Decoder-Optimised Progressive Edge Growth Algorithms for the Design of LDPC Codes with Low Error Floors

C.T. Healy and R. C. de Lamare

***Abstract*—A novel construction for irregular low-density parity-check (LDPC) codes based on a modification of the Progressive Edge Growth (PEG) algorithm is presented. Edge placement of the PEG algorithm is enhanced by use of the Sum-Product algorithm in the design of the parity-check matrix. The proposed algorithm is highly flexible in block length and rate. The codes constructed by the proposed methods are tested in the AWGN channel and significant performance improvements are achieved. The flexibility of the proposed decoder optimisation operation is then shown by its use in modifying the Improved PEG (IPEG) algorithm to achieve further performance gains.**

I. Introduction

Low-density parity-check (LDPC) codes are a class of capacity approaching codes first introduced by Gallager [1], extended to the irregular degree distribution case by Luby et al. [2] and provided with an analytical tool, density evolution (DE), for optimising the degree distribution under certain conditions by Richardson et al. [3]. For DE, a notable assumption, with respect to realising practical codes, is that the decoding neighbourhood of a given variable node (VN) is tree-like [3]. This is true in the case of infinite length codes and the assumption approximately holds for codes with large block length. However, for practical codes of medium to short length, this assumption is not verified. As a consequence, the assumption of independence of messages passed under sum-product decoding breaks down. A major focus in the search for practical finite length codes is the mitigation of the effects of the cycles which break down the independence assumption. Approaches include maximising the girth and improving the connectivity of the cycles in finite length codes, as typified by the ACE metric presented by Tian et al. [4].

Among those codes capable of best performance at practical lengths are codes designed by the Progressive Edge Growth algorithm [5]. The PEG algorithm is a greedy edge placement construction method for the parity-check matrix of an LDPC code which places edges in the Tanner graph of the code such that when a cycle is created, that cycle is of the maximum possible length under the current settings of the matrix. This algorithm produces LDPC codes with large girth and with particularly large local girth in the lower weight VN subgraph of the parity-check matrix, leading to improved performance. The PEG algorithm is a particularly versatile code construction algorithm, in both length and rate. In addition, it may be applied to the construction of structured LDPC codes, as was demonstrated in previous work of the authors [6].

C.T. Healy and R. C. de Lamare are with the Communications Research Group, Department of Electronics, University of York, York Y010 5DD, United Kingdom. Emails: cth503@ohm.york.ac.uk rcdl500@ohm.york.ac.uk

The significant performance gains provided by both the local girth maximisation technique of the PEG algorithm and the graph connectivity maximisation technique of the ACE design procedure naturally lead to the combination of the methods. In this case, a set of check nodes is identified which will produce cycles of equal maximum possible length and the check node chosen is that which has maximum connectivity according to the ACE metric. Significant performance gain has been demonstrated with codes generated by this method [7] [8] [9] particularly in the error floor region.

In this work, we propose an approach to improve the design of PEG-based techniques which involves use of decoder-based optimisation with the sum-product algorithm (SPA). During construction of the parity-check matrix, the PEG algorithm regularly provides a number of check node (CN) candidates, each of which is at the maximum distance possible from the current VN, and so will lead to the creation of cycles of equal length when the edge is placed. The proposed decoder-optimised (DO) PEG algorithm compares the performance of the code under the current graph setting for each of the candidate CNs. The edge which produces the best performance according to the SPA is then selected. The proposed DOPEG and DOIPEG algorithms provide codes with improved performance in the error floor region at the cost of increased complexity in code construction.

II. Definitions and Notation

The variable node degree sequence D_v is defined as the set of column weights of the size $m \times n$ LDPC parity-check matrix (PCM) $\mathbf{H}$ designed, and is prescribed by the variable node degree distribution $\lambda(x)$ as described in [3]. D_v is arranged in non-decreasing order. The PEG algorithm constructs the PCM by operating progressively on variable nodes to place the edges required by D_v. The PEG algorithm chooses a check node c_i to connect to the variable node of interest v_j by expanding a subgraph from v_j up to maximum depth l. The set of check nodes found in this subgraph is denoted $N_{v_j}^l$ while the set of check nodes of interest, those not currently found in the subgraph, are denoted $\overline{N_{v_j}^l}$. For the PEG algorithm, a check node is chosen at random from the minimum weight check nodes of this set. This is the set upon which the DO operation is performed in the DOPEG modification, and upon which the ACE comparison is performed in the IPEG modification.

III. Decoder-Optimised PEG and IPEG Algorithms

With high regularity, the original PEG algorithm is faced with a set of check nodes which are equivalent according to the metric by which it compares candidates, that is the

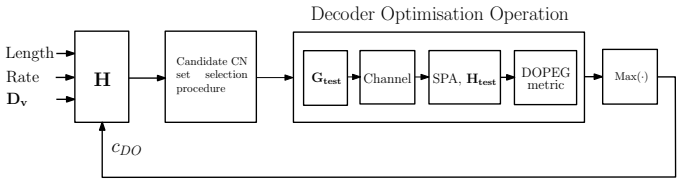

Fig. 1. Block diagram of the proposed decoder-based optimisation.

distance from the current variable node of interest in a tree expanded from that VN. Likewise for the IPEG algorithm, albeit with less regularity, a set of CNs will be found to be equivalent in both distance metric of the PEG algorithm and graph connectivity as measured by the ACE metric. For both algorithms the strategy in this case is to choose a CN at random from these sets of equivalent candidates. The motivation for this work was to find the candidate from each of these sets which provides the best performance. To this end, the decoder-based optimisation was developed.

The candidate code for check node c_i is formed in the following way: $\mathbf{H}_{\text{test}}$, the PCM of the candidate code, is simply the constructed PCM under the current setting with the addition of a '1' placed in the position [c_i,v_j]. The node v_j is the current variable node of interest. The DO operation is used for $j \geq m+1$, ensuring that the candidate codes generated have more VNs than CNs. Each candidate code is used in decoding a preset number of all-zero codeword vectors as follows: The codewords are subjected to additive white Gaussian noise (AWGN) over a range of values of signal-to-noise ratio (SNR) and the candidate PCM is used to decode through log-domain SPA decoding. The performance of each candidate code is evaluated and the code which provides the best performance indicates which candidate check node to choose for edge placement. This is outlined in Fig. 1. When the block 'candidate CN set selection procedure' delivers $\overline{N_{v_j}^l}$ of the PEG algorithm, this block diagram represents the DOPEG. When this block delivers the set of nodes in $\overline{N_{v_j}^l}$ with equal maximum ACE metric then the DOIPEG is represented.

The graph of the chosen candidate code forms a subgraph of the final graph. Since the graphs of the candidate codes are almost identical, differing in only one entry with that entry creating a cycle of equal length in each case, the difference in performance of the candidate codes is determined by the connectivity of the cycles each candidate edge creates. At each placement choosing the edge with the best subgraph connectivity as indicated by the DO comparison leads to a graph with better overall connectivity, and an improved performance.

A. Description of Metric Calculation

For each candidate code, the soft-output bit log-likelihood ratios (LLRs) of the decoder are given by

$$L(Q_i) = L(s_i) + \sum_{j \in C_i} L(r_{ji}), \tag{1}$$

where $L(s_i)$ is the channel output LLR for the coded bit s_i and $L(r_{ji})$ is the LLR passed from CN *j* to VN *i* in a half-iteration of the SPA algorithm. C_i is the set of CNs connected to VN *i*. Our goal is to produce a convergence metric CVM for each candidate node CN_α, $\alpha = 1,\ldots,X$ where *X* is the cardinality of the set of minimum weight CNs of $\overline{N_{s_j}^l}$. To this end we define the $Z \times X$ matrix $\mathbf{C}_V$

$$\mathbf{C}_{\text{V}}(\beta,\alpha) = \sum_{t=1}^{Y}\sum_{i=1}^{N} \left(w \cdot |L(Q_i)|\right), \tag{2}$$

$$\text{w} = \begin{cases} 1 & \text{if sgn}(L(Q_i)) = s_i \\ -1 & \text{otherwise} \end{cases}. \tag{3}$$

The variables used in (2) are: *N* is the length of the candidate codeword. *Y* is the number of noise vectors applied code at each SNR point for each candidate code. The integer β indicates the SNR point, $\beta = 1,\ldots,Z$, where *Z* is the total number of SNR points operated over. The convergence metric $\text{CVM}(\alpha)$ for candidate CN_α is then the overall average sum for each candidate at each SNR point.

TABLE I
PSEUDOCODE FOR THE PROPOSED DOPEG ALGORITHM

For $j = 1$ to n
 For $k = 1$ to $D_v(j)$
  If $k == 0$
   Place edge (c_{min}, v_j), c_{min} chosen randomly from the minimum weight CNs of the current graph.
  Else
   Expand tree from v_j until the cardinality of $N_{v_j}^l$ stops increasing but is less than m **or** $\overline{N_{v_j}^l} \neq \emptyset$ but $\overline{N_{v_j}^{l+1}} = \emptyset$.
   If $j < m+1$
    Place edge (c_{min}, v_j), c_{min} chosen randomly from the minimum weight CNs of $\overline{N_{v_j}^l}$.
   Else
    For p = 1 to Length($\overline{N_{v_j}^l}$)
     PCM $\mathbf{H_{test}}$ formed from $\mathbf{H}$ under current graph setting up to column v_j, with edge in position $(\overline{N_{v_j}^l}(p), v_j)$
     Use $\mathbf{H_{test}}$ to decode in the presence of AWGN over the selected SNR range using the log-domain SPA decoder with soft output.
     Compute convergence metrics CVM as described in Section III-A
     Identify CN $c_{DO} = \arg \max_{\alpha} CVM(\alpha)$.
     Place edge in position (c_{DO}, v_j)
    End For
   End If
  End If
 End For
End For

B. Proposed DOIPEG Extension to the DOPEG Algorithm

The IPEG modification of the PEG algorithm provides a method for selecting a candidate CN from the set $\overline{N_{v_j}^l}$ which has greater graph connectivity, leading to improved performance. Taking a similar approach, we extend the DOPEG algorithm, by use of the ACE metric, to the DOIPEG algorithm. The ACE comparison of candidate check nodes is carried out before the DO stage. As such, the DO operation

is carried out on a refined set of CNs which are at equal maximum distance from the VN of interest v_j and which have equal ACE metric, which gives an approximate measure of graph connectivity. These CNs, viewed as equivalent by the IPEG algorithm, are compared by means of the DO operation as in the DOPEG algorithm. The CN which provides the best performance is selected. The gain achieved by codes constructed by the DOIPEG algorithm over IPEG codes is intuitively consistent, the ACE metric gives an inexact measure of the extrinsic message degree (EMD) and so the set of CNs compared by the DO operation may provide differing levels of connectivity. The CN which provides the greatest connectivity will produce an intermediate code with the best performance and will be selected, thus the final graph will have improved structure.

In Table II the DOIPEG modification of the DOPEG algorithm is outlined by means of the pseudocode for the combined ACE metric and DO operations. The section of pseudocode below replaces lines 14 - 22 in the pseudocode of Table I to provide that of the full DOIPEG algorithm.

TABLE II
PSEUDOCODE FOR THE DO STAGE OF THE DOIPEG ALGORITHM

For each CN $c_q \in \overline{N_{v_j}^l}$
  Calculate the metric $ACE_{c_q,\mathrm{path}} = \sum_{v_p}(w_{v_p} - 2)$, where w_{v_p} is
  the weight of column v_p and the summation is taken over all VNs in
  the path from v_p to c_q. This summation is carried out for each path,
  and the final ACE metric associated with c_q is the minimum of
  $ACE_{c_q,\mathrm{path}}$
End For
The set Φ_{v_j} is the set of CNs with equal maximum ACE metrics and
equal maximum distance from the node v_j
If cardinality of Φ_{v_j} == 1
  Edge placed in position (Φ_{v_j}, v_j)
Else
  For p = 1 to Length(Φ_{v_j})
    PCM $\mathbf{H_{test}}$ formed from $\mathbf{H}$ under current graph setting up to
    column v_j, with edge in position $(\overline{N_{v_j}^l}(p), v_j)$
    Use $\mathbf{H_{test}}$ to decode over the SNR range with the log-domain SPA
    decoder with soft output.
    Compute metric as described in Section III-A
  End For
End If

IV. SIMULATION RESULTS

We consider the irregular rate 1/2 code with maximum VN degree 8, from the DE optimal degree distribution of [3] Table II.

$$\lambda_1(x) = .30013x + .28395x^2 + .41592x^7 \qquad (4)$$

The resulting VN degree sequence D_v was modified so that the number of weight-2 VNs was smaller than the number of CNs. As discussed in [3] this ensures that no cycles exist which are composed only of weight-2 VNs. BPSK modulation over the AWGN channel was considered. The log-domain SPA decoder was used in the receiver. In both plots, at least 100 block errors were gathered per point. For the results gathered for Fig. 2, the decoder was operated to a maximum of 50 iterations while for Fig. 3 the decoder was operated to a maximum of 10 iterations. It has been verified that the use of a lower maximum number of iterations does not alter the hierarchy of code performances.

In Fig. 2, the error performances of codes of block length 250 are compared. The PEG and IPEG codes are constructed according to [5], [7], respectively. The DOPEG and DOIPEG codes were constructed with the DO stage operating with identical input parameters, five distinct AWGN vectors were applied to the all-zero codeword at each SNR point in the range specified. Applying a number of independent noise vectors ensures that the difference in performance metric obtained for each candidate check node accurately represents the effect that candidate has on the graph structure, rather than resulting from random variations of the noise applied. The log-SPA decoder was operated to a maximum of 50 iterations. The SNR range operated over was [1:0.05:2]. In the development of the proposed algorithm it was found that the DO operation was successful provided the SNR range selected was in the waterfall region of the error performance of the chosen code ensemble. As expected the DOPEG algorithm provides significant performance improvements over the PEG algorithm. However, both codes are outperformed by the IPEG algorithm. The DOIPEG algorithm provides a performance gain over the IPEG in the error floor region.

Fig. 3 shows that the gain achieved by the proposed construction algorithms is consistent for different block lengths. The results were gathered for an SNR of 4dB with log-SPA decoding used. The DO parameters of the constructed codes were identical to the codes of Fig. 2. The increased gain with block length observed is due to the DO operation having the greatest effect in the error floor region. Direct comparison of gains achieved at different block lengths should be avoided, as the SNR point chosen will lie in a different region of the BER curve for codes of different lengths, and the results show that the DO operation consistently provides gains and is flexible in code length.

As Fig. 3 shows, the DOPEG codes are outperformed by the IPEG codes for the irregular LDPC codes considered. However, if a structure is imposed on the graph of the code which caused the ACE metric of the IPEG algorithm to become less accurate or to fail entirely, it is likely that the DOPEG construction could still be a suitable design tool. Cases where the structure of the graph is limited include quasi-cyclic LDPC codes [10] and irregular repeat accumulate (IRA) codes [11] for faster encoding. Work in these areas is ongoing.

The proposed algorithms require considerably greater complexity in code construction. A comparison of construction times of the existing and proposed algorithms is given in Table III, for each block length and construction algorithm the time is given in seconds as measured using the *tic*/*toc* functions of MATLAB and generated on the same system. It should be noted that the DOPEG algorithm in particular has a very large cost in terms of construction time when compared to the PEG algorithm it is based on. The increase for the DOIPEG over the IPEG algorithm is significantly less, owing to the fact that

it operates on a smaller set of candidate check nodes.

The increased effort in code construction is justified by the performance gains achieved by the DO-based construction methods over codes which themselves perform excellently. It should also be noted that while the DOPEG and DOIPEG methods require increased computation, this cost applies only in the code construction phase. In transmission, the codes generated provide improved performance with no extra cost in complexity.

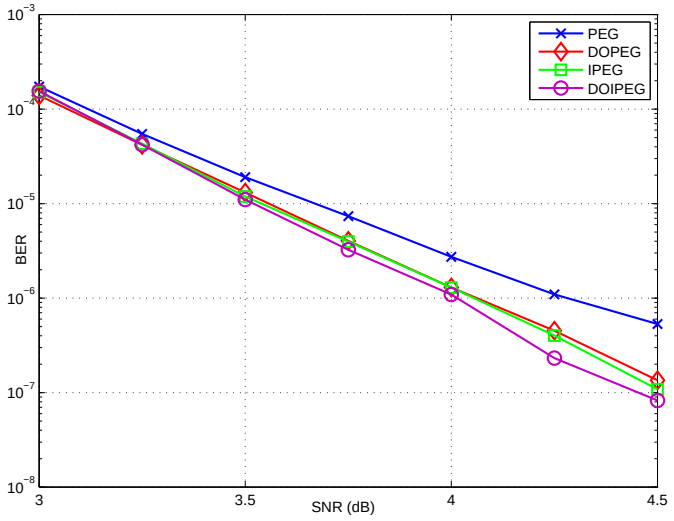

Fig. 2. BER curves for the proposed code constructions at block length 250

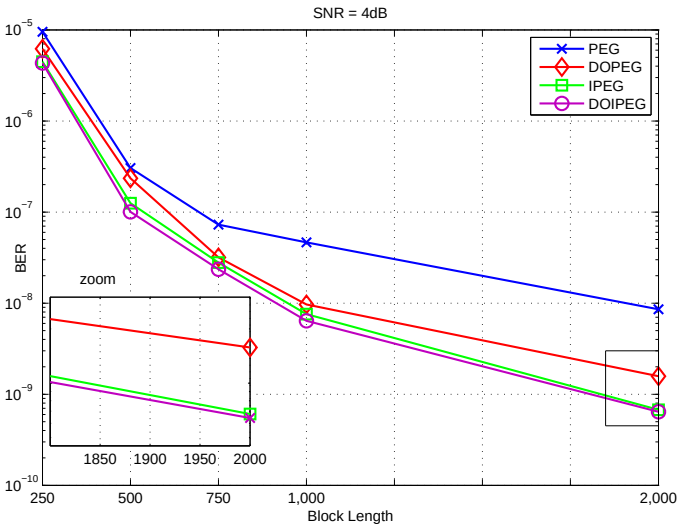

Fig. 3. Performance gains achieved at a number of different block lengths

V. Conclusions

The proposed DOPEG and DOIPEG methods are flexible in length and rate, and are capable of generating irregular LDPC codes with improved performance in the error floor region. As has been stated previously, the gains achieved are intuitively consistent, since the candidate CN with best subgraph connectivity will exhibit the best intermediate code performance and better subgraph connectivity will lead to improved performance for the final code.

TABLE III
Code generation times, in seconds, for the algorithms presented.

N	PEG	IPEG	DOPEG	DOIPEG
250	27.6	33.4	4.9×10^3	482
500	182.3	199.9	4.1×10^4	2.5×10^3
1000	1.3×10^3	1.5×10^3	2.3×10^5	1.1×10^4

The novel code constructions are proposed for codes of short to medium block length. As was discussed in Section IV, the computational cost of the algorithms for larger lengths becomes too high, while the performance improvements over the base codes, PEG and IPEG, will reduce in line with the concentration theorem [12], as indeed will the gains of these codes over random code constructions. The codes of Fig. 3 are indicative of the lengths recommended for code construction by the proposed method. For these lengths the graph structure, short cycles and their connections, has a significant impact on code performance.

References

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. 20, pp. 21–28, Jan. 1962.

[2] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," *Proc. 29th Annual ACM Symp. Theory of Computing*, pp. 150–159, 1997.

[3] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[4] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 50, no. 8, pp. 1242–1247, Aug. 2004.

[5] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 50, no. 1, pp. 21–28, Jan. 2005.

[6] A. Uchoa, C. Healy, R. de Lamare, and R. Souza, "Design of LDPC codes based on progressive edge growth techniques for block fading channels," *Communications Letters, IEEE*, vol. 15, no. 11, pp. 1221 –1223, november 2011.

[7] H. Xiao and A. H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Commun. Lett.*, vol. 8, no. 12, pp. 715–717, Dec. 2004.

[8] D. Vukobratovic, A. Djurendic, and V. Senk, "ACE spectrum of LDPC codes and generalized ACE design," *IEEE Conference on Comms., ICC*, pp. 665–670, Jun. 2007.

[9] D. Vukobratovic and V. Senk, "Generalized ACE constrained progressive edge-growth LDPC code design," *IEEE Commun. Lett.*, vol. 12, no. 1, pp. 32–34, Jan. 2008.

[10] M. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.

[11] A. K. H. Jin and R. McEliece, "Irregular repeat-accumulate codes," *Proc.2nd Int Symp. on Turbo Codes and Related Topics*, pp. 1–8, Sep. 2000.

[12] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.