# PEG-Like Algorithm for LDPC Codes

Gan Srirutchataboon, Ambar Bajpai, Lunchakorn Wuttisittikulkij
Department of electrical Engineering
Chulalongkorn University
Bangkok Thailand

Piya kovintavewat
Faculty of Science and Technology
Nakhon Pathom Rajabhat University
Nakhon Pathom Thailand

***Abstract*—Progressive Edge-Growth (PEG) algorithm is one of the promising methods to construct a parity-check matrix (or H matrix) with a large girth for low-density parity-check (LDPC) codes. However, generating a large H matrix based on a PEG algorithm usually requires a lot of computations because of its complexity. This paper proposes an alternative method based on a topology matrix to construct the H matrix, which has lower complexity than the PEG algorithm. We refer to the proposed method as a "PEG-like" algorithm. Results indicate that the proposed method can provide the same H matrix as the PEG algorithm does but with lower complexity**

***Index* Low-density parity-check (LDPC), non-binary topology matrix, parity-check matrix, progressive edge-growth (PEG)**

## I. INTRODUCTION

Low-density parity-check (LDPC) codes have recently become an important class of codes used in many practical applications. Newly emerging wireless networking standards such as wireless LANs (IEEE 802.11n), WiMax (IEEE 802.16e), and digital video broadcasting (DVB-T2) adopt LDPC codes to combat against channel errors.

It is commonly known that LDPC codes can achieve performance close to the Shannon's limit [1] with moderate complexity using suboptimal iterative decoding. LDPC codes were originally invented by Gallager in 1962 [2], and rediscovered in 1996 by Mackay and Neal [3] and Wiberg [4]. Since then, there has been extensive research and development of LDPC codes. Various new classes of LPDC codes have been found such as [5], [6], [7], [8]. Different methods for constructing a parity check matrix (**H**) have also been introduced. Among them, a progressive edge-growth (PEG) algorithm [5] is perceived promising, as it yields a large girth and can be applied to generate linear-time encodeable LDPC codes. Subsequent researches [8-10] seek to improve the PEG performance from different aspects. For example, Zhou *et al.* [8] proposed an extended PEG algorithm for constructing the **H** matrix to achieve high code rate (i.e., 0.93), while maintaining the same girth. Prompakdee *et al.* [9] presented a method to generate the **H** matrix based on Quasi Cyclic (QC) method for the PEG algorithm with maximized girth property. In [10], a modified PEG algorithm was introduced to improve the PEG performance in the waterfall region.

However, constructing a large **H** matrix based on the PEG algorithm normally requires a relatively long computational time, especially when the size of **H** matrix is very large and the required bit node degree is high. The increase in complexity of PEG is mainly due to the time spent on the process of subgraph spreading. Therefore, this paper aims to reduce the complexity of subgraph spreading by introducing a new method for constructing the **H** matrix based on topology matrix. The new algorithm will be shown to create the same **H** matrix as generated by the PEG algorithm but with lower complexity. This is why we refer to the proposed algorithm as a PEG-like algorithm.

The rest of this paper is organized as follows. Section II explains the definitions and notations of LDPC codes. Then, Section III briefly explains the PEG algorithm, and Section IV presents the proposed method. Simulation results are given in Section V. Finally, Section VI concludes this paper.

## II. DEFINITIONS AND NOTATIONS

An *M*-by-*N* parity-check **H** matrix (*M* rows and *N* columns) of an LDPC code can be represented by a Tanner graph [7], where *M* is the number of parity-check equations, *N* is the number of coded bits, and $K = N - M$ is the number of message bits. The Tanner graph is a bipartite graph, which composes of the set $(V, E)$, where $V = V_c \cup V_s$, $V_c = \{c_0, c_1, ..., c_{M-1}\}$ is the set of check nodes, $V_s = \{s_0, s_1, ..., s_{N-1}\}$ is the set of bit nodes, and *E* is the set of edges, $(c_i, s_j) \in E$ corresponding to a nonzero element at the *i*-th row and the *j*-th column in the **H** matrix, where $0 \le i \le M-1$, and $0 \le j \le N-1$.

Additionally, let the degrees of check and bit nodes be define as $D_c = \{d_{c_0}, d_{c_1}, ..., d_{c_{M-1}}\}$ and $D_s = \{d_{s_0}, d_{s_1}, ..., d_{s_{N-1}}\}$, respectively, where $E_{s_j}^k$ denote the edges on $s_j$ with $0 \le k \le d_{s_j} - 1$. Fig. 1 shows the Tanner graph for $D_s$ = {2, 2, 2, 2, 2, 2, 2, 2, 2} and $D_c$ = {4, 4, 4, 4, 4}, where the bit □ represents the check node $c_i$, and the o represents the bit node $s_j$.

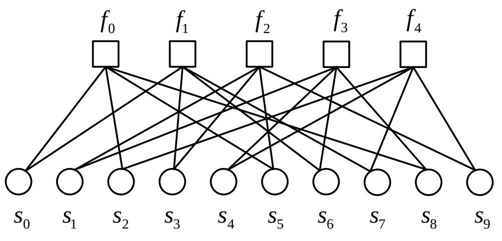

Fig. 1. A Tanner graph with $D_s$ = {2, 2, 2, 2, 2, 2, 2, 2, 2, 2} and $D_c$ = {4, 4, 4, 4, 4}.

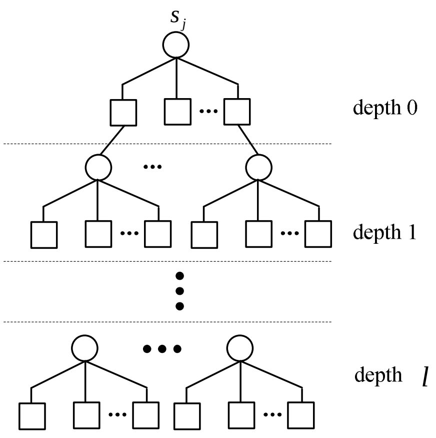

Fig. 2. A sub-graph spreading from bit node $s_j$.

## III. PEG ALGORITHM

This section briefly explains how the PEG algorithm works [6], whose process is to connect an edge between a check node and a bit node by using the spreading of the sub-graph as depicted in Fig. 2. For a given bit node $s_j$, we define its neighbourhood within depth $l$, $N_{s_j}^l$, as the set consisting of all check nodes reached by sub-graph spreading. Specifically, the first edge that wants to connect to a bit node $s_j$ can be chosen randomly from a check node with the lowest degree. For the next edge, we must first spread the sub-graph from $s_j$ and then select the check node from the lowest depth(at depth $l$) of this graph that has the lowest degree. However, if the sub-graph does not cover all check nodes, we must choose the check node with the lowest degree that is not within the sub-graph( $\bar{N}_{s_j}^l$ ). The PEG algorithm can be summarized as follows.

1. Assign the degree of a bit node $s_j$, e.g., $d_{s_0} = 2$, and the set of edges incident to this bit node as $E_{s_j} = \{E_{s_j}^0, E_{s_j}^1\}$.
2. Add an edge to this bit node $s_j$. First, if the edge that wants to be added to this bit node $s_j$ is $E_{s_j}^0$, we can choose the set of $V_c$ with the lowest degree randomly. If the edge is not $E_{s_j}^0$, we must expand the sub-graph up to depth $l$. Then, the two event a) and b) can happened:

   a) Given the set of $V_c$ within depth $l$ denoted as $N_{s_j}^l$, if the number of $N_{s_j}^l$ is less than $M$, we must choose the set of $V_c$ that is not in depth $l$, denoted as $\bar{N}_{s_j}^l$, which has the lowest degree randomly; and

   b) If the cardinality of set $N_{s_j}^l$ is equal to $M$, we will choose the check node with the lowest degree at depth $l$. Repeat this step until the $k$-th equal to $d_{s_j}$
3. Go back to Step 2 for adding edges to the next bit node until $j = N-1$, where $0 \le j \le N-1$.

Thus, the set of $N_{s_j}^l$ and $\bar{N}_{s_j}^l$ must satisfy $N_{s_j}^l \cup \bar{N}_{s_j}^l = V_c$, where $V_c \setminus N_{s_j}^l = \bar{N}_{s_j}^l$ and $V_c \setminus \bar{N}_{s_j}^l = N_{s_j}^l$. Finally, if we choose any check node at depth $l$, it can be shown that the number of girths from the bit node $s_j$ will be equal to $2(l+2)$.

Below is a pseudo-code for the PEG algorithm.

```
For $j = 0$ to $n-1$
   For $k = 0$ to $d_{s_j} - 1$
       If $k = 0$
          $E_{s_j}^0 \leftarrow$ Choose the check node with the degree
          randomly
       Else
          $E_{s_j}^k \leftarrow$ Use the above Step 2(a) if the number of
               $N_{s_j}^l$ is less than $M$
               Use the Step 2(b) if the number of $\bar{N}_{s_j}^l$ is
               equal to $M$
       End
   End
End
```

## IV. PROPOSED ALGORITHM

Here, we propose a new method to construct the **H** matrix, whose girths of every bit node are close or identical to that generated by the PEG algorithm. The aim of the proposed algorithm is to reduce the complexity in terms of processing time for generating a large **H** matrix. By definition, any two bit nodes in a Tanner graph are never directly connected. Both bit nodes can reach each other only through at least one check node. In the PEG algorithm, an important process that consumes large processing time is the sub-graph spreading, where the search starting from an initial bit node to all other nodes across the current graph is required.

If the **H** matrix maps in to a topology matrix, we do not need to concern about the bit nodes anymore; hence, we can save computation time. In PEG, if the sub-graph is spread from the bit node $s_j$, we must search in rows and columns of the **H** matrix. This process requires $\sum_{i=1}^{\text{column weight - 1}} 2(l_i+1)$ times for spreading the sub-graph to reach the depth $l$. In other words, if the bit node is ignored, the process to expand the sub-graph will be reduced, and it can be expressed in terms of a topology matrix. In addition, the process of searching in the topology matrix usually requires $\sum_{i=1}^{\text{column weight - 1}} (l_i+1)$ times.

This is an efficient concept to reduce the time for creating the same **H** matrix as the one obtained from the PEG algorithm.

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 6 | 9 | 3 |
| 1 | 0 | 4 | 7 | 8 |
| 6 | 4 | 0 | 2 | 10 |
| 9 | 7 | 2 | 0 | 5 |
| 3 | 8 | 10 | 5 | 0 |

Fig. 3. An example of the topology matrix ($T_{M\times M}$) for the 5×5 **H** matrix with a column weight of 2.

## *A. Algorithm Description*

For sub-graph spreading, if the bit nodes are ignored we can reduce the **H** matrix of size $M \times N$ to a $T_{M\times M}$ matrix where each row and each column in $T_{M\times M}$ represents each check node in **H**. Fig. 3 displays an example of $T_{M\times M}$ (i.e., $M$ = 5), where $t_{pq}$ denotes an element at the $p$-th row and the $q$-th column, $0 \le p$, and $q \le M - 1$. If $t_{pq} \ne 0$, it means that the $p$-th and the $q$-th check nodes are connected via the $t_{pq}$-th bit node. The proposed algorithm using the $T_{M\times M}$ matrix can be described as follows. For a given bit node $s_j$,

i) If it is the first time connecting a check node to the bit node $s_j$, select the $p$-th row of $T_{M\times M}$ with the most number of $0'$ and assign an edge to connect this selected check node to the bit node $s_j$. This check node is also used as a starting point for the sub-graph spreading in the next edge assignment.

ii) If it is not the first time connecting a check node to the bit node $s_j$, sub-graph spreading must be carried out first by using matrix $T_{M\times M}$ as follows: search through each row of matrix $T_{M\times M}$ only where the starting check node (or nodes) is for nonzero elements, determine all adjacent check nodes from the columns where those nonzero elements are. All these adjacent check nodes are then used as the next starting point for sub-graph spreading. This sub-graph spreading procedure repeated in the same manner until no further new check nodes can be reached or all check nodes are reached. For the former, select a check node to connect to the bit node $s_j$ from those unreached check nodes with the lowest degree. For the latter, select a check node to connect to the bit node $s_j$ from the farthest group of reachable check nodes whose degree is the lowest. Next, update the matrix $T_{M\times M}$ by adding $s_j$ to all elements that connect all corresponding check nodes together via the bit node $s_j$. Last, all connected check nodes through bit node $s_j$ are used as the starting point for sub-graph spreading of the next edge assignment.

The proposed algorithm can be shown in a pseudo-code:

```
For $j = 1$ to $n - 1$
    For $k = 1$ to $d_{sj} - 1$ do
        If $k = 1$
            case i)
        Else
            case ii)
        End
    End
End
```

This algorithm can create the **H** matrix with $M$ rows and $N$ columns, where $N \le \binom{M}{2}$ for avoiding a girth of 4. From this equation, our algorithm can create the **H** matrix with highest code rate as $R_{MAX} = 1 - \frac{2}{(M-1)}$. Fig. 3 shows an example of the topology matrix with a column weight of 2. The number of each column represents the number of bit nodes that connect the $p$-th row and the $q$-th row together.

## V. RESULT AND DISCUSSION

Consider the ($N$, $K$) **H** matrix, where $N$ is the length of coded bits, $K$ is the number of message bits, and $M = N - K$ is the number of parity bits. To evaluate the performance of the proposed algorithm, we simulate the system based on an additive white Gaussian noise (AWGN) channel model, where a $K$-bit binary input sequence $a_k \in \{0, 1\}$ is encoded by an LDPC decoder and is mapped to an $N$-bit coded sequence $b_k \in \{\pm 1\}$. Then, the received sequence is given by $y_k = b_k + n_k$, where $n_k$ is AWGN with zero mean and variance $\sigma^2$. At the receiver, the received sequence $y_k$ is decoded by an LDPC decoder implemented based on a message passing algorithm with $x$ internal iterations. Note that each bit-error rate (BER) point is obtained by using as many data packets as possible to obtain at least 1000 erroneous bits. In simulation, the signal-to-noise ratio is defined as SNR = $10\log_{10}(1/\sigma^2)$ in decibel (dB).

Fig. 4 compares the BER performance between the PEG and the proposed algorithms based on (504, 252), (504, 1008), and (1016, 2032) **H** matrices for a column weight of 2. Clearly, the PEG-like algorithm performs similar to the PEG algorithm as expected. Next, we compare the complexity between the PEG and the proposed algorithms for the (1016, 2032) **H** matrix with a column weight of 2, as depicted in Fig. 4,

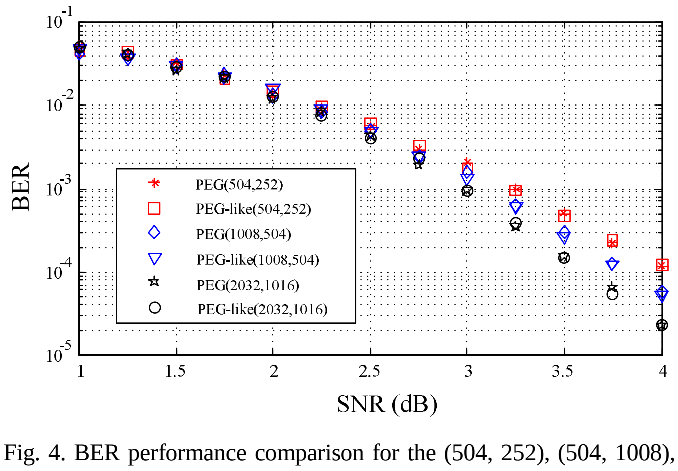

Fig. 4. BER performance comparison for the (504, 252), (504, 1008), (1016, 2032) **H** matrices with a code rate of 1/2.

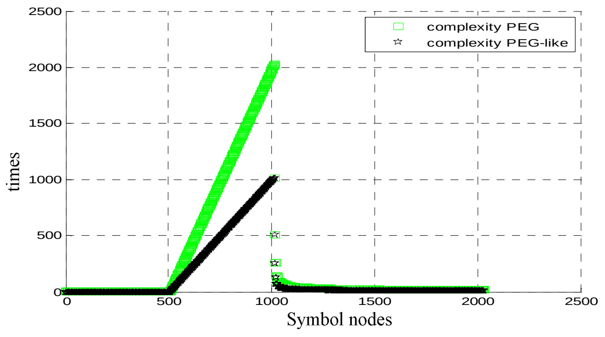

Fig. 5. Complexity comparison for the (1016, 2032) **H** matrix with a bit-node degree of 2.

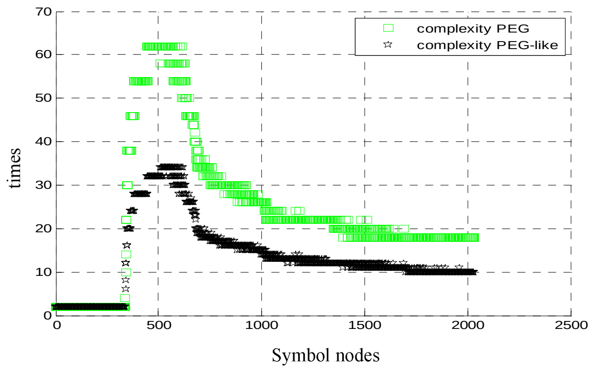

Fig. 6. Complexity comparison for the (1016, 2032) **H** matrix with a bit-node degree of 3.

where the x-axis represents the *i*-th bit node, and the y-axis denotes the searching time required to complete the loop at the *i*-th bit node. Clearly, the PEG and PEG-like algorithms do not spread a sub-graph at the beginning of the process, resulting in less and comparable complexity. Nonetheless, after the edges begin to connect among another, the complexity increases rapidly because the sub-graph spreading from the bit node $s_j$ will have the increase number of check nods in $\overline{N}_{s_j}^{l}$. Afterwards, if the sub-graph spreading from the bit node $s_j$ can cover all check nodes up to depth $l$, the candidate choice for check nodes decreases; this leads to less complexity as shown in Fig. 5. Eventually, the searching time for check nodes will be reduced.

In addition, to confirm that the PEG-like algorithm has less complexity than the regular PEG algorithm, we also compare their complexity for the (1016, 2032) **H** matric with a column weight of 3. Again, the sub-graph does not spread at the beginning of the process (up to 400 bits) as shown in Fig. 6, leading to similar complexity as for the case of a column weight of 2. After that the complexity in terms of searching time of the check nodes will increase significantly because of an increased depth $l$. Subsequently, when the sub-graph spreads from a bit node $s_j$ that covers many check nodes, a candidate choice for the check nodes decreases. Thus, this causes the complexity factor to reduce gradually in terms of searching times.

## VI. CONCLUSION

To reduce the complexity of the PEG algorithm, we proposed the PEG-like algorithm based on the topology matrix to create a parity-check matrix for LDPC codes. In our algorithm procedure, the topology matrix employs less time for expanding the sub-graph because we do not search the bit nodes. Thus, the PEG-like algorithm can reduce the time in sub-graphing procedure, while resulting in the same parity-check matrix as the PEG algorithm generates.

## REFERENCES

[1] C. E. Shannon, “A mathematical theory of communication,” The Bell System Technical Journal, vol. 27, 1948.

[2] R. Gallager, “Low-density parity-check codes,” IRE Transactions on Information Theory, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[3] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes,” Electron. Lett., vol. 32, pp. 1645–1646, Aug. 1996.

[4] N. Wiberg, “Codes and decoding on general graphs,” Dissertation no.440

[5] [1] Yu Kou; Shu Lin; Fossorier, M.P.C., "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *Information Theory, IEEE Transactions on*, vol.47, no.7, pp.2711,2736, Nov 2001.

[6] Xiao-Yu Hu; Eleftheriou, E.; Arnold, D.-M., "Regular and irregular progressive edge-growth tanner graphs," Information Theory, IEEE Transactions on , vol.51, no.1, pp.386,398, Jan. 2005.

[7] Zhiheng Zhou; Xiangxue Li; Dong Zheng; Kefei Chen; Jianhua Li, "Extended PEG Algorithm for High Rate LDPC Codes," Parallel and Distributed Processingwith Applications, 2009 IEEE International Symposium on , vol., no., pp.494,498, 10-12 Aug. 2009

[8] Prompakdee, P.; Phakphisut, W.; Supnithi, P., "Quasi Cyclic-LDPC codes based on PEG algorithm with maximized girth property," Intelligent Signal Processingand Communications Systems (ISPACS), 2011 International Symposium on , vol., no., pp.1,4, 7-9 Dec. 2011

[9] [6] Richter, G., "An Improvement of the PEG Algorithm for LDPC Codes in the Waterfall Region," Computer as a Tool, 2005. EUROCON 2005.The International Conference on , vol.2, no., pp.1044,1047, 21-24 Nov. 2005

[10] R. M. Tanner, "A recursive approach to low complexity codes", IEEE Trans. Inf. Theory, vol. IT-27, no. 6, pp.533 -547