

# Error Correction Capability of Column-Weight-Three LDPC Codes

Shashi Kiran Chilappagari, *Student Member, IEEE*, and Bane Vasic, *Senior Member, IEEE*

## Abstract

In this paper, we investigate the error correction capability of column-weight-three LDPC codes when decoded using the Gallager A algorithm. We prove that the necessary condition for a code to correct  $k \geq 5$  errors is to avoid cycles of length up to  $2k$  in its Tanner graph. As a consequence of this result, we show that given any  $\alpha > 0$ ,  $\exists N$  such that  $\forall n > N$ , no code in the ensemble of column-weight-three codes can correct all  $\alpha n$  or fewer errors. We extend these results to the bit flipping algorithm.

## Index Terms

Low-density parity-check codes, Gallager A algorithm, trapping sets, error correction capability

## I. INTRODUCTION

Gallager in [1] showed that for  $\gamma \geq 3$  and  $\rho > \gamma$ , there exist  $(n, \gamma, \rho)$  regular low-density parity-check (LDPC) codes for which the bit error probability tends to zero asymptotically whenever we operate below the threshold. Richardson and Urbanke in [2] derived the capacity of LDPC codes for various message passing algorithms and described density evolution, a deterministic algorithm to compute thresholds. Zyablov and Pinsker [3] analyzed LDPC codes under a simpler decoding algorithm known as the bit

Manuscript received October 17, 2007. This work is funded by NSF under Grant CCF-0634969, ITR-0325979 and INSIC-EHDR program.

S. K. Chilappagari and B. Vasic are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721 USA (e-mails: shashic@ece.arizona.edu, vasic@ece.arizona.edu).

flipping algorithm and showed that almost all the codes in the regular ensemble with  $\gamma \geq 5$  can correct a constant fraction of worst case errors. Sipser and Spielman in [4] used expander graph arguments to analyze bit flipping algorithm. Burshtein and Miller in [5] applied expander based arguments to show that message passing algorithms can also correct a fixed fraction of worst case errors when the degree of each variable node is at least five. Feldman *et al.* [6] showed that linear programming decoder [7] is also capable of correcting a fraction of errors. Recently, Burshtein in [8] showed that regular codes with variable nodes of degree four are capable of correcting a linear number of errors under bit flipping algorithm. He also showed tremendous improvement in the fraction of correctable errors when the variable node degree is at least five.

In this paper, we consider the error correction capability of the ensemble  $\mathcal{C}^n(3, \rho > 3)$  of  $(3, \rho)$  regular LDPC codes as defined in [2] when decoded using the Gallager A algorithm. We analyze decoding failures using the notion of trapping sets and prove that a code with girth  $g \geq 10$  cannot correct all  $g/2$  or fewer errors. Using this result, we prove that for any  $\alpha > 0$ , for sufficiently large block length  $n$ , no code in the  $\mathcal{C}^n(3, \rho)$  ensemble can correct  $\alpha$  fraction of errors. This result settles the problem of error correction capability of column-weight-three codes. The rest of the paper is organized as follows. In Section II we establish the notation and describe the Gallager A algorithm. We then characterize the failures of the Gallager A decoder with the help of fixed points. We also introduce the notions of trapping sets, failure sets and critical number. In Section III we investigate the relation between error correction capacity and girth of the code. We extend the results to bit flipping algorithm in Section IV and conclude in Section V.

## II. DECODING ALGORITHMS AND TRAPPING SETS

### A. Graphical Representations of LDPC Codes

LDPC codes [1] are a class of linear block codes which can be defined by sparse bipartite graphs [9]. Let  $\mathcal{G}$  be a bipartite graph with two sets of nodes:  $n$  variable nodes and  $m$  check nodes. The check

nodes (variable nodes) connected to a variable node (check node) are referred to as its neighbors. The degree of a node is the number of its neighbors. This graph defines a linear block code  $\mathcal{C}$  of length  $n$  and dimension at least  $n - m$  in the following way: The  $n$  variable nodes are associated to the  $n$  coordinates of codewords. A vector  $\mathbf{v} = (v_1, v_2, \dots, v_n)$  is a codeword if and only if for each check node, the sum of its neighbors is zero. Such a graphical representation of an LDPC code is called the Tanner graph [10] of the code. The adjacency matrix of  $\mathcal{G}$  gives  $H$ , a parity check matrix of  $\mathcal{C}$ . An  $(n, \gamma, \rho)$  regular LDPC code has a Tanner graph with  $n$  variable nodes each of degree  $\gamma$  (column weight) and  $n\gamma/\rho$  check nodes each of degree  $\rho$  (row weight). This code has length  $n$  and rate  $r \geq 1 - \gamma/\rho$  [9]. In the rest of the paper we consider codes in the  $(3, \rho)$ ,  $\rho > 3$ , regular LDPC code ensemble. Note that the column weight and row weight are also referred to as left degree and right degree in literature. It should also be noted that the Tanner graph is not uniquely defined by the code and when we say the Tanner graph of an LDPC code, we only mean one possible graphical representation. The girth  $g$  is the length of the shortest cycle in  $\mathcal{G}$ . In this paper,  $\bullet$  represents a variable node,  $\square$  represents an even degree check node and  $\blacksquare$  represents an odd degree check node.

### B. Hard Decision Decoding Algorithms

Gallager in [1] proposed two simple binary message passing algorithms for decoding over the binary symmetric channel (BSC); Gallager A and Gallager B. See [9] for a detailed description of 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 (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  $j^{\text{th}}$  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 all messages from  $v$  except to  $c$ ,  $\omega_j(:, c)$  denote the set of all messages to  $c$ .  $\omega_j(:, \setminus v, c)$ ,  $\varpi_j(c, :)$ ,  $\varpi_j(c, : \setminus v)$ ,  $\varpi_j(:, v)$  and  $\varpi_j(:, \setminus c, v)$  are defined similarly.

The Gallager A algorithm can be defined as follows.

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

At the end of each iteration, an estimate of each variable node is made based on the incoming messages and possibly the received value. The decoded word at the end of  $j^{\text{th}}$  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  $\mathbf{r}^{(M)}$ .

*A Note on the Decision Rule:* Different rules to estimate a variable node after each iteration are possible and it is likely that changing the rule after certain iterations may be beneficial. However, the analysis of various scenarios is beyond the scope of this paper. For column-weight-three codes only two rules are possible.

- Decision Rule A: if all incoming messages to a variable node from neighboring checks are equal, set the variable node to that value; else set it to received value
- Decision Rule B: set the value of a variable node to the majority of the incoming messages; majority always exists since the column-weight is three

We adopt Decision Rule A throughout this paper.

### C. Trapping Sets of Gallager A Algorithm

We now characterize failures of the Gallager A decoder using fixed points and trapping sets [11]. Consider an LDPC code of length  $n$  and let  $\mathbf{x}$  be the binary vector which is the input to the Gallager A decoder. Let  $\mathcal{S}(\mathbf{x})$  be the support of  $\mathbf{x}$ . The support of  $\mathbf{x}$  is defined as the set of all positions  $i$  where  $\mathbf{x}(i) \neq 0$ . Without loss of generality, we assume that the all zero codeword is sent over BSC and that

the input to the decoder is the error vector. Hence, throughout this paper a message of 1 is alternatively referred to as an incorrect message, a received value of 1 is referred to as an initial error.

*Definition 1:* [11] A decoder failure is said to have occurred if the output of the decoder is not equal to the transmitted codeword.

*Definition 2:*  $\mathbf{x}$  is called a *fixed point* if

$$\omega_j(v, c) = \mathbf{x}(v), \quad \forall j > 0$$

That is, the message passed from variable nodes to check nodes along the edges are the same in every iteration. Since the outgoing messages from variable nodes are same in every iteration, it follows that the incoming messages from check nodes to variable nodes are also same in every iteration and so is the estimate of a variable after each iteration. In fact, the estimate after each iteration coincides with the received value. It is clear from above definition that if the input to the decoder is a fixed point, then the output of the decoder is the same fixed point.

*Definition 3:* [12] Let  $\mathbf{x}$  be a fixed point. Then  $\mathcal{S}(\mathbf{x})$  is known as a trapping set. A  $(V, C)$  trapping set  $\mathcal{T}$  is a set of  $V$  variable nodes whose induced subgraph has  $C$  odd degree checks.

*Theorem 1:* Let  $\mathcal{C}$  be a code in the ensemble of  $(3, \rho)$  regular LDPC codes. Let  $\mathcal{T}$  be a set consisting of  $V$  variable nodes with induced subgraph  $\mathcal{I}$ . Let the checks in  $\mathcal{I}$  be partitioned into two disjoint subsets;  $\mathcal{O}$  consisting of checks with odd degree and  $\mathcal{E}$  consisting of checks with even degree. Let  $|\mathcal{O}| = C$  and  $|\mathcal{E}| = S$ .  $\mathcal{T}$  is a trapping set iff : (a) Every variable node in  $\mathcal{I}$  is connected to at least two checks in  $\mathcal{E}$  and (b) No two checks of  $\mathcal{O}$  are connected to a variable node outside  $\mathcal{I}$ .

*Proof:* See Appendix I ■

We note that Theorem 1 is a consequence of Fact 3 in [11]. We also remark that Theorem 1 can be extended to higher column weight codes but in this paper we restrict our attention to column-weight-three codes.

If the variable nodes corresponding to a trapping set are in error, then a decoder failure occurs. However, not all variable nodes corresponding to trapping set need to be in error for a decoder failure to occur.

*Definition 4:* [12] The minimal number of variable nodes that have to be initially in error for the decoder to end up in the trapping set  $\mathcal{T}$  will be referred to as *critical number* for that trapping set.

*Definition 5:* A set of variable nodes which if in error lead to a decoding failure is known as a *failure set*.

*Remarks*

- 1) To “end up” in a trapping set  $\mathcal{T}$  means that, after a possible finite number of iterations, the decoder will be in error, on at least one variable node from  $\mathcal{T}$ , at every iteration [11].
- 2) The notion of a failure set is more fundamental than a trapping set. However, from the definition, we cannot derive necessary and sufficient conditions for a set of variable nodes to form a failure set.
- 3) A trapping set is a failure set. Subsets of trapping sets can be failure sets. More specifically, for a trapping set of size  $V$ , there exists at least one subset of size equal to the critical number which is a failure set.
- 4) The critical number of a trapping set is not fixed. It depends on the outside connections of checks in  $\mathcal{E}$ . However, the maximum value of critical number of a  $(V, C)$  trapping set is  $V$ .

*Example 1:* Fig.1(a) shows a subgraph induced by a set of three variable nodes  $\{v_1, v_2, v_3\}$ . If no two odd degree check nodes from  $\{c_4, c_5, c_6\}$  are connected to a variable outside the subgraph, then by Theorem 1, Fig.1(a) is a  $(3, 3)$  trapping set. On the other hand, if two odd degree checks, say  $c_5$  and  $c_6$ , are connected to another variable node, say  $v_4$ , the subgraph resembles Fig. 1(b). Assuming no other connections, Fig.1(b) is a  $(4, 2)$  trapping set. We make the following observations:

- 1) The three variable nodes in a  $(3, 3)$  trapping set form a six cycle. However, not all six cycles are  $(3, 3)$  trapping sets. Apart from the subgraph induced by variable nodes, the outside connections should be known to determine whether a given subgraph is a trapping set or not. The  $(4, 2)$  trapping set in Fig.1(b) illustrates this point.
- 2) The critical number of a  $(3, 3)$  trapping set is three. There exist  $(4, 2)$  trapping sets with critical

number three and it is highly unlikely that a  $(4, 2)$  trapping set does not contain a failure set of size three. However, we can show by a counterexample that this is indeed possible.

- 3) A  $(V, C)$  trapping set is not unique i.e., two trapping sets with same  $V$  and  $C$  can have different underlying topological structures (induced subgraphs). So, when we talk of a trapping set, we refer to a specific topological structure. In this paper, the induced subgraph is assumed to be known from the context.
- 4) To avoid a trapping set in a code, it is sufficient to avoid topological structures isomorphic to the subgraph induced by the trapping set. For example to avoid  $(3, 3)$  trapping sets of Fig.1(a), it is sufficient to avoid six cycles. It is possible that a code has six cycles but no  $(3, 3)$  trapping sets. In this case all six cycles are part of  $(4, 2)$  or other trapping sets.

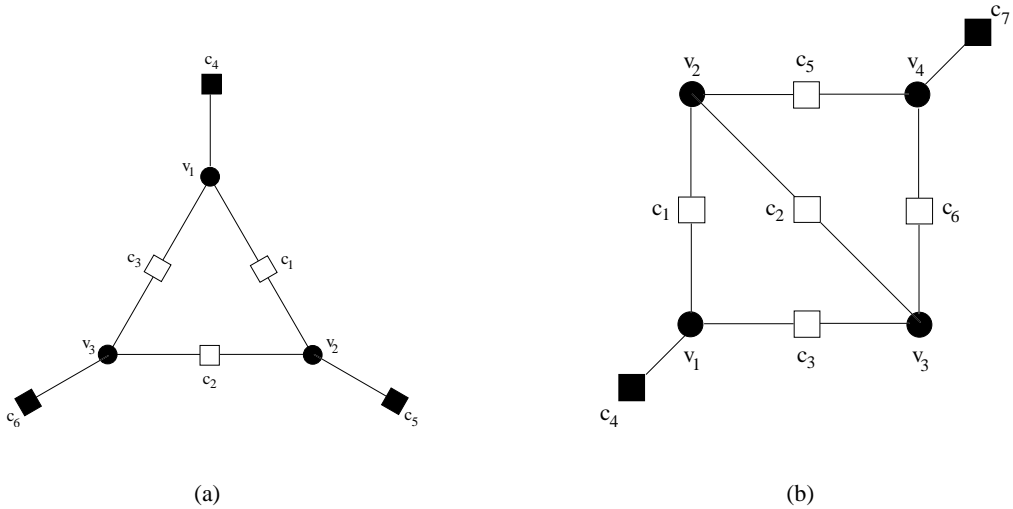


Fig. 1. Examples of trapping sets (a) a  $(3, 3)$  trapping set (b) a  $(4, 2)$  trapping set

### III. ERROR CORRECTION CAPABILITY AND GIRTH OF THE CODE

Burshtein and Miller in [5] applied expander based arguments to message passing algorithms. They analyzed ensembles of irregular graphs and showed that if the degree of each variable node is at least five, then message passing algorithms can correct a fraction of errors. Codes with column weight three and four cannot achieve the expansion required for these arguments. Recently, Burshtein in [8] developed a new technique to investigate the error correction capability of regular LDPC codes and showed that at

sufficiently large block lengths, almost all codes with column weight four are also capable of correcting a fraction of errors under bit flipping algorithm. For column-weight-three codes he notes that such a result cannot be proved. This is because a non negligible fraction of codes have parallel edges in their Tanner graphs and such codes cannot correct a single worst case error.

In this paper we prove a stronger result by showing that for any given  $\alpha > 0$ , at sufficiently large block lengths  $n$ , no code in the  $\mathcal{C}^n(3, \rho)$  ensemble can correct all  $\alpha n$  or fewer errors under Gallager A algorithm and show that this holds for the bit flipping algorithm also.

*Lemma 1:* [8] A code whose Tanner graph has parallel edges cannot correct a single worst case error.

*Proof:* See [8]. The proof is for bit flipping algorithm, but also applies to Gallager A algorithm. ■

*Lemma 2:* Let  $\mathcal{C}$  be an  $(n, 3, \rho)$  regular LDPC code with girth  $g = 4$ . Then  $\mathcal{C}$  has at least one failure set of size two or three.

*Proof:* See Appendix II. ■

*Lemma 3:* Let  $\mathcal{C}$  be an  $(n, 3, \rho)$  regular LDPC code with girth  $g = 6$ . Then  $\mathcal{C}$  has least one failure set of size three or four.

*Proof:* Since  $g = 6$ , there is at least one six cycle. Without loss of generality, we assume that  $\{v_1, v_2, v_3\}$  together with the three even degree checks  $\{c_1, c_2, c_3\}$  and the three odd degree checks  $\{c_4, c_5, c_6\}$  form a six cycle as in Fig.1(a). If no two checks from  $\{c_4, c_5, c_6\}$  are connected to a variable node, then  $\{v_1, v_2, v_3\}$  is a  $(3, 3)$  trapping set and hence a failure set of size three. On the contrary, assume that  $\{v_1, v_2, v_3\}$  do not form a  $(3, 3)$  trapping set. Then there exists  $v_4$  which is connected to at least two checks from  $\{c_4, c_5, c_6\}$ . If  $v_4$  is connected to all the three checks,  $\{v_1, v_2, v_3, v_4\}$  is a codeword of weight four and it is easy to see that  $\{v_1, v_2, v_3\}$  is a failure set. Now assume that  $v_4$  is connected to only two checks from  $\{c_4, c_5, c_6\}$ . Without loss of generality, let the two checks be  $c_5$  and  $c_6$ . Let the third check connected to  $v_4$  be  $c_7$  as shown in Fig.1(b). If  $c_4$  and  $c_7$  are not connected to a common variable node then  $\{v_1, v_2, v_3, v_4\}$  is a  $(4, 2)$  trapping set and hence a failure set of size four. If  $c_4$  and  $c_7$  are connected to say  $v_5$ , we have two possibilities: (a) The third check is  $c_8$  and (b) The third check of  $v_5$  is  $c_2$  (the third



check cannot be  $c_1$  or  $c_3$  as this would introduce a four cycle). We claim that in both cases  $\{v_1, v_2, v_3, v_4\}$  is a failure set. The two cases are discussed below.

Case (a): Let  $S(\mathbf{r}) = \{v_1, v_2, v_3, v_4\}$ .

$$\omega_1(v, :) = \begin{cases} 1, & v \in \{v_1, v_2, v_3, v_4\} \\ 0, & \text{otherwise} \end{cases}$$

The messages in the second half of first iteration are,

$$\varpi_1(c_1, v) = \begin{cases} 1, & v \in \{v_1, v_2\} \\ 0, & \text{otherwise} \end{cases}$$

Similar equations hold for  $c_2, c_3, c_5, c_6$ . For  $c_4$  we have

$$\varpi_1(c_4, v) = \begin{cases} 0, & v = v_1 \\ 1, & \text{otherwise} \end{cases}$$

Similar equations hold for  $c_7$ . At the end of first iteration, we note that  $v_2$  and  $v_3$  receive all incorrect messages,  $v_1, v_4$  and  $v_5$  receive two incorrect messages and all other variable nodes receive at most one incorrect message. We therefore have  $\mathbf{r}^{(1)} = \mathbf{r}$  and  $S(\mathbf{r}^{(1)}) = \{v_1, v_2, v_3, v_4\}$ . The messages sent by variable nodes in the second iteration are,

$$\begin{aligned} \omega_2(v, :) &= 1, v \in \{v_1, v_2, v_3, v_4\} \\ \omega_2(v_5, c_8) &= 1, \\ \omega_2(v_5, \{c_4, c_7\}) &= 0, \\ \omega_2(v, :) &= 0, v \in \{v_1, \dots, v_n\} \setminus \{v_1, v_2, v_3, v_4, v_5\}. \end{aligned}$$

The messages passed in second half of second iteration are same as in second half of first iteration, except that  $\varpi(c_8, : \setminus v_5) = 1$ . At the end of second iteration, we note that  $v_2$  and  $v_3$  receive all incorrect messages,  $v_1, v_4$  and  $v_5$  receive two incorrect messages and all other variable nodes receive at most one incorrect message. The situation is same as at the end of first iteration. The algorithm runs for  $M$  iterations and the decoder outputs  $\mathbf{r}^{(M)} = \mathbf{r}$  which implies that  $\{v_1, v_2, v_3, v_4\}$  is a failure set.

Case (b): The proof is along the same lines as for Case (a). The messages for first iteration are the same. The messages in the first half of second iteration are,

$$\begin{aligned}\omega_2(v, :) &= 1, v \in \{v_1, v_2, v_3, v_4\} \\ \omega_2(v_5, c_2) &= 1, \\ \omega_2(v_5, \{c_4, c_7\}) &= 0, \\ \omega_2(v, :) &= 0, v \in \{v_1, \dots, v_n\} \setminus \{v_1, v_2, v_3, v_4, v_5\}.\end{aligned}$$

The messages passed in second half of second iteration are same as in second half of first iteration, except that  $\varpi(c_2, \setminus \{v_2, v_3, v_5\}) = 1$  and  $\varpi(c_2, \{v_2, v_3, v_5\}) = 0$ . At the end of second iteration,  $v_1, v_2, v_3, v_4$  and  $v_5$  receive two incorrect messages and all other variable nodes receive at most one incorrect message and hence  $\mathbf{r}^{(2)} = \mathbf{r}$ . The messages passed in first half of third iteration (and therefore subsequent iterations) are same as the messages passed in first half of second iteration. The algorithm runs for  $M$  iterations and the decoder outputs  $\mathbf{r}^{(M)} = \mathbf{r}$  which implies that  $\{v_1, v_2, v_3, v_4\}$  is a failure set. ■

*Lemma 4:* Let  $\mathcal{C}$  be an  $(n, 3, \rho)$  regular LDPC code with girth  $g = 8$ . Then  $\mathcal{C}$  has at least one failure set of size four or five.

*Proof:* See Appendix II. ■

*Remark:* It might be possible that Lemmas 2–4 can be made stronger by further analysis, i.e., it might be possible to show that a code with girth four has a failure set of size two, a code with girth six has failure set of size three and a code with girth eight has a failure set of size four. However, these weaker lemmas are sufficient to establish the main theorem.

*Lemma 5:* Let  $\mathcal{C}$  be an  $(n, 3, \rho)$  regular LDPC code with girth  $g \geq 10$ . Then the set of variable nodes  $\{v_1, v_2, \dots, v_{g/2}\}$  involved in the shortest cycle is a trapping set of size  $g/2$ .

*Proof:* Since  $\mathcal{C}$  has girth  $g$ , there is at least one cycle of length  $g$ . Without loss of generality, assume that  $\{v_1, v_2, \dots, v_{g/2}\}$  form a cycle of minimum length as shown in Fig.2. Let the even degree checks be  $\mathcal{E} = \{c_1, c_2, \dots, c_{g/2}\}$  and the odd degree checks be  $\mathcal{O} = \{c_{g/2+1}, c_{g/2+2}, \dots, c_g\}$ . Note that each variable

node is connected to two checks from  $\mathcal{E}$  and one check from  $\mathcal{O}$  and  $c_{g/2+i}$  is connected to  $v_i$ . We claim that no two checks from  $\mathcal{O}$  can be connected to a common variable node.

The proof is by contradiction. Assume  $c_i$  and  $c_j$  ( $g/2 + 1 \leq i < j \leq g$ ) are connected to a variable node  $v_{ij}$ . Then  $\{v_i, \dots, v_j, v_{ij}\}$  form a cycle of length  $2(j - i + 2)$  and  $\{v_j, \dots, v_{g/2}, v_1, \dots, v_i, v_{ij}\}$  form a cycle of length  $2(g/2 - j + i + 2)$ . Since  $g \geq 10$ ,

$$\min(2(j - i + 2), 2(g/2 - j + i + 2)) < g.$$

This implies that there is a cycle of length less than  $g$ , which is a contradiction as the girth of the graph is  $g$ .

By Theorem 1,  $\{v_1, v_2, \dots, v_{g/2}\}$  is a trapping set. ■

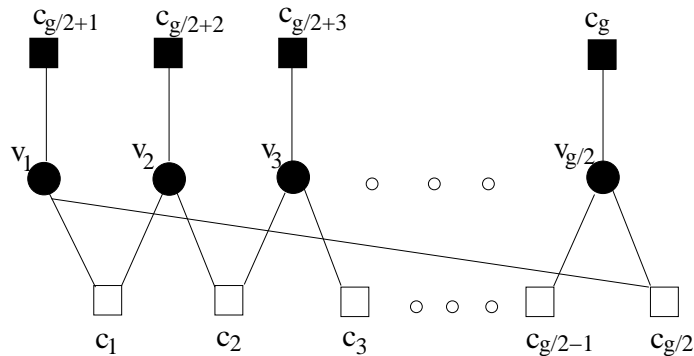


Fig. 2. Illustration of a cycle of length  $g$

*Corollary 1:* For a code to correct all  $k \geq 5$  or fewer errors, it is necessary to avoid all cycles up to length  $2k$ .

We now state and prove the main theorem.

*Theorem 2:* Consider the standard  $(3, \rho)$  regular LDPC code ensemble. Let  $\alpha > 0$ . Let  $N$  be the smallest integer satisfying

$$\alpha N > 2 \left( \frac{\log N}{\log(2(\rho - 1))} + 1 \right)$$

$$\alpha N \geq 5.$$

Then, for  $n > N$ , no code in the  $\mathcal{C}^n(3, \rho)$  ensemble can correct all  $\alpha n$  or fewer errors.

*Proof:* First observe that for any  $n > N$ , we have

$$\alpha n > 2 \left( \frac{\log n}{\log(2(\rho - 1))} + 1 \right). \quad (1)$$

From [Theorem C.1 [1]] and [Lemma C.1 [1]], we have the girth  $g$  of any code in  $\mathcal{C}^n(3, \rho)$  is bounded by

$$g \leq 4 \left( \frac{\log n}{\log(2(\rho - 1))} + 1 \right) \quad (2)$$

For  $n > N$ , Equations (1) and (2) imply that for any code in the  $\mathcal{C}^n(3, \rho)$  ensemble, the girth is bounded by

$$g < 2\alpha n.$$

The result now follows from Corollary 1. ■

#### IV. EXTENSION TO THE BIT FLIPPING ALGORITHM

The bit flipping algorithm does not belong to the class of message passing algorithms. However, the definitions from Section II and the results from Section III can be generalized to the parallel bit flipping algorithm [4]. Without loss of generality we assume that the all zero codeword is sent. We begin with a few definitions.

*Definition 6:* [4] A variable node is said to be corrupt if it is different from its original sent value. In our case, a variable node is corrupt if it is 1. A check node is said to be satisfied if it is connected to even number of corrupt variables and unsatisfied otherwise.

*Definition 7:* Let  $\mathbf{r}$  be the input to the parallel bit flipping decoder.  $S(\mathbf{r})$  is a trapping set for bit flipping algorithm if the set of corrupt variables after every iteration is  $S(\mathbf{r})$ .

*Theorem 3:* Let  $\mathcal{T}$  be a set of variable nodes satisfying the conditions of Theorem 1. Then  $\mathcal{T}$  is a trapping set for the bit flipping algorithm.

*Proof:* Let  $S(\mathbf{r}) = \mathcal{T}$ . Then  $\mathcal{T}$  is the set of corrupt variable nodes. Observe that a variable flips if it is connected to at least two unsatisfied checks. Since no variable is connected to two unsatisfied checks, the set of corrupt variable nodes is unchanged and by definition  $\mathcal{T}$  is a trapping set. ■

We note that Theorem 3 is also a consequence of Fact 3 from [11].

*Corollary 2:* A trapping set for Gallager A is also a trapping set for bit flipping algorithm.

It can be shown that Lemmas 1–5 and Theorem 2 also hold for the bit flipping algorithm.

## V. CONCLUSION

In this paper we have investigated the error correction capability of column-weight-three codes under Gallager A and extended the results to bit flipping algorithm. Future work includes investigation of sufficient conditions to correct a given number of errors for column-weight-three as well as higher column weight codes.

## APPENDIX I

*Proof of Theorem 1:* Let  $\mathbf{r}$  be the input to the decoder with  $S(\mathbf{r}) = \mathcal{T}$ . Then,

$$\omega_1(v, :) = \begin{cases} 1, & v \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases}$$

Let a check node  $c_o \in \mathcal{O}$ . Then,

$$\varpi_1(c_o, v) = \begin{cases} 0, & v \in \mathcal{T} \\ 1, & \text{otherwise} \end{cases}$$

Let a check node  $c_e \in \mathcal{E}$ . Then,

$$\varpi_1(c_e, v) = \begin{cases} 1, & v \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases}$$

For any other check node  $c$ ,  $\varpi_1(c, v) = 0$ . By the conditions of the theorem, at the end of first iteration, any  $v \in \mathcal{T}$  receives at least two 1's and any  $v \notin \mathcal{T}$  receives at most one 1. So, we have

$$\omega_2(v, :) = \begin{cases} 1, & v \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases}$$

By definition,  $\mathcal{T}$  is a trapping set.

To see that the conditions stated are necessary observe that for a variable node to send the same messages as in the first iteration, it should receive at least two messages which coincide with the received value.

■

## APPENDIX II

*Proof of Lemma 2:* Let  $\{v_1, v_2\}$  be the variable nodes that form a four cycle with even degree checks  $\{c_1, c_2\}$  and odd degree checks  $\{c_3, c_4\}$ . If  $c_3$  and  $c_4$  are not connected to a common variable node, then  $\{v_1, v_2\}$  is a  $(2, 2)$  trapping set and hence a failure set of size two. Now assume that  $c_3$  and  $c_4$  are connected to a common variable node  $v_3$ . Then,  $\{v_1, v_2, v_3\}$  is a  $(3, 1)$  trapping set and therefore a failure set of size three.

■

*Proof of Lemma 4:* Let  $\mathcal{T}_1 = \{v_1, v_2, v_3, v_4\}$  be the variable nodes that form an eight cycle (see Fig.3(a)). If no two checks from  $\{c_5, c_6, c_7, c_8\}$  are connected to a common variable node, then  $\mathcal{T}_1$  is a  $(4, 4)$  trapping set and hence a failure set of size four. On the other hand, if  $\mathcal{T}_1$  is not a trapping set, then there must be at least one variable node which is connected to two checks from  $\{c_5, c_6, c_7, c_8\}$ . Assume that  $c_5$  and  $c_7$  are connected to  $v_5$  and the third check of  $v_5$  is  $c_9$  (see Fig.3(b)). We claim that  $\mathcal{T}_2 = \mathcal{T}_1 \cup \{v_5\}$  is a failure set. Let  $\mathcal{E}$  and  $\mathcal{O}$  be as defined in Theorem 1.

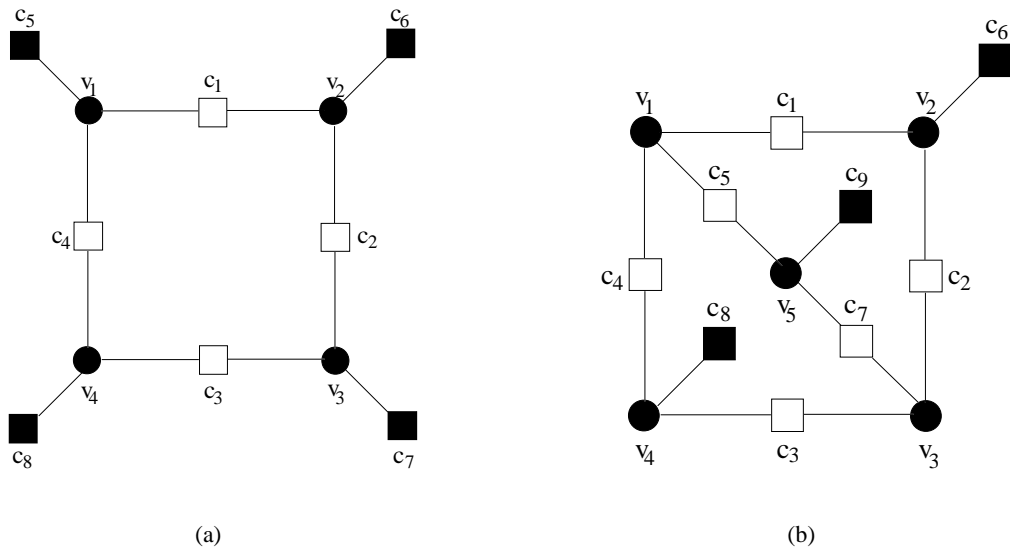


Fig. 3. Subgraphs isomorphic to (a) a  $(4, 4)$  trapping set (b) a  $(5, 3)$  trapping set

**Case 1:** No two checks from  $\mathcal{O} = \{c_6, c_8, c_9\}$  are connected to a common variable node. Then  $\mathcal{T}_2$  is a  $(5, 3)$  trapping set and hence a failure set of size five.

**Case 2:** All the three checks in  $\mathcal{O}$  are connected to a common variable node, say  $v_6$ . Then  $\mathcal{T}_2 \cup \{v_6\}$  is a codeword of weight six and it is easy to see that  $\mathcal{T}_2$  is a failure set.

**Case 3:** There are variable nodes connected to two checks from  $\mathcal{O}$ . There can be at most two such variable nodes (if there are three such variable nodes, they will form a cycle of length less than or equal to six violating the condition that the graph has girth eight). Note that if  $S(\mathbf{r}) = \mathcal{T}_2$ , the decoder has a chance of correcting only if a check node in  $\mathcal{E}$  receives an incorrect message from a variable node outside  $\mathcal{T}_2$  in some  $j^{\text{th}}$  iteration. We now prove that this is not possible. Indeed in the first iteration

$$\omega_1(v, :) = \begin{cases} 1, & v \in \mathcal{T}_2 \\ 0, & \text{otherwise} \end{cases}$$

By similar arguments as in the proof for Theorem 1, it can be seen that the only check nodes which send incorrect messages to variable nodes outside  $\mathcal{T}_2$  are  $c_6, c_8$  and  $c_9$ . There are now two subcases.

**Subcase 1:** There is one variable node connected to two checks from  $\mathcal{O}$ . Let  $v_6$  be connected to  $c_6$  and  $c_8$ . It can be seen that the third check connected to  $v_6$  cannot belong to  $\mathcal{E}$  as this would violate the girth condition. So, let the third check be  $c_{10}$ . In the first half of second iteration, we have

$$\omega_2(v, c) = \begin{cases} 1, & v \in \mathcal{T}_2 \text{ or } (v, c) = (v_6, c_{10}) \\ 0, & \text{otherwise} \end{cases}$$

The only check nodes which send incorrect messages to variable nodes outside  $\mathcal{T}_2$ , are  $c_6, c_8, c_9$  and  $c_{10}$ . The variable node  $v_6$  is connected to  $c_6$  and  $c_8$ . If  $c_9$  and  $c_{10}$  are not connected to any common variable node, we are done. On the other hand, let  $c_9$  and  $c_{10}$  be connected to a variable node, say  $v_7$ . The third check of  $v_7$  cannot be in  $\mathcal{E}$ . Proceeding as in the case of proof for Lemma 3, we can prove that  $\mathcal{T}_2$  is a failure set by observing that there cannot be a variable node outside  $\mathcal{T}_2$  which sends an incorrect message to a check in  $\mathcal{E}$ .

**Subcase 2:** There are two variable nodes connected to two checks from  $\mathcal{O}$ . Let  $c_6$  and  $c_8$  be connected to  $v_6$  and  $c_6$  and  $c_9$  connected to  $v_7$ . Proceeding as above, we can conclude that  $\mathcal{T}_2$  is a failure set.

#### ACKNOWLEDGMENT

This work is funded by NSF under Grant CCF-0634969, ITR-0325979 and INSIC-EHDR program. The authors would like to thank Anantharaman Krishnan for illustrations.

#### REFERENCES

- [1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [2] 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.
- [3] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity for Gallager low-density codes," *Problems of Information Transmission*, vol. 11, pp. 18–28, 1976.
- [4] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.
- [5] D. Burshtein and G. Miller, "Expander graph arguments for message-passing algorithms," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 782–790, Feb. 2001.
- [6] J. Feldman, T. Malkin, R. A. Servedio, C. Stein, and M. J. Wainwright, "LP decoding corrects a constant fraction of errors," *IEEE Trans. Inform. Theory*, vol. 53, no. 1, pp. 82–89, Jan. 2007.
- [7] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 3, pp. 954–972, March 2005.
- [8] D. Burshtein, "On the error correction of regular LDPC codes using the flipping algorithm," in *International Symposium on Information Theory*, June 24–29 2007, pp. 226–230.
- [9] 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.
- [10] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sept. 1981.
- [11] T. J. Richardson, "Error floors of LDPC codes," in *41st Annual Allerton Conf. on Communications, Control and Computing*, 2003, pp. 1426–1435.
- [12] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *International Conference on Communications*, vol. 3, June 11–15 2006, pp. 1089–1094.