QUASI-CYCLIC LOW-DENSITY PARITY-CHECK CODES BASED ON DECODER OPTIMISED PROGRESSIVE EDGE GROWTH FOR SHORT BLOCKS

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

Department of Electronics, University of York, York, UK

ABSTRACT

A novel construction for quasi-cyclic (QC) regular and irregular low-density parity-check (LDPC) codes based on a modification of the QC Progressive Edge Growth (PEG) algorithm is presented. Edge placement of the PEG-based 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, in particular when compared with algebraic constructions. The codes constructed by the proposed methods are tested in the AWGN channel and performance improvements are achieved. The proposed QC-LDPC codes provide an inherent trade-off between code performance and encoding/decoding complexity.

***Index Terms*—** Channel coding, Low-density parity-check codes, Iterative decoding

1. INTRODUCTION

Low-density parity-check (LDPC) codes are a class of capacity approaching codes first introduced by Gallager [1]. Key developments leading to practical implementations of LDPC codes include the bipartite graph known as the Tanner graph [2] and the fact that decoding with complexity linear in code length is achievable by means of the sum-product algorithm (SPA) [3]. Irregular LDPC codes, which have a parity-check matrix with varying row and column weights were introduced by Luby et al. [4] and found to perform excellently. Richardson et al. [5] then developed density evolution (DE), an analytical tool for optimising the degree distribution which defines the row and column weights of the irregular parity-check matrix.

A notable limitation of DE for codes of more practical short and medium lengths is the assumption that the decoding neighbourhood of a given variable node (VN) is tree-like [5]. While this assumption is true for codes of infinite length and approximately holds for large block length codes, at short to medium lengths it is not verified. At these lengths short cycles found in the graph of the code have significant negative effect on code performance as a result of their impact on the independence of messages passed in SPA decoding. A number of approaches exist which attempt to lessen the effects of these short cycles. The concept of approximate cycle extrinsic message degree (ACE) [6] emphasises the importance of connectivity between short cycles in how they effect code performance. A particularly useful tool in constructing codes with excellent performance is the progressive edge growth (PEG) algorithm [7]. This is a greedy edge placement construction algorithm which maximises the length of the cycle created at each edge placement. The PEG algorithm is particularly suited to the design of LDPC codes with short block lengths. It is flexible in length and rate and may be applied, through modifications, to construct codes with particular beneficial structure such as irregular repeat accumulate (IRA) codes [8] which allow efficient encoding and the PEG-Root-Check LDPC codes for the block fading channel [9], towards which the authors contributed.

The ability to decode LDPC codes with complexity linear in code length is due to the sparsity of the parity-check matrix. In general the generator matrix of an LDPC code is not sparse so encoding is more costly due to the required matrix multiplication. Quasi-cyclic (QC) LDPC codes allow low-complexity encoding as well as decoding [10]. The generator and parity-check matrices have an imposed structure which may be exploited such that encoding may be performed with shift registers [11] and decoding with SPA decoder may be further parallelised.

In [12] the PEG algorithm is used to construct QC-LDPC codes with improved performance over random QC constructions. The proposed algorithm, termed QC-DO-PEG, relies on the concept of using the decoder to improve the design of the code. This concept was developed previously in [13] for unstructured LDPC codes. This leads leads to overall improved graph connectivity and to an improvement in BER performance.

The rest of this paper is structured as follows. In Section 2 we formulate the problem and provide the necessary background on QC-LDPC codes and the PEG algorithm. In Section 3 the proposed code design algorithm is presented and in Section 4 the simulation results are given. Section 5 provides concluding remarks.

2. PROBLEM STATEMENT

The goal was to design cost effective LDPC codes for short block lengths. QC-LDPC codes allow linear encoding complexity by use of shift registers but suffer particularly at short block lengths from short cycles. The QC-PEG algorithm uses the tree expansion of the original PEG algorithm to choose the placement positions of the QC sub-matrices in the parity-check matrix of the code in order to increase the girth of the code. The proposed QC-DO-PEG algorithm further improves performance by using the SPA decoder in code construction to optimise edge placement. Essential concepts and notation in QC-LDPC codes and the PEG algorithm are outlined in the following.

QC-LDPC parity-check matrices (PCMs) are structured as

$$\mathbf{H}_{QC} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,t} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,t} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{c,1} & \mathbf{A}_{c,2} & \cdots & \mathbf{A}_{c,t} \end{bmatrix}, \qquad (1)$$

each $\mathbf{A}_{a,b}$ is a $Q \times Q$ matrix, either a circulant permutation matrix or a null matrix. In [12] the position of and the first entry in each $\mathbf{A}_{a,b}$ is determined by a modified PEG algorithm which operates on every *Q-th* column. The remaining entries in $\mathbf{A}_{a,b}$ are found by cyclic shift. In the proposed algorithm the sub-matrix positions and entry positions within the sub-matrix are determined by the PEG based tree expansion in combination with our proposed decoder optimisation.

In the PEG algorithm and its modification for constructing QC-LDPC codes, the neighbourhood to depth *l* of a VN v_j is the set of all CNs in the tree expanded from VN v_j with *l* levels and is denoted $N_{v_j}^l$. The set of all CNs which are not in the neighbourhood of v_j at depth *l* is then denoted $\overline{N_{v_j}^l}$. The QC-PEG algorithm expands the tree from v_j to depth l such that either the cardinality of $N_{v_j}^l$ stops increasing or $\overline{N_{v_j}^l} \neq 0$ but $\overline{N_{v_j}^{l+1}} = 0$. In the second case a cycle will be created, and this cycle will have the greatest length possible under the current graph setting.

D_v is defined as the degree sequence of weights of the columns of sub-matrices. $\overline{N_{v_j}^{mw,l}}$ is the set of CNs in $\overline{N_{v_j}^l}$ which have minimum weight under the current graph setting.

3. PROPOSED QC-DO-PEG CODE DESIGN

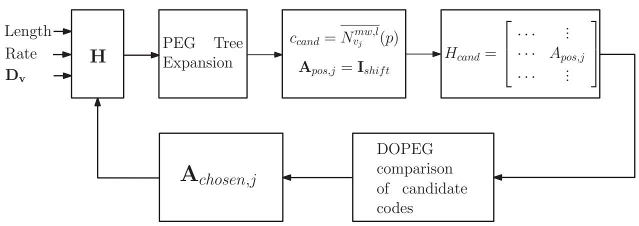

Fig. 1. Block Diagram of the QC-DO-PEG Construction

With high regularity the QC-PEG algorithm provides a set of candidate check nodes $\overline{N_{v_j}^{mw,l}}$ with cardinality greater than 1. In this case a CN is chosen at random from this set of minimum weight CNs and the rest of the edges in the sub-matrix are placed by a downward cyclic shift of the first column defined by this choice. In the proposed QC-DO-PEG algorithm the decoder optimisation is used to make a choice when there is more than one CN in the set $\overline{N_{v_j}^{mw,l}}$, each corresponding to a distinct circulant permutation matrix in a particular position with a particular cyclic shift in the first column. An intermediate code is constructed for each candidate circulant permutation matrix and the performance of these codes is compared by encoding and operating the SPA decoder in the presence of AWGN for a given number of message vectors and over a given range of signal-to-noise ratios (SNRs). The intermediate code with best graph connectivity will provide the best performance. The candidate sub-matrix corresponding to this code is then chosen for placement and the algorithm progresses to the next set of edges for placement. By optimising the connectivity of the cycle created at each placement, the code which is constructed by this method has better overall graph connectivity.

From the set $\overline{N_{v_j}^{mw,l}}$ the goal is to identify the CN which provides the best performance. For each $c_g \in \overline{N_{v_j}^{mw,l}}, g = 1 : Length(\overline{N_{v_j}^{mw,l}})$, the candidate code is described by $\mathbf{H}_{cand.}$:

$$\mathbf{H}_{cand.} = \begin{bmatrix} \mathbf{B}_{curr.,1} & \mathbf{B}_{curr.,2} & \cdots & \mathbf{B}_{curr.,j-1} & \mathbf{B}_{cand.} \end{bmatrix} \tag{2}$$

The $Q \cdot c \times Q$ matrices $\mathbf{B}_{curr.,b}, b = 1 : j-1$ are the columns of sub-matrices of the code under the current graph setting, exluding the column of interest.

$$\mathbf{B}_{cand.} = \begin{bmatrix} \mathbf{A}_{1,j} \\ \vdots \\ \mathbf{A}_{circpos_g,j} \\ \vdots \\ \mathbf{A}_{c,j} \end{bmatrix} \tag{3}$$

The $\mathbf{A}_{a,j}$, $a = 1 : c \ / \ (\text{circpos}_g)$ are the sub-matrices of the column of interest under the current graph setting. $\mathbf{A}_{circpos_g,j}$ is the sub-matrix specified by c_g. It has a "1" entry in the position *shift*$_g$ in its first column, subsequent columns are downward cyclic shifts of this column. The indices $circpos_g$ and $shift_g$ are given by:

$$\text{circpos}_g = \lceil g/Q \rceil, \tag{4}$$

$$\text{shift}_g = ((g - (\text{circpos}_g - 1) \cdot Q - 1) \mod Q) + 1 \tag{5}$$

The performance of each candidate code is then tested by encoding randomly generated message vectors and decoding in the presence of AWGN over a range of values of SNR.

3.1. Calculation of Metric of Convergence

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

$$L(T_i) = L(s_i) + \sum_{j \in C_i} L(r_{ji}), \tag{6}$$

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 c_g, $g = 1,\ldots, X$, where *X* is the cardinality of the set $\overline{N_{s_j}^{mw,l}}$. We define the $Z \times X$ matrix:

$$\mathbf{D}(h, g) = \sum_{t=1}^{Y} \sum_{i=1}^{N} (w \cdot |L(T_i)|), \tag{7}$$

$$\text{w} = \begin{cases} 1 & \text{if } \text{sgn}(L(T_i)) = s_i \\ -1 & \text{otherwise} \end{cases} . \tag{8}$$

where *N* is the length of the candidate codeword, *Y* is the number of message vectors transmitted for each candidate code at each SNR point and h indicates the SNR point, $h = 1, \ldots, Z$ and Z is the total number of SNR points. The convergence metric CVM(g) for candidate CN c_g is then the overall average sum for each candidate at each SNR point.

3.2. Pseudocode for the QC-DO-PEG Algorithm

In Algorithm 1 the pseudocode for the proposed code construction algorithm is presented. The $\lceil \cdot \rceil$ operator indicates the smallest integer greater than the value it operates on. The Length(.) function is the cardinality of the set it operates on.

From Algorithm 1, it can be seen that the proposed construction differs from the QC-PEG algorithm in steps 21 and 22. Step 21 will be the major contributor to the increase in complexity. This step involves a matrix inversion to derive the candidate encoder, and for each test message transmitted a matrix multiplication and use of the SPA decoder. These steps are required for each candidate at each edge placement. As a reference, for the codes presented in Fig. 2 the MATLAB tic/toc functions returned construction times of approximately 3 seconds and 348 seconds for the QC-PEG and

QC-DO-PEG constructions, respectively. For this reason the construction method is viable for short to medium length codes only. While this may seem to be an excessive increase in complexity, it should be noted that code construction is in most cases carried out off-line and in transmission the codes cost no more in terms of complexity than a QC-PEG code or any QC-LDPC code with the same block length and sub-matrix size.

4. SIMULATION RESULTS

We consider irregular rate $R = \frac{1}{2}$ codes with maximum VN degree 8, from the variable node degree distribution

$$\lambda_1(x) = 0.2703x + 0.2973x^2 + 0.4324x^7, \tag{9}$$

which was derived from the DE optimal degree distribution presented in [5] Table II, subject to the constraint imposed by the QC structure of the codes to be generated and the additional constraint that the number of weight-2 variable nodes be less than the total number of check nodes. This last constraint ensures that no cycles exist which are composed entirely of weight-2 nodes, a particularly damaging scenario in terms of performance under SPA decoding.

BPSK modulation over the AWGN channel was considered. The log-domain SPA decoder was used in the receiver and at least 80 block errors were gathered per point in each BER plot. For comparison, a random irregular QC-LDPC construction and the QC-PEG construction of [12] were used. The random construction consists of randomly choosing both sub-matrix position and the position of the first entry within the sub-matrix. The code produced then had length 4 cycles removed, as is standard practice with randomly generated codes.

In Fig. 2 the BER performance of codes of length 256, rate $R = \frac{1}{2}$ and with sub-matrix size $Q = 8$ is compared. In Fig. 3 the BER performance of codes of length 512, $R = \frac{1}{2}$ and with sub-matrix size $Q = 16$ is compared. For both Fig. 2 and Fig. 3, as expected, the two PEG-based constructions outperform the random construction by a considerable margin. This is due to the large number of short cycles in the graph of the random QC code and the fact that there is no restriction on cycles formed of only weight-2 variable nodes. It is seen that the two PEG-based codes perform almost exactly as well in the low SNR region. The benefits of decoder optimisation in the QC-DO-PEG are seen in the error floor region of the plots. Since the cycles created by the QC-PEG and QC-DO-PEG are, on average, of equal length this gain is a result of the improved connectivity of cycles in the graph of the QC-DO-PEG code. This is consistent with the results presented in [6] concerning error floor performance.

For the QC-DO-PEG as for the QC-PEG code it was found that BER performance improved as the sub-matrix size decreased, with the limiting case of $Q = 1$ corresponding to the original PEG algorithm. This provides a trade-off between code performance and encoding/decoding complexity, with smaller Q allowing more freedom in the QC-PEG and QC-DO-PEG to produce better performing codes and larger Q restricting the gains of the PEG tree expansion while lowering encoding and decoding complexity.

5. CONCLUSIONS

In this paper a construction method for regular and irregular QC-LDPC codes was presented. The codes generated by this method outperform those of [12] which were shown to perform favourably when compared to both randomly and algebraically constructed LDPC codes. The QC-LDPC codes generated by this method are

Algorithm 1 QC-DO-PEG

- **for** $j = 1 : t$ **do**
  - **for** $k = 1 : D_v(j)$ **do**
    - **if** $k == 0 \ \ \& \ \ j > \frac{N}{2}$ **then**
      - Choose candidate c_{ind} at random from the set $\overline{N_{v_j}^{mw,l}}$.
      - **for** $m = 0 : Q - 1$ **do**
        - circpos $= \lceil ind/Q \rceil$
        - shift $= ((ind - (circpos - 1) \cdot Q + m - 1) \bmod Q) + 1$
        - Place edge in the position $(c_{((circpos-1) \cdot Q)+shift}, v_{j \cdot Q+m})$
      - **end for**
    - **else**
      - Expand the tree from the VN v_j to depth l s.t. $N_{v_j}^l$ stops expanding **or** $\overline{N_{v_j}^l} \neq 0$ but $\overline{N_{v_j}^l} = 0$.
      - **if** $j < c + 1$ **then**
        - Choose candidate c_{ind} at random from the set $\overline{N_{v_j}^{mw,l}}$.
        - **for** $m = 0 : Q - 1$ **do**
          - circpos $= \lceil ind/Q \rceil$
          - shift $= ((ind - (circpos - 1) \cdot Q + m - 1) \bmod Q) + 1$
          - Place edge in the position $(c_{((circpos-1) \cdot Q)+shift}, v_{j \cdot Q+m})$
        - **end for**
      - **else**
        - **for** $p = 1 :$ Length$(\overline{N_{v_j}^{mw,l}})$ **do**
          - Form the PCM $\mathbf{H}_{cand.}$ as $\mathbf{H}$ under the current graph setting with a circulant permutation matrix in the position defined by its first entry $(c_{cand.}, v_{j \cdot Q+1})$ where $c_{cand.} = \overline{N_{v_j}^{mw,l}}(p)$. Use $\mathbf{H}_{cand.}$ to encode, decode in the presence of AWGN using log-domain SPA decoder with soft output. Compute convergence metrics CVM as described in Section 3.1.
          - Identify CN $c_{ind} = \arg\ \max_g(\mathrm{CVM}(g))$.
          - **for** $m = 0 : Q - 1$ **do**
            - circpos $= \lceil ind/Q \rceil$
            - shift $= (1 + (ind - (circpos - 1) \cdot Q + m - 1) \bmod Q) + 1$
            - Place edge in the position $(c_{((circpos-1) \cdot Q)+shift}, v_{j \cdot Q+m})$
          - **end for**
        - **end for**
      - **end if**
    - **end if**
  - **end for**
- **end for**

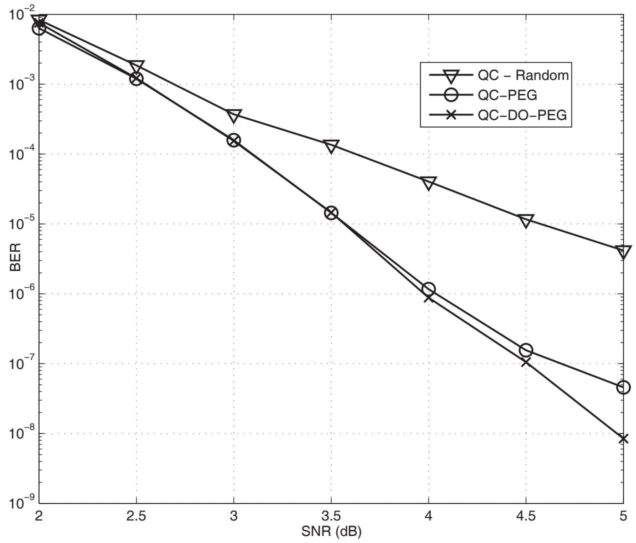

Fig. 2. Performance of codes of length 256, rate 1/2, $Q = 8$

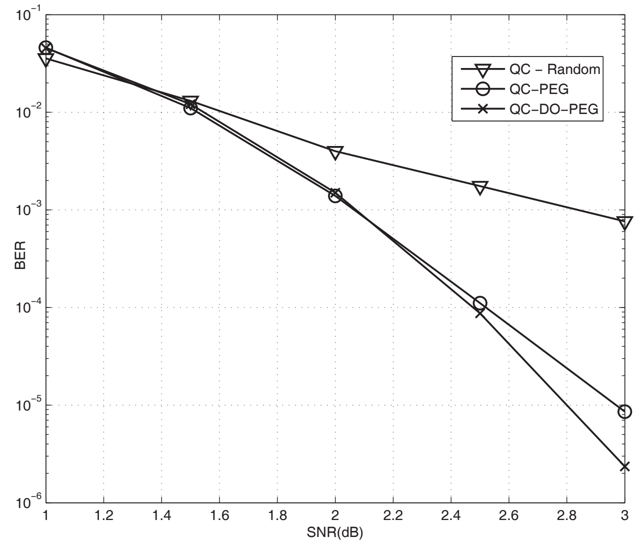

Fig. 3. Performance of codes of length 512, rate 1/2, $Q = 16$

capable of more efficient encoding than unstructured LDPC codes while remaining more flexible in length and rate than those codes produced by algebraic constructions. The construction method presented is more computationally costly than that presented in [12] but this cost is seen to provide definite improvement in BER performance. For this improvement there is no extra cost in the implementation of these codes over the previous QC-PEG codes.

6. REFERENCES

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

[2] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, pp. 533–547, Sept. 1981.

[3] D.J. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Junfeng Hu and Hailin Zhang, "Construction of short block length ira codes based on modified ipeg algorithm," in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, oct. 2008, pp. 1 –4.

[9] A.G.D. Uchoa, C. Healy, R.C. de Lamare, and R.D. 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.

[10] M.P. 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] Zongwang Li, Lei Chen, Lingqi Zeng, S. Lin, and W.H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *Communications, IEEE Transactions on*, vol. 54, no. 1, pp. 71 – 81, jan. 2006.

[12] Zongwang Li and B.V.K.V. Kumar, "A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph," *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2, pp. 1990– 1994, nov. 2004.

[13] C.T. Healy and R.C. de Lamare, "Decoder-optimised progressive edge growth algorithms for the design of LDPC codes with low error floors," *Communications Letters, IEEE, to appear*.