Construction of Short-Length LDPC Codes with Low Error Floor

X. Zheng[1], F. C. M. Lau[1], C. K. Tse[1] and Y. He[2]
1. Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong
2. College of Information Engineering, Shenzhen University, Shenzhen
Email: {xia.zheng, encmlau, encktse}@polyu.edu.hk; yjhe@szu.edu.cn

***Abstract*— It has been known that stopping sets and trapping sets are the main contributors to error floors exhibited by short-length low-density parity-check (LDPC) codes. In this work, a new metric called "approximate cycle set extrinsic message degree (ACSE)" is defined to help removing stopping sets with small size and bad trapping sets. Based on the new metric, we propose a code-construction algorithm that produces LDPC codes with very low error floors. Finally, we compare the error rates between codes produced by the proposed algorithm and MacKay code.**

I. Introduction

Low-density parity-check (LDPC) codes have been known to be very good error-correcting codes. Moreover, with a long code length and a large number of decoding iterations, LDPC codes can achieve performance as close as 0.04 dB from the Shannon limit [1]. In all practical scenarios, the code length must be finite and the decoder can only perform a certain number of iterations. Thus, in the associated bipartite graphs for finite-length LDPC codes, achieving cycle-free becomes impossible. Yet, to ensure that the decoder iterations are as independent as possible, short-cycles should be avoided [2].

In [3], Hu *et al*. have proposed a very effective algorithm to construct good short-length LDPC codes with large cycle (girth). In the method, namely progressive edge-growth (PEG) algorithm, connections between the variable nodes and check nodes are established in an edge-by-edge manner. Compared with random graph codes and other known good codes, codes constructed with the PEG algorithm are shown to produce lower error rates in the waterfall signal-to-noise ratio (SNR) regions.

At high SNR regions, the code performance is mainly determined by the low-weight codewords and the near codewords [4]. Moreover, it has been found that small-size stopping sets may contribute to low-weight codewords and near codewords, resulting errors at high SNR regions. In [5], it has further been discovered that short cycles with few connections to all other nodes are likely contributors to small stopping sets and are most harmful. With the use of a parameter called "approximate cycle extrinsic message degree (ACE)" that measures the connectivity of a cycle to all other nodes, Tian *et al*. have been able to remove many small stopping sets during the construction of LDPC codes [5]. But the algorithm mainly considers individual cycles in which nodes are of low degrees. In [6], another algorithm to detect stopping sets has been proposed. Unfortunately, the method cannot detect some typical small stopping sets. Since the above methods aim at removing small stopping sets, they have not considered the distributions of cycle lengths, which have been shown to affect the performance of the codes in the waterfall region.

To design codes with good performance at both the waterfall region and the high SNR region, code construction methods based on combining PEG and stopping-set checks have been proposed [7], [8]. Another way to find codes performing well at both the waterfall region and the high SNR region is to conduct an extensive search among a large pool PEG-constructed LDPC codes. In particular, the MacKay codes, which are the best codes known today, have been found based on the above searching method [9]. As would be expected, lots of resources and simulation times need to be spent in search of each good code.

In this paper, we propose a new technique that enables the removal of small-size stopping sets during the construction of LDPC codes. The method is based on the measure of the extrinsic message degrees of a cycle set, which contains one or more cycles connected by a common node. The proposed algorithm, when used together with PEG, has been able to construct short codes with very good error performance. Finally, we compare error performance of our codes with other well known codes.

II. Factors Limiting Error Performance at High SNR Regions

Before we proceed, we review some definitions and elaborate how stopping sets and trapping sets contribute to errors at the high SNR regions.

A. Stopping Sets

Definition 1: (Cycle) *A cycle in a bipartite graph is a path, consisting of edges connecting the two sets of nodes, that originates from and terminates at the same node*. Moreover, each edge can only be used once in the cycle. Also, the length of the cycle is given by the number of edges making up the cycle.

Definition 2: (Stopping set) *A stopping set S of an LDPC code is a subset of variable nodes with the condition that all neighbors of S are connected to S with 2 or more connections*.

This work was supported in part by a research grant (G-YG34) provided by Hong Kong Polytechnic University.

978-1-4244-2342-2/08/$25.00 ©2008 IEEE.

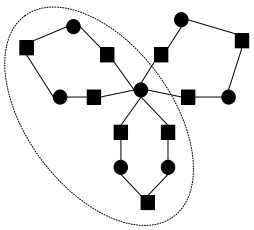
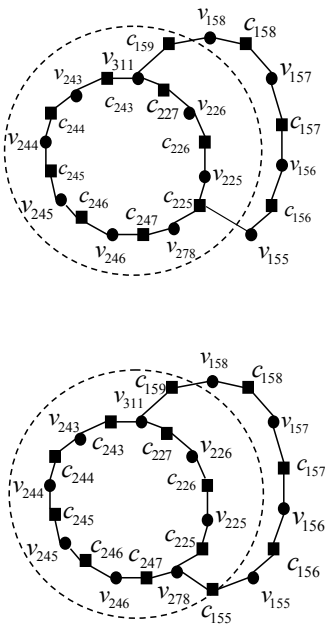


Fig. 1. A stopping set of size 7 is formed by combining 3 cycles with ACE values 4. A $[5, 2]$ trapping set is formed by the nodes in the dotted oval. Filled circles and filled squares represent, respectively, variables nodes and check nodes.



(b)

Fig. 2. (a) A stopping set of size 12 is formed by combining 2 cycles with ACE values equaling 1. A $[8, 1]$ trapping set is formed by the nodes in the dashed circle. (b) A stopping set of size 12 is formed by combining 2 cycles with ACE values equaling 2. A $[8, 2]$ trapping set is formed by the nodes in the dashed circle. Filled circles and filled squares represent, respectively, variables nodes and check nodes. The subscripts denote the variable- or check-node numbers.

Consider the neighboring check nodes of a stopping set S. Suppose each of these neighbors has an even number of connections with S. Then a valid codeword is formed when the variable nodes in the stopping set are set to one while all other variable nodes are zeros. As a consequence, the weight of such a codeword, i.e., number of ones, equals the size of the stopping set. Thus, if the minimum size among all possible stopping sets is small, it is likely that the minimum weight of the codeword will also be small. Since a lower minimum weight codeword and a larger number of such codewords will both give rise to a higher error floor at the high SNR region [4], the minimum stopping-set size of the code should be made as large as possible in order to lower the error floor. In reality, it is not practical to consider all possible stopping sets when designing a code. Suppose there is a codeword of length N. The number of stopping sets with size s is about C_s^N. Taking $N = 1000$ and $s = 5$ will produce almost 10^{15} different stopping sets. If the code length is longer, say 10^4, the number of stopping sets will be extremely large and it would be an impossible task to consider all of them. Instead of detecting all the stopping sets, Tian *et al*. [5] have introduced the concept of ACE and has proposed an ACE-based algorithm to eliminate small stopping sets indirectly.

Definition 3: (Extrinsic message degree (EMD)) *An extrinsic check node of a variable-node set is a check node that is singly connected to this set, and the EMD of a variable-node set is the number of extrinsic check nodes of the variable-node set [5].*

Definition 4: (Approximate cycle EMD (ACE)) *The ACE of a cycle with length* $2l$ *equals* $\sum_{i=1}^{i=l}(d_i - 2)$*, where* d_i *denotes the degree of the* i*-th variable node in the cycle. The ACE of a degree-*d_i *variable node is taken as* $d_i - 2$ *while that of any check node is equal to* 0 *[5].* Short cycles with low ACE values are found to be potential contributors to small stopping sets [5]. Compared with searching for and removing all the small stopping sets, computing the ACE values of short cycles passing through the variable nodes is much simpler and faster. Thus, in an attempt to remove small stopping sets indirectly, an algorithm to eliminate short cycles with low ACE values has been proposed [5].

The algorithm attempts to construct codes such that the ACE values of all cycles are larger than some given threshold, denoted by ζ. When $\zeta \geq 3$, the algorithm will have considered the variable nodes with low degrees (2 or 3) very well. Moreover, even when two cycles consisting of such low-degree nodes are combined through shared variables nodes or check nodes, the chance that the combined cycles forming a small stopping set is low. But the algorithm has not considered cycles involving high-degree variable nodes thoroughly. As shown in Fig. 1, although the ACE value of each of the three cycles is 4, a small stopping set of size 7 is formed when the three cycles are considered as a whole. Moreover, all the check nodes are connected to the variable nodes twice, implying that the stopping set forms a codeword of weight 7. As a consequence, if Fig. 1 forms a subgraph of a code graph, it can be concluded that the minimum weight codeword is no more than 7. In addition, for some given degree distributions, the average variable-node degree may be too low to generate any code that satisfies $\zeta \geq 3$. Under such a circumstance, the occurrences of combined cycles involving low-degree variable nodes to form small stopping sets become frequent. Figure 2 illustrates the scenarios when two common nodes combine two cycles into small stopping sets of size 12.

B. Trapping Sets

In addition to small stopping sets, trapping sets contribute to the error floor.

Definition 5: (Trapping set) *A* $[w, u]$ *trapping set of a code is a vector with hamming weight* w *and syndrome weight* u *[4].* Trapping sets with both small w and relatively smaller u

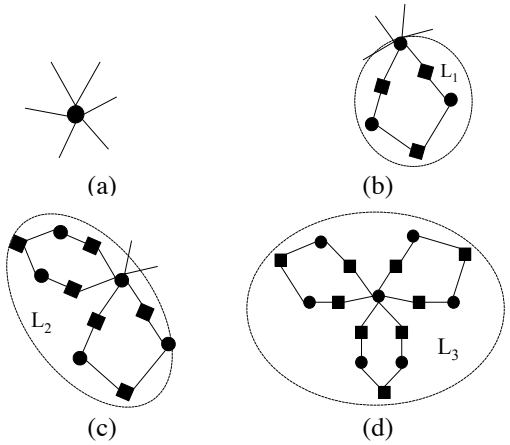


Fig. 3. Cycle sets and ACSE values. (a) A variable node with degree 6. (b) Cycle set L_1 contains one cycle. ACSE of L_1 equals 4. (c) Cycle set L_2 contains two cycles. ACSE of L_2 equals 2. (d) Cycle set L_3 contains three cycles. ACSE of L_3 equals 0. Filled circles and filled squares represent, respectively, variables nodes and check nodes.

are of high potential to give rise to errors. First, a small value of w makes the error pattern likely to occur. Second, an even smaller u indicates that the u check sums that will be affected, should errors occur in the w bits, may not provide sufficient information for the decoder to correct the error pattern. In Fig. 1, a $[5, 2]$ trapping set is formed by the nodes in the dashed circle. Also, $[8, 1]$ and $[8, 2]$ trapping sets, are found in Figs. 2(a) and (b), respectively.

III. Proposed Code Construction Algorithm

A. Approximate cycle set EMD (ACSE)

Definition 6: (Cycle set) *A cycle set consists of one or more cycles which are linked together by a common (root) node.*

Definition 7: (Approximate cycle set EMD (ACSE)) *The ACSE of a cycle set is defined as the number of extrinsic-message edges stretched out from the cycle set to the check nodes singly connected to the set. It is used to measure the connectivity of a cycle set to all other nodes.* Figure 3 illustrates the concept of a cycle set and its ACSE value. A variable node with degree 6 is the common node linking up the cycles. It can be readily shown that the ACES values of the cycle sets L_1, L_2 and L_3 are, respectively, 4, 2 and 0.

Consider a variable node, denoted by v, and expand its connections to a depth of d to form a subgraph. If all possible cycle sets, under the condition that the variable node v is the common node, have to be found and their ACSE values evaluated, it will be a complex and time-consuming process. Instead, we estimate the minimum ACSE of cycle sets with different sizes as follows. Note that all cycles refer to those beginning and terminating at v.

1) Set $k = 1$. Search among all cycles from the subgraph and select the one with the smallest ACE value. Denote the selected cycle by l_k. Set the cycle set $L_k = \{l_k\}$. Then the ACSE value of L_k equals the ACE value of l_k.
2) Increment k by 1. Search among all remaining cycles from the subgraph and add the one, denoted by l_k, to L_{k-1} such that the ACSE of the new cycle set $L_k = \{l_1, l_2, \ldots, l_k\}$ is the smallest.
3) Repeat Step 2 until the cycles in the cycle set L_k have included all edges emanated from v.

B. PEG-ACSE Code Construction Algorithm

First, the variable nodes are assigned with degrees according to a given degree distribution. Then the variable nodes are sorted according to their degrees in a non-decreasing manner. That is to say, denoting d_i as the degree for the ith variable node, then $d_i \leq d_j$ for $i < j$. To construct a code graph with large stopping sets, we need to make sure that low degree nodes are involved in cycles which are as large as possible. It is particularly crucial for degree-2 nodes because they contribute no extrinsic connections to cycles.

Denote the block length, check length and the number of degree-2 nodes by, respectively, N, M and P. Also, define $N^{(d)}(v_i)$ as the check-node set reached by a subgraph spreading from a variable node v_i at the depth d. Then we construct $\min\{M, P\}$ degree-2 nodes in a zigzag manner to guarantee that the cycles formed by the degree-2 nodes are as large as possible. For the remaining nodes, connections are made based on the following algorithm.

- Set the ACSE threshold value ζ with a positive integer.
- For $i = (\min\{M, P\} + 1)$ to N
  - Fail=0;
  - While Fail==0
    - For $m = 1$ to d_i
      - Connect the edges of the variable node v_i to the check nodes from the set $N^{(d)}(v_i)$ using the PEG algorithm [3].
    - End ($m = 1$ to d_i)
    - With v_i as the root node, estimate the minimum ACSE of cycle sets with different sizes using the algorithm in Sect. III-A.
    - If any of the minimum ACSE values is smaller than the threshold ζ
      - Fail=0;
    - Else
      - Fail=1;
    - End (if...else)
  - End (while)
- End ($i = (\min\{M, P\} + 1)$ to N)

Note that the threshold ζ should be carefully chosen. After performing an extensive study, we find that stopping sets of small size and trapping sets $[w, u]$ with small w and u cannot be eliminated effectively when $\zeta \leq 3$. But for $\zeta \geq 6$, it is very difficult to construct code matrix that can satisfy the threshold requirement.

IV. Results and Discussions

Two degree distribution sets (each containing a variable-node degree distribution and a check-node degree distribution) that have been optimized based on the density evolution

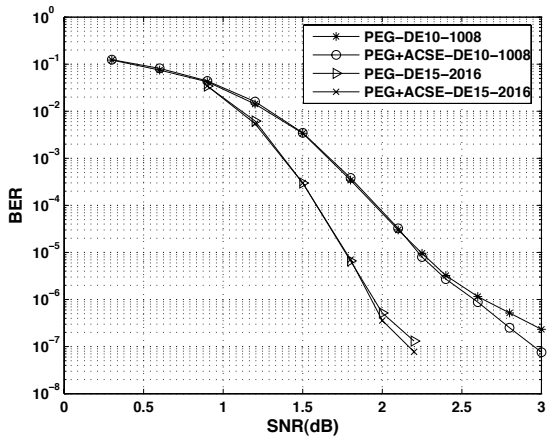


Fig. 4. The BER performance of LDPC codes with length 1008 and 2016. Coding rate equals $1/2$. Two sets of degree distributions are used — $\lambda_1(x)$ (denoted by DE15) and $\lambda_2(x)$ (denoted by DE10). Codes are constructed using PEG-only and the proposed PEG-ACSE algorithms.

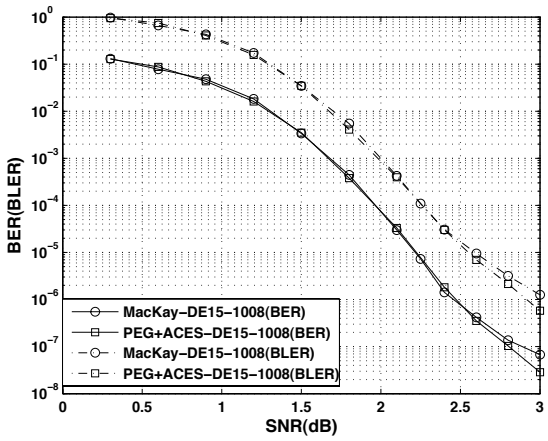


Fig. 5. The BER and BLER performance of PEG-ACSE-constructed LDPC code and MacKay code. Code length is 1008 and code rate equals $1/2$. MacKay code, constructed by PEG-only algorithm, is the best known code with parameters (1008, 504) over an AWGN channel.

algorithm are used in our studied [10]. The corresponding variable-node degree distributions are given by

$$\begin{aligned}\lambda_1(x) &= 0.23802x + 0.20907x^2 + 0.03492x^3 \\ &\quad + 0.12015x^4 + 0.01587x^6 + 0.00480x^{13} \\ &\quad + 0.37627x^{14} \end{aligned} \quad (1)$$

and

$$\begin{aligned}\lambda_2(x) &= 0.27165x + 0.25105x^2 + 0.30938x^3 \\ &\quad + 0.00104x^4 + 0.43853x^9, \end{aligned} \quad (2)$$

and they have maximum variable-node degrees 15 and 10, respectively. Codes of length 1008 and 2016 with coding rate $1/2$ are then constructed based on PEG-only and PEG-ACSE algorithms. A depth of $d = 11$ is used for both construction methods. Moreover, in the PEG-ACSE algorithm, ACSE thresholds of $\zeta = 4$ and $\zeta = 5$ are used for codes with lengths 1008 and 2016, respectively. In addition, we assume an AWGN channel and a belief propagation decoder which will iterate a maximum of 50 times for each received codeword.

Fig. 4 plots the bit error rate (BER) performance of the codes with length 1008 and 2016. It can be observed that the PEG-ACSE-constructed codes have similar BER error performance as PEG-constructed codes at waterfall regions (SNR less than 2 dB). Moreover, PEG-ACSE-constructed codes outperform PEG-constructed codes at high SNR regions (SNR greater than 2 dB). Finally, we compare the error rates of PEG-ACSE-constructed codes with MacKay codes [9], which have been known to be the best performing codes today. In Fig. 5, we plot the BER and block error rate (BLER) curves for codes with length 1008 and rate $1/2$. When SNR is less than 2.5 dB, both the PEG-ACSE-constructed code and the MacKay code produce similar error rates. At higher SNRs, the PEG-ACSE-constructed code outperforms the MacKay code. We can therefore conclude that the proposed PEG-ACSE code construction algorithm is an effective way of creating short LDPC codes with very good error performance.

V. Conclusion

In this paper, we have presented a new metric called approximate cycle set EMD (ACSE) that measures the connectivity of a cycle set to all other nodes. By setting a threshold ACSE value during the construction of codes, we can easily form codes with very good error performance. Simulation results have shown that PEG-ACSE-constructed codes can achieve lower error floor than PEG-constructed codes and the MacKay codes.

References

[1] S.-Y. Chung, G. D. Forney, T. J. Richardson and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters,* vol. 5, no. 2, pp. 58–60, Feb. 2001.

[2] Y. Mao and A. Banihashemi, "A Heuristic search for good low-density parity-check codes at short block lengths," *Proc., IEEE Intl. Conf. Comm.*, Helsinki, Finland, Jun. 2001, pp. 41–44.

[3] X. Y. Hu, E. Eleftheriou and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inform. Theory,* vol. 51, no. 1, pp. 386–398, Jan. 2005.

[4] T. Richardson, "Error floors of LDPC codes," *Proc., 41st Annu. Allerton Conf. on Communication, Control, and Computing,* Urbana-Champaign, USA, Oct. 2003, pp. 1426–1435.

[5] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC codes construction," *IEEE Trans. Communication*, vol. 52, pp. 1242–1247, Aug. 2004.

[6] A. Ramamoorthy and R. Wesel, "Construction of short block length irregular low-density parity-check codes", *Proc., IEEE Intl. Conf. Comm.*, Paris, France, Jun. 2004, vol. 1, pp. 410–414.

[7] H. Xiao and A. H. Banihashemi, "Improved Progressive-Edge-Growth (PEG) Construction of Irregular LDPC Codes," *IEEE Communications Letters,* vol. 8, no. 12, pp. 715–717, Dec. 2004.

[8] G. Richter and A. Hof, "On a Construction Method of Irregular LDPC Codes Without Small Stopping Sets," *Proc., IEEE Intl. Conf. Comm.*, Istanbul, Turkey, Jun. 2006, vol. 3, pp. 1119–1124.

[9] D. J. C, MacKay, *Encyclopedia of sparse graph codes*, available from: http://wol.ra.phy.cam.ac.uk/mackay/codes/data.html.

[10] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory,* vol. 47, pp. 599–618, Feb. 2001.