Girth of the Tanner Graph and Error Correction Capability of LDPC Codes

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

***Abstract*—We investigate the relation between the girth and the guaranteed error correction capability of γ-left regular LDPC codes. For column-weight-three codes, we give upper and lower bounds on the number of errors correctable by the Gallager A algorithm. For higher column weight codes, we find the number of variable nodes which are guaranteed to expand by a factor of at least $3\gamma/4$, hence giving a lower bound on the guaranteed correction capability under the bit flipping (serial and parallel) algorithms. We also establish upper bounds by studying the sizes of smallest possible trapping sets.**

Keywords: Low-density parity-check codes, Gallager A algorithm, bit flipping algorithms, error correction capability

I. Introduction

Low-density parity-check (LDPC) codes [1] are a class of capacity approaching codes under low complexity iterative decoding algorithms [2]. Gallager showed in [1] that the minimum distance of an LDPC code increase linearly with the length of the code. Linear increase of the minimum distance with the code length, however, does not guarantee the correction of a linear fraction of errors under iterative decoding algorithms. A similar statement can be made about guaranteed correction of a fixed number of errors in a finite length code.

Work on guaranteed error correction capability of LDPC codes started with Zyablov and Pinsker [3] who showed that almost all codes in the regular ensemble with $\gamma \geq 5$ can correct a constant fraction of worst case errors under the bit flipping algorithm. Siper and Spielman [4] showed that the bit flipping algorithms, parallel and serial, can correct a fraction of errors if the underlying Tanner graph is a good expander. They used expander graphs, a method that is generalized by Burshtein and Miller [5] who showed that message passing algorithms can also correct a fixed fraction of worst case errors when the degree of each variable node is more than five.

The main limitation of the above approaches is that expander arguments cannot be applied to codes with column-weight smaller than five. Recently, Burshtein [6] 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. He notes that for column-weight-three codes such a result cannot be proved using the method from [6]. The reason for this is that a non-negligible fraction of codes have parallel edges in their Tanner graphs, and such codes cannot correct a single worst case error.

Another limitation of the above approaches is that the fraction of nodes having the required expansion is small, requiring codes to have large length to guarantee correction of a fixed number of errors. Determining the expansion of a given graph is known to be non-deterministic polynomial-time (NP) hard [7] and the spectral gap methods can only guarantee an expansion factor of $1/2$ or less. While a random graph is a good expander with high probability, no explicit construction of codes having the required expansion is known. However if expansion properties can be related to the parameters of the Tanner graph, such as girth and column-weight, which are known or can be easily determined, then the bounds on guaranteed error correction capability can be established as a function of these parameters. This is the general idea of our approach as reviewed in this paper.

In [8], a stronger version of the statement by Burshtein regarding the error correction capability of column-weight-three LDPC codes was proved. By studying the smallest length cycles in the Tanner graph of a code, it was shown in [8] that an LDPC code whose Tanner graph has girth $g \geq 10$ cannot correct all error patterns up to weight $g/2$. As a consequence, it was established that for any $\alpha > 0$, at sufficiently large block length n, no code in the $C^n(3, \rho)$ ensemble can correct a α fraction of errors. In [9], it was shown 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 the upper bound established in [8] is tight. For higher column weight codes, the relation between the girth and the guaranteed error correction capability of left regular LDPC codes when decoded using the bit flipping algorithms was investigated in [10]. Specifically, a lower bound on the size of variable node sets which expand by a factor of at least $3\gamma/4$ was derived and the sizes of smallest possible trapping sets were studied to establish an upper bound on the guaranteed error correction capability. In this review paper, we summarize the results from [8], [9], [10] and provide an overview and intuition of methods used to obtain these results.

The rest of the paper is organized as follows. In Section II, we provide a brief introduction to LDPC codes, decoding algorithms and trapping sets. In Section III, we consider the

Manuscript received September 26, 2008. This work is funded by NSF under Grant CCF-0634969, ECCS-0725405 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.

Parts of this work were submitted to IEEE Transactions on Information Theory and presented at the International Symposium on Information Theory (ISIT'08)

guaranteed error correction capability of column-weight three codes. In Section IV, we derive the upper and lower bounds on the guaranteed error correction capability of higher column-weight LDPC codes. We conclude with a few remarks in Section V.

II. Preliminaries

A. Notations

A binary LDPC code $\mathcal{C}$ is a linear block code which maps a message block of f information bits to a binary n-tuple. A graphical representation $G = (V \cup C, E)$ of $\mathcal{C}$, also called the Tanner graph [11], is a bipartite graph with two sets of nodes: n variable (bit) nodes $V = \{1, 2, \dots, n\}$ and m check (constraint) nodes $C = \{1, 2, \dots, m\}$; and a set of edges E. The check nodes (variable nodes resp.) connected to a variable node (check node resp.) are referred to as its neighbors. A check node is satisfied if the modulo two sum of its neighbors is zero and unsatisfied otherwise. A vector $\mathbf{w} = (w_1 w_2 \dots w_n)$ is a codeword if and only if all the check nodes are satisfied. The support of $\mathbf{w}$, denoted as supp($\mathbf{w}$), is defined as the set of all variable nodes (bits) $v \in V$ such that $w_v \neq 0$. The adjacency matrix of G gives H, a parity-check matrix of $\mathcal{C}$. The degree $\chi(u)$ of a node $u \in V \cup C$ is the number of its neighbors. A node with degree one is called a leaf or a pendant node. A (γ, ρ) regular LDPC code $\mathcal{C}$ has a Tanner graph G in which all variable nodes have degree γ, and all check nodes have degree ρ. This code has rate $r > 1 - \gamma/\rho$ [11]. The degree of a variable node (check node resp.) is also referred to as the left degree (right degree resp.) or the column weight (row weight resp.).

The length of the shortest cycle in the graph G is called the girth g of G. The girth of a bipartite graph is even. 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. 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 [2] 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 output symmetric channels (see [2]), 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. Additionally, for column-weight-three codes, 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. Decoding Algorithms

LDPC codes can be decoded with low complexity iterative algorithms. In this paper, we consider transmission over the binary symmetric channel (BSC). We define *"flipping"* as the event of a bit changing its value from "0" to "1" or vice versa. The BSC with transition probability p flips a transmitted bit with probability p.

On the BSC, LDPC codes can be decoded with a simple hard decision decoding algorithm known as the parralel bit flipping algorithm, which is defined as below.

Parallel Bit Flipping Algorithm

- In parallel, flip the variable nodes that connect to more unsatisfied than satisfied neighbors.
- Repeat until no such variable node remains.

The parallel bit flipping algorithm is iterative in nature but does not belong to the class of message passing algorithms (see [5] for more details). Gallager in [1] proposed two simple binary message passing algorithms for decoding over the BSC, Gallager A and Gallager B. We shall first describe the Gallager B algorithm and then describe the Gallager A algorithm as a special case of the Gallager B algorithm.

Gallager B Algorithm

The Gallager B algorithm on the BSC operates by passing the messages along the edges of the Tanner graph of an LDPC code. Every round of message passing (iteration) starts with sending messages from variable nodes and ends by sending messages from check nodes to variable nodes. Assume that the decoder performs D iterations. Let $\mathbf{y} = (y_1 y_2 \dots y_n)$ be the input to the decoder and let $\mathbf{x}^l = (x_1^l x_2^l \dots x_n^l)$, $l \leq D$ be the output vector at the l^{th} iteration. 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. 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. A threshold $b_{v,j}$ is determined for every iteration j and the degree of variable node v. The Gallager B algorithm can then be defined as follows.

$$\omega_j(v,c) = \begin{cases} y_v, & \text{if } j = 1 \\ \kappa, & \text{if } |\{c' | c' \neq c, \varpi_{j-1}(c', v) = \kappa\}| \geq b_{v,j} \\ y_v, & \text{otherwise} \end{cases}$$

$$\varpi_j(c,v) = \left(\sum_{\kappa_j \in \omega_j(: \setminus v, c)} \kappa_j \right) \bmod 2$$

where $\kappa \in \{0,1\}$.

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 decoder is run until a valid codeword is found or the maximum number of iterations D is reached, whichever is earlier. The output of the decoder is either a codeword or $\mathbf{x}^D$.

The Gallager A algorithm can be obtained from the Gallager B algorithm by setting the threshold $b_{j,v} = \chi(v) - 1$ for all j.

C. Decoding Failure and Trapping Sets

As mentioned above, we assume that the all zero codeword is transmitted. Consider a LDPC code of length n. A variable node v is said to be *eventually correct* if there exists a positive integer q such that for all $l \geq q$, $v \notin \text{supp}(\mathbf{x}^l)$.

Definition 1: A decoder failure is said to have occurred if there does not exist $l \leq D$ such that $\text{supp}(\mathbf{x}^l) = \emptyset$.

Definition 2: Let $\mathbf{T}(\mathbf{y})$ denote the set of variable nodes that are not eventually correct. If $\mathbf{T}(\mathbf{y}) \neq \emptyset$, let $a = |\mathbf{T}(\mathbf{y})|$ and b be the number of odd degree check nodes in the sub-graph induced by $\mathbf{T}(\mathbf{y})$. We say $\mathbf{T}(\mathbf{y})$ is an (a,b) trapping set.

Definition 3: Let $\mathcal{T}$ be a trapping set and let $\mathbf{Y}(\mathcal{T}) = \{\mathbf{y} | \mathbf{T}(\mathbf{y}) = \mathcal{T}\}$. The critical number ξ of trapping set $\mathcal{T}$ is the minimal number of variable nodes that have to be initially in error for the decoder to end up in the trapping set $\mathcal{T}$, i.e. $\xi = \min_{\mathbf{Y}(\mathcal{T})} |\text{supp}(\mathbf{y})|$.

Definition 4: Let $\mathcal{T}$ be a trapping set. If $\mathbf{T}(\mathbf{y}) = \mathcal{T}$ then $\text{supp}(\mathbf{y})$ is a failure set of $\mathcal{T}$.

Definition 5: For transmission over a BSC, $\mathbf{y}$ is a fixed point of the decoding algorithm if $\text{supp}(\mathbf{y}) = \text{supp}(\mathbf{x}^l)$ for all l.

It follows that for transmission over the BSC, if $\mathbf{y}$ is a fixed point then $\mathbf{T}(\mathbf{y}) = \text{supp}(\mathbf{y})$ is a trapping set.

III. Column Weight Three Codes

In this section, we give the upper and lower bounds on the number of errors correctable by an iterative decoder for column-weight-three binary LDPC codes. We focus on the message passing decoding algorithms, specifically Gallager A and Gallager B, which are the same for column-weight-three codes and hence will be referred to as Gallager A algorithm. The problem of determining the guaranteed error correction capability of column-weight-three LDPC codes under the bit flipping algorithms is simpler and can be solved in a similar procedure.

A. The Upper Bound

We obtain the upper bound 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. We point out that this also holds for the bit flipping algorithm. The following theorem gives sufficient conditions for a set of variable nodes to be a trapping set of column-weight-three LDPC codes.

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 $|\mathcal{E}| = S$. $\mathbf{y}$ is a fixed point if : (a) $\text{supp}(\mathbf{y}) = \mathcal{T}$ and (b) Every variable node in $\mathcal{I}$ is connected to at least two checks in $\mathcal{E}$ and (c) No two checks of $\mathcal{O}$ are connected to a variable node outside $\mathcal{I}$.

Proof: See [8]. ■

We proceed by stating the following lemmas. The proof for Lemma 1 can be found in [6]. Note that this proof is for bit flipping algorithms, but also applies to the Gallager A algorithm. We give the proof of the last statement in Lemma 2. The proofs for the other statements are given in [8].

Lemma 1: [6] A code whose Tanner graph has parallel edges cannot correct a single worst case error.

Lemma 2: Let $\mathcal{C}$ be an $(n,3,\rho)$ regular LDPC code with girth g. Then the following holds:

- if $g = 4$ then $\mathcal{C}$ has at least one failure set of size two or three.
- if $g = 6$ then $\mathcal{C}$ has at least one failure set of size three or four.
- if $g = 8$ then $\mathcal{C}$ has at least one failure set of size four or five.
- if $g \geq 10$ then the set of $g/2$ variable nodes $\{v_1, v_2, \ldots, v_{g/2}\}$ involved in the shortest cycle is the support of a fixed point.

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.1. Let the even degree checks be $\mathcal{E} = \{c_1, c_2, \ldots, c_{g/2}\}$ and the odd degree checks be $\mathcal{O} = \{c_{g/2+1}, c_{g/2+2}, \ldots, c_g\}$. 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 $\{v_i, \ldots, v_j, v_{ij}\}$ 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 the support of a fixed point. ■

Remarks: Theorem 1 can be extended for the bit flipping algorithms as can be seen in Section IV-C. It follows that for column-weight-three LDPC codes, a trapping set under the Gallager A algorithm is also a trapping set under the bit flipping algorithms.

B. The Lower Bound

We show that a column-weight-three LDPC code with Tanner graph of girth $g \geq 10$ can correct all error patterns

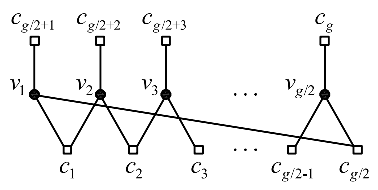

Fig. 1. Illustration of a cycle of length g

of weight $g/2 - 1$ or less under the Gallager A algorithm. Equivalently, there are no trapping sets with critical number less than $g/2$. In the following subsection, we give a detailed analysis of the Gallager A algorithm in the first k iterations.

1) The first k iterations: The following lemma describes the messages passed by the Gallager A algorithm in a column-weight-three code.

Lemma 3: (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.
- $\omega_j(v,:\backslash c) = \{1\}$ and $\omega_j(v,c) = 0$ if $\varpi_{j-1}(:\backslash c,v) = \{0\}$ 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.
- $\omega_j(v,:\backslash c) = \{0\}$ and $\omega_j(v,c) = 1$ if $\varpi_{j-1}(:\backslash c,v) = \{1\}$ 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. ■

Now let v be a variable node which sends an incorrect message along the edge $e = (v,c)$ in the $(k+1)^{th}$ 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 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 $e = (v,c)$ 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. 2(a), Fig. 2(b) and Fig. 2(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. 2 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 4: (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 $d = 2,4,\ldots,2(k-1)$, 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 $B(\mathcal{N}_e^{2k}) \geq 2^k$. 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: See [9] ■

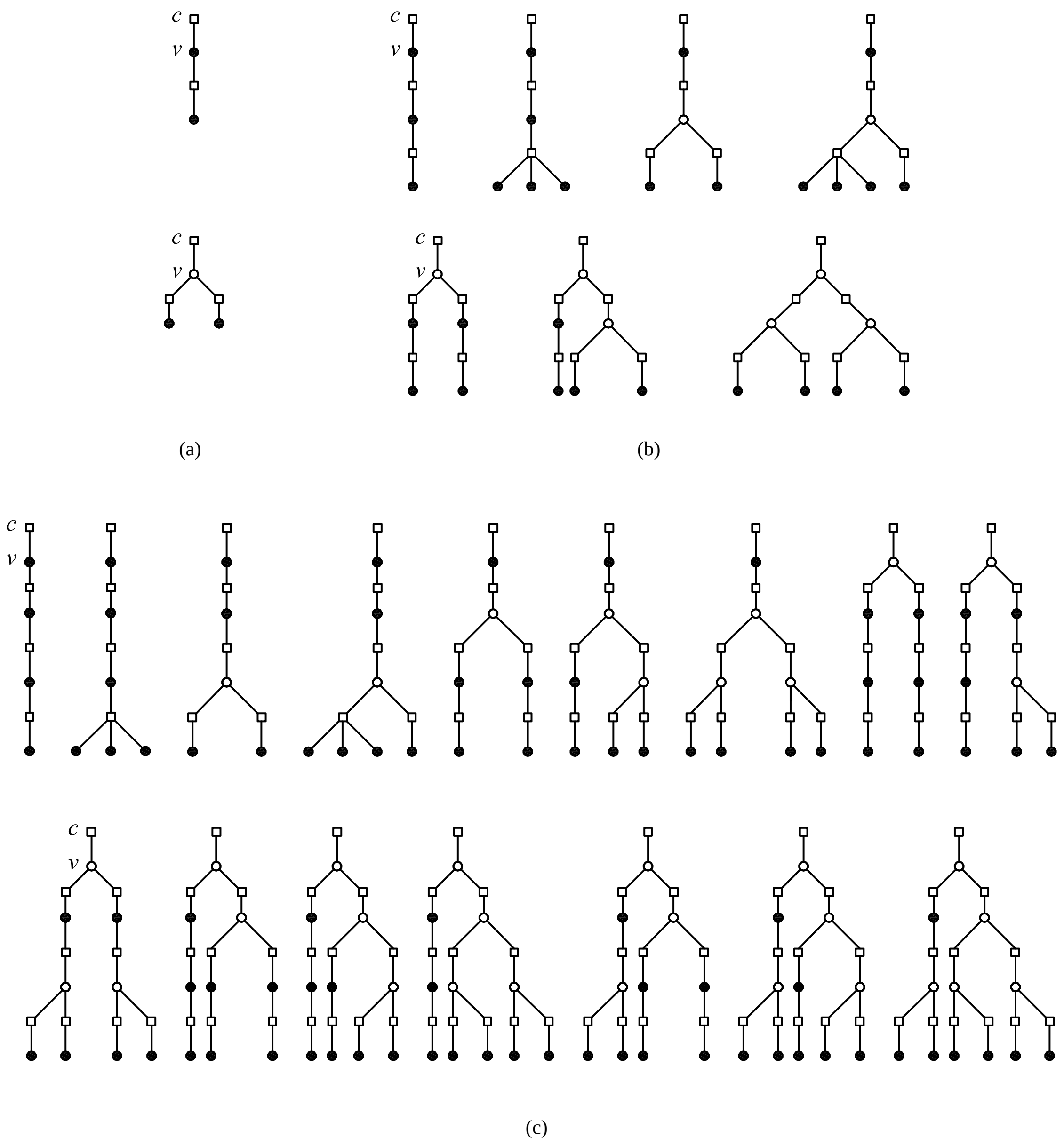

Fig. 2. Possible configurations of at most $2k+1$ bad variable nodes in the neighborhood of a variable node v sending an incorrect message to check node c in the $(k+1)^{th}$ iteration for (a) $k=1$, (b) $k=2$ and (c) $k=3$.

2) The main theorem:

Theorem 2: 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

Proof: 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 give an overview of the proof when $g/2$ is even. The proof for odd $g/2$ can be found in [9].

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 $N_{e_1}^1$ and $N_{e_2}^1$ 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. 3(a) for an illustration). All possible configurations of bad variable nodes in $\mathcal{N}_{e_3}^{2k}$ and $\mathcal{N}_{e_4}^{2k}$ can be determined using the method outlined in Section III-B1. 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 $\max\left(B(\mathcal{N}_{e_3}^{2k-2})+B(\mathcal{N}_{e_4}^{2k}),B(\mathcal{N}_{e_3}^{2k})+B(\mathcal{N}_{e_4}^{2k-2})\right)$. We are interested only in the valid configurations, i.e., at most

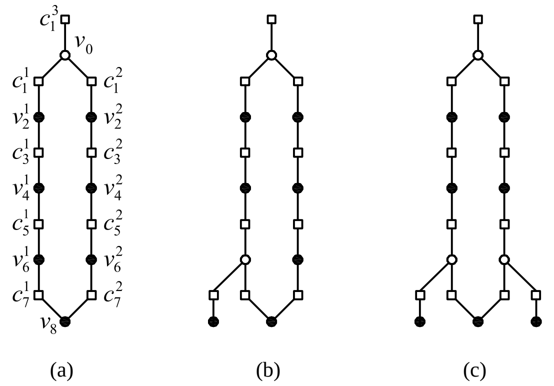

Fig. 3. Configurations of at most 7 bad variable nodes, free of cycles of length less than 16, which do not converge in 4 iterations.

$g/2-1$ 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.

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. 2(c) illustrates different configurations of bad variable nodes in $\mathcal{N}_e^6$. As remarked earlier, Fig. 2(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 4 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. 3 shows all such possible configurations.

Case 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 the $(k+1)^{th}$ 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 5: v_2^1 and v_2^2 are bad variable nodes.

Proof: See [9] ■

We now have the following theorem. Their proofs are given in [9].

Theorem 3: 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.

Theorem 3 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 3 is not true in general. However, for $g\geq 16$, we will show in Theorem 4 that any configuration for girth g can be extended to a configuration for girth $g+4$.

Theorem 4: For $g/2$ even and $k\geq 3$, there are only three valid configurations which do not converge in $(k+1)$ iterations.

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

Case 3: $g=12$**.** See [9]. ■

IV. Column Weight Four Codes and Higher

A. Expansion and Error Correction Capability

Sipser and Spielman [4] analyzed the performance of the bit flipping algorithms using the expansion properties of the underlying Tanner graph of the code. We summarize the results from [4] below for the sake of completeness. We start with the following definitions from [4].

Definition 6: Let $G=(U,E)$ with $|U|=n_1$. Then *every set of at most* m_1 *nodes expands by a factor of* δ if, for all sets $S\subset U$

$$|S|\leq m_1 \Rightarrow |\{y:\exists x\in S \text{ such that } (x,y)\in E\}|>\delta|S|.$$

We consider bipartite graphs and expansion of variable nodes only.

Definition 7: A graph is a $(\gamma,\rho,\alpha,\delta)$ expander if it is a (γ,ρ) regular bipartite graph in which every subset of at most α fraction of the variable nodes expands by a factor of at least δ.

The following theorem from [4] relates the expansion and error correction capability of an (n,γ,ρ) LDPC code with Tanner graph G when decoded using the parallel bit flipping decoding algorithm.

Theorem 6: [4, Theorem 11] Let G be a $(\gamma,\rho,\alpha,(3/4+\epsilon)\gamma)$ expander over n variable nodes, for any $\epsilon>0$. Then, the simple parallel decoding algorithm will correct any $\alpha_0<\alpha(1+4\epsilon)/2$ fraction of errors after $\log_{1-4\epsilon}(\alpha_0 n)$ decoding rounds.

Notes:

1) The serial bit flipping algorithm can also correct $\alpha_0<\alpha/2$ fraction of errors if G is a $(\gamma,\rho,\alpha,(3/4)\gamma)$ expander.
2) The results hold for any left regular code as expansion is needed for variable nodes only.

From the above discussion, it is observed that finding the number of variable nodes which are guaranteed to expand by a factor of at least $3\gamma/4$, gives a lower bound on the guaranteed error correction capability of LDPC codes.

B. Column Weight, Girth and Expansion

In this section, we prove our main theorem which relates the column weight and girth of a code to its error correction capability. We show that the size of variable node sets which have the required expansion is related to the well known Moore bound [12, p.180]. We start with a few definitions required to establish the main theorem.

Definition 8: The *reduced graph* $H_r = (V \cup C_r, E'_r)$ of $H = (V \cup C, E')$ is a graph with vertex set $V \cup C_r$ and edge set E'_r given by

$$\begin{aligned} C_r &= C \setminus C_p,\ C_p = \{c \in C : \text{c is a pendant node}\} \\ E'_r &= E' \setminus E'_p,\ E'_p = \{(v_i, c_j) \in E : c_j \in C_p\}. \end{aligned}$$

The *Moore bound* [12, p.180] denoted by $n_0(\chi, g)$ is a lower bound on the least number of vertices in a χ-regular graph with girth g. It is given by

$$\begin{aligned} n_0(\chi, g) = n_0(\chi, 2r+1) &= 1 + \chi \sum_{i=0}^{r-1} (\chi - 1)^i,\ g \text{ odd} \\ n_0(\chi, g) = n_0(\chi, 2r) &= 2 \sum_{i=0}^{r-1} (\chi - 1)^i,\ g \text{ even.} \end{aligned}$$

In [13], it was shown that a similar bound holds for irregular graphs.

Theorem 7: [13] The number of nodes $n(\overline{\chi}, g)$ in a graph of girth g and average degree at least $\overline{\chi} \geq 2$ satisfies

$$n(\overline{\chi}, g) \geq n_0(\overline{\chi}, g).$$

Note that $\overline{\chi}$ need not be an integer in the above theorem.

We now state and prove the main theorem.

Theorem 8: Let G be a γ-left regular Tanner graph G with $\gamma \geq 4$ and $g(G) = 2g'$. Then for all $k < n_0(\gamma/2, g')$, any set of k variable nodes in G expands by a factor of at least $3\gamma/4$.

Proof: Let $G^k = (V^k \cup C^k, E^k)$ denote the subgraph induced by a set of k variable nodes V^k. Since G is γ-left regular, $|E^k| = \gamma k$. Let $G^k_r = (V^k \cup C^k_r, E^k_r)$ be the reduced graph. We have

$$\begin{aligned} |C^k| &= |C^k_r| + |C^k_p| \\ |E^k| &= |E^k_p| + |E^k_r| \\ |E^k_p| &= |C^k_p| \\ |C^k_p| &= \gamma k - |E^k_r|. \end{aligned}$$

We need to prove that $|C^k| > 3\gamma k/4$.

Let $f(k, g')$ denote the maximum number of edges in an arbitrary graph of order k and girth g'. By Theorem 2, for all $k < n_0(\gamma/2, g')$, the average degree of a graph with k nodes and girth g' is less than $\gamma/2$. Hence, $f(k, g') < \gamma k/4$. We now have the following lemmas. Their proofs are given in [10].

Lemma 6: The number of edges in G^k_r cannot exceed $2f(k, g')$ i.e.,

$$|E^k_r| \leq 2f(k, g').$$

We now find a lower bound on $|C^k|$ in terms of $f(k, g')$.

Lemma 7: $|C^k| \geq \gamma k - f(k, g')$.

The theorem now follows as

$$f(k, g') < \gamma k/4$$

and therefore

$$|C^k| > 3\gamma k/4.$$

■

Corollary 1: Let $\mathcal{C}$ be an LDPC code with column-weight $\gamma \geq 4$ and girth $2g'$. Then the bit flipping algorithm can correct any error pattern of weight less than $n_0(\gamma/2, g')/2$.

C. Cage Graphs and Trapping Sets

In this section, we first give necessary and sufficient conditions for a given set of variables to be a trapping set. We then state the definition of a class of interesting graphs known as cage graphs [14] and establish a relation between cage graphs and trapping sets. We then give an upper bound on the error correction capability based on the sizes of cage graphs.

Theorem 9: Let $\mathcal{C}$ be an LDPC code with γ-left regular Tanner graph G. 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. $\mathbf{y}$ is a fixed point if : (a) supp$(\mathbf{y}) = \mathcal{T}$ and (b) Every variable node in $\mathcal{I}$ has at least $\lceil \gamma/2 \rceil$ neighbors in $\mathcal{E}$, and (c) No $\lfloor \gamma/2 \rfloor + 1$ checks of $\mathcal{O}$ share a neighbor outside $\mathcal{I}$.

Proof: See [10] ■

Remark: Theorem 9 is a consequence of Fact 3 from [15].

To determine whether a given set of variables is a trapping set, it is necessary to not only know the induced subgraph but also the neighbors of the odd degree checks. However, in order to establish general bounds on the sizes of trapping sets given only the column weight and the girth, we consider only condition (b) of Theorem 9 which is a necessary condition. A set of variable nodes satisfying condition (b) is known as a *potential trapping set*. A trapping set is a potential trapping set that satisfies condition (c). Hence, a lower bound on the size of the potential trapping set is a lower bound on the size of a trapping set. It is worth noting that a potential trapping set can always be extended to a trapping set by successively adding a variable node till condition (c) is satisfied.

Definition 9: [14] A (χ, g)*-cage graph*, $G(\chi, g)$, is a χ-regular graph with girth g having the minimum possible number of nodes.

A lower bound, $n_l(\chi, g)$, on the number of nodes $n_c(\chi, g)$ in a (χ, g)-cage graph is given by the Moore bound. An upper bound $n_u(\chi, g)$ on $n_c(\chi, g)$ (see [14] and references therein) is given by

$$\begin{aligned} n_u(3, g) &= \begin{cases} \frac{4}{3} + \frac{29}{12}\, 2^{g-2} & \text{for g odd} \\ \frac{2}{3} + \frac{29}{12}\, 2^{g-2} & \text{for g even} \end{cases} \\ n_u(\chi, g) &= \begin{cases} 2(\chi - 1)^{g-2} & \text{for g odd} \\ 4(\chi - 1)^{g-3} & \text{for g even} \end{cases}. \end{aligned}$$

Theorem 10: Let $\mathcal{C}$ be an LDPC code with γ-left regular Tanner graph G and girth $2g'$. Let $|\mathcal{T}(\gamma, 2g')|$ denote the size of smallest possible potential trapping set of $\mathcal{C}$ for the bit flipping algorithm. Then,

$$|\mathcal{T}(\gamma, 2g')| = n_c(\lceil \gamma/2 \rceil, g').$$

Proof: See [10] ■

Theorem 11: There exists a code $\mathcal{C}$ with γ-left regular Tanner graph of girth $2g'$ which fails to correct $n_c(\lceil \gamma/2 \rceil, g')$ errors.

Proof: See [10] ■

Remark: We note that for $\gamma = 3$ and $\gamma = 4$, the above bound is tight. Observe that for $\chi = 2$, the Moore bound is $n_0(\chi, g) = g$ and that a cycle of length $2g$ with g variable nodes is always a potential trapping set. In fact, for a code with $\gamma = 3$ or 4, and Tanner graph of girth greater than eight, a cycle of the smallest length is always a trapping set (see [8] for the proof).

V. Summary

In this paper, we have reviewed relations between the girth and error correction capability of LDPC codes. Specifically, we have given the best possible upper and lower bounds on the number of errors correctable by a Gallager A decoder. Clearly, the error correction capability grows only linearly in the girth for column-weight-three codes. For higher column weight LDPC codes, we have given a lower bound on the guaranteed error correction capability by finding bounds on the number of nodes that have the required expansion. We also studied the sizes of the smallest possible trapping sets to establish an upper bound.

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, no. 1, 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] D. Burshtein, "On the error correction of regular LDPC codes using the flipping algorithm,," *IEEE Trans. Inform. Theory*, vol. 54, no. 2, pp. 517–530, Feb. 2008.

[7] N. Alon, "Spectral techniques in graph algorithms," in *LATIN '98: Proceedings of the Third Latin American Symposium on Theoretical Informatics*. London, UK: Springer-Verlag, 1998, pp. 206–215.

[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, D. V. Nguyen, and B. Vasic, "Error correction capability of column-weight-three LDPC codes: Part II," submitted to *IEEE Trans. Inform. Theory*. [Online]. Available: http://arxiv.org/abs/0807.3582

[10] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, "On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes," May 2008. [Online]. Available: http://arxiv.org/abs/0805.2427

[11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, Sept. 1981.

[12] N. Biggs, *Algebraic graph theory*. Cambridge: Cambridge University Press, 1993.

[13] N. Alon, S. Hoory, and M. Linial, "The Moore bound for irregular graphs," *Graphs and Combinatorics*, vol. 18, no. 1, pp. 53–57, 2002.

[14] E. W. Weisstein, "Cage graph." [Online]. Available: http://mathworld.wolfram.com/CageGraph.html

[15] T. J. Richardson, "Error floors of LDPC codes," in *Proc. of 41st Annual Allerton Conf. on Communications, Control and Computing*, 2003, pp. 1426–1435.