Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Construction of Irregular LDPC Code Using ACE Algorithm

Sushil Sonavane, Prof. D.P.Rathod, Sandip Sukhdeve, Ashish Patil
Electrical Dept.,
VJTI, Mumbai
India

Abstract-The Low Density Parity Check (LDPC) codes have been widely acclaimed in recent years for their near-capacity performance, they have not found their way into many important applications. For some cases, this is due to their increased decoding complexity relative to the classical coding techniques. For other cases, this is due to their inability to reach very low bit error rates at low signal-to-noise ratios (SNRs), a consequence of the error rate floor phenomenon associated with iterative LDPC decoders. The construction of finite-length irregular LDPC codes with low error floors is currently an attractive research problem A new metric, called extrinsic message degree (EMD), measures cycle connectivity in bipartite graphs of LDPC codes. Using an easily computed estimate of EMD, we propose a Viterbi-like algorithm that selectively avoids small cycle clusters that are isolated from the rest of the graph. , a simple but powerful heuristic design algorithm, the approximate cycle extrinsic message degree (ACE) constrained design algorithm, has recently been proposed. This algorithm is different from conventional girth conditioning by emphasizing the connectivity as well as the length of cycles.

Keywords—Error floor, extrinsic message degree (EMD), graph cycles, irregular low-density parity-check (LDPC) codes, stopping sets, unstructured graph construction.

I. INTRODUCTION

Low-density parity-check (LDPC) codes have generated much interest recently LDPC codes were first presented by Gallager [1] in 1962 and have created much interest recently when rediscovered and shown to perform remarkably close to the Shannon limit. Gallager described regular codes, defined by parity-check matrices with constant column and row weights, which were constructed pseudo-randomly to avoid 4-cycles [2]. Recently, Luby et al. extended Gallager's results to consider irregular codes, that is, codes with non-constant row and column weights in , and showed that these codes are capable of outperforming regular codes [3].The irregular codes are constructed via a pseudo-random process which usually involves discarding codes which contain 4-cycles. MacKay *et al.* [2] first reported the tradeoff between the threshold SNR and the error floor BER for irregular LDPC codes versus regular LDPC codes. A similar tradeoff has been found for turbo codes [5]. Richardson *et al.* A significant amount of work has been done recently in the area of finite-length behavior of the iterative decoding of LDPC codes over the BEC channel. Finite-length behavior of LDPC codes, decoded iteratively over the BEC , is determined by the subgraphs of an LDPC code graph called stopping sets [6]. Small stopping sets deteriorate the performance of LDPC codes in the error-floor region. The design method that optimizes irregular LDPC codes with respect to their minimum stopping set size (i.e, their stopping distance [7]) is yet unknown, which is one of the most challenging open problems in the LDPC code design [4]. However, the recently proposed ACE constrained design algorithm [8] heuristically produces LDPC codes with improved error-floor performance by influencing stopping sets via conditioning cycles. This paper builds upon the ACE constrained LDPC code design. We introduce the ACE spectrum as an efficient tool for quick evaluation of finite-length irregular LDPC codes. Randomly realized finite-length irregular LDPC codes with block sizes on the order of 10^4 [1] approach their density evolution *threshold* closely (within 0.8dB) at rate 1/2, outperforming their regular counterparts [3] by about 0.6dB. The common approach has been to indirectly improve *dmin* through code conditioning techniques such as the removal of short cycles (girth conditioning [9], [10]). However, not all short cycles are equally harmful. Standard girth conditioning severely constrains code structure by removing all cycles shorter than a specified length even though many of these can do little harm, as will be explained. Under belief iterative decoding, small stopping sets [11] in random codes lead to high error floors. However, these can be alleviated by generating codes from an expurgated ensemble that has large stopping sets. In this project we use a technique that addresses only cycles that are likely contributors to small stopping sets. In this paper, we show that such a technique lowers the error floors of irregular LDPC codes significantly while only slightly degrading the performance in the waterfall region.

II. The Cycle Ace Metric

The design of optimal irregular LDPC codes for the BEC reduces to the design of irregular LDPC code graphs with their stopping distance being maximized. However, finding the stopping distance of an LDPC code graph is shown to be an NP-hard problem [12]. In this section, we review the notion of stopping set and the idea of influencing its size by influencing the cycles in the code graph using the ACE metric.

A Tanner graph, or LDPC code graph G(H), is derived from an LDPC code parity-check matrix H in a usual way, where variable (or left) nodes correspond to columns, check (or right) nodes correspond to rows, and edges correspond to ones in the parity-check matrix. We start with the definition of a stopping set [4].

Definition 2.1: A set of variable nodes of an LDPC code graph G(H), such that every check node neighbor of this set is connected to this set at least twice, is called a stopping set. (In this paper, however, with the term stopping set we refer to the subgraph of the original LDPC code graph that is induced by the variable nodes in the stopping set.) It is pointed out in [6] that a stopping set (observed as a subgraph) contains one or several interconnected cycles. Since every stopping set possesses a cycle, it is possible to influence the size of the stopping sets by influencing cycles. The traditional approach to remove all short cycles is essentially good, since it is an easy way to remove some of the small stopping sets. The well-known progressive edge growth (PEG) algorithm [8] proceeds in this fashion. However, for short-length LDPC codes this approach is not effective enough since the obtained girths (minimum cycle lengths) are small. The cycles of the same length in the code graph differ with respect to the probability that they are part of a small stopping set. Based on this observation, all cycles of the same length can be classified if they can be evaluated using an appropriate metric. The metric proposed in [6], named the extrinsic message degree (EMD) of a cycle, measures the level of connectivity of a cycle with the rest of the graph via its variable nodes.

Definition 2.2: For a given cycle C in the LDPC code graph G(H), let VC be the set of variable nodes in C and C(VC) be the set of check node neighbors of VC. We can divide the set C(VC) into three disjoint subsets:

• Ccyc(VC): the subset of C(VC) that belongs to the cycle C. Each node from Ccyc(VC) is connected to the set VC exactly twice,
• Ccut(VC): the subset of C(VC) that are not in the cycle C, but are connected to the set VC at least twice,
• Cext(VC): the subset of C(VC) connected to the set VC once.

This definition leads to another one, related to edges:

Definition 2.3: For a given cycle C in the LDPC code graph G(H) and the corresponding set VC, let E(VC) be the set of all edges incident on VC. We can divide the set E(VC) into three disjoint subsets:

• Ecyc(VC): the subset of cycle edges in E(VC) incident on the check nodes in Ccyc(VC),

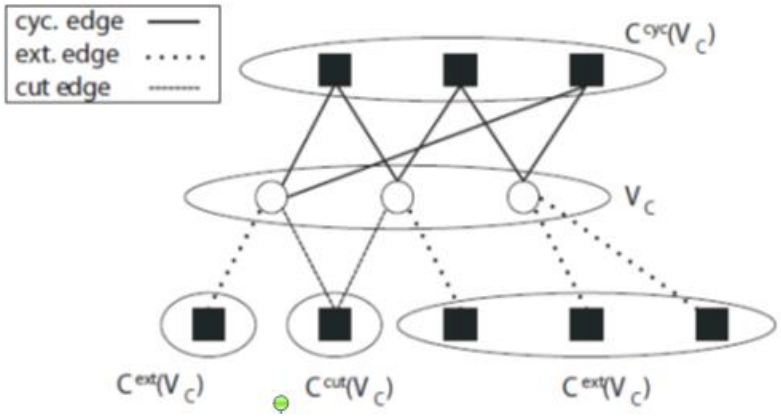

Fig.1. cycle, cut and extrinsic edges of cycle C.

• Ecut(VC): the subset of cut edges in E(VC) incident on the check nodes in Ccut(VC) (note that these edges cut the cycle C into subcycles),
• Eext(VC): the subset of extrinsic edges in E(VC) incident on the check nodes in Cext(VC).

(See Figure 1 for an example related with previous definitions.)

Definition 2.4: The extrinsic message degree of the cycle C in the LDPC code graph G(H), denoted EMD(C), is defined as:

$$EMD(C) = |Eext(VC)| \quad (1)$$

where |Eext(VC)| is the cardinality of Eext(VC).

If a cycle in the LDPC code graph has low EMD, its communication with the rest of the graph is limited. This limits the amount of new evidence about the values of variable nodes in VC that could be collected from the rest of the graph. In the extreme case, when the EMD of the cycle is zero, variable nodes in the cycle are isolated from the rest of the graph and VC is the stopping set1. It is not an easy task to find the EMD of the cycle in the code graph, since it requires additional steps to determine if the edge is an extrinsic edge or a cut edge. If this difference is neglected and both extrinsic and cut edges are taken into account in the cycle metric, the simplified version of the EMD metric, named the ACE metric [6], is obtained.

Definition 2.5: The approximated cycle EMD (ACE) of the cycle C in the LDPC code graph G(H), denoted ACE(C), is given as:

$$ACE(C) = |Eext(VC)| + |Ecut(VC)| \quad (2)$$

It is easy to calculate ACE(C) as:

$$ACE(C) = \sum_{v \in Vc} (d(v) - 2) \quad (3)$$

where the sum is taken over all variable nodes in the cycle C, and d(v) is the degree of the variable node v.

We refer to a cycle C of length l (the number of edges in the cycle) and ACE(C) value η as to an (l, η)-cycle. The ACE constrained design algorithm, introduced in [6], is developed to design (dmax, ηACE) ACE constrained LDPC codes where, by construction, each (l, η)-cycle satisfies that if $l \leq 2d_{max}$, then $\eta \geq \eta_{ACE}$. The simulation results presented in [6] confirm that (dmax, ηACE) ACE constrained LDPC codes reach considerably lower error floors than randomly constructed codes, for selected values of (dmax, ηACE). However, increasing either one or both of the parameters from the pair (dmax, ηACE) leads to computational problems in the ACE constrained design algorithm, since finding (dmax, ηACE)- compliant graphs becomes a hard task (or unsolvable task, for sufficiently large values of (dmax, ηACE) pair.)

III. SCHEME OF IMPLEMENTATION

The well-known matrix and bipartite graph descriptions of a rate 1/3 (9, 3) code are given in Fig. 2. One column in the parity check matrix corresponds to one variable in the bipartite graph. For convenience, we will use 'column' and 'variable' interchangeably in this paper. The parity check matrix *H* is constructed such that the *H*2 portion of it is invertible, which guarantees that *H* is full-rank. For systematic encoding, *H*1 corresponds to information bits and *H*2 corresponds to parity bits.

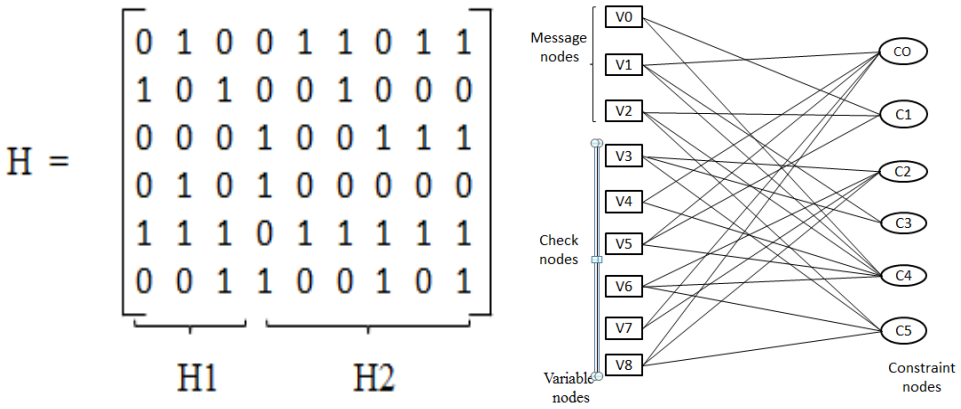

Fig. 2. Matrix and graph description of a (9,3) code

The Gaussian elimination process ultimately guarantees that the H matrix has full rankb y ensuring that the n−k columns of H2 will be linearly independent. For degree-2 variable nodes, independence entails freedom from cycles so that all degree-2 parity check nodes will be cycle-free. Now we introduce the ACE detection method, based on support trees, is directly related to the graph structure.

The tree depiction of ACE detection (η = 0) is given in Fig. 3. Here, variable and constraint node labels refer literally to those of the example code in Fig. 1 and the support tree that extends four levels below root node v0 is portrayed. We define pt to be the ACE of a path between root node v0 and an arbitrary node μt (it can be either a variable node or a constraint node).

Recall also that ACE(μt) = degree(μt) − 2 if μt is a variable, and
ACE(μt) = 0 if μt is a constraint.

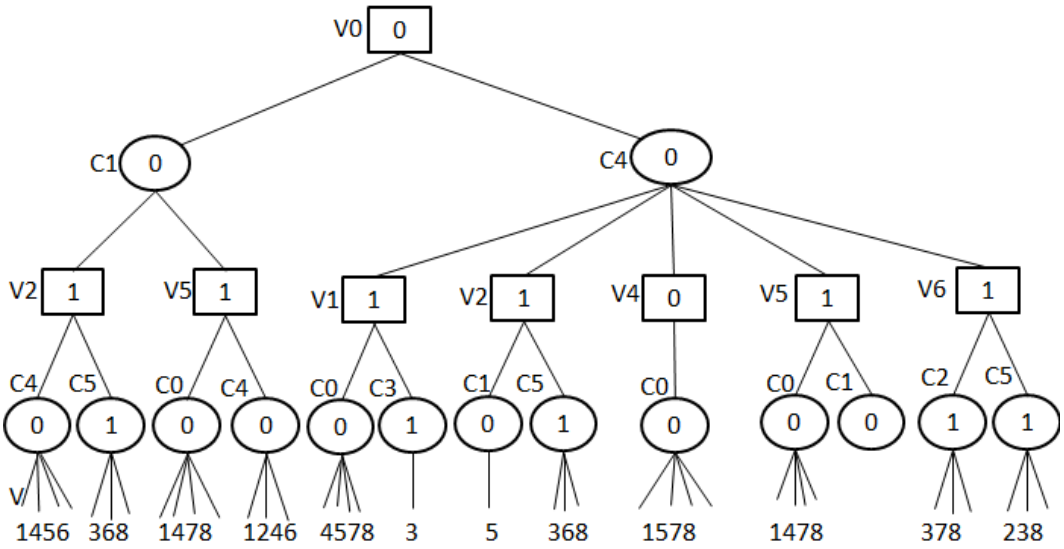

Fig. 3. Illustration of an ACE search tree associated with v0 in the example code of Fig.1. Bold lines represent survivor paths. ACE values are indicated on the interior of circles (variables) or squares (constraints), except on the lowest level where they are described with a table.

To explain the above algorithm, we need to recognize that a node should propagate descendants (be active) only if the path leading to this node has the lowest ACE value that any path to this node has had thus far. Therefore linear cost is

achieved instead of an exponential cost. Initially all the path ACEs can be set to (which means 'unvisited'). Note that cycles occur when a node is revisited, is simultaneously visited, or both. A cycle ACE equals the sum of the previously lowest path ACE to a node and the current path ACE to the node minus the doubly counted root and child ACE. Handling multiple simultaneous arrivals to the same node is a trivial extension where ACE minimization is performed sequentially across all arrivals. In the example shown in Fig. 5, bold lines at each level describe the current set of active paths. In this example 'ties' are assigned the path whose parent has the lowest index. For instance the path (v0-c1-v2-c5) with ACE =1 survives while, (v0-c4-v2-c5), (v0-c4-v1-c3), (v0-c4-v6-c2) each also having ACE = 1, perish. For an example of pruning occurring due to cycle detection on differing levels of the tree, observe that the path (v0-c1-v2-c4) with ACE = 1 does not survive since c5 was visited at Level-1 and was accordingly assigned ACE = 0.

Note that cycle detection is implicitly performed in the above algorithm. Unvisited variables are initialized to $P(\mu t)=\infty$ When a variable is first visited, $P(\mu t)$ is assigned a finite value. Upon subsequent visits to the same variable, ill-conditioned cycle detection is alarmed if the condition (Ptemp + $P(\mu t)$ - ACE (v) - ACE (μt)) is not satisfied.

IV. CONCLUSIONS

We have discussed the effect of graph connectivity on error floors, and have introduced the ACE algorithm to construct irregular LDPC codes that have a specific ACE property for short cycles, while maintaining the density-evolution-prescribed degree distribution. The class of LDPC codes with extremal ACE spectrum properties is introduced as the subset of the irregular LDPC code ensemble. The irregular LDPC codes from this class have a potential to offer both excellent waterfall and error-floor performance.

ACKNOWLEDGMENT

We would like to express my special thanks of gratitude to our guide Prof. D.P. Rathod sir who gave us the golden opportunity to write this important paper on the topic load balancing on cloud data centres, which also helped us in doing a lot of Research and we came to know about so many new things, we are really thankful to him.

REFERENCES

[1] T. Richardson, M. Shokrollahi, and R. Urbanke, "*Design of capacity-approaching irregular low-density parity-check codes*," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[2] R. G. Gallager, Low-Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963.

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann, 1988.

[4] C. Di, D. Proietti, E. Teletar, T. J. Richardson, and R. L. Urbanke, *"Finite length analysis of low-density parity-check codes on the Binary erasure channel*," *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1570-1579,June 2002.

[5] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "*Comparison of constructions of irregular Gallager codes,*" *IEEE Trans. Commun.*, vol. 47, pp. 1449–1454, Oct. 1999. 47, pp. 569–584, Feb. 2001..

[6] A. Vardy and M. Schwartz, "*On the stopping distance and the stopping redundancy of codes,*" *IEEE Trans. Inform. Theory*, vol. 52, no.3, pp. 922-932, Mar. 2006.

[7] C. C. Wang, S. R. Kulkarni, and H. V. Poor, "*Exhausting error-prone patterns in LDPC codes,*" preprint, submitted to *IEEE Trans. Inform.Theory* [Online]. Available: arXiv:cs.IT/0609046.

[8] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "*Selective avoidance of cycles in irregular LDPC code construction,*" *IEEE Trans. Comm..*, vol. 52, no. 8, pp. 1242-1248, Aug. 2004.

[9] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "*Comparison of constructions of irregular Gallager codes,*" *IEEE Trans. Comm.* vol.47, pp. 1449–1454, Oct. 1999.

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

[11] T. Richardson, M. Shokrollahi, and R. Urbanke, "*Design of capacity-approaching irregular low-density parity-check codes,*" *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[12] K. M. Krishnan and P. Shankar, *"Computing the stopping distance of a Tanner graph is NP-hard," IEEE Trans. Inform. Theory*, vol.53, no. 6, pp. 2278-2280, June 2007.

[13] Dejan Vukobratovi´c, Vojin Šenk, "*Evaluation and Design of Irregular LDPC Codes Using ACE Spectrum*", IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 57, NO. 8, AUGUST 2009.

[14] Tao Tian, Chris Jones, John D. Villasenor, and Richard D. Wesel, *"Construction of Irregular LDPC Codes with Low Error Floors"*, Electrical Engineering Department, University of California, Los Angeles, CA, 90095, USA.

