# On a Construction Method of Irregular LDPC Codes Without Small Stopping Sets

G. Richter and A. Hof

University of Ulm
Department of Telecommunications and Applied Information Theory
Albert-Einstein-Allee 43, D-89081 Ulm, Germany
{gerd.richter,axel.hof}@uni-ulm.de

***Abstract*— In this paper, we present a construction method based on the progressive edge-growth (PEG) algorithm to design irregular low-density parity-check (LDPC) codes without small stopping sets. We show how to choose the connections in the PEG algorithm when having multiple choices to connect a variable node with a check node. Since preventing small stopping sets also prevents a low minimum distance, our construction method also leads to LDPC codes with a higher minimum distance. Furthermore, we show by simulation that our construction method improves the performance over the binary erasure channel and over the additive white Gaussian noise channel for a low erasure probability and a high signal-to-noise ratio, respectively.**

***Keywords* – Irregular LDPC codes, PEG algorithm, error ¤oor, stopping sets, minimum distance**

## I. Introduction

Low-density parity-check (LDPC) codes were originally invented by Gallager in 1963 [1]. He showed that LDPC codes are capable of reaching a performance close to the channel capacity at low complexity, when they are decoded by an iterative decoding algorithm, the so-called belief propagation or sum-product algorithm. After being forgotten for more than 30 years, LDPC codes were rediscovered in 1996 by Mackay and Neal [2] and also by Wiberg [3]. Because of their good performance and their comparably low decoding complexity, they became serious competitors to turbo codes [4].

Gallager considered only regular LDPC codes, i.e., codes that are represented by a sparse parity-check matrix with a constant number of ones in each row and in each column. Later, it was shown that the performance of LDPC codes in the waterfall region can be improved by using irregular LDPC codes [5], [6], [7].

The drawback of irregular LDPC codes is that they exhibit an error ¤oor that is caused by small stopping sets for the binary erasure channel (BEC) and by codewords with small Hamming weight and by small trapping sets for the additive white Gaussian noise (AWGN) channel [8]. Hence, many different construction methods were introduced to lower the error ¤oor of irregular LDPC codes, e.g., the progressive-edge growth (PEG) algorithm [9], maximizing the approximate cycle extrinsic message degree (ACE) [10], a combination of these two [11], or an improvement of the ACE algorithm [12].

In this paper, we modify the PEG algorithm to construct LDPC codes without small stopping sets and with a large minimum distance. During the PEG algorithm it often happens that there are multiple choices to connect a variable node with a check node. In the standard case, one chooses either the one with the smallest index or just one at random. We modify the PEG in such a way, that we forbid to connect a variable node with a check node, that could lead to a small stopping set or a small minimum distance, and choose another.

The paper is organized as follows: In Section II some basic de£nitions are given, while Section III recalls the PEG algorithm. After describing, how to choose the check nodes in the PEG algorithm to prevent small stopping sets in Section IV, we describe ef£cient algorithms to detect such small stopping sets in Section V. Section VI compares our construction method with existing construction algorithms and shows simulation results. Finally, we conclude our paper in Section VII.

## II. Definitions and Notations

Every binary LDPC code of length $n$ and dimension $k$ can be represented as Tanner graph [13]. This graph consists of two sets of nodes connected by edges. One set, the variable nodes, corresponds to the $n$ columns of the parity-check matrix $\mathbf{H}$ and the other set, the check nodes, represents the rows of the parity-check matrix. The number of rows in $\mathbf{H}$ is denoted by $m$, where $m \geq n-k$. A one in the parity-check matrix in row $j$ and in column $i$ corresponds to an edge between the $i$-th variable node $v_i$ and the $j$-th check node $c_j$. A check node $c_j$ is called a neighbor of a variable node $v_i$, if there exists an edge between $c_j$ and $v_i$. The number of edges incident to $v_i$ is called the variable node degree $d(v_i)$, which is equal to the number of ones in column $i$. Similarly, the number of edges connected with $c_j$ is called the check node degree $d(c_j)$ and is equal to the number of ones in row $j$. This is demonstrated in Example 1.

**Example 1** Assume the parity-check matrix $\mathbf{H}$ of an LDPC code of length $n = 10$ and dimension $k = 5$ is given by

$$H = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The Tanner graph representing this LDPC code is depicted in Fig. 1.

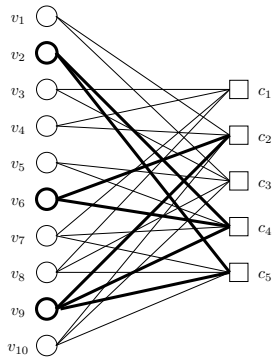


Fig. 1. An irregular LDPC code

We enumerate the edges of a variable node in an arbitrary way and call $e_{v_i}^k, k = \{1, \ldots, d(v_i)\}$ the $k$-th edge of $v_i$. We de£ne $\mathcal{N}_{v_i}^l$ as the set consisting of all check nodes reached by a tree spreading from $v_i$ within depth $l$ and we de£ne $\overline{N_{v_i}^l}$ as the complementary set. Let $d_{v_{\max}}$ and $d_{c_{\max}}$ denote the maximum variable node and check node degree, respectively, and let $\lambda_i$ and $\rho_i$ represent the fraction of edges emanating from variable and check nodes of degree $i$. Then we can de£ne

$$\lambda(x) = \sum_{i=2}^{d_{v_{\max}}} \lambda_i x^{i-1} \text{ and } \rho(x) = \sum_{i=2}^{d_{c_{\max}}} \rho_i x^{i-1}$$

as the *variable node degree distribution* and the *check node degree distribution*, respectively.

A cycle in a Tanner graph is de£ned as follows:

**De£nition 1** *A cycle is a path from a variable node $v_i$ back to itself, if any edge in the path is used only once. The length of the cycle is the number of edges contained in this cycle.*

It is shown in [10] that some cycles are more harmful than others. To distinguish these cycles the ACE metric is introduced in [10], which is de£ned as follows:

**De£nition 2** *The ACE of a length $2\ell$ cycle is $\sum_i (d_i - 2)$, where $d_i$ is the degree of the $i$-th variable node in this cycle. An LDPC code has ACE property $(\ell_{\mathrm{ACE}}, \nu_{\mathrm{ACE}})$, if all the cycles whose length are $2\ell_{\mathrm{ACE}}$ or less have ACE values of at least $\nu_{\mathrm{ACE}}$.*

Di *et al.* [14] pointed out the crucial role of stopping sets in the erasure decoding algorithm for LDPC codes [15]. A stopping set is de£ned as follows:

**De£nition 3** *A stopping set $\mathcal{S}$ is a subset of $\mathcal{V}$, the set of variable nodes, such that all neighbors of the variable nodes in $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice. The size of a stopping set $s$ is de£ned as the cardinality of $\mathcal{S}$.*

It can be seen in Fig. 1 that the set $\{v_2, v_6, v_9\}$ is a stopping set. It is shown in [14] that the set of erasures, which remains when the iterative erasure decoding algorithm stops is equal to the unique maximum stopping set. Since every codeword with small Hamming weight is caused by a stopping set with small size, preventing small stopping sets also helps to increase the minimum distance of LDPC codes.

Furthermore, we de£ne the check node distance that is similar to the minimum row distance (MRD) used in [12].

**De£nition 4** *The check node distance $d_c(i, j)$ of the check nodes $c_i$ and $c_j$ is de£ned as*

$$d_c(i,j) = \begin{cases} |i-j| \text{ for } i, j \leq n_2 + 1 \\ \infty \text{ otherwise,} \end{cases} \quad (1)$$

*where $n_2$ is the number of degree-2 variable nodes.*

## III. PEG Algorithm

In this section, we recall the PEG algorithm introduced by Hu *et al.* in [9]. Here, we describe the version, where the check node is chosen randomly from all possible candidates.

Similar as in [9], we summarize the PEG algorithm as follows:

- Sort the variable nodes such that $d(v_i) \leq d(v_j)$ for $i < j$.
- For $i = 1$ to $n$
  - For $k = 1$ to $d(v_i)$
    - If $k = 1$
      - Connect the £rst edge $e_{v_i}^1$ of $v_i$ randomly with a check node, which has the lowest check node degree under the current graph setting.
    - Else
      - Expand a tree from $v_i$ up to depth $l$ under the current graph setting such that $\overline{N_{v_i}^l} \neq \emptyset$ but $\overline{N_{v_i}^{l+1}} = \emptyset$, or the cardinality of $N_{v_i}^l$ stops increasing but is less than $m$.
      - Connect the edge $e_{v_i}^k$ randomly with a check node from the set $\overline{N_{v_i}^l}$ having the lowest degree under the current setting.
    - End (if...else)
  - End (for $k = 1$ to $d(v_i)$)
- End (for $i = 1$ to $n$)

Note that sorting the variable nodes in the beginning of the PEG algorithm prevents having small cycles between variable nodes with low degree. Thus, the resulting LDPC codes have automatically a large ACE property.

## IV. Modifications of the PEG algorithm

In this section, we describe our modi£cations of the PEG algorithm. These modi£cations can be applied when the number of degree-2 variable nodes is smaller than $m$.

As already mentioned in [9], we connect the degree-2 nodes in a zigzag manner, which is often called the staircase construction. Thus, we can guarantee that there are no cycles between degree-2 variable nodes. Note that exactly $d_c(i, j)$ degree-2 variable nodes are needed to connect $c_i$ with $c_j$, when $i$ and $j$ are at most $n_2 + 1$. If $i$ or $j$ is larger than $n_2 + 1$,

then there exists no connection between $c_i$ and $c_j$ with only degree-2 variable nodes.

Our observations showed that the most stopping sets with small size contain two degree-3 variable nodes and degree-2 variable nodes. Also many stopping sets with small size contain four degree-3 variable nodes and degree-2 variable nodes. Hence, we forbid to choose connections to check nodes that lead to stopping sets of size smaller than the desired minimum stopping set size $\hat{s}$ with 4 or less degree-3 variable nodes.

Thus, if we choose the connection for a degree-3 variable node, we mark all the check nodes from the set $\overline{N_{v_i}^l}$ as not selectable, if the connection probably leads to a small stopping set. If there are no selectable check nodes in the set $\overline{N_{v_i}^l}$, we choose a check node from the set $\overline{N_{v_i}^{l-1}}$ that does probably not lead to a small stopping set. In the following, we show how to mark the check nodes as not selectable.

Stopping sets with size smaller than $\hat{s}$ and only one degree-3 variable node $v_{i_1}$ can only occur, when the sum of the check node distances $d_c(j_1, j_3) = d_c(j_1, j_2) + d_c(j_2, j_3) < \hat{s} - 1$, where $j_1 < j_2 < j_3$ and $c_{j_1}, c_{j_2}, c_{j_3}$ are neighbors of $v_{i_1}$. This is demonstrated in Fig. 2. The stopping set in this £gure contains one degree-3 variable node and $d_c(j_1, j_3)$ degree-2 variable nodes.

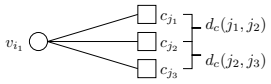


Fig. 2. Stopping set with one degree-3 variable node

Thus, we forbid to choose check nodes, for which the distances satisfy $d_c(j_1, j_2) < \lceil(\hat{s} - 1)/2\rceil$ or $d_c(j_2, j_3) < \lceil(\hat{s} - 1)/2\rceil$. This prevents also stopping sets with two degree-3 variable nodes as shown in Fig. 3 and some stopping sets with more than two degree-3 variable nodes.

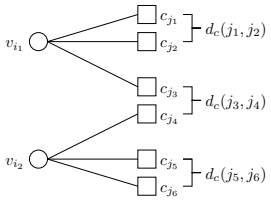


Fig. 3. Stopping set with two degree-3 variable nodes

Furthermore, Fig. 4 shows another stopping set that can occur with two degree-3 variable nodes.

When connecting the third edge of $v_{i_1}$ we choose only a check node $c_{j_5}$, for which the sum of the check node distances $d_c(j_1, j_2) + d_c(j_3, j_4) + d_c(j_5, j_6)$ in Fig. 4 is not smaller than $\hat{s} - 2$ for the 3 check nodes that are connected with any other degree-3 variable node.

We prevent the PEG algorithm to produce stopping sets with three degree-3 variable nodes of size smaller than $\hat{s}$ by marking the check nodes as not selectable, for which the sum

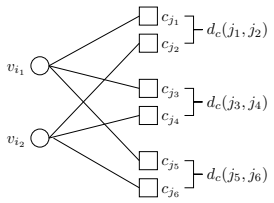


Fig. 4. Stopping set with two degree-3 variable nodes

of two distances in Fig. 4 is smaller than $\lfloor(\hat{s} - 2)/2\rfloor$. This also reduces the probability to create small stopping sets with more than three degree-3 variable nodes.

After marking all the check nodes as not selectable as described above, the only possible stopping set with four degree-3 variable nodes of size smaller than $\hat{s}$ that can occur is shown in Fig. 5.

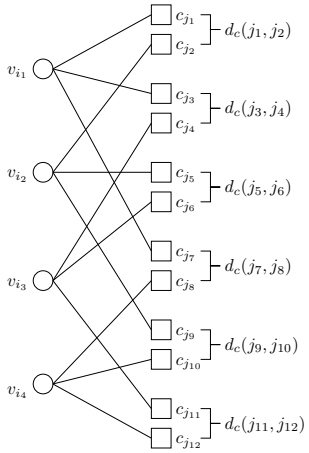


Fig. 5. Stopping set with four degree-3 variable nodes

Hence, when connecting the last edge of any degree-3 variable node $v_{i_1}$, we forbid to choose a check node $c_{j_7}$ that leads to a stopping set of size smaller than $\hat{s}$. The connection creates such a stopping set, if the sum of the six distances in Fig. 5 is smaller than $\hat{s} - 4$.

The last change to the original PEG algorithm is how to choose the connections to the remaining check nodes. Because most stopping sets with more than four degree-3 variable nodes contain at least two cycles as depicted in Fig. 6, we choose only connections to check nodes, where the cycle distance $c = \min\left(d_c(j_1, j_2) + d_c(j_3, j_4) + d_c(j_5, j_6)\right) > \lfloor(\hat{s} - 5)/2\rfloor$ between $v_{i_1}$ and any other two degree-3 variable nodes. If this is not possible, we choose the connections such that $c$ is maximized. This reduces the probability to produce small stopping sets with more than four degree-3 variable nodes and also the probability to produce small trapping sets [8]. If we have more check nodes with $c > \lfloor(s-5)/2\rfloor$ or with the same $c$ (which is usually the case), we just select one check node randomly having the lowest degree under the current graph setting. This ensures that the check node degree grows in a concentrated fashion as in the original PEG algorithm.

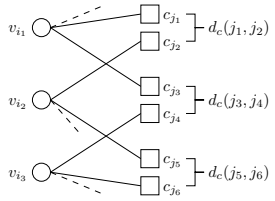


Fig. 6. Cycle between three degree-3 variable nodes

For any other variable nodes $v_i$ with $d(v_i) > 3$, we select a check node of the set $N_{v_i}^l$ having the lowest check node degree under the current graph setting. Additionally, we can take care that the minimum check node distance between the check node connected with the new edge and all the check nodes connected with the other edges of this variable node is not too small. This lowers the probability to create small stopping sets with variable nodes with $d(v_i) > 3$. It is also possible to avoid small stopping sets with variable nodes with $d(v_i) > 3$, but this results in a larger complexity.

## V. Algorithms to Detect Small Stopping Sets

In this section, we describe ef£cient algorithms, which can detect, if a connection leads to a stopping set as depicted in Section IV of size smaller than $\hat{s}$. The complexity of all these algorithms does not depend on the codelength $n$, but only on the size of the desired minimal stopping set $\hat{s}$. All stopping sets can be detected by spanning trees and adding some distances.

The stopping sets shown in Fig. 2 and in Fig. 3 can be easily detected by calculating the check node distances of the check nodes connected with one variable node.

To detect a stopping set of size smaller than $\hat{s}$ as shown in Fig. 4, we span a one-level tree from the degree-3 variable node $v_{i_1}$, which is connected to the check nodes $c_{j_1}$, $c_{j_3}$, and $c_{j_5}$. This tree contains all check nodes $c_l$, for which $d_c(j_1, l) \leq \hat{s}-3$, $d_c(j_3, l) \leq \hat{s}-3$, or $d_c(j_5, l) \leq \hat{s}-3$. Every check node in this tree is labeled with the check node distance. Furthermore, all degree-3 variable nodes, which are connected to the check nodes $c_l$ are included in this tree. The variable nodes get the same labels as the check nodes they are connected to. This is shown in Fig. 7.

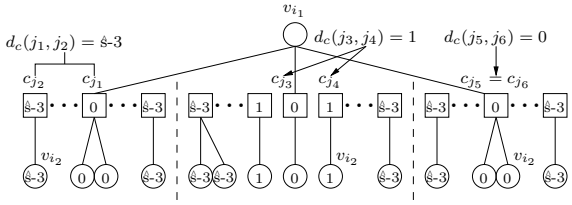


Fig. 7. Algorithm to detect stopping sets with two degree-3 variable nodes

This £gure can be divided into three parts. If a variable node is found in all three parts of the tree and the sum of the labels is smaller than $(\hat{s} - 2)$, a stopping set of size smaller than $\hat{s}$ as depicted in Fig. 4 is detected. In the example of Fig. 7, a stopping set of size $\hat{s}$ is found.

By spanning a two-level tree from $v_{i_1}$ that is connected with two of its edges to $c_{j_1}$ and $c_{j_3}$, we can check, if the cycle distance $c$ from Fig. 6 is smaller than $t = \lfloor(\hat{s} - 5)/2\rfloor$. The tree consists of all check nodes $c_l$, for which $d_c(j_1, l) \leq t$ or $d_c(j_3, l) \leq t$. Again every check node $c_l$ and every degree-3 variable node that is connected to $c_l$ is labeled with the check node distance. This tree can be divided into two parts and can be seen in Fig. 8.

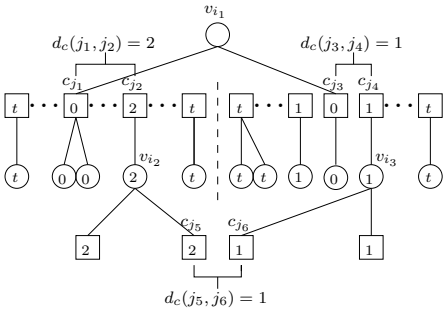


Fig. 8. Algorithm to calculate the cycle distance $c$

For every combination of a variable node of the left part in the tree and a variable node of the right part of the tree, for which the sum of the labels is at most $t$, we calculate $d_c(j_5, j_6)$ and $c = d_c(j_1, j_2) + d_c(j_3, j_4) + d_c(j_5, j_6)$. In the example of Fig. 8, a cycle with $c = 4$ is found.

For the detection of a stopping set with four degree-3 variable nodes as shown in Fig. 5, we £rst determine the variable nodes, for which the cycle distance $c \leq \hat{s} - 5$, before we connect the third edge. After connecting the third edge to the check node $c_{j_7}$, we span a tree that contains all check nodes $c_l$ with $d_c(j_7, l) \leq \hat{s} - 5$ and the variables nodes $v_h$ connected to $c_l$. Then we check for every combination of $v_h$ and the variable nodes included in a cycle with $c \leq \hat{s} - 5$, if a small stopping set is found. This is shown in Fig. 9.

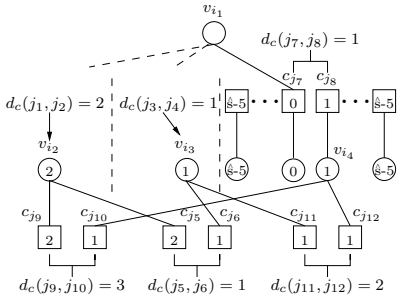


Fig. 9. Algorithm to detect stopping sets with four degree-3 variable nodes

The sum of the six distances, which is 10 in this £gure, is equal to the size of the stopping set minus 4.

## VI. Simulation Results

In this section, we compare LDPC codes constructed with the modi£ed PEG algorithm described in Section IV with LDPC codes constructed with the original PEG algorithm

introduced in [9] and with the combination of the PEG algorithm and the ACE condition investigated in [11]. Since all these algorithms are pseudorandom, the smallest stopping set size $s_{\min}$ and the minimum distance $d$ can vary a lot for the same construction. Therefore, we construct 50 LDPC codes with different random seeds for each algorithm and determine the number of stopping sets of small size with the two error impulse algorithm described in [16]. After that we compare the frame error rates (FERs) of LDPC codes constructed with the original PEG algorithm and with the modi£ed PEG algorithm for a transmission over the BEC and the AWGN channel, respectively. For the transmission over the BEC we used the ef£cient erasure decoding algorithm described in [15]. For the AWGN channel all simulations were done with the shuf¤ed belief propagation decoder described in [17], with a maximum number of 500 iterations, and with the linear approximation described in [18].

For the £rst simulations, we used an LDPC code ensemble with length $n = 1000$, rate $R = 1/2$, and variable node degree distribution $\lambda(x) = 0.283x^1 + 0.281x^2 + 0.436x^8$.

Table I shows the number of stopping sets in 50 LDPC codes and the average minimum distance $\overline{d}$ of these 50 codes constructed with the different algorithms. The £rst and the second row stand for the PEG algorithm and the combination of PEG and ACE, respectively. The last row represents the modi£ed PEG algorithm described in Section IV with $\hat{s} = 18$.

| Size | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | $\overline{d}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| PEG | 1 | 7 | 15 | 23 | 53 | 93 | 156 | 246 | 448 | 12.0 |
| P+A | - | 3 | 10 | 18 | 23 | 66 | 122 | 199 | 380 | 12.8 |
| Mod | - | - | - | - | - | 1 | 5 | 20 | 70 | 16.7 |

TABLE I

NUMBER OF STOPPING SETS OF LDPC CODES ($R = 1/2$, $n = 1000$)

While the improved PEG algorithm investigated in [11] only shows a slight improvement, the average minimum distance $\overline{d}$ is increased from 12.0 to 16.7 by using the modi£ed PEG algorithm described in Section IV. Also the number of stopping sets with size smaller than $s = 18$ is reduced from 1042 to 96. There are no stopping sets of size smaller than $s = 18$ that contain less than £ve degree-3 variable nodes in the 50 LDPC codes constructed with the modi£ed PEG algorithm. The best code constructed with the original PEG algorithm and with the combination of PEG and ACE has $s_{\min} = d = 15$. The best LDPC code constructed with the modi£ed PEG algorithm has $s_{\min} = d = 18$.

Fig. 10 shows the FERs of the LDPC codes with the smallest and the largest $s_{\min}$ constructed with the PEG algorithm and with the modi£ed PEG algorithm for a transmission over the BEC.

We can see that the best LDPC code constructed with the modi£ed PEG algorithm outperforms the two codes constructed with the original PEG algorithm. Even the worst LDPC code constructed with the modi£ed PEG algorithm

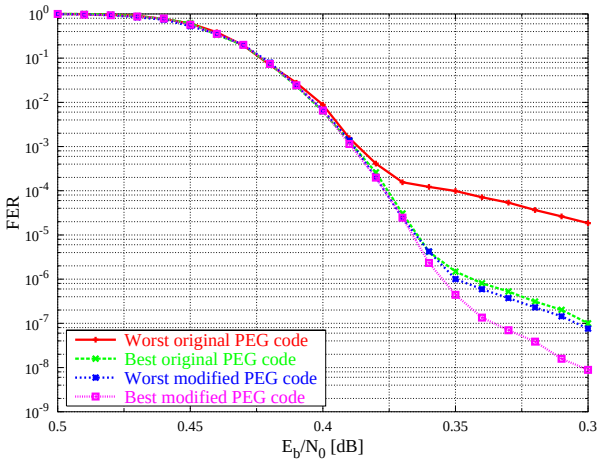


Fig. 10. FERs over the BEC ($R = 1/2$, $n = 1000$)

shows a slightly better FER than the best LDPC code constructed with the original PEG algorithm for the simulated region.

In Fig. 11, the performance of the same four LDPC codes for a transmission over the AWGN channel can be seen.

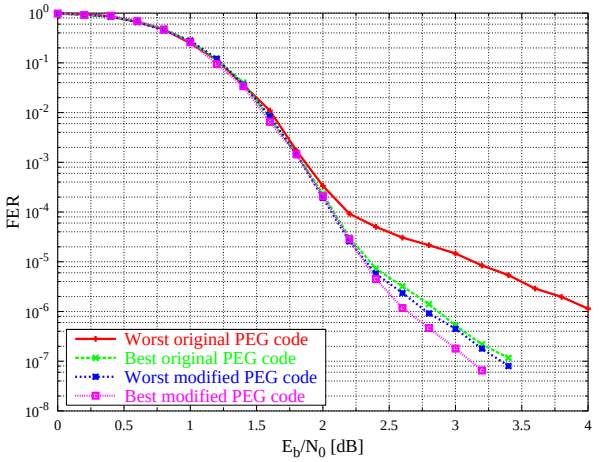


Fig. 11. FERs over the AWGN channel ($R = 1/2$, $n = 1000$)

As for the transmission over the BEC, the two LDPC codes constructed with the modi£ed PEG algorithm outperform the two codes constructed with the original PEG algorithm.

Similar to Table I, Table II shows the stopping set size distribution of 50 LDPC codes with $n = 2000$, $R = 3/4$, and $\lambda(x) = 0.1245x^1 + 0.4460x^2 + 0.4078x^{10} + 0.0213x^{11}$ constructed with the different algorithms.

Again, we see that the modi£ed PEG algorithm lowers the number of stopping sets with size smaller than $s = 12$ and increases the average minimum distance from $\overline{d} = 6.4$ to $\overline{d} = 9.3$. The best LDPC code constructed with the original PEG algorithm has $s_{\min} = d = 7$, while the best code constructed with the modi£ed PEG algorithm has $s_{\min} = d = 10$.

In Fig. 12 and in Fig. 13, one can see the performance of the

| Size | 5 | 6 | 7 | 8 | 9 | 10 | 11 | $\bar{d}$ |
|---|---|---|---|---|---|---|---|---|
| PEG | 2 | 37 | 118 | 322 | 808 | 2497 | 8107 | 6.4 |
| P+A | - | - | 34 | 171 | 553 | 1698 | 5563 | 7.5 |
| Mod | - | - | - | 5 | 49 | 396 | 2160 | 9.3 |

TABLE II

NUMBER OF STOPPING SETS OF LDPC CODES ($R = 3/4$, $n = 2000$)

LDPC codes with the smallest and the largest $s_{\min}$ constructed with the original PEG algorithm and with the modi£ed PEG algorithm for a transmission over the BEC and the AWGN channel, respectively.

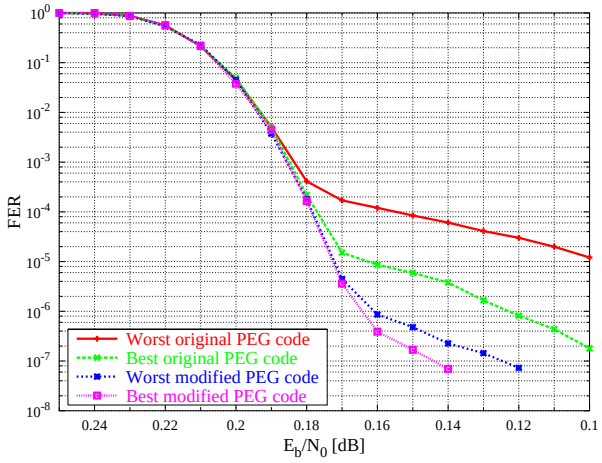


Fig. 12. FERs over the BEC ($R = 3/4$, $n = 2000$)

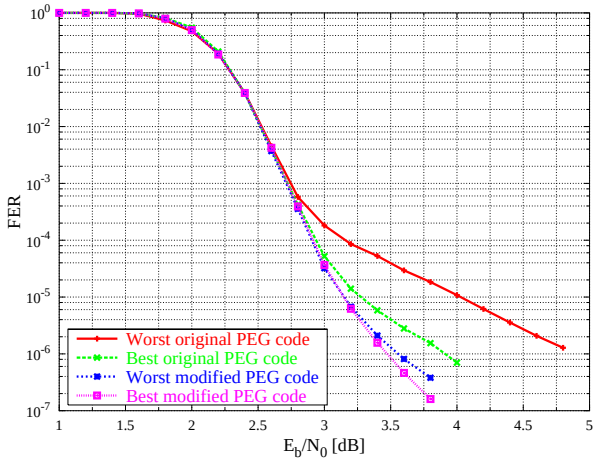


Fig. 13. FERs over the AWGN channel ($R = 3/4$, $n = 2000$)

It is observed that even the worst of the 50 LDPC codes constructed with the modi£ed PEG algorithm shows quite a large improvement compared to the best of the 50 LDPC codes constructed with the original PEG algorithm for a low erasure probability or a high SNR. This is due to the fact that the worst LDPC code constructed with the modi£ed PEG algorithm has $s_{\min} = d = 8$, which is larger than the distance and the smallest stopping set size of the best LDPC code constructed with the original PEG algorithm.

## VII. CONCLUSIONS

In this paper, we constructed irregular LDPC codes without small stopping sets by modifying the PEG algorithm. Therefore, we showed how to choose the connections to the check nodes when there are multiple choices. Thus, we prevent the algorithm to produce stopping sets of small size. We demonstrated by the number of stopping sets with small sizes as well as by simulations that LDPC codes constructed with the modi£ed PEG algorithm outperform LDPC codes constructed with the original PEG algorithm.

## ACKNOWLEDGMENTS

This work was supported by the German research council Deutsche Forschungsgemeinschaft (DFG) under Grant Bo 867/12. The authors would like to acknowledge the DFG for their support.

## REFERENCES

[1] R. G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, Cambridge, 1963.

[2] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low-density parity-check codes. *Electron. Lett.*, 32:1645–1646, August 1996.

[3] N. Wiberg. *Codes and Decoding on General Graphs*. PhD thesis, Linköping University, Sweden, 1996.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes(1). In *IEEE International Conference on Communications*, pages 1064–1070, Geneva, 1993.

[5] D. J. C. MacKay, S. T. Wilson, and M. C. Davey. Comparison of constructions of irregular codes. *36th Allerton Conference Communications, Control, and Computing*, September 1998.

[6] T. J. Richardson, M. A. Shokrollahi, and R. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inf. Theory*, 47(2):617–637, February 2001.

[7] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Trans. Inf. Theory*, 47(2):585–598, February 2001.

[8] T. Richardson. Error ¤oors of LDPC codes. In *41st Annual Allerton Conference on Communications, Control, and Computing*, Allerton, England, October 2003.

[9] X. Y. Hu, E. Eleftheriou, and D. M. Arnold. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Trans. Inf. Theory*, 51(1):386–398, January 2005.

[10] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel. Selective avoidance of cycles in irregular LDPC code construction. *IEEE Trans. Inf. Theory*, 51(1):386–398, January 2004.

[11] H. Xiao and A. H. Banihashemi. Improved progressive-edge growth (PEG) construction of irregular codes. *IEEE Communication Letters*, 8(12):715–717, December 2004.

[12] L. Dinoi, F. Sottile, and S. Benedetto. Design of variable-rate irregular LDPC codes with low error ¤oor. In *IEEE International Conference on Communications*, Seoul, Korea, May 2005.

[13] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory*, IT-27:533–547, September 1981.

[14] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inf. Theory*, 48(6):1570–1579, June 2002.

[15] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Ef£cient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47(2):569–584, February 2001.

[16] G. Richter. Finding small stopping sets in the Tanner graphs of LDPC codes. In *4th International Symposium on Turbo Codes and Related Topics*, Munich, Germany, April 2006.

[17] J. Zhang and M. P. C. Fossorier. Shuf¤ed iterative decoding. *IEEE Trans. Commun.*, pages 209–213, February 2005.

[18] G. Richter, G. Schmidt, M. Bossert, and E. Costa. Optimization of a reduced-complexity decoding algorithm for LDPC codes by density evolution. In *IEEE International Conference on Communications*, Seoul, Korea, May 2005.