

Error Correction Capability of Column-Weight-Three LDPC Codes: Part II

Shashi Kiran Chilappagari, *Student Member, IEEE*, Dung Viet Nguyen, *Student Member, IEEE*, Bane Vasic, *Senior Member, IEEE*, and Michael W. Marcellin, *Fellow, IEEE*

Abstract—The relation between the girth and the error correction capability of column-weight-three LDPC codes is investigated. Specifically, it is shown that the Gallager A algorithm can correct $g/2 - 1$ errors in $g/2$ iterations on a Tanner graph of girth $g \geq 10$.

Keywords: Low-density parity-check codes, Gallager A algorithm, error floor, girth

I. INTRODUCTION

Iterative message passing algorithms for decoding low-density parity-check (LDPC) codes [1] operate by passing messages along the edges of a graphical representation of the code known as the Tanner graph [2]. These algorithms are optimal when the underlying graph is a tree (see [3],[4] for general theory of LDPC codes), but in the presence of cycles, the decoding becomes sub-optimal and there exist low-weight patterns known as near codewords [5] or trapping sets [6] uncorrectable by the decoder. It is now well established that the trapping sets lead to error floor in the high signal-to-noise (SNR) region (see [7] for a list of references). While it is generally known that high girth codes have better performance in the error floor region, the exact relation between the girth and the slope of the frame error rate (FER) curve in the error floor region is unknown.

In this paper, we consider transmission over the binary symmetric channel (BSC) and the corresponding hard decision decoding algorithms. We focus on column-weight-three codes which are of special importance as their decoders have very low complexity and are interesting for a wide range of applications. We show that a column-weight-three LDPC code with Tanner graph of girth $g \geq 10$ can correct all error patterns of weight $g/2 - 1$ or less under the Gallager A algorithm, thereby showing that there are no trapping sets with critical number less than $g/2$ (see [8] pp. 4-6, for missing definitions). In [7], we showed that the slope of the FER curve in the error floor region is determined by the minimal critical number. In [8], we proved that a column-weight-three LDPC code with Tanner graph of girth $g \geq 10$ always has a trapping set of size $g/2$ with critical number $g/2$ for the Gallager A algorithm. It is worth noting that for codes with Tanner graphs of girth $g = 6$ and $g = 8$, girth alone cannot guarantee correction of

two and three errors, respectively. It can be easily shown that codes with girth six Tanner graphs can correct two errors if and only if the Tanner graph does not contain a codeword of weight four. For codes with Tanner graphs of girth eight, we established necessary and sufficient conditions to correct three errors [9]. Thus, with the results presented here, the problem of determining the slope of the FER curve in the error floor region for column-weight-three codes under the Gallager A algorithm is now completely solved.

The rest of the paper is organized as follows. In Section II, we establish the notation, define the Gallager A algorithm and analyze the algorithm for the first k ($g/4 - 1 \leq k < g/4$) iterations. In Section III, we prove our main result and we conclude in Section IV with a few remarks.

II. PRELIMINARIES

A. Notation

Let \mathcal{C} be an LDPC code with column weight three and length n . The Tanner graph G of \mathcal{C} consists of two sets of nodes: the set of variable nodes V with $|V| = n$ and the set of check nodes C . The check nodes (variable nodes) connected to a variable node (check node) are referred to as its neighbors. An edge e is an unordered pair $\{v, c\}$ of a variable node v and a check node c and is said to be incident on v and c . A directed edge \vec{e} is an ordered pair (v, c) or (c, v) corresponding to the edge $e = \{v, c\}$. With a moderate abuse of notation, we denote directed edges by simple letters (without arrows) but specify the direction. The girth g is the length of the shortest cycle in G . For a given node u , the neighborhood of depth d , denoted by \mathcal{N}_u^d , is the induced subgraph consisting of all nodes reached and edges traversed by paths of length at most d starting from u (including u). The directed neighborhood of depth d of a directed edge $e = (v, c)$ denoted by \mathcal{N}_e^d , is defined as the induced subgraph containing all edges and nodes on all paths e_1, \dots, e_d starting from v such that $e_1 \neq e$ (see [3] for definitions and notation). In a Tanner graph with girth g , we note that \mathcal{N}_u^t is a tree when $t \leq g/2 - 1$. Also, if $e_1 = (v, c)$ and $e_2 = (c, v)$, then $\mathcal{N}_{e_1}^i \cap \mathcal{N}_{e_2}^j = \phi$ for $i + j < g - 1$. Let k denote the number of independent iterations as defined in [1]. The original value of a variable node is its value in the transmitted codeword. We say a variable node is good if its received value is equal to its original value and bad otherwise. A message is said to be correct if it is equal to the original value of the corresponding variable node and incorrect otherwise. In this paper, \circ denotes a good variable node, \bullet denotes a bad variable node and \square denotes a check node. For

Manuscript received July 22, 2008. This work is funded by NSF under Grant CCF-0634969, ECCS-0725405, ITR-0325979 and by the INSIC-EHDR program.

S. K. Chilappagari, D. V. Nguyen, B. Vasic and M. W. Marcellin are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona, 85721 USA. (emails: {shashic, nguyendv, vasic, marcellin}@ece.arizona.edu.

output symmetric channels (see [3]), without loss of generality, we can assume that the all zero codeword is transmitted. We make this assumption throughout the paper. Hence, a bad variable node has received value 1 and an incorrect message has a value of 1. A configuration of bad variable nodes is a subgraph in which the location of bad variables relative to each other is specified. A valid configuration \mathcal{C}_g is a configuration of at most $g/2 - 1$ bad variable nodes free of cycles of length less than g . The set of bad variable nodes in \mathcal{N}_e^d is denoted by $\mathcal{B}(\mathcal{N}_e^d)$ and $|\mathcal{B}(\mathcal{N}_e^d)|$ is denoted by $B(\mathcal{N}_e^d)$. The number of bad variable nodes at depth d in \mathcal{N}_e^d is denoted by b_e^d .

B. Hard Decision Decoding Algorithms

Gallager in [1] proposed two simple binary message passing algorithms for decoding over the BSC; Gallager A and Gallager B. See [10] for a detailed description of the Gallager B algorithm. For column-weight-three codes, which are the main focus of this paper, these two algorithms are the same. Every round of message passing (iteration) starts with sending messages from variable nodes to check nodes (first half of the iteration) and ends by sending messages from check nodes to variable nodes (second half of the iteration). Let \mathbf{r} , a binary n -tuple be the input to the decoder. Let $\omega_j(v, c)$ denote the message passed by a variable node v to its neighboring check node c in j^{th} iteration and $\varpi_j(c, v)$ denote the message passed by a check node c to its neighboring variable node v . Additionally, let $\omega_j(v, :)$ denote the set of all messages from v , $\omega_j(v, : \setminus c)$ denote the set of messages from v to all its neighbors except to c and $\omega_j(:, c)$ denote the set of all messages to c . Let $|\varpi(:, v) = m|$ denote the number of incoming messages to v which are equal to $m \in \{0, 1\}$. The terms $\omega_j(:, \setminus v, c)$, $\varpi_j(c, :)$, $\varpi_j(c, : \setminus v)$, $\varpi_j(:, : v)$ and $\varpi_j(:, \setminus c, v)$ are defined similarly. The Gallager A algorithm can then be defined as follows.

$$\begin{aligned} \omega_1(v, c) &= \mathbf{r}(v) \\ \omega_j(v, c) &= \begin{cases} 1, & \text{if } \varpi_{j-1}(:, \setminus c, v) = \{1\} \\ 0, & \text{if } \varpi_{j-1}(:, \setminus c, v) = \{0\} \\ \mathbf{r}(v), & \text{otherwise} \end{cases} \\ \varpi_j(c, v) &= \left(\sum \omega_j(:, \setminus v, c) \right) \bmod 2 \end{aligned}$$

At the end of each iteration, an estimate of each variable node is made based on the incoming messages and possibly the received value. In this paper, we assume that the estimate of a variable node is taken to be the majority of the incoming messages (see [8] for details). The decoded word at the end of the j^{th} iteration is denoted as $\mathbf{r}^{(j)}$. The decoder is run until a valid codeword is found or a maximum number of iterations M is reached, whichever is earlier. The output of the decoder is either a codeword or $\mathbf{r}^{(M)}$.

C. The first k iterations

We begin with a lemma describing the messages passed by the Gallager A algorithm in a column-weight-three code.

Lemma 1: (i) If v is a bad variable node, then we have $\omega_1(v, :) = \{1\}$ and

- $\omega_j(v, :) = \{1\}$ if $|\varpi_{j-1}(:, v) = 1| \geq 2$, i.e., v sends incorrect messages to all its neighbors if it receives two or more incorrect messages from its neighboring checks in the previous iteration.
- $\omega_j(v, : \setminus c) = \{1\}$ and $\omega_j(v, c) = 0$ if $\varpi_{j-1}(:, \setminus c, v) = \{0\}$ and $\varpi_{j-1}(c, v) = 1$, i.e., v sends one correct message and two incorrect messages if it receives one incorrect message from its neighboring checks in the previous iteration. The correct message is sent along the edge on which the incorrect message is received.
- $\omega_j(v, :) = \{0\}$ if $\varpi_{j-1}(:, v) = \{0\}$, i.e., v sends all correct messages if it receives all correct messages from its neighboring checks in the previous iteration.

(ii) If v is a good variable node, then we have $\omega_1(v, :) = \{0\}$ and

- $\omega_j(v, :) = \{0\}$ if $|\varpi_{j-1}(:, v) = 0| \geq 2$, i.e., v sends all correct messages if it receives two or more correct messages from its neighboring checks in the previous iteration.
- $\omega_j(v, : \setminus c) = \{0\}$ and $\omega_j(v, c) = 1$ if $\varpi_{j-1}(:, \setminus c, v) = \{1\}$ and $\varpi_{j-1}(c, v) = 0$, i.e., v sends one incorrect message and two correct messages if it receives two incorrect messages from its neighboring checks in the previous iteration. The incorrect message is sent along the edge on which the correct message is received.
- $\omega_j(v, :) = \{1\}$, if $\varpi_{j-1}(:, v) = \{1\}$, i.e., v sends all incorrect messages if it receives all incorrect messages from its neighboring checks in the previous iteration.

(iii) For a check node c , we have,

- $\varpi_j(c, v) = \omega_j(v, c) \oplus 1$, if $|\omega_j(:, c) = 1|$ is odd, i.e., c sends incorrect messages along the edges on which it received correct messages and correct messages along the edges on which it received incorrect messages, if the total number of incoming incorrect messages from its neighboring variable nodes is odd.
- $\varpi_j(c, v) = \omega_j(v, c)$, if $|\omega_j(:, c) = 1|$ is even, i.e., c sends incorrect messages along the edges on which it received incorrect messages and correct messages along the edges on which it received correct messages, if the total number of incoming incorrect messages from its neighboring variable nodes is even.

(iv) A variable node is estimated incorrectly at the end of an iteration if it receives at least two incorrect messages.

Proof: Follows from the description of the Gallager A algorithm. \blacksquare

Now let v be a variable node which sends an incorrect message along the edge $e = (v, c)$ in the $(k+1)^{\text{th}}$ iteration. The message along e depends only on the variable nodes and check nodes in \mathcal{N}_e^{2k} . Under the assumption that \mathcal{N}_e^{2k} is a tree, the above observations provide a method to find all the possible configurations of bad variable nodes in \mathcal{N}_e^{2k} . We have the following two cases:

(a) v is a bad variable node: In this case, there must be at least one variable node in \mathcal{N}_e^2 which sends an incorrect message in the k^{th} iteration.

(b) v is a good variable node: In this case, there must be at least two variable nodes in \mathcal{N}_e^2 which send an incorrect

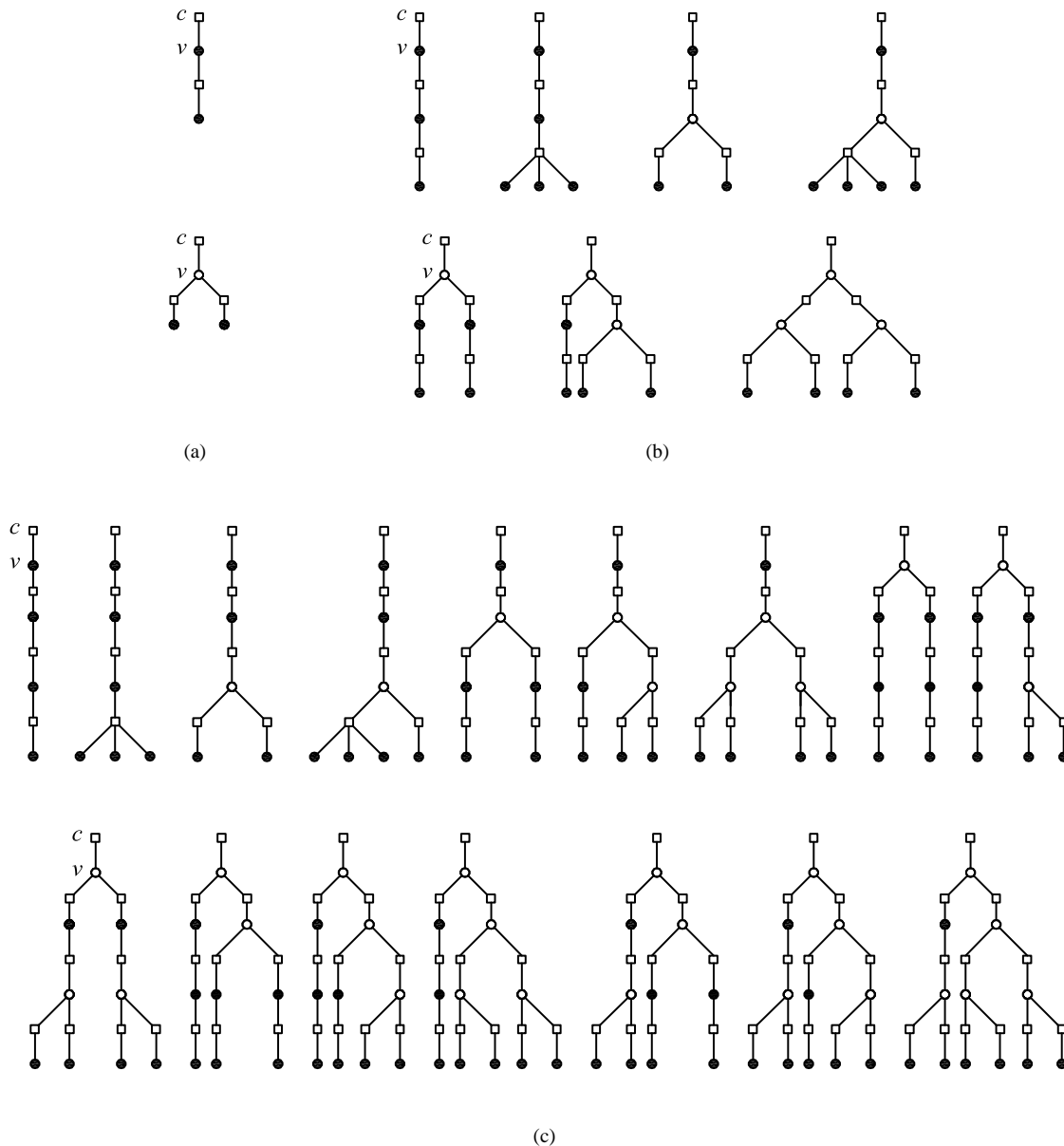


Fig. 1. Possible configurations of at most $2k + 1$ bad variable nodes in the neighborhood of a variable node v sending an incorrect message to check node c in the $(k + 1)^{th}$ iteration for (a) $k = 1$, (b) $k = 2$ and (c) $k = 3$.

message in the k^{th} iteration.

This is repeated k times until we reach the first iteration, at which point only the nodes and edges in \mathcal{N}_e^{2k} would have been explored and all of these are guaranteed to be distinct as \mathcal{N}_e^{2k} is a tree. Since only bad variables send incorrect messages in the first iteration, we can calculate the number of bad variables in each configuration. Specifically, let v be a variable node which sends an incorrect message along the $e = (v, c)$ in the second iteration. If v is a good variable node, then \mathcal{N}_e^2 must have at least two bad variable nodes. If v is bad, \mathcal{N}_e^2 must have at least one bad variable node. Following this approach, we have Fig. 1(a), Fig. 1(b) and Fig. 1(c) which show the possible configurations of bad variable nodes in \mathcal{N}_e^{2k} so that v sends an incorrect message in the $(k + 1)^{th}$ iteration, for $k = 1, k = 2$ and $k = 3$, respectively.

Remarks: (i) Fig. 1 shows configurations with at most $2k + 1$ bad variable nodes.

(ii) We do not illustrate configurations in which a bad variable node receives two incorrect messages in the k^{th} iteration, so that it sends an incorrect message in the $(k + 1)^{th}$ iteration. However, all such configurations can be found by considering configurations involving good variable nodes and converting a good variable node to a bad one. This increases the number of bad variable nodes in the configuration. As will be seen later, such configurations are not relevant for establishing our main result.

The above observations help establish bounds on $B(\mathcal{N}_e^{2k})$, which we state in the following lemma.

Lemma 2: (i) If v is a bad variable node sending an incorrect message on $e = (v, c)$ in the $(k + 1)^{th}$ iteration and

\mathcal{N}_e^{2k} is a tree, then $B(\mathcal{N}_e^{2k}) \geq k + 1$. If $B(\mathcal{N}_e^{2k-2}) = 1$, i.e., $b_e^d = 0$ for $d = 2, 4, \dots, 2(k-1)$, then $B(\mathcal{N}_e^{2k}) \geq 2^{(k-1)} + 1$. If $B(\mathcal{N}_e^{2k-2}) = 2$, then $B(\mathcal{N}_e^{2k}) \geq 2^{(k-2)} + 2$.

(ii) If v is a good variable node sending an incorrect message on $e = (v, c)$ in the $(k+1)^{th}$ iteration and \mathcal{N}_e^{2k} is a tree, then $B(\mathcal{N}_e^{2k}) \geq 2k$. If $B(\mathcal{N}_e^{2k-2}) = 0$, then $B(\mathcal{N}_e^{2k}) \geq 2k$. If $B(\mathcal{N}_e^{2k-2}) = 1$, then $B(\mathcal{N}_e^{2k}) \geq 2^{(k-1)} + 2^{(k-2)} + 1$. If $B(\mathcal{N}_e^{2k-2}) = 2$, then $B(\mathcal{N}_e^{2k}) \geq 2^{(k-1)} + 2$.

Proof: The proof is by induction on k . It is easy to verify the bounds for $k = 2$. Let the bounds be true for some $k \geq 2$. Let v_0 be a bad variable node sending an incorrect message on $e = (v_0, c)$ in the $(k+2)^{th}$ iteration. Further, assume that \mathcal{N}_e^{k+2} is a tree. Then, \mathcal{N}_e^k has at least one check node c_1 which sends an incorrect message along the edge $e_1 = (c_1, v_0)$ in the $(k+1)^{th}$ iteration. This implies that \mathcal{N}_e^k has at least one variable node v_2 sending an incorrect message in the $(k+1)^{th}$ iteration along the edge $e_2 = (v_2, c_1)$. Since a path of length 2 exists between v_0 and v_1 , $\mathcal{N}_e^k \subset \mathcal{N}_e^{k+2}$.

If v_2 is a bad variable node, then $B(\mathcal{N}_e^{2k}) \geq k + 1$ and consequently $B(\mathcal{N}_e^{2k+2}) \geq k + 2$. If v_2 is a good variable node, then $B(\mathcal{N}_e^{2k}) \geq 2k$ and consequently $B(\mathcal{N}_e^{2k+2}) \geq 2k + 1 > k + 1$.

If $b_e^d = 0$ for $d = 2, 4, \dots, 2k$, then v_2 is a good variable node such that $B(\mathcal{N}_e^{2k-2}) = 0$ which implies that $B(\mathcal{N}_e^{2k}) \geq 2k$ by the induction hypothesis. Hence, $B(\mathcal{N}_e^{2k+2}) \geq 2k + 1$.

If $B(\mathcal{N}_e^{2k}) = 2$ then either (a) v_2 is a bad variable node with $b_e^d = 0$ for $d = 2, 4, \dots, 2(k-1)$ which implies that $B(\mathcal{N}_e^{2k}) \geq 2^{(k-1)} + 1$ by the induction hypothesis. Hence, $B(\mathcal{N}_e^{2k+2}) \geq 2^{(k-1)} + 2$, or (b) v_2 is a good variable node with $B(\mathcal{N}_e^{2k-2}) = 1$ which implies that $B(\mathcal{N}_e^{2k}) \geq 2^{(k-1)} + 2^{(k-2)} + 1$ by the induction hypothesis. Hence, $B(\mathcal{N}_e^{2k+2}) \geq 2^{(k-1)} + 2^{(k-2)} + 2 > 2^{(k-1)} + 2$.

By the principle of mathematical induction, the bounds are true for all k when v_0 is a bad variable node. The proofs are similar for the case when v_0 is a good variable node. ■

III. THE MAIN THEOREM

In this section, we prove that a column-weight-three code with Tanner graph of girth $g \geq 10$ can correct $g/2 - 1$ errors in $g/2$ iterations of the Gallager A algorithm. The proof proceeds by finding, for a particular choice of k , all configurations of $g/2 - 1$ or less bad variable nodes which do not converge in $k + 1$ iterations and then prove that these configurations also converge in subsequent iterations. When $g/2$ is even, we use $k = g/4 - 1$ (or $g/2 - 1 = 2k + 1$) and when $g/2$ is odd, we use $k = (g-2)/4$ (or $g/2 - 1 = 2k$). We deal with these cases separately.

A. $g/2$ is even

Let v_0 be a variable node which receives two incorrect messages along the edges $e_1 = (c_1^1, v_0)$ and $e_2 = (c_1^2, v_0)$ at the end of $(k+1)^{th}$ iteration. This implies that $\mathcal{N}_{e_1}^1$ and $\mathcal{N}_{e_2}^1$ each has a variable node, v_2^1 and v_2^2 respectively, that sends an incorrect message in the $(k+1)^{th}$ iteration to check nodes c_1^1 and c_1^2 , respectively. Let $e_3 = (v_2^1, c_1^1)$, $e_4 = (v_2^2, c_1^2)$, $e_5 = (c_1^1, v_2^1)$, and $e_6 = (c_1^2, v_2^2)$ (see Fig. 2(a) for an illustration). All possible configurations of bad variable nodes

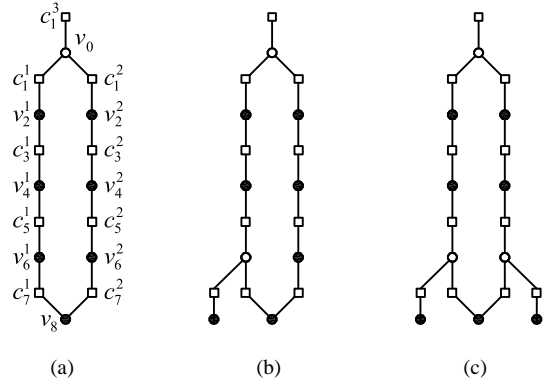


Fig. 2. Configurations of at most 7 bad variable nodes, free of cycles of length less than 16, which do not converge in 4 iterations.

in $\mathcal{N}_{e_3}^{2k}$ and $\mathcal{N}_{e_4}^{2k}$ can be determined using the method outlined in Section II-C. Since there exists a path of length 3 between v_2^2 and c_1^1 , we have $\mathcal{N}_{e_4}^i \subset \mathcal{N}_{e_5}^{i+3}$. Also, $\mathcal{N}_{e_3}^i \cap \mathcal{N}_{e_5}^j = \phi$ for $i+j < g-1 = 4k+3$. Therefore, $\mathcal{N}_{e_3}^i \cap \mathcal{N}_{e_4}^j = \phi$ for $i+j < 4k$. This implies that $\mathcal{N}_{e_3}^{2k}$ and $\mathcal{N}_{e_4}^{2k}$ can have a common node only at depth $2k$. The total number of bad variable nodes in $\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k}$, $B(\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k})$, in any configuration is therefore lower bounded by $B(\mathcal{N}_{e_3}^{2k-2}) + B(\mathcal{N}_{e_4}^{2k-2}) + \max(b_{e_3}^{2k}, b_{e_4}^{2k})$ or equivalently $\max(B(\mathcal{N}_{e_3}^{2k-2}) + B(\mathcal{N}_{e_4}^{2k}), B(\mathcal{N}_{e_3}^{2k}) + B(\mathcal{N}_{e_4}^{2k-2}))$. We are interested only in the valid configurations, i.e., at most $g/2 - 1$ bad variable nodes, free from cycles of length less than g . We divide the discussion into three parts: (1) we find all the possible valid configurations for the case when $g = 16$; (2) we then proceed iteratively for $g > 16$; (3) We consider the case $g = 12$ separately as the arguments for $g \geq 16$ do not hold for this case.

1) $g = 16$: Let v be a variable node which sends an incorrect message in the iteration $k+1 = g/4 = 4$ along edge $e = (v, c)$, given that there are at most seven bad variables and \mathcal{N}_e^7 is a tree. Fig. 1(c) illustrates different configurations of bad variable nodes in \mathcal{N}_e^6 . As remarked earlier, Fig. 1(c) does not show configurations in which a bad variable node has to receive two incorrect messages in an iteration to send an incorrect message along the third edge in the next iteration. It can be seen in the proof of Theorem 2 that these cases do not arise in valid configurations.

Let $v_0, v_2^1, v_2^2, e_1, e_2, e_3, e_4$ be defined as above with $k = 3$. Using the arguments outlined above (and the constraint that $g = 16$), all possible configurations such that $B(\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k}) \leq 7$ can be found. Fig. 2 shows all such possible configurations.

2) $g \geq 20$: Let \mathcal{C}_g be a valid configuration in which there exists a variable node v_0 which receives two incorrect messages along the edges $e_1 = (c_1^1, v_0)$ and $e_2 = (c_1^2, v_0)$ at the end of $(k+1)^{th}$ iteration. This implies that $\mathcal{N}_{e_1}^1$ and $\mathcal{N}_{e_2}^1$ each has a variable node, v_2^1 and v_2^2 , respectively, that sends an incorrect message in the $(k+1)^{th}$ iteration. We have the following lemma.

Lemma 3: v_2^1 and v_2^2 are bad variable nodes.

Proof: The proof is by contradiction. We know that the total number of bad variables in any configuration is lower bounded by

$\max(B(\mathcal{N}_{e_3}^{2k-2}) + B(\mathcal{N}_{e_4}^{2k}), B(\mathcal{N}_{e_3}^{2k}) + B(\mathcal{N}_{e_4}^{2k-2}))$ and that $k > 3$. We have two cases.

(a) v_2^1 and v_2^2 are both good variable nodes: We first note that in any valid configuration, $B(\mathcal{N}_{e_3}^{2k-2}) \geq 2$. Otherwise, we have $B(\mathcal{N}_{e_3}^{2k-2}) = 1$, and from Lemma 2, $B(\mathcal{N}_{e_3}^{2k}) \geq 2^{(k-1)} + 2^{(k-2)} + 1 > 2k + 1$ or, we have $B(\mathcal{N}_{e_3}^{2k-2}) = 0$, and $B(\mathcal{N}_{e_3}^{2k}) \geq 2^k + 2 > 2k + 1$. Both cases are a contradiction as we have at most $2k + 1$ bad variable nodes. Hence, $B(\mathcal{N}_{e_3}^{2k-2}) \geq 2$.

Now, $B(\mathcal{N}_{e_4}^{2k}) \geq 2k$, and hence we have $B(\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k}) \geq B(\mathcal{N}_{e_4}^{2k}) + B(\mathcal{N}_{e_3}^{2k-2}) \geq 2k + 2$, which is a contradiction.

(b) v_2^1 is a bad variable node and v_2^2 is a good variable node. The opposite case is identical.

First we claim that in any valid configuration, $B(\mathcal{N}_{e_3}^{2k-2}) \geq 2$. Since v_2^1 is a bad variable node, $B(\mathcal{N}_{e_3}^{2k-2}) \neq 0$. Assume that $B(\mathcal{N}_{e_3}^{2k-2}) = 1$. Then $B(\mathcal{N}_{e_3}^{2k}) \geq 2^{(k-1)} + 1$. Again, $B(\mathcal{N}_{e_4}^{2k-2}) \geq 2$ implies that $B(\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k}) \geq 2^{(k-1)} + 3 > 2k + 1$ (as $k > 3$), which is a contradiction. Hence, $B(\mathcal{N}_{e_3}^{2k-2}) \geq 2$.

Now, $B(\mathcal{N}_{e_3}^{2k-2}) \geq 2$ and $B(\mathcal{N}_{e_4}^{2k}) \geq 2k$, implies that $B(\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k}) \geq 2k + 2$ which is a contradiction.

Hence, v_2^1 and v_2^2 are both bad variable nodes.

■

We now have the following theorem:

Theorem 1: If \mathcal{C}_g is a configuration which does not converge in $(k + 1)$ iterations, then there exists a configuration \mathcal{C}_{g-4} which does not converge in k iterations.

Proof: In the configuration \mathcal{C}_g , v_2^1 and v_2^2 are bad variable nodes which send incorrect messages to check nodes c_1^1 and c_1^2 , respectively, in the $(k + 1)^{th}$ iteration. This implies that $\mathcal{N}_{e_3}^1$ and $\mathcal{N}_{e_4}^1$ each has a check node, c_3^1 and c_3^2 , respectively, that sends an incorrect message in k^{th} iteration to v_2^1 and v_2^2 , respectively. Now consider a configuration \mathcal{C}_{g-4} constructed from \mathcal{C}_g by removing the nodes $v_2^1, v_2^2, c_1^1, c_1^2$ and the edges connecting them to their neighbors and introducing the edges (v_0, c_3^1) and (v_0, c_3^2) (see Fig. 3(b)). If \mathcal{C}_g has at most $2k + 1$ bad variable nodes and no cycles of length less than g , then \mathcal{C}_{g-4} has at most $2(k-1) + 1$ bad variable nodes and no cycles of length less than $g - 4$. In \mathcal{C}_{g-4} variable node v_0 receives two incorrect messages at the end of k iterations and hence \mathcal{C}_{g-4} is a valid configuration which does not converge in k iterations.

■

Theorem 1 gives a method to construct valid configurations of bad variable nodes for girth g from valid configurations for girth $g + 4$. Also, if \mathcal{C}_g^1 and \mathcal{C}_g^2 are two distinct valid configurations, then the configurations \mathcal{C}_{g-4}^1 and \mathcal{C}_{g-4}^2 constructed from \mathcal{C}_g^1 and \mathcal{C}_g^2 , respectively, are distinct. Hence, the number of valid configurations for girth g is greater than or equal to the number of valid configurations for girth $g + 4$. Note that the converse of Theorem 1 is not true in general. However, for $g \geq 16$, we will show in Theorem 2 that any configuration for girth g can be extended to a configuration for girth $g + 4$.

Theorem 2: For $g/2$ even and $k \geq 3$, there are only three valid configurations which do not converge in $(k + 1)$ iterations.

Proof: For $k = 3$, we have $g = 16$ and there are only three valid configurations as given in Fig. 2. So, for $g \geq 16$ and $g/2$ even, there can be at most three valid configurations.

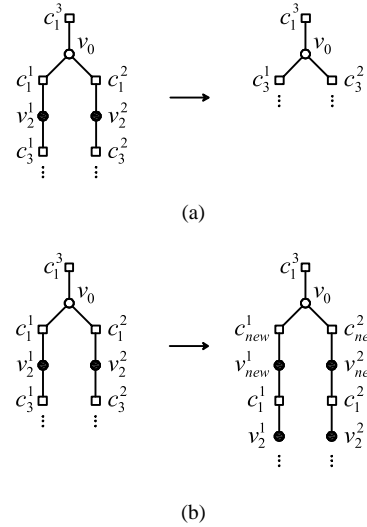


Fig. 3. (a) Construction of \mathcal{C}_{g-4} from \mathcal{C}_g , (b) Construction of \mathcal{C}_{g+4} from \mathcal{C}_g .

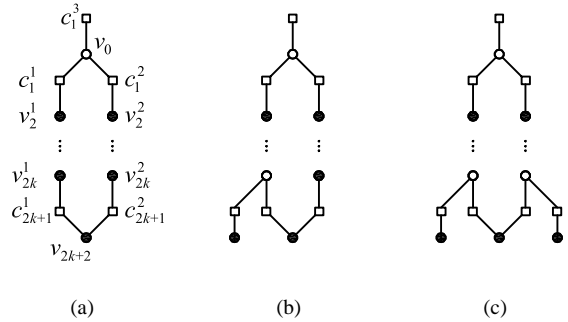


Fig. 4. Configurations of at most $2k + 1$ bad variable nodes free of cycles of length less than $4k + 4$ which do not converge in $(k + 1)$ iterations.

Each valid configuration for $g = 16$, can be extended to a configuration \mathcal{C}_{20} for $g = 20$ by the addition of two bad variable nodes v_{new}^1 and v_{new}^2 in the following way. Remove the edges (v_0, c_1^1) and (v_0, c_1^2) . Add bad variable nodes v_{new}^1 and v_{new}^2 and check nodes c_{new}^1 and c_{new}^2 . Introduce the edges (v_0, c_{new}^1) , (v_0, c_{new}^2) , (v_{new}^1, c_{new}^1) , (v_{new}^2, c_{new}^2) , (v_{new}^1, c_1^1) and (v_{new}^2, c_1^2) (see Fig. 3(b) for an illustration). It can be seen that \mathcal{C}_{20} is a valid configuration for girth $g = 20$. In general, the configurations constructed using the above method from the valid configurations for $g \geq 16$ are valid configurations for $g + 4$. Fig. 4 illustrates the three configurations for all $k \geq 3$.

■

Remark: In all valid configurations \mathcal{C}_g with $g \geq 16$, no bad variable node receives two incorrect messages at the end of the $(k + 1)^{th}$ iteration.

Theorem 3: All valid configurations \mathcal{C}_g converge to a codeword in $g/2$ iterations.

Proof: We prove the theorem for one configuration for $g = 16$ only. The proof is similar for other configurations. At the end of fourth iteration, let v_0 receive two incorrect messages (see Fig 2(a)). It can be seen that there cannot exist another variable node (either good or bad) which receives two incorrect messages without violating the $g = 16$ constraint.

Also, v_8 receives all correct messages and $v_2^1, v_2^2, v_4^1, v_4^2, v_6^1, v_6^2$ receive one incorrect message each from $c_3^1, c_3^2, c_5^1, c_5^2, c_7^1, c_7^2$, respectively. In the fifth iteration, we have

$$\begin{aligned}\omega_5(v_0, c_1^3) &= 1, \\ \omega_5(v, : \setminus c) &= \{1\}, \quad (v, c) \in \{(v_2^1, c_3^1), (v_2^2, c_3^2), \\ &\quad (v_4^1, c_5^1), (v_4^2, c_5^2), (v_6^1, c_7^1), (v_6^2, c_7^2)\}, \\ \omega_5(v, c) &= 0, \quad \text{otherwise.}\end{aligned}$$

In the sixth iteration, we have

$$\begin{aligned}\omega_6(v_0, c_1^3) &= 1, \\ \omega_6(v, : \setminus c) &= \{1\}, \quad (v, c) \in \{(v_2^1, c_3^1), (v_2^3, c_3^2), (v_4^1, c_5^1), \\ &\quad (v_4^2, c_5^2)\}, \\ \omega_6(v, c) &= 0, \quad \text{otherwise.}\end{aligned}$$

In the seventh iteration, we have,

$$\begin{aligned}\omega_7(v_0, c_1^3) &= 1, \\ \omega_7(v, : \setminus c) &= \{1\}, \quad (v, c) \in \{(v_2^1, c_3^1), (v_2^2, c_3^2)\} \\ \omega_7(v, c) &= 0, \quad \text{otherwise.}\end{aligned}$$

Finally in the eighth iteration, we have,

$$\begin{aligned}\omega_8(v_0, c_1^3) &= 1, \\ \omega_8(v, c) &= 0, \quad \text{otherwise}\end{aligned}$$

At the end of eighth iteration, no variable node receives two incorrect messages and hence the decoder converges to a valid codeword. ■

3) $g = 12$: In this case, $k = 2$ and the incoming messages at the end of the second iteration are independent. We need to prove that any code with Tanner graph with $g = 12$, can correct all error patterns of weight less than six. Let v be a variable node which sends an incorrect message in the third iteration along edge $e = (v, c)$ given that there are at most 5 bad variables and \mathcal{N}_e^5 is a tree. Fig. 1(c) illustrates different configurations of bad variable nodes in \mathcal{N}_e^4 .

Fig. 5 shows all possible configurations of five or less bad variable nodes which do not converge to a codeword at the end of three iterations. However, all the configurations converge to a codeword in six iterations. The proofs for configurations in Fig. 5(a)-(h) are similar to the proof for configuration in Fig. 2(a) and are omitted. Since, configuration (i) has only four bad variable nodes, a complete proof for convergence requires considering all possible locations of the fifth bad variable node, but other than that the structure of the proof is identical to that of the proof for the configuration in Fig. 2(a). It is worth noting that in this case, there exist configurations in which a bad variable receives two incorrect messages at the end of the third iteration. However, all the configurations eventually converge to a codeword.

B. $g/2$ is odd

In this case, we have $k = (g - 2)/4$ and we need to prove that the code is capable of correcting all error patterns of weight $g/2 - 1 = 2k$ or less. The methodology of the proof is similar to the proof in the case when $g/2$ is even. In this case,

we have $\mathcal{N}_{e_3}^i \cap \mathcal{N}_{e_4}^j = \phi$ for $i + j < 4k - 2$. This implies that $\mathcal{N}_{e_3}^{2k}$ and $\mathcal{N}_{e_4}^{2k}$ can have a common node at depth $2k - 1$. Therefore, in any configuration, $B(\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k})$ is lower bounded by $\max(B(\mathcal{N}_{e_3}^{2k-4}) + B(\mathcal{N}_{e_4}^{2k}), B(\mathcal{N}_{e_3}^{2k}) + B(\mathcal{N}_{e_4}^{2k-4}))$. The valid configurations in this case are the ones which satisfy $B(\mathcal{N}_{e_3}^{2k} \cup \mathcal{N}_{e_4}^{2k}) \leq 2k$. We again deal with $g \geq 14$ and $g = 10$ separately.

1) $g \geq 14$:

Lemma 4: For $g = 14$, there is only one configuration of six bad variable nodes which does not converge in four iterations.

Proof: Using arguments outlined above and the configurations in Fig. 1(b) along with the constraint that $g \geq 14$, we conclude that there is only one configuration which does not converge in four iterations, which is shown in Fig. 6(a). ■

Lemma 5: If \mathcal{C}_g with $g \geq 14$ is a valid configuration which does not converge in $k + 1$ iterations, then v_1^1 and v_1^2 are bad variable nodes

Proof: Similar to the proof of Lemma 3. ■

Theorem 4: If \mathcal{C}_g is a valid configuration which does not converge in $k + 1$ iterations, then there exists a valid configuration \mathcal{C}_{g-4} which does not converge in k iterations.

Proof: Similar to the proof of Theorem 1. ■

Theorem 5: For $k \geq 3$, there is only one valid configuration which does not converge in $k + 1$ iterations.

Proof: For $k = 3$, we have $g = 14$ and there is only one configuration. For $k = 4$, the number of valid configurations cannot be more than one. The valid configuration for $g = 14$, can be extended to a configuration for $g = 18$ (in the same manner as in Theorem 2). In general, the valid configuration for girth g can be extended to a valid configuration for girth $g + 4$. Fig. 6(b) shows \mathcal{C}_g for all $g \geq 14$. ■

Theorem 6: The configuration \mathcal{C}_g converges to a codeword in $g/2$ iterations.

Proof: Similar to the proof of Theorem 3. ■

2) $g = 10$: In this case, $k = 2$ and there are three configurations which do not converge at the end of the third iteration. Fig. 7 shows the three configurations. It can be shown that these configurations converge in five iterations.

IV. DISCUSSION

In this paper, we have established a relation between the girth and error correction capability of column-weight-three LDPC codes. The result presented is the best possible bound as it is known that codes with girth $g \geq 10$ cannot be guaranteed to correct $g/2$ errors. While it is disappointing that the error correction capability grows only linearly in the girth for column-weight-three codes, the methodology of our proof can be applied to higher column-weight codes to (hopefully) obtain better results.

REFERENCES

- [1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [2] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, Sept. 1981.
- [3] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

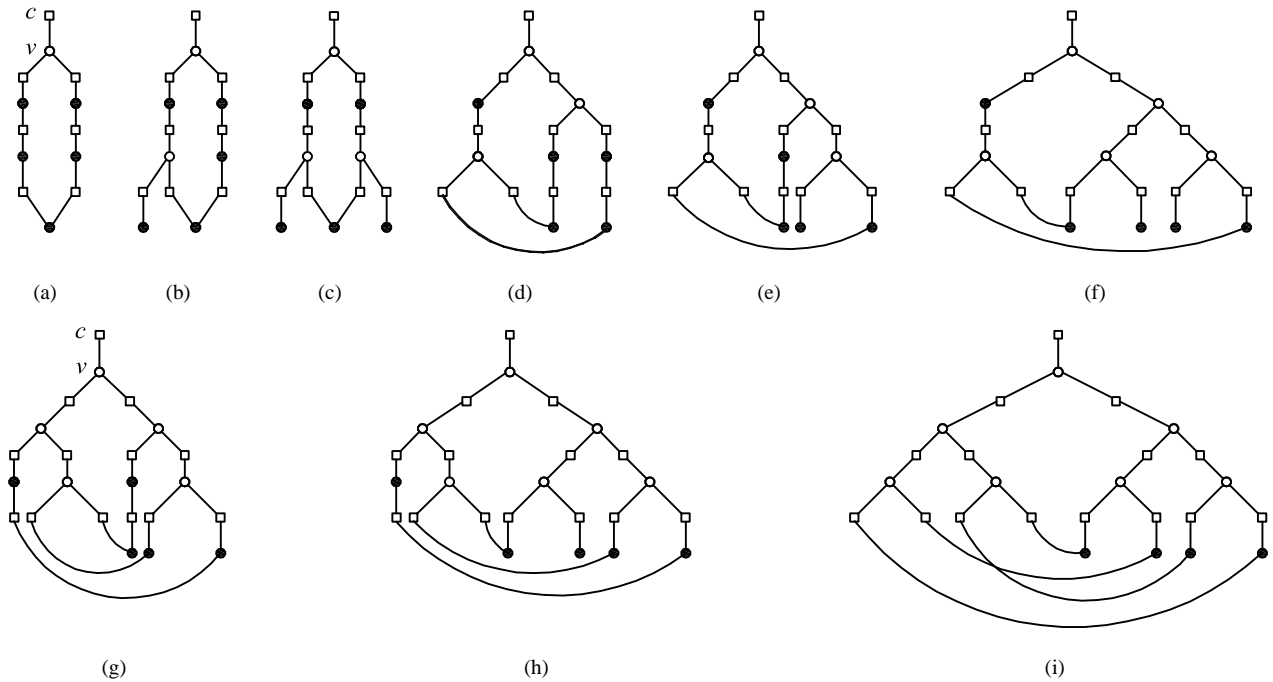


Fig. 5. Configurations of at most 5 variable nodes free of cycles of length less than 12 which do not converge in 3 iterations.

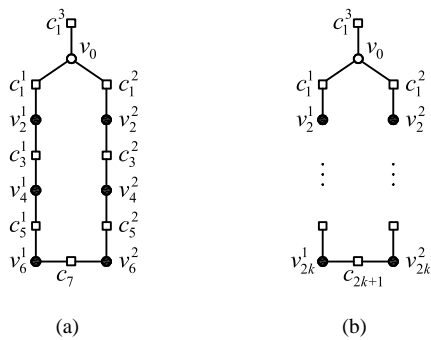


Fig. 6. (a) Configuration of at most 6 bad variable nodes free of cycles of length less than 14 which does not converge in 4 iterations (b) Configuration of at most $2k$ bad variable nodes free of cycles of length less than $4k + 2$ which does not converge in $k + 1$ iterations.

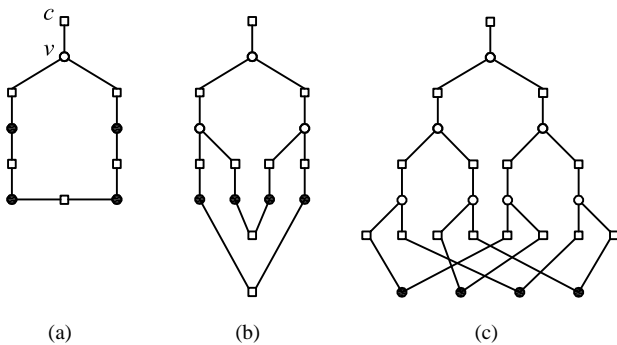


Fig. 7. Configurations of at most 4 variable nodes free of cycles of length less than 10 which do not converge in 3 iterations.

- [4] T. J. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.
- [5] D. J. C. MacKay and M. J. Postol, "Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes," in *Proceedings of MFCSIT2002, Galway*, ser. Electronic Notes in Theoretical Computer Science, vol. 74. Elsevier, 2003. [Online]. Available: <http://www.inference.phy.cam.ac.uk/mackay/abstracts/margulis.html>
- [6] T. J. Richardson, "Error floors of LDPC codes," in *Proc. of 41st Annual Allerton Conf. on Communications, Control and Computing*, 2003, pp. 1426–1435.
- [7] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Eliminating trapping sets in low-density parity check codes by using Tanner graph covers," to appear in *IEEE Trans. Inform. Theory*. [Online]. Available: <http://arxiv.org/abs/0805.1662>
- [8] S. K. Chilappagari and B. Vasic, "Error correction capability of column-weight-three LDPC codes," submitted to *IEEE Trans. Inform. Theory*. [Online]. Available: <http://arxiv.org/abs/0710.3427>
- [9] S. K. Chilappagari, A. R. Krishnan, and B. Vasic, "LDPC codes which can correct three errors under iterative decoding," in *Proc. of IEEE Information Theory Workshop*, May 5-9, 2008.
- [10] A. Shokrollahi, "An introduction to low-density parity-check codes," in *Theoretical aspects of computer science: advanced lectures*. New York, NY, USA: Springer-Verlag New York, Inc., 2002, pp. 175–197.