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

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

***Abstract*—In this paper, the error-correction capability of column-weight-three low-density parity-check (LDPC) codes when decoded using the Gallager A algorithm is investigated. It is proved that a necessary condition for a code to correct all error patterns with up to $k \geq 5$ errors is to avoid cycles of length up to $2k$ in its Tanner graph. As a consequence of this result, it is shown 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 αn or fewer errors. The results are extended to the bit flipping algorithms.**

***Index Terms*—Error-correction capability, Gallager A algorithm, low-density parity-check (LDPC) codes, Tanner graph, trapping sets.**

I. Introduction

GALLAGER [1] showed that for $\gamma \geq 3$ and $\rho > \gamma$, there exist (n, γ, ρ) 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 [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 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 [4] used expander graph arguments to analyze the serial and parallel bit flipping algorithms. Burshtein and Miller [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 six. Feldman *et al.* [6] showed that the linear programming decoder [7] is also capable of correcting a fraction of errors. Recently, Burshtein [8] showed that regular codes with variable nodes of degree four are capable of correcting a linear fraction of errors under the parallel 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 [1]. We analyze decoding failures using the notions of trapping sets and inducing sets and prove that a code with a Tanner graph [9] of girth $g \geq 10$ cannot correct all error patterns with up to $g/2$ 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 α fraction of errors. This result settles the problem of error-correction capability of column-weight-three LDPC codes. The rest of this 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 trapping sets [10] and inducing sets. We also introduce the notions of failure sets, fixed sets, and fixed points. In Section III, we investigate the relation between error-correction capability and girth of the Tanner graph of the code. We extend the results to the bit flipping algorithms in Section IV and conclude in Section V.

Manuscript received October 18, 2007; revised October 27, 2008. Current version published April 22, 2009. This work was supported by the National Science Foundation (NSF) under Grants CCF-0634969, ITR-0325979, and INSIC-EHDR.

The authors 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).

Communicated by T. J. Richardson, Associate Editor for Coding Theory.

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 G be a bipartite graph with two sets of nodes: the set of variable nodes V with $|V| = n$ and the set of check nodes C with $|C| = m$. 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 with the n coordinates of codewords. A vector $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ is a codeword if and only if for each check node, the modulo two sum of its neighbors is zero. Such a graphical representation of an LDPC code is called the Tanner graph [9] of the code. The adjacency matrix of G gives H, a parity check matrix of $\mathcal{C}$. An (n, γ, ρ) regular LDPC code has a Tanner graph with n variable nodes each of degree γ (column weight) and $n\gamma/\rho$ check nodes each of degree ρ (row weight). This code has length n and rate $r \geq 1 - \gamma/\rho$[1]. In the rest of this paper, we consider codes in the $(3, \rho)$, $\rho > 3$, regular LDPC code ensemble. Note that in the literature, the column weight and row weight are also referred to as left degree and right degree, respectively. 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 G. In this paper, ● represents a variable node, □ represents an even-degree check node, and ■ represents an odd-degree check node.

B. Hard Decision Decoding Algorithms

Gallager [1] proposed two simple binary message passing algorithms for decoding over the binary symmetric channel (BSC): Gallager A and Gallager B. See [11] 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 the variable nodes sending messages to their neighboring check nodes (first half of the iteration) and ends by the check nodes sending messages to their neighboring 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 jth iteration and $\varpi_j(c,v)$ denote the message passed by a check node c to its neighboring variable node v. Let $\mathcal{N}(c)$ denote the set of neighbors of check node c. Additionally, let $\omega_j(v,:)$ denote the set of all messages from v, $\omega_j(v,:\backslash c)$ denote the set of all messages from v except to c, and $\omega_j(:,c)$ denote the set of all messages to c. The terms $\omega_j(:\backslash v,c)$, $\varpi_j(c,:)$, $\varpi_j(c,:\backslash v)$, $\varpi_j(:,v)$, and $\varpi_j(:\backslash c,v)$ are defined similarly. The Gallager A algorithm can be defined as follows:

$$\omega_1(v,c) = r_v$$

$$\omega_j(v,c) = \begin{cases} 1, & \text{if } \varpi_{j-1}(:\backslash c,v) = \{1\} \\ 0, & \text{if } \varpi_{j-1}(:\backslash c,v) = \{0\} \\ r_v, & \text{otherwise} \end{cases}$$

$$\varpi_j(c,v) = \left(\sum_{u\in\mathcal{N}(c)\backslash v} \omega_j(u,c) \right) \bmod 2.$$

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 codeword estimate of the decoder at the end of jth 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 $\mathbf{r}^{(M)}$.

1) 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 check nodes are equal, set the variable node to that value; else set it to its 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 but note that all the results can be established for Decision Rule B also.

C. Trapping Sets of Gallager A Algorithm

We now characterize the failures of the Gallager A decoder using the notions of trapping sets [10], inducing sets, and fixed sets. We adopt the definitions of the terms eventually correct variable nodes, trapping sets, and failure sets from [10]. For output symmetric channels, without loss of generality, we can assume that the all zero codeword is transmitted [2]. We make this assumption throughout this paper. The support of a vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, denoted by $\operatorname{supp}(\mathbf{y})$, is defined as the set of all variable nodes v for which $y_v \neq 0$. Since the transmitted codeword is assumed to be the all zero codeword, the support of the input to the decoder is simply the set of variable nodes flipped by the channel (or in other words the set of variable nodes initially in error).

Let $\mathbf{y}$ denote the input to the Gallager A decoder and $\mathbf{y}^{(l)}$ denote the codeword estimate of the decoder at the end of lth iteration. A variable node v is said to be *eventually correct* if there exists a positive integer l_c such that for all $l_c \leq l \leq M$, $y_v^l = 0$. For an input $\mathbf{y}$, the *failure set* $\mathbf{T}(\mathbf{y})$ is defined as the set of variable nodes that are not eventually correct. The decoding on the input $\mathbf{y}$ is successful if and only if $\mathbf{T}(\mathbf{y}) = \emptyset$. If $\mathbf{T}(\mathbf{y}) \neq \emptyset$, then we say that $\mathbf{T}(\mathbf{y})$ is a *trapping set* and $\operatorname{supp}(\mathbf{y})$ is an *inducing set*. The size of an inducing set is its cardinality. Since the failure sets of two different input vectors can be the same trapping set, we denote a trapping set simply by $\mathcal{T}$. A trapping set $\mathcal{T}$ is said to be an (a,b) trapping set if it has a variable nodes and b odd-degree check nodes in the subgraph induced by $\mathcal{T}$.

Remarks:

1) If the Tanner graph G of a code $\mathcal{C}$ has an inducing set of size k, then the code cannot correct all error patterns with up to k errors. We use this fact as the basis for our main theorem.
2) An (a,b) trapping set is not unique, i.e., two trapping sets with the same a and b 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.
3) In order to show that a given set of variable nodes $\mathcal{T}$ is a trapping set, we should exhibit a vector $\mathbf{y}$ for which $\mathbf{T}(\mathbf{y}) = \mathcal{T}$.

The definitions above do not provide the necessary and sufficient conditions for a set of variable nodes to be an inducing set or trapping set. Hence, we define the notion of a *fixed set* for which such conditions can be derived and note that any fixed set is an inducing set and a trapping set.

Definition 1: Let $\mathcal{F}$ be a set of variable nodes and let $\mathbf{y}$ be a vector such that $\operatorname{supp}(\mathbf{y}) = \mathcal{F}$. Consider running the Gallager A decoder with $\mathbf{y}$ as the input. If $\omega_j(v,c) = \mathbf{y}_v, \forall\, j > 0$, then $\mathcal{F}$ is known as a fixed set and $\mathbf{y}$ is known as a fixed point.

That is, the messages passed from variable nodes to check nodes along the edges are the same in every iteration. Since the outgoing messages from variable nodes are the same in every iteration, it follows that the incoming messages from check nodes to variable nodes are also the same in every iteration and so is the estimate of a variable node after each iteration. In fact, the estimate after each iteration coincides with the received value. It is clear from the above definition that if the input to the decoder is a fixed point, then the output of the decoder is the same fixed point. It follows that for transmission over the BSC, if $\mathbf{y}$ is a fixed point and $\mathbf{T}(\mathbf{y}) \neq \emptyset$, then $\mathbf{T}(\mathbf{y}) = \operatorname{supp}(\mathbf{y})$ is a trapping set as well as an inducing set. We now provide necessary and sufficient conditions for a set of variable nodes to be a fixed set.

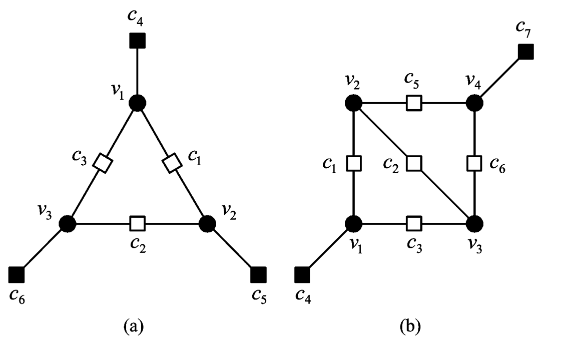

Fig. 1. Subgraphs induced by (a) $(3,3)$ trapping set and (b) $(4,2)$ trapping set.

Theorem 1: Let $\mathcal{C}$ be a code with Tanner graph G in the ensemble of $(3,\rho)$ regular LDPC codes. Let $\mathcal{F}$ be a set of variable nodes with induced subgraph $\mathcal{I}$. Let the check nodes in $\mathcal{I}$ be partitioned into two disjoint subsets: $\mathcal{O}$ consisting of check nodes with odd degree and $\mathcal{E}$ consisting of check nodes with even degree. $\mathcal{F}$ is a fixed set iff: a) every variable node in $\mathcal{I}$ is connected to at least two check nodes in $\mathcal{E}$ and b) no two check nodes of $\mathcal{O}$ are connected to a common variable node outside $\mathcal{I}$.

Proof: See Appendix I $\square$

We note that Theorem 1 is a consequence of Fact 3 from [10].

Since every fixed set is an inducing set, the conditions a) and b) of Theorem 1 are sufficient for a given set of variable nodes to be an inducing set and a trapping set. We illustrate the above definitions with an example.

Example 1: Fig. 1(a) shows a subgraph induced by a set of three variable nodes $\{v_1,v_2,v_3\}$ in a Tanner graph of girth six. 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, $\{v_1,v_2,v_3\}$ is a fixed set, and therefore, an inducing set of size three. If $\mathbf{y}$ with $\text{supp}(\mathbf{y})=\{v_1,v_2,v_3\}$ is the input vector to the Gallager A decoder, then $\mathbf{T}(\mathbf{y})=\{v_1,v_2,v_3\}$. This implies that $\{v_1,v_2,v_3\}$ is a trapping set and according to the terminology $\{v_1,v_2,v_3\}$ is a $(3,3)$ trapping set, i.e., the induced subgraph has three variable nodes and three odd-degree (here degree one) check nodes.

If two odd-degree check nodes, say c_5 and c_6, are connected to another variable node, say v_4, then the subgraph induced by $\{v_1,v_2,v_3,v_4\}$ is depicted in Fig. 1(b). Assuming that the check nodes c_4 and c_7 are not connected to a common variable node, $\{v_1,v_2,v_3,v_4\}$ is a fixed set and an inducing set of size four. If $\mathbf{y}$ with $\text{supp}(\mathbf{y})=\{v_1,v_2,v_3,v_4\}$ is the input vector to the Gallager A decoder, then $\mathbf{T}(\mathbf{y})=\{v_1,v_2,v_3,v_4\}$. Consequently, Fig. 1(b) depicts the subgraph induced by a $(4,2)$ trapping set.

Now, assume that no two check nodes in $\{c_1,\dots,c_7\}$ in Fig. 1(b) are connected to a common variable node in $V\setminus\{v_1,v_2,v_3,v_4\}$. Let $\mathbf{y}$ with $\text{supp}(\mathbf{y})=\{v_1,v_2,v_3\}$ be the input to the decoder. In this case, $\mathbf{T}(\mathbf{y})=\{v_1,v_2,v_3\}$. On the other hand if $\mathbf{y}$ with $\text{supp}(\mathbf{y})=\{v_1,v_2,v_4\}$ is the input to the decoder, then $\mathbf{T}(\mathbf{y})=\{v_1,v_2,v_3,v_4\}$. This illustrates that an inducing set of size three can result in a trapping set of size four. We also note that a fixed set can contain inducing sets as its subsets. In this example, $\{v_1,v_2,v_3,v_4\}$ is a fixed set and its subsets $\{v_1,v_2,v_3\}$, $\{v_2,v_3,v_4\}$, $\{v_1,v_2,v_4\}$, and $\{v_1,v_3,v_4\}$ are inducing sets.

III. Error-Correction Capability and Girth of the Code

Burshtein and Miller [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 six, then the 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 [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 the parallel bit flipping algorithm. For column-weight-three codes, he notes that such a result cannot be proved. This is because a nonnegligible 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 αn or fewer errors under the Gallager A algorithm and show that this holds for the bit flipping algorithms also.

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

Proof: The ensemble of $(3,\rho)$ regular codes consists of codes whose Tanner graphs consist of a variable node connected to a check node via multiple edges. Such a Tanner graph is said to contain parallel edges. In [8], it was shown that such a code cannot correct a single worst case error. The proof is for the parallel bit flipping algorithm, but also applies to the Gallager A algorithm (see [8] for details). $\square$

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

Proof: See Appendix II. $\square$

Lemma 3: Let $\mathcal{C}$ be an $(n,3,\rho)$ regular LDPC code with Tanner graph G of girth $g=6$. Then, $\mathcal{C}$ has least one inducing 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 check nodes $\{c_1,c_2,c_3\}$ and the three odd-degree check nodes $\{c_4,c_5,c_6\}$ form a six cycle as in Fig. 1(a). The check nodes c_4,c_5, and c_6 are distinct, since otherwise the girth would be less than six. If no two check nodes from $\{c_4,c_5,c_6\}$ are connected to a common variable node, then $\{v_1,v_2,v_3\}$ is a fixed set and hence an inducing set of size three. On the contrary, assume that $\{v_1,v_2,v_3\}$ is not a fixed set. Then, there exists a variable node v_4, which is connected to at least two check nodes from $\{c_4,c_5,c_6\}$. If v_4 is connected to all the three check nodes, then $\{v_1,v_2,v_3,v_4\}$ is the support of a codeword of weight four and it is easy to see that $\{v_1,v_2,v_3\}$ is an inducing set. Now assume that v_4 is connected to only two check nodes from $\{c_4,c_5,c_6\}$. Without loss of generality, let the two check nodes be c_5 and c_6. Let the third check node 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 fixed set and hence an inducing set of size four. If c_4 and c_7 are connected to say v_5, we have two possibilities: a) the third check node neighbor of v_5 is c_8 and b) the third check node neighbor of v_5 is c_2 (the third check node neighbor of v_5 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 an inducing set. The two cases are discussed below.

Case a) Let $\operatorname{supp}(\mathbf{y}) = \{v_1, v_2, v_3, v_4\}$. Then

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

The messages in the second half of the 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 the 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. Therefore, we have $\mathbf{y}^{(1)} = \mathbf{y}$ and $\operatorname{supp}(\mathbf{y}^{(1)}) = \{v_1, v_2, v_3, v_4\}$. The messages sent by variable nodes in the second iteration are

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

The messages passed in the second half of the second iteration are the same as in the second half of first iteration, except that $\varpi(c_8, : \backslash v_5) = 1$. At the end of the 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 the same as at the end of first iteration. The algorithm runs for M iterations and the decoder outputs $\mathbf{y}^{(M)} = \mathbf{y}$, which implies that $\{v_1, v_2, v_3, v_4\}$ is an inducing set.

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

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

The messages passed in the second half of the second iteration are the same as in the second half of the first iteration, except that $\varpi(c_2, : \backslash \{v_2, v_3, v_5\}) = 1$ and $\varpi(c_2, \{v_2, v_3, v_5\}) = 0$. At the end of the 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 $\mathbf{y}^{(2)} = \mathbf{y}$. The messages passed in the first half of the third iteration (and therefore subsequent iterations) are the same as the messages passed in the first half of the second iteration. The algorithm runs for M iterations and the decoder outputs $\mathbf{y}^{(M)} = \mathbf{y}$ which implies that $\{v_1, v_2, v_3, v_4\}$ is an inducing set. □

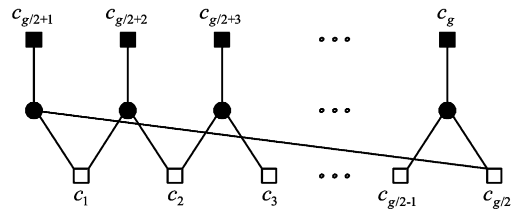

Fig. 2. Illustration of a cycle of length g.

Lemma 4: Let $\mathcal{C}$ be an $(n, 3, \rho)$ regular LDPC code with Tanner graph G of girth $g = 8$. Then, $\mathcal{C}$ has at least one inducing 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 Tanner graph of girth four always has an inducing set of size two, a code with Tanner graph of girth six has an inducing set of size three, and a code with Tanner graph of girth eight has an inducing 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 Tanner graph of girth $g \geq 10$. Then, the set of variable nodes $\{v_1, v_2, \ldots, v_{g/2}\}$ involved in the shortest cycle is a fixed set (as well as an inducing 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 $\{v_1, v_2, \ldots, v_{g/2}\}$ form a cycle of minimum length as shown in Fig. 2. Let the even-degree check nodes be $\mathcal{E} = \{c_1, c_2, \ldots, c_{g/2}\}$ and the odd-degree check nodes be $\mathcal{O} = \{c_{g/2+1}, c_{g/2+2}, \ldots, c_g\}$. Note that each variable node is connected to two check nodes from $\mathcal{E}$ and one check node from $\mathcal{O}$ and $c_{g/2+i}$ is connected to v_i. We claim that no two check nodes from $\mathcal{O}$ can be connected to a common variable node outside $\{v_1, v_2, \ldots, v_{g/2}\}$.

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, $\{v_i, \ldots, v_j, v_{ij}\}$ form a cycle of length $2(j - i + 2)$ and $\{v_j, \ldots, v_{g/2}, v_1, \ldots, v_i, v_{ij}\}$ 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, $\{v_1, v_2, \ldots, v_{g/2}\}$ is a fixed set, and consequently, an inducing set of size $g/2$. It is worth noting that $\{v_1, v_2, \ldots, v_{g/2}\}$ is a $(g/2, g/2)$ trapping set. □

Corollary 1: For a code in the standard $(3, \rho)$ regular LDPC code ensemble to correct all error patterns up to $k \geq 5$ errors, it is necessary to avoid all cycles up to length $2k$ in its Tanner graph representation.

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 any $n > N$, no code in the $\mathcal{C}^n(3, \rho)$ ensemble can correct all αn 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 [1, Th. C.1] and [1, Lemma C.1], we have that 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 $n > N$, (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 Algorithms

The serial and parallel bit flipping algorithms [3], [4] do 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 bit flipping algorithms. Without loss of generality, we assume that the all zero codeword is sent. We begin with a few definitions.

Definition 2 [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 variable nodes and unsatisfied otherwise.

Definition 3: Let $\mathbf{y}$ be the input to the bit flipping decoder (serial or parallel). $\text{supp}(\mathbf{y})$ is a fixed set for the bit flipping algorithms if the set of corrupt variable nodes after every iteration is $\text{supp}(\mathbf{y})$.

Theorem 3: Let $\mathcal{F}$ be a set of variable nodes satisfying the conditions of Theorem 1. Then, $\mathcal{F}$ is a fixed set for the bit flipping algorithms.

Proof: Let $\text{supp}(\mathbf{y}) = \mathcal{F}$. Then, $\mathcal{F}$ is the set of corrupt variable nodes at the beginning of the decoding. Observe that a variable node flips if it is connected to at least two unsatisfied check nodes. Since no variable node is connected to two unsatisfied check nodes, the set of corrupt variable nodes is unchanged and by definition $\mathcal{F}$ is a fixed set. □

We note that Theorem 3 is also a consequence of Fact 3 from [10].

Corollary 2: A fixed set for the Gallager A algorithm is also a fixed set for the bit flipping algorithms.

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

V. Conclusion

In this paper, we have investigated the error-correction capability of column-weight-three codes under Gallager A algorithm and extended the results to the bit flipping algorithms. Recently, we have shown in [12] that a column-weight-three LDPC code with Tanner graph of girth $g \geq 10$ can correct all error patterns with up to $g/2-1$ errors thereby showing that the bounds established in this paper are tight. Future work includes investigation of necessary and sufficient conditions to correct a given number of errors for higher column weight codes.

Appendix I

Proof of Theorem 1: We first show that the conditions of the theorem are sufficient. Let $\mathcal{F}$ be a set of variable nodes with induced subgraph $\mathcal{I}$ satisfying conditions a) and b). Let $\mathbf{y}$ be the input to the decoder with $\text{supp}(\mathbf{y}) = \mathcal{F}$. Then

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

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

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

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

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

For any other check node c, $\varpi_1(c, v) = 0$. By the conditions of the theorem, at the end of first iteration, any $v \in \mathcal{F}$ receives at least two 1's and any $v \notin \mathcal{F}$ receives at most one 1. So, we have

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

The messages passed in the subsequent iterations are the same as the messages passed in the first iteration, and by definition, $\mathcal{F}$ is a fixed 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. The following two cases arise.

1) The subgraph induced by v_1 and v_2 has three even-degree check nodes c_1, c_2, and c_3. In this case, $\{v_1, v_2\}$ is a weight-two codeword, and therefore, an inducing set of size two.
2) The subgraph induced by v_1 and v_2 has two even-degree check nodes $\{c_1, c_2\}$ and two odd-degree check nodes $\{c_3, c_4\}$. If c_3 and c_4 are not connected to a common

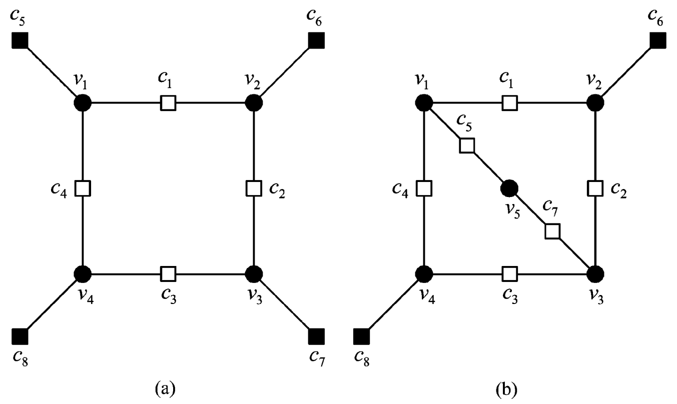

Fig. 3. Subgraphs induced by (a) $(4,4)$ trapping set (b) $(5,3)$ trapping set.

variable node, then $\{v_1, v_2\}$ is a fixed set, and hence, an inducing 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 fixed set, and therefore, an inducing set of size three. □

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

Case 1: No two check nodes from $\mathcal{O} = \{c_6, c_8, c_9\}$ are connected to a common variable node. Then, $\mathcal{T}_2$ is a fixed set and hence an inducing set of size five.

Case 2: All the three check nodes 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 an inducing set.

Case 3: There are variable nodes connected to two check nodes 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 $\mathrm{supp}(\mathbf{y}) = \mathcal{T}_2$, 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 jth 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 that 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 check nodes from $\mathcal{O}$. Let v_6 be connected to c_6 and c_8. It can be seen that the third check node connected to v_6 cannot belong to $\mathcal{E}$ as this would violate the girth condition. So, let the third check node be c_{10}. In the first half of the 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 that 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 node neighbor 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 an inducing set by observing that there cannot be a variable node outside $\mathcal{T}_2$, which sends an incorrect message to a check node in $\mathcal{E}$.

Subcase 2: There are two variable nodes connected to two check nodes 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 an inducing set.

ACKNOWLEDGMENT

The authors would like to thank A. R. Krishnan and D. V. Nguyen for illustrations and R. Radhakrishnan for useful discussions.

REFERENCES

[1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.
[2] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. 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," *Probl. Inf. Transm.*, vol. 11, pp. 18–28, 1976.
[4] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inf. 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. Inf. 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. Inf. 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. Inf. Theory*, vol. 51, no. 3, pp. 954–972, Mar. 2005.
[8] D. Burshtein, "On the error correction of regular LDPC codes using the flipping algorithm," in *Proc. Int. Symp. Inf. Theory*, Jun. 24–29, 2007, pp. 226–230.
[9] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.
[10] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. Commun. Control Comput.*, 2003, pp. 1426–1435.
[11] A. Shokrollahi, "An introduction to low-density parity-check codes," in *Theoretical Aspects of Computer Science: Advanced Lectures*. New York: Springer-Verlag, 2002, pp. 175–197.
[12] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, "Error correction capability of column-weight-three LDPC codes: Part II," 2008 [Online]. Available: http://arxiv.org/abs/0807.3582

Shashi Kiran Chilappagari (S'05–M'09) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology, Madras, India, in 2004 and the Ph.D. degree in electrical engineering from the University of Arizona, Tucson, in 2008.

Currently, he is a Research Engineer at the Department of Electrical and Computer Engineering, University of Arizona. In summer 2006, he was an intern in the Systems Group at the IBM Zurich Research Laboratory, Zurich, Switzerland. In summer 2008, he interned at the Los Alamos National Labs, Los Alamos, NM. His research interests include error control coding, data compression, and information theory.

Bane Vasic (S'92–M'93–SM'02) received the B.Sc., M. Sc., and Ph.D. degrees in electrical engineering from the University of Nis, Nis, Yugoslavia (now Serbia), in 1989, 1991, and 1994, respectively.

From 1996 to 1997, he worked as a Visiting Scientist at the Rochester Institute of Technology and Kodak Research, Rochester, NY, where he was involved in research in optical storage channels. From 1998 to 2000, he was with Lucent Technologies, Bell Laboratories (Bell-Labs). He was involved in research in iterative decoding and low-density parity-check codes, as well as development of codes and detectors implemented in Bell-Labs chips. Currently, he is a Professor at the Electrical and Computer Engineering Department, University of Arizona, Tucson. His research interests include coding theory, information theory, communication theory, and digital communications and recording.

Dr. Vasic is a Member of the Editorial Board for the IEEE TRANSACTIONS ON MAGNETICS. He served as Technical Program Chair of the IEEE Communication Theory Workshop in 2003 and as Co-Organizer of the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) Workshops on Optical/Magnetic Recording and Optical Transmission, and Theoretical Advances in Information Recording in 2004. He was Co-Organizer of the Los Alamos Workshop on Applications of Statistical Physics to Coding Theory in 2004, the Communication Theory Symposium within the IEEE International Conference on Communications (ICC 2006), and IEEE Communication Theory Workshop (CTW 2007).