually building up to decoding failure. As such, it is fundamental to avoid small cycles. The girth of a code refers to the length of the smallest cycle in its corresponding bipartite graph and it is important to maximize the girth of the particular **H** matrix size with objective of obtaining better performance such as in [6-7].

The construction of bipartite graphs with good girth properties is crucial for good performance. In [2], the **H** matrix is created through random constructions. Another class is based on algebraic constructions [8-10] where LDPC codes can be commonly encoded in linear time with linear complexity on account of special structural property. In [11], the proposed algorithm based on graph theory, prevents the formation of a cycle by detecting the matrix related to the subgraph created from the original bipartite graph. Among these algorithms is the PEG algorithm [12] which constructs the **H** matrix through an edge by edge progression by optimizing the local girth after each edge addition for any particular code dimension and node degree distribution. The PEG algorithm demonstrates improved decoding performance through iterative decoding.

In [13], a modification of the PEG algorithm called the RandPEG is proposed. The technique improves the girth g achieved by the PEG algorithm, and when the girth cannot be increased, the RandPEG reduces the number of cycles of length g. The authors in [14] put forward a technique to construct the **H** matrix based on Quasi Cyclic (QC) method for the PEG algorithm with maximized girth property. Other works in [15-17] were also based on the PEG algorithm for QC codes. In [18], the PEG algorithm was extended to optimize high code rate of LDPC codes at the expense of a relatively high decoding error floor. Though the performance is decreased in the AWGN channel at low SNR, the authors assert that the high rate codes reduce transmission cost. The technique in [19] utilized a density metric instead of the conventional distance metric for the selection of nodes resulting in the creation of high rate codes. The concept of decoder optimization for the selection of check nodes in PEG-based construction techniques is presented in [20] with improved performance without extra cost in computational complexity. In [21], an algorithm was provided to the original PEG one to avoid the creation of dominant elementary trapping sets during edge selection in

This work was financially supported by the Tertiary Education Commission (TEC) of Mauritius.

Digital Object Identifier 10.4316/AECE.2016.02012

the attempt of lowering the error floor performance. A more recent work in [22] attempts to reduce the complexity of the PEG algorithm by decreasing the time for expanding the subgraph by not searching the bit nodes hence generating the same **H** matrix as in PEG.

In this work, we introduce a new type of PEG construction for LDPC codes based on the Betweenness Centrality (BC) metric borrowed from social networks terminology. The idea is inspired from [23] where the BC metric was used to enhance the routing performance in wireless mesh networks. A considerable gain in performance was achieved when the degree metric was based on the betweenness metric [23] (see (1)).

$$C_{B,i}=\sum_{j<k}\frac{g_{j,k}(i)}{g_{j,k}}, i\neq j, i\neq k \qquad (1)$$

where $C_{B,i}$, the betweenness centrality of node i is defined as the percentage of shortest paths across all possible pairs of nodes that pass through node i. Let $g_{j,k}$ be the number of shortest paths in from node j to node k and $g_{j,k}(i)$ be the number of shortest paths from node j to node k that contain node i.

Given that the bipartite graph describing the LDPC is analogous to a network of nodes, a modified version of the PEG method was adapted based on the use of the BC metric. Results obtained demonstrate the efficacy of the method such that for a given target girth g, the algorithm aims in constructing graphs with the minimal size so that a graph of girth g exists, and most graphs built surpassed those obtained by the standard PEG algorithm [12] or the RandPEG algorithm [13]. The graphs obtained can in turn be used to design ultra sparse non-binary (NB) LDPC codes giving good performance at small to moderate codeword lengths and high Galois field orders [13].

The paper is organised as follows: In section II, a description of the method is provided. Section III presents the results and discussion and we conclude with section IV.

II. MAXIMIZING THE RATE OF A GIVEN CODE

An (n,k) LDPC code is a linear block code mapping a source sequence $\mathbf{s}$ of k bits into a codeword $\mathbf{c}$ of n bits through $\mathbf{c} = \mathbf{G}^T\mathbf{s}$. $\mathbf{G}^T$ is the generator transpose associated to $\mathbf{H}$ in such a way that $\mathbf{HG}^T = \mathbf{0}$.

In a bipartite graph, the notation $N^d(p)$ refers to the set of nodes having a depth or a path of length d from a particular node p where p can be a bit node c_i, $0 \le i \le n-1$ or a check node f_s, $0 \le s \le k-1$. In Fig.1, $N^1(c_0) = \{f_3, f_6, f_9\}$, $N^1(f_4) = \{c_5, c_7, c_{11}\}$, $N^2(f_7) = \{f_4, f_9\}$ and $N^3(c_0) = \{f_3, f_6, f_7\}$.

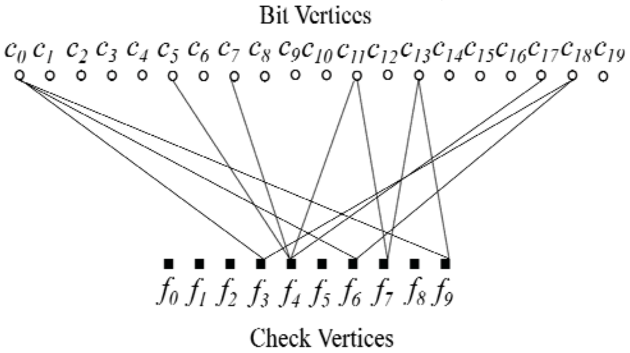
Bit Vertices
c_0 c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{13} c_{14} c_{15} c_{16} c_{17} c_{18} c_{19}
f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9
Check Vertices

Figure 1. Subgraph of a (20,10) regular LDPC code. Note: Not all edges are shown in the bipartite graph

A bipartite graph needs to be built with m check nodes, λ edges per bit node with the desired girth. To start with, a null bipartite graph is created with m check nodes. For each bit node c_i, $\lambda = 2$ edges are added so that the overlap of two columns of the parity-check matrix **H** is ≤ 1 [2]. The overlap between two columns refers to the number of positions where they both have a non-zero entry. For instance, an overlap of 2 in **H** reflects a cycle of length 4 which should be avoided. Algorithm 1 in Fig. 2 describes the technique employed by the new algorithm. Algorithm 1 builds a graph with column-weight two only and $\lambda = 2$.

For girth g,

$$V_1 \subseteq \left\{ f_j : \forall f_j \notin \left(\bigcup_{d=1}^{g-1} N^d\left(c_i\right) \right), j = 0,\ldots,m-1 \right\} \qquad (2)$$

$$V_2 \subseteq \arg\min_{f_j} \left\{ N^1\left(f_j\right), j = 0,\ldots,m-1 \right\} \qquad (3)$$

$$V_3 \subseteq \arg\min_{f_j} \left\{ C_B\left(f_j\right), \forall f_j \in V_2 \right\} \qquad (4)$$

$$C_B\left(f_j\right) = \sum_{k<l} g_{k,l}\left(f_j\right) \qquad (5)$$

where $g_{k,l}(f_j) = 1$ if f_j lies on the shortest path between f_k and f_l and $g_{k,l}(f_j) = 0$ otherwise.

```
for (j = 1 to j = λ) do
  1. Build sets $V_1$, $V_2$ and $V_3$ according
     to (2), (3) and (4) respectively.
    if ($V_1$ = ϕ) then
      if (j = 1) then
        2.  Stop the whole process
            and execute step 6.
      end if
      if (j = 2) then
        3.  Stop process for $c_i$ and
            proceed to add edges to
            $c_{i+1}$.
      end if
    else
      4.  Select any check node ($f_s$)
          from set $V_3$.
      5.  Update connectivity of all
          nodes up to g - 1 for $c_i$ and
          $f_s$ according to Algorithm 2.
      6.  Update the BC metric of all
          check nodes.
    end if
  end for
 7. Prune all bit nodes having less
than λ edges.
```

Figure 2. Algorithm 1 Adding edges to bit node c_i

A. Steps 1 to 4 in Algorithm 1

The process of adding the first edge to a bit node c_i consists of successive screening operations. For a certain girth g, at each step, check nodes contributing to a cycle of $g - 2$ are eliminated and at the end, the edge is chosen randomly in a set containing the check nodes having the minimum BC metric.

V_1 from (2) is a set that discards check nodes that would have caused a cycle of length $g - 2$. For constructing an **H** matrix of girth $g = 6$ for example, check nodes contributing to a cycle of length 4 should be avoided. In the same way, those leading to a cycle of length 4 and 6 should be excluded for $g = 8$. The set V_2 from (3) takes the elements of V_1 and returns the nodes having the least check node degree d_c. This step ensures the resulting graph is regular, with the same numbers connections for each check node.

The set V_3 from (4) then retrieves the check nodes having the least BC metric from V_2 using (5). To illustrate how the set V_3 calculates the least BC metric of the nodes, a (16,8) LDPC code is taken as example. The subgraph of the code is shown in Fig. 3 after the first edge has been added to c_6.

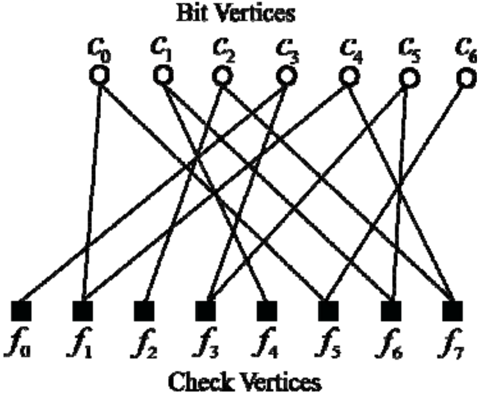

Figure 3. Subgraph of a (16,8) regular LDPC code after addition of the first edge in c_6

The procedure of adding edges to c_6 and c_7 will be described in subsections B and C respectively. Before that, the k+1 by k+1 Shortest Path (SP) matrix $M_{SP}(c_i, \lambda_j)$ is introduced where c_i and λ_j represent the i^{th} bit node to which the j^{th} edge has already been added. The SP matrix is derived from Step 5 in Algorithm 1 from Fig. 2 and from Algorithm 2 from Fig. 4 which will be explained in subsection D. This matrix is crucial in finding the shortest path which exists between any two check nodes f_j. In this work, if the distance of the shortest path between f_j is greater or equal to 4, the list of check nodes that lie on the shortest path between f_j is found. For instance, a path of length 4 or 6 between two check nodes f_j will have only one or two check nodes lying between them respectively. As the shortest path matrix is a k+1 by k+1 matrix, only the upper triangular part of the matrix is considered as the lower part is exactly the same. To connect an edge λ_j to a certain bit node c_i, the SP matrix generated by the previous λ_{j-1} addition is regarded.

B. Addition of the second edge λ_2 to c_6

For the addition of the second edge λ_2 to c_6, the SP matrix when the first edge λ_1 has already been added, is considered. In this case, the SP matrix $M_{SP}(c_6, \lambda_1)$ =

$$\left[\begin{array}{ccccccccc} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ f_0 & -1 & -1 & -1 & 2 & 6 & -1 & 4 & -1 \\ f_1 & -1 & -1 & 4 & -1 & -1 & 2 & -1 & 2 \\ f_2 & -1 & 4 & -1 & -1 & -1 & 6 & -1 & 2 \\ f_3 & 2 & -1 & -1 & -1 & 4 & -1 & 2 & -1 \\ f_4 & 6 & -1 & -1 & 4 & -1 & -1 & 2 & -1 \\ f_5 & -1 & 2 & 6 & -1 & -1 & -1 & -1 & 4 \\ f_6 & 4 & -1 & -1 & 2 & 2 & -1 & -1 & -1 \\ f_7 & -1 & 2 & 2 & -1 & -1 & 4 & -1 & -1 \end{array}\right]$$

Matrix MSP(c_6, λ_1) demonstrates the shortest path that exists between any two check nodes f_j when the first edge λ_1 has already been added to c_6. For instance, there exists a path of length 4 between check nodes f_0 and f_6. As such, there is only one check node which lies between f_0 and f_6 contributing to that distance (note that the connected bit nodes on the path are not considered). To find out that check node requires searching for a check node which lies at a path of length 2 from both f_0 and f_6. In the example mentioned, this particular check node is found to be f_3 (see Fig. 3).

```
$N^1(c_i) = N^1(c_i) \cup f_s$
$N^1(f_s) = N^1(f_s) \cup c_i$
for ($d$ = 2 to $d$ = $g$ - 1) do
    $N^d(c_i) = N^d(c_i) \cup N^{d-1}(f_s)$
    $N^d(f_s) = N^d(f_s) \cup N^{d-1}(c_i)$
    $N^d(N^{d-1}(c_i)) = N^d(N^{d-1}(c_i)) \cup f_s$
    $N^d(N^{d-1}(f_s)) = N^d(N^{d-1}(f_s)) \cup c_i$
end for
for ($d$ = 3 to $d$ = $g$ - 1) do
    for ($a$ = 3 to $a$ = $d$) do
        if ($d \geq a$) then
            $N^d(N^{a-2}(f_s)) = N^d(N^{a-2}(f_s)) \cup N^{d-a+1}(c_i)$
            $N^d(N^{a-2}(c_i)) = N^d(N^{a-2}(c_i)) \cup N^{d-a+1}(f_s)$
        end if
    end for
end for
```

Figure 4. Algorithm 2 Updating subgraphs of c_i and f_s

We next demonstrate how the BC metric is updated. The latter is best represented as a cube of size k by k by k. Each layer of the cube represents a check node's participation in the shortest paths between all shortest paths across the bipartite graph. All values of the cube are initialised to 0. Concerning the example of node f_3 which lies on the shortest path between f_0 and f_6, we next consider an expanded and partially filled BC matrix of Fig. 5. Since f_3 is the relevant node, the layer m = 3 is considered. The value of the cells lying at the intersection of check node f_0 (represented by row 0) and check node f_6 (represented by column 6), and at the intersection of check node f_6 (represented by row 6) and check node f_0 (represented by column 0) are incremented by 1 in the algorithm (represented by the highlighted cells in Fig. 5).

In the SP matrix $M_{SP}(c_6, \lambda_1)$, a path of length 6 exists between check nodes f_0 and f_4. The two check nodes lying at a distance of length 4 from both f_0 and f_4 are f_3 and f_6. In Fig. 5 for layers m = 3 and m = 6, the cells values which lie at the intersection of check node f_0 and check node f_4, and at the intersection of check node f_4 and check node f_0 are incremented by 1 in the algorithm (represented by the highlighted cells in Fig. 5). The same mechanism is applied to all paths greater or equal to 4 in matrix $M_{SP}(c_6, \lambda_1)$.

For each layer m in the BC matrix, the number of ones represented by the highlighted cells is counted and since the cell values appear twice, the calculated number is divided by two to obtain the BC metric. Fig.5 represents only part of all the updates of the matrix. The full one is shown in Table I where the check node degree d_c (calculated from Fig.3) and the BC metric for each check node f_j after λ_1 has been added to c_6 are represented. For the next edge selection therefore, the check node degree, d_c is first taken into consideration and then the BC metric is used to discriminate between the remaining check nodes (this order ensures that the generated code is a perfectly regular one).

From Table I, the check nodes having the least d_c are f_0, f_2 and f_4 and are therefore the potential candidates for the selection of the second node λ_2 to be connected to c_6.

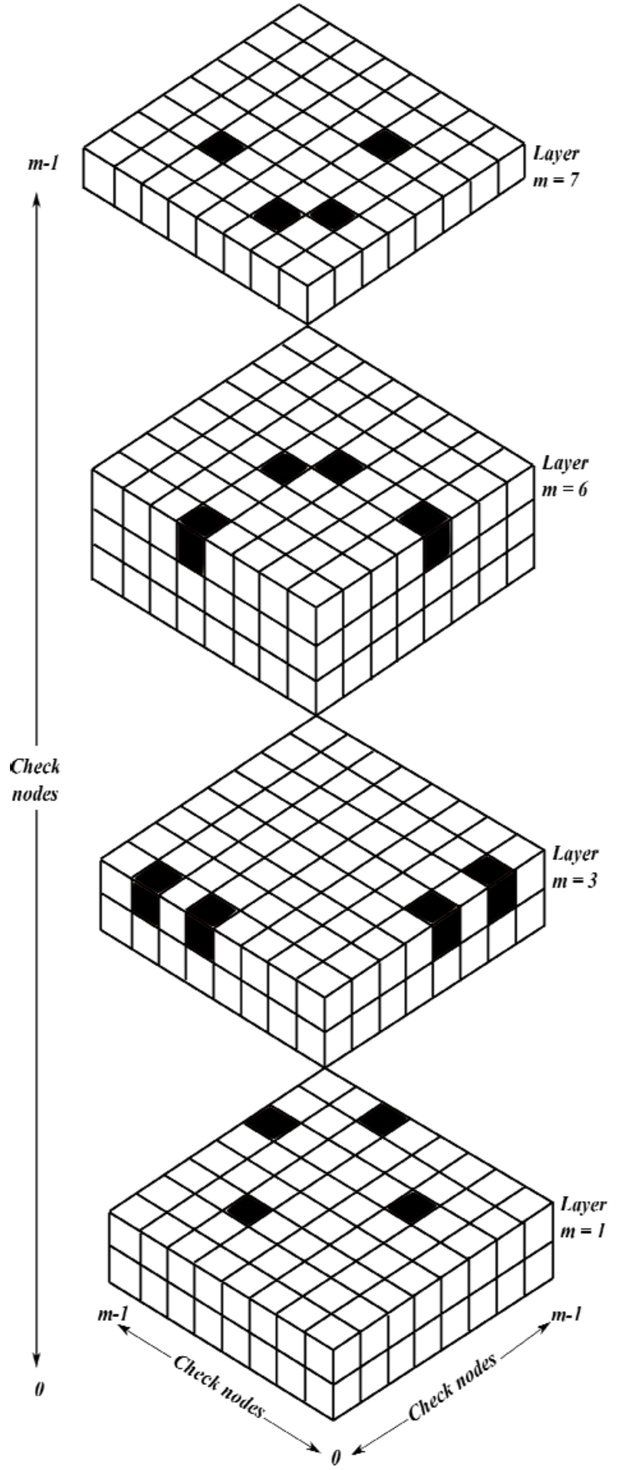

Figure 5. Partially filled BC matrix for addition of second edge λ_2 in c_6 for (16,8) regular LDPC code

TABLE I. CHECK NODE DEGREE d_c AND BC METRIC FOR EACH CHECK NODE AFTER ADDITION OF λ_1 TO c_6

Check node	Check node degree, d_c	BC metric
f_0	1	6
f_1	2	2
f_2	1	6
f_3	2	2
f_4	1	6
f_5	2	6
f_6	2	2
f_7	2	2

The check node with the least BC metric is then considered. As the BC metric of check nodes f_0, f_2 and f_4 is 6, a random selection is performed and f_0 is chosen as λ_2 to c_6 as shown in Fig. 6.

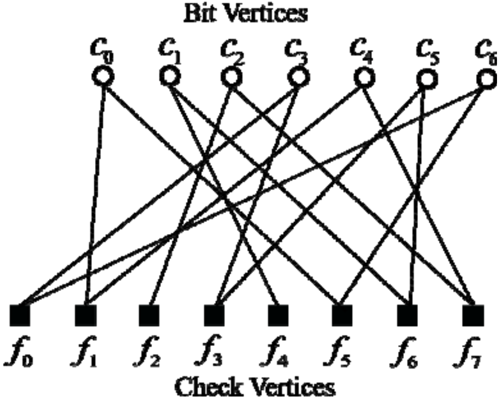

Figure 6. Subgraph of a (16,8) regular LDPC code after addition of the second edge λ_2 in c_6

C. Addition of the first edge λ_1 to c_7

For the addition of the first edge λ_1 to c_7, the SP matrix $M_{SP}(c_6, \lambda_2)$ when λ_2 has already been added to c_6, is considered. $M_{SP}(c_6, \lambda_2)$ =

$$\left[\begin{array}{ccccccccc} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ f_0 & -1 & 4 & 8 & 2 & 6 & 2 & 4 & 6 \\ f_1 & 4 & -1 & 4 & 6 & -1 & 2 & 8 & 2 \\ f_2 & 8 & 4 & -1 & -1 & -1 & 6 & -1 & 2 \\ f_3 & 2 & 6 & -1 & -1 & 4 & 4 & 2 & 8 \\ f_4 & 6 & -1 & -1 & 4 & -1 & 8 & 2 & -1 \\ f_5 & 2 & 2 & 6 & 4 & 8 & -1 & 6 & 4 \\ f_6 & 4 & 8 & -1 & 2 & 2 & 6 & -1 & -1 \\ f_7 & 6 & 2 & 2 & 8 & -1 & 4 & -1 & -1 \end{array}\right]$$

It can be seen from matrix $M_{SP}(c_6, \lambda_2)$ that there exists a path of length 8 between check nodes f_0 and f_2. From Fig. 6, the check nodes lying between check nodes f_0 and f_2 are f_1, f_5 and f_7. In the expanded and partially filled BC matrix (at first initialised to 0) in Fig. 7, in the layers m = 1, 5 and 7, the cells values which lie at the intersection of check nodes f_0 and f_2 (represented by row 0 and column 2, respectively), and at the intersection of f_2 and f_0 (represented by row 2 and column 0, respectively), are incremented by 1 in the algorithm (represented by the highlighted cells in Fig. 7). The same mechanism is applied to all path length greater or equal to 4 in matrix $M_{SP}(c_6, \lambda_2)$.

For each layer m in the BC matrix, the number of ones represented by the highlighted cells is counted and since the cell values appear twice, the calculated number is divided by two to obtain the BC metric. As Fig. 7 represents only part of all the updates of the matrix, the full one is found in Table II for each check node f_j after λ_2 has been added to c_6. As the check node degree d_c is first considered, the check nodes having the least d_c are f_2 and f_4. To discriminate between check nodes f_2 and f_4, the one with the least BC is selected for edge addition. As, the BC metric of both check nodes is 12, the edge f_4 is chosen as λ_1 to c_7 at random.

D. Step 5 in Algorithm 1

In step 5 of Algorithm 1 in Fig. 2, the connectivity of all nodes up to g - 1 for c_i and f_s is updated. The newly formed edge c_i - f_s is connected by two subgraphs. As such, all bit

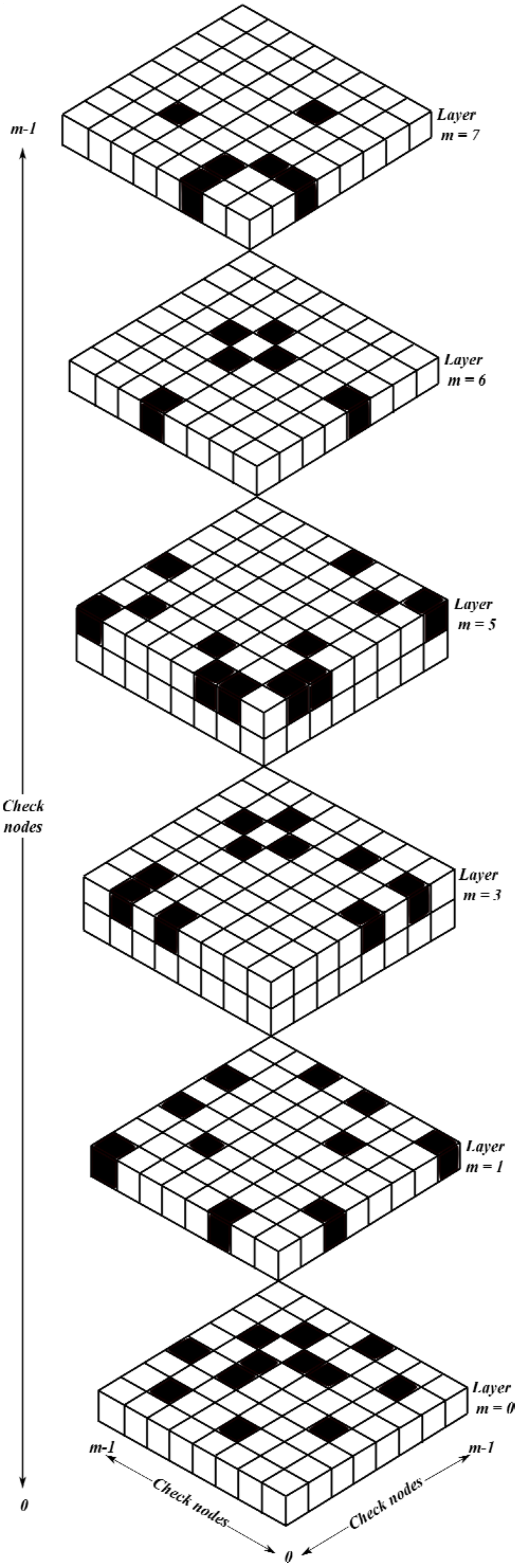

Figure 7. Partially filled BC matrix for addition of the first edge λ_1 in c_7 for (16,8) regular LDPC code

TABLE II. CHECK NODE DEGREE d_c AND BC METRIC FOR EACH CHECK NODE AFTER ADDITION OF λ_2 TO c_6

Check node	Check node degree, d_c	BC Metric
f_0	2	18
f_1	2	14
f_2	1	12
f_3	2	14
f_4	1	12
f_5	2	18
f_6	2	12
f_7	2	12

and check nodes that form part of the subgraph at a depth of g - 1 need to be notified of c_i - f_s.

When an edge is added between a certain bit and check node in the bipartite graph, the respective subgraph of either node contains many other subgraphs which hold potential connections for the formation of cycles. Fig. 8 shows the subgraphs of the newly formed edge c_i - f_s.

Connections of c_i and f_s with other bit and check nodes in the bipartite graph can be seen in Fig. 8. Algorithm 2 [19] in Fig. 4 provides the update of all nodes up to g - 1 for c_i and f_s as in step 5 in Algorithm 1 in Fig. 2.

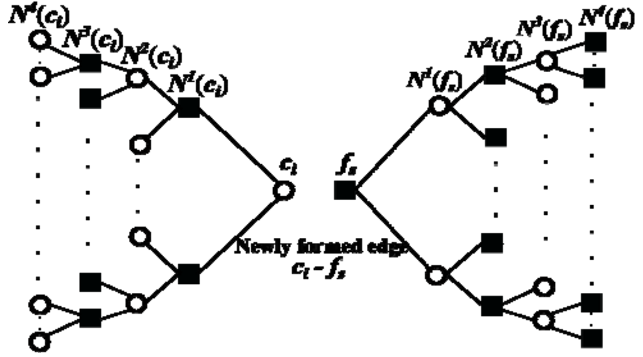

Figure 8. Subgraphs of c_i and f_s

E. Step 6 in Algorithm 1

In step 6 of Algorithm 1 in Fig. 2, the BC metric of all nodes in the bipartite graph is updated so that the node with the least BC metric can be chosen during the subsequent edge connection process. In our algorithm, when the BC metric is optimized, all nodes are equally connected in the graph allowing for a better transfer of decoding metrics via the edge connection.

F. Step 7 in Algorithm 1

It may occur that the algorithm adds only one edge for some bit nodes as a result of its sub-optimality. If λ = 2, for some bit nodes, the algorithm adds only one edge. However, for the next bit node c_{i+1}, the algorithm has the capability of adding the set of λ = 2 edges.

As the BC is based on the number of shortest paths that pass through a certain node, one which has a high BC metric is likely to be such a node that connects two subgraphs in the bipartite graph. Nodes having a high BC metric have a high probability of being cut-points due to the fact that they lie on a high number of shortest paths among other nodes in the bipartite graph. They are related to the degree of compactness and connectedness, and their removal entails a partition of disconnected group of nodes (subgraphs) in the graph. The reason for selecting an edge in V_3 from (4) having the least BC metric in step 4 in Algorithm 1 in Fig. 2, is to encourage edge selection associated with disconnected group of nodes.

Step 7 of Algorithm 1 in Fig. 2 deals with the elimination of bit nodes having less than λ edges as only bit nodes with λ edges need to be kept. A bipartite graph is obtained with m check nodes, n bit nodes and exactly λ edges per bit node.

III. APPLICATION AND PERFORMANCE EVALUATION

A. Design of ultra-sparse graphs

Tables III, IV and V show the smallest graph size, that is the codeword length that the LDPC code with the BC metric could construct for regular (2, d_c) graphs of girth g = 6 to 10, g = 12 to 16 and g = 18 to 22 respectively where d_c is the check node degree. It is noted that the column weight is 2 although the technique could be applied to any graph connectivity. When the value of the graph size achieves the lower bound, this is indicated by a star (*) super-script. Alternatively, the lower bound value is super-scripted with parenthesis. Our results were compared with those obtained with RandPEG [13] and standard PEG [12] which are indicated on the first square brackets and second square brackets respectively in Tables III, IV and V. For instance, considering the cell d_c = 5 and g = 10 in Table III, [110] represents the graph the standard PEG algorithm could construct, [90] represents the graph built by the RandPEG and **85** denotes the best regular (2, d_c) graph of girth g achieved so far by the LDPC code based on the BC metric. In the same cell, the superscripted value $^{(65)}$ represents the value of the theoretical lower bound.

TABLE III. THE SMALLEST GRAPH SIZE OBTAINED FOR CHECK NODE DEGREE d_c AND GIRTH g = 6 TO g = 10

d_c	$g = 6$	$g = 8$	$g = 10$
3	6* [6] [6]	9* [9] [9]	15* [15] [18]
4	10* [10] [10]	16* [16] [20]	38$^{(34)}$[38][42]
5	15* [15] [15]	25* [25] [35]	**85**$^{(65)}$[90][110]
6	21* [21] [21]	36* [36] [48]	**177**$^{(111)}$[189][225]
7	28* [28] [28]	49* [49] [70]	**308**$^{(175)}$[385][441]
8	36* [36] [36]	64* [64] [116]	**496**$^{(260)}$[728][812]
9	45* [45] [45]	81* [81] [162]	
10	55* [55] [55]	100* [100] [230]	
...	...*	...*	
50	1275* [1275] [1275]	2500*	

TABLE IV. THE SMALLEST GRAPH SIZE OBTAINED FOR CHECK NODE DEGREE d_c AND GIRTH g = 12 TO g = 16

d_c	$g = 12$	$g = 14$	$g = 16$
3	21*[21][27]	36* [36] [36]	45* [45] [72]
4	52* [52] [104]	**200**[260][292]	

TABLE V. THE SMALLEST GRAPH SIZE OBTAINED FOR CHECK NODE DEGREE d_c AND GIRTH g = 18 TO g = 22

d_c	$g = 18$	$g = 20$	$g = 22$
3	**96**$^{(69)}$[114][150]	**156**$^{(93)}$[201][285]	**351**$^{(141)}$[447][558]

The RandPEG algorithm is built from a randomization approach in which the objective function was used to discriminate among other edge connections. Results obtained by the RandPEG are found in [13].

For all values of d_c up to 50, the algorithm constructs graphs for g = 6 and 8 as shown in Table III. For g = 10, 14, 18, 20 and 22 the algorithm could construct smaller graphs than those obtained by the standard PEG algorithm [12] or the RandPEG algorithm [13]. These results are highlighted in bold in Tables III, IV and V. For large graphs with increasing girth and d_c, the present algorithm outperforms consistently better. To the best of the authors' knowledge, the graphs achieved by our algorithm are the best (2, d_c) graphs of girth g.

For all graph sizes our algorithm could construct, the matrices were constructed three times and an average number of trials was calculated. Fig. 9 shows the average number of trials against d_c for g = 6, 8, 10, 12 and 14. It can be observed that the average number of trials increases with d_c.

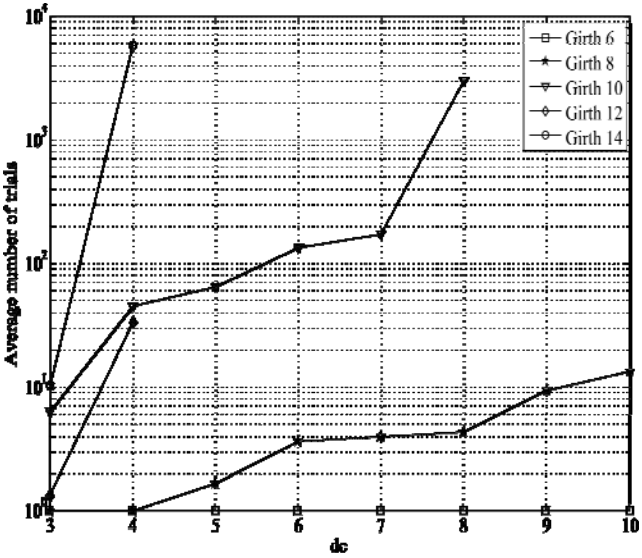

Figure 9. Graph of average number of trials against d_c for the LDPC code with the BC metric

Fig. 10 illustrates the average number of trials against column size for further graph sizes the LDPC code with the BC metric has been able to construct with girth 10, 14, 18, 20 and 22 against that of the RandPEG algorithm. It can be deduced that as the graph size decreases, it becomes more and more challenging for the matrices with a specific girth to be constructed, thus requiring an increase in the number of trials for successful matrix construction.

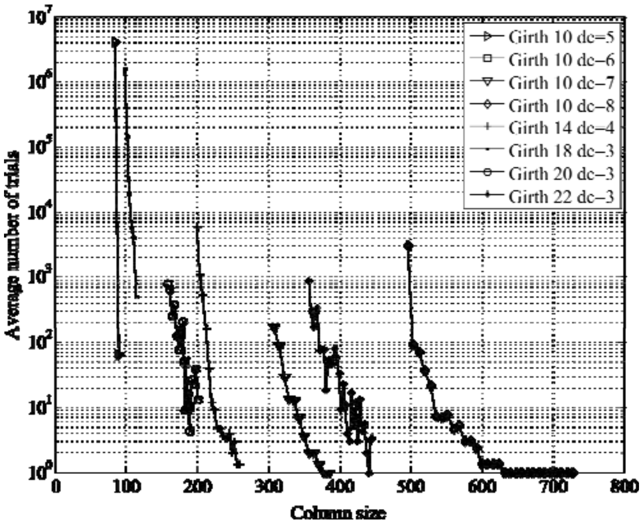

Figure 10. Graph of average number of trials against column size for girth g = 10, 14, 18, 20 and 22

B. Application to non-binary LDPC codes and design of protographs

The graphs built from our algorithm can be used in many applications such as building good non-binary LDPC codes in high order fields. The column weight two graphs constructed from our algorithm could be utilized for nonbinary coefficients in *GF*(64).

The presented algorithm could be used to build protographs which serve as a blueprint for the construction of LDPC codes [24]. A protograph is a bipartite graph with a relatively small number of nodes. It consists of variable nodes and check nodes. A larger graph can be obtained from a protograph through a copy and permute procedure [24]. The copies are overlaid in such a way that the same-type vertices neighbour each other. However, the subgraphs are disconnected. Connection of the subgraphs is achieved through swapping of endpoints of edges [24]. This procedure results in a derived graph from a prototype LDPC code. In [25] and [26], new QC protograph LDPC code constructions are presented. In this work, the presented algorithm can be used as protographs to build larger codes. The protograph will consist of a bipartite graph of small size constructed as per Algorithm 1 in Fig. 2. After updating the BC metric of all nodes of the protograph, it will become an optimized version which can be duplicated a certain number of times. The duplicated graphs will then be permuted according to some structural rules. The derived graph will become optimized in itself from the protographs.

C. Error-correction performance

This subsection deals with the error-correction performance of the LDPC code with the BC metric over other LDPC codes with results shown in Fig. 11-13 as graphs of Block Error Rate (BER) against (E_b/N_o) where (E_b/N_o) is the ratio of energy per information symbol to noise spectral density.

Simulations were carried out using the AWGN channel with the SPA as decoding algorithm. For a (504,252) LDPC code, our algorithm outperformed the PEG code [12] and the MacKay code [2] with a coding gain of 0.3 dB and 0.6 dB respectively at a BER of $1.00\text{x}10^{-6}$ in Fig. 11. The performance of our algorithm was compared with the DPEG code in [19]. For a (200,100) LDPC code, coding gains of 0.15 dB and 0.55 dB were achieved by the new code over the DPEG and MacKay codes respectively at a BER of $5.03\text{x}10^{-7}$ shown in Fig. 12. For a larger (400,200) code, a coding gain of 0.15 dB is obtained over the DPEG code at a BER of $7.45\text{x}10^{-7}$ in Fig. 13. It is therefore observed that the presented algorithm gives better error-correction performance.

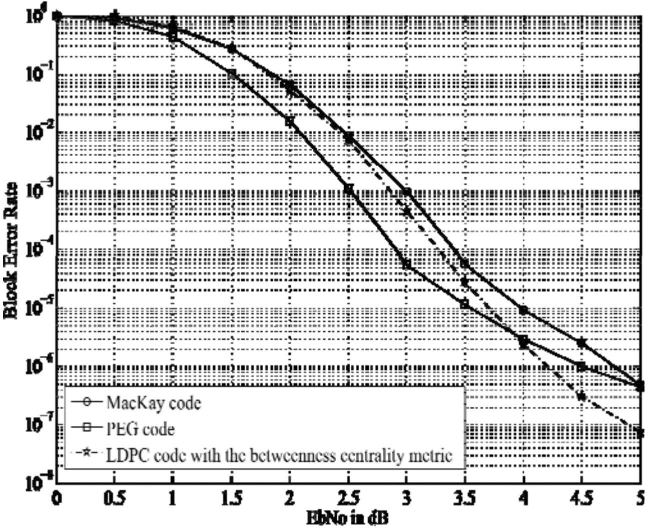

Figure 11. Graph of BER against (E_b/N_o) in dB for (504,252) girth 8 MacKay, PEG and LDPC code with the BC metric

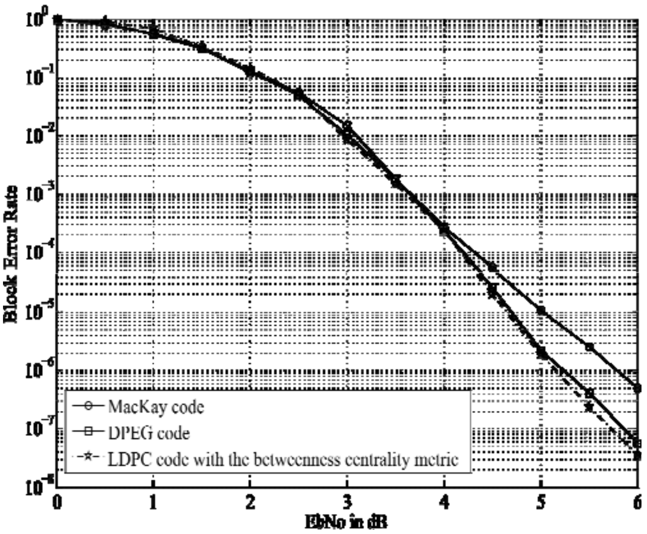

Figure 12. Graph of BER against (E_b/N_o) in dB for (200,100) girth 8 MacKay, DPEG and LDPC code with the BC metric

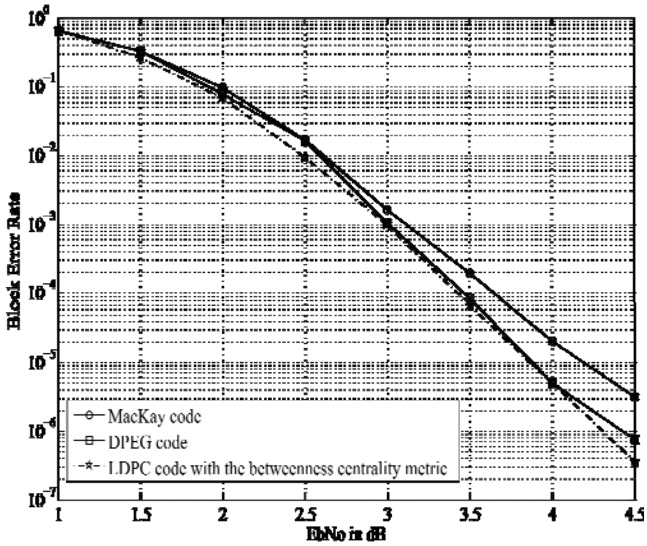

Figure 13. Graph of BER against (E_b/N_o) in dB for (400,200) girth 8 MacKay, DPEG and LDPC code with the BC metric

IV. CONCLUSION

In this work, a new PEG algorithm construction based on the BC metric is proposed. The smallest graph size the new code could construct surpasses those obtained from the RandPEG algorithm in [13]. The new code can also be used in non-binary LDPC codes and serve as protographs to build larger optimized codes. In addition, the technique proves to be competitive in terms of error-correcting performance. When compared to the MacKay [2] and DPEG [19] codes, our algorithm gives better performance over high SNR.

REFERENCES

[1] R. Gallager, “Low-density parity-check codes,” IRE Transactions on Information Theory, vol. 8, no. 1, pp. 21–28, 1962. doi: 10.1109/TIT.1962.1057683

[2] D. MacKay, “Good error-correcting codes based on very sparse matrices,” IEEE Transactions on Information Theory, vol. 45, no. 2, pp. 399–431, 1999. doi: 10.1109/18.748992

[3] J. M. F. Moura, J. Lu, H. Zhang, "Structured low-density parity-check codes," in IEEE Signal Processing Magazine, vol. 21, no.1, pp. 42-55, 2004. doi: 10.1109/MSP.2004.1267048

[4] C. A. Cole, S. G. Wilson, E. K. Hall, T. R. Giallorenzi, "Analysis and Design of Moderate Length Regular LDPC Codes with Low Error Floors," in 40th Annual Conference on Information Sciences and

Systems, Princeton, NJ, 2006, pp. 823-828. doi: 10.1109/CISS.2006.286581

[5] C. -C. Wang, S. R. Kulkarni, H. V. Poor, “Finding all small errorprone substructures in ldpc codes,” IEEE Transactions on Information Theory, vol. 55, no. 5, pp. 1976–1999, 2009. doi: 10.1109/TIT.2009.2015993

[6] J. Campello, D. S. Modha, S. Rajagopalan, “Designing ldpc codes using bit-filling,” in IEEE International Conference on Communications, Helsinki, 2001, pp. 55-59. doi: 10.1109/ICC.2001.936272

[7] J. Campello, D. S. Modha, “Extended bit-filling and ldpc code design,” in IEEE Global Telecommunications Conference (GLOBECOM '01), San Antonio, TX, 2001, pp. 985-989. doi: 10.1109/GLOCOM.2001.965565

[8] Y. Kou, S. Lin, M. P. C. Fossorier, “Low-density parity-check codes based on finite geometries: A rediscovery and new results,” IEEE Transactions on Information Theory, vol. 47, no. 7, pp. 2711–2736, 2001. doi: 10.1109/18.959255

[9] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, D. J. Costello, “Ldpc block and convolutional codes based on circulant matrices,” IEEE Transactions on Information Theory, vol. 50, no. 12, pp. 2966–2984, 2004. doi: 10.1109/TIT.2004.838370

[10] H. -Y. Liu, X. -Y. Lin, L. -R. Ma, J. Chen, “On the stopping distance and stopping redundancy of finite geometry ldpc codes,” IEICE Trans. Fundam. Electron. Commun. Comput. Sci., vol. E91-A, no. 8, pp. 2159–2166, 2008. doi: 10.1093/ietfec/e91-a.8.2159

[11] Y. Cui, X. Si, Y. Shen, “A novel algorithm of constructing ldpc codes with graph theory,” in IEEE Conference on Cybernetics and Intelligent Systems, Chengdu, Sept. 2008, pp. 602–605. doi: 10.1109/ICCIS.2008.4670752

[12] X. -Y. Hu, E. Eleftheriou, D. Arnold, “Regular and irregular progressive edge-growth tanner graphs,” IEEE Transactions on Information Theory, vol. 51, no. 1, pp. 386–398, 2005. doi: 10.1109/TIT.2004.839541

[13] A. Venkiah, D. Declercq, C. Poulliat, “Design of cages with a randomized progressive edge-growth algorithm,” IEEE Communications Letters, vol. 12, no. 4, pp. 301–303, 2008. doi: 10.1109/LCOMM.2008.071843

[14] P. Prompakdee, W. Phakphisut, P. Supnithi, “Quasi cyclic-ldpc codes based on peg algorithm with maximized girth property,” in International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), Chiang Mai, Dec. 2011, pp. 1–4. doi: 10.1109/ISPACS.2011.6146165

[15] Z. Fan, W. Zhang, X. Liu, H. Cheng, “An improved algorithm for constructing qc-ldpc codes based on the peg algorithm,” in Fourth International Conference on Communications and Networking in China (ChinaCOM), Aug. 2009, pp. 1–4. doi: 10.1109/CHINACOM.2009.5339908

[16] Y. Huang, Y. Cheng, Y. Zhang, H. Han, “Construction of non-binary quasi-cyclic ldpc codes based on peg algorithm,” in 12th IEEE International Conference on Communication Technology (ICCT), Nanjing, Nov. 2010, pp. 266–268. doi: 10.1109/ICCT.2010.5689251

[17] L. Huang, Y. Wang, P. Gong, “An improved construction method of qc-ldpc codes based on the peg algorithm,” in Third Pacific-Asia Conference on Circuits, Communications and System (PACCS), Wuhan, July 2011, pp. 1–4. doi: 10.1109/PACCS.2011.5990282

[18] Z. Zhou, X. Li, D. Zheng, K. Chen, J. Li, “Extended peg algorithm for high rate ldpc codes,” in IEEE International Symposium on Parallel and Distributed Processing with Applications, Chengdu, Aug. 2009, pp. 494–498. doi: 10.1109/ISPA.2009.80

[19] P. C. Catherine, K. M. S. Soyjaudah, “A density-based progressive edge-growth matrix creation technique for ldpc codes,” in 6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Brest, Sept. 2010, pp. 211–215. doi: 10.1109/ISTC.2010.5613841

[20] C. T. Healy, R. C. de Lamare, “Decoder optimised progressive edge growth algorithm,” in IEEE 73rd Vehicular Technology Conference (VTC Spring), Yokohama, May 2011, pp. 1–5. doi: 10.1109/VETECS.2011.5956769

[21] S. Khazraie, R. Asvadi, A. H. Banihashemi, “A peg construction of finite-length ldpc codes with low error floor,” IEEE Communications Letters, vol. 16, no. 8, pp. 1288–1291, 2012. doi: 10.1109/LCOMM.2012.060112.120844

[22] G. Srirutchataboon, A. Bajpai, L. Wuttisittikulkij, P. Kovintavewat, “Peg-like algorithm for ldpc codes,” in International Electrical Engineering Congress (IEECON), Chonburi, March 2014, pp. 1–4. doi: 10.1109/iEECON.2014.6925956

[23] M. Kas, S. Appala, C. Wang, K. M. Carley, L. R. Carley, O. K. Tonguz, “What if wireless routers were social? approaching wireless mesh networks from a social networks perspective,” in IEEE Wireless Communications, vol. 19, no. 6, pp. 36–43, Dec. 2012. doi: 10.1109/MWC.2012.6393516

[24] T. V. Nguyen, A. Nosratinia, "Rate-compatible short-length protograph ldpc codes," IEEE Communications Letters, vol. 17, no. 5, pp. 948-951, 2013. doi: 10.1109/LCOMM.2013.031313.122046

[25] I. Diop, S. M. Farssi, M. Ba, H. B. Diouf, “Construction of codes protographes ldpc quasi-cycliques based on an arithmetic progression,” in Second International Conference on Innovative Computing Technology (INTECH), Casablanca, Sept. 2012, pp. 194–199. [Online]. doi: 10.1109/INTECH.2012.6457749

[26] M. Karimi, A. H. Banihashemi, “On the girth of quasi-cyclic protograph ldpc codes,” IEEE Transactions on Information Theory, vol. 59, no. 7, pp. 4542–4552, 2013. doi: 10.1109/TIT.2013.2251395