An Algorithm for Counting Short Cycles in Bipartite Graphs

Thomas R. Halford, *Student Member, IEEE,* and Keith M. Chugg, *Member, IEEE*

***Abstract*—Let $G = (\mathcal{U} \cup \mathcal{W}, \mathcal{E})$ be a bipartite graph with disjoint vertex sets $\mathcal{U}$ and $\mathcal{W}$, edge set $\mathcal{E}$, and girth g. This correspondence presents an algorithm for counting the number of cycles of length g, $g+2$, and $g+4$ incident upon every vertex in $\mathcal{U} \cup \mathcal{W}$. The proposed cycle counting algorithm consists of integer matrix operations and its complexity grows as $\mathcal{O}(gn^3)$ where $n = \max(|\mathcal{U}|, |\mathcal{W}|)$.**

***Index Terms*—Bipartite graphs, cycles, girth, graphical models of codes, loops.**

I. Introduction

The rediscovery of Gallager's low-density parity-check (LDPC) codes [2] by Spielman *et al.* [3] and MacKay *et al.* [4] along with the work of Wiberg, Loeliger, and Koetter [5], [6] sparked interest in the subject of graphical models of codes. *Good* graphical code models are those that imply near-optimal decoding algorithms with practically realizable complexity. A number of folk theorems have arisen in the literature regarding the graph-theoretic properties of good graphical code models [7]. For example, it is widely accepted that good LDPC and concatenated convolutional code designs have graphical models without too many short cycles [8]–[11]. Large girth, however, is not enough to ensure a good graphical model. As is shown in Section VI, the performance of two LDPCs with the same girth but a different number of short cycles can be dramatically different. The regularity of the cycle structure of a graph (i.e., how random the graph appears) also affects graphical code model quality. For example, the introduction of irregularity to LDPC designs is known to improve performance [12]. To summarize, good graphical models of codes tend to have large girth, a small number of short cycles, and a cycle structure that is not overly regular.

Given the growing body of evidence connecting the performance of a code and the properties of its associated graphical model, characterizing the cycle structure of a graphical model is of great interest. Exactly counting cycles and paths in arbitrary graphs is known to be hard [13]. Alon, Yuster, and Zwick presented methods for counting cycles of length less than 8 in [14]; however, their methods are prohibitively complex for longer cycles. The present work addresses the restricted problem of counting the cycles of length g, $g+2$, and $g+4$ in bipartite graphs with girth g. Only bipartite graphs are considered because the graphs used to represent codes (e.g., Tanner graphs [15], factor graphs [16]) are bipartite.

The remainder of this correspondence is organized as follows. Section II presents a brief review of the required graph theory and introduces the notation used in Sections III–V. Section III presents a recursion used to count paths and thus cycles in bipartite graphs. Section IV presents the proof technique used to solve said recursion. Section V presents the specific expressions required to count cycles of length $g, g+2$ and $g+4$ in bipartite graphs with girth g. Section VI presents a number of examples of the counting algorithm. The complexity of the proposed algorithm is discussed in Section VII. A sample of the proofs of the expressions presented in Section V is appended. The cycle counting technique presented in this correspondence was described previously in [1].

Manuscript received March 4, 2005; revised September 7, 2005. The work of T. R. Halford was supported in part by the Powell Foundation. The material in this correspondence was presented in part at the Communication Theory Workshop, Capri, Italy, May 2004.

The authors are with the Communication Sciences Institute, University of Southern California, Los Angeles, CA 90089-2565 USA (e-mail: halford@usc.edu; chugg@usc.edu).

Communicated by M. Sudan, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2005.860472

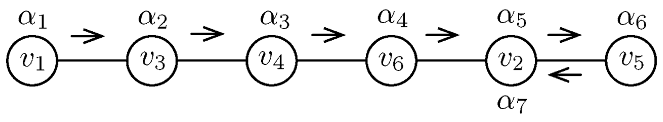

Fig. 1. A $(4,2)$-lollipop walk where $\alpha_7 = \alpha_5$ while $\alpha_1, \alpha_2, \ldots, \alpha_6$ are distinct.

II. Review of Graph Theory and Notation

Throughout this correspondence, the shorthand $[a,b]$ is used to abbreviate the set

$$\{a, a+1, \ldots, b-1, b\} \subset \mathbb{Z}$$

where $a < b \in \mathbb{Z}$ (the set of integers). Similary, $[c,d]_e$ and $[e,f]_o$ are used to abbreviate sets of consecutive even and odd integers, respectively, where $c < d \in 2\mathbb{Z}$ and $e < f \in 2\mathbb{Z}+1$.

A. Graph Theory

A graph $G(\mathcal{V}, \mathcal{E})$ consists of a finite nonempty set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$, which is any subset of the pairs

$$\{\{u,v\} : u, v \in \mathcal{V}, u \neq v\}.$$

A *walk* of length n in $G(\mathcal{V}, \mathcal{E})$ is a sequence of vertices $\alpha_1, \alpha_2, \ldots, \alpha_n, \alpha_{n+1}$ such that $\alpha_i \in \mathcal{V}$ and $\{\alpha_i, \alpha_{i+1}\} \in \mathcal{E}$ for all $i \in [1,n]$. A walk of length n is termed *closed* if $\alpha_1 = \alpha_{n+1}$; that is, if α_1 and α_{n+1} correspond to the same vertex $v \in \mathcal{V}$. A walk is a *trail* if the edges that form the walk are distinct; a trail is a *path* if the vertices are distinct. A closed walk of length n is a *cycle* if $\alpha_1, \alpha_2, \ldots, \alpha_n$ are distinct; that is, if $\alpha_i \neq \alpha_j$ for all $i \neq j \in [1,n]$. The lengths of the shortest and longest cycles of a graph are respectively denoted its *girth* g and *circumference* c [17]. A graph with no cycles is a *tree* and has $g = \infty$ and $c = 0$.

This work introduces the term $(m, n-m)$-*lollipop*[1] walk to describe length n walks where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are distinct and $\alpha_{n+1} = \alpha_{m+1}$ for some $m \in [1,n]$. Cycles of length $2m$ are thus $(0,2m)$-lollipop walks. Figs. 1 and 2 illustrate a $(4,2)$-lollipop walk and a $(2,4)$-lollipop walk, respectively.

A graph $G(\mathcal{V}, \mathcal{E})$ is bipartite with vertex classes $\mathcal{U} = \{u_i\}_{i=1}^{|\mathcal{U}|}$ and $\mathcal{W} = \{w_i\}_{i=1}^{|\mathcal{W}|}$ if $\mathcal{V} = \mathcal{U} \cup \mathcal{W}, \mathcal{U} \cap \mathcal{W} = \emptyset$ and each edge joins a vertex in $\mathcal{U}$ to one in $\mathcal{W}$. All cycles in bipartite graphs contain an even number of edges. Consequently, all $(m, n-m)$-lollipop walks in a bipartite graph satisfy $n - m \equiv 0 \bmod 2$. Associated with the bipartite graph $G(\mathcal{U} \cup \mathcal{W}, \mathcal{E})$ is the $|\mathcal{U}| \times |\mathcal{W}|$ edge matrix $E = [e_{ij}]$ where $e_{ij} = 1$ if $(u_i, w_j) \in \mathcal{E}$ and 0 otherwise.

Finally, the number of cycles of length $2m$ in a bipartite graph is denoted N_{2m}.

[1]This terminology is nonstandard and should not be confused with the lollipop graph introduced by Thomason in [18].

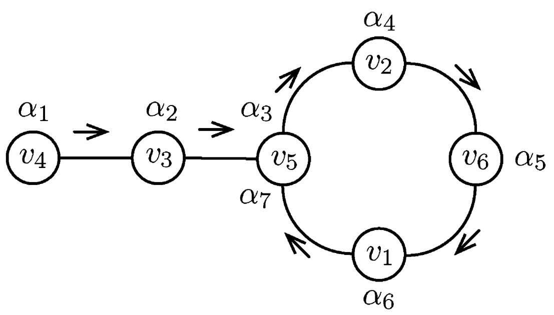

Fig. 2. A $(2,4)$-lollipop walk where $\alpha_7 = \alpha_3$ while $\alpha_1, \alpha_2, \ldots, \alpha_6$ are distinct.

B. Matrix Notation

The following matrix notation is used throughout this correspondence. Let A and B be matrices. The matrix product is denoted AB while A^T is the transpose of A. If $A = [a_{ij}]$ and $B = [b_{ij}]$ have the same dimensions then the direct (element-wise) product is defined by

$$C = A \circ B = [c_{ij}], \qquad \text{where } c_{ij} = a_{ij} b_{ij}. \tag{1}$$

If A is a square matrix then $\mathrm{Tr}(A)$ is the trace of A and the $\mathcal{Z}[\cdot]$ operator is defined by

$$\mathcal{Z}[A] = A - A \circ I \tag{2}$$

where I is the identity. Thus, $\mathcal{Z}[A]$ is A with the diagonal elements replaces by zeros. The following notation is also employed for arbitrary matrices $A = [a_{ij}]$ where $k \in \mathbb{Z}$:

$$B = \binom{A}{k} = [b_{ij}], \quad \text{where } b_{ij} = \binom{a_{ij}}{k} = \frac{a_{ij}!}{k!(a_{ij} - k)!} \tag{3}$$

$$B = A + k = [b_{ij}], \quad \text{where } b_{ij} = a_{ij} + k \tag{4}$$

$$B = kA = [b_{ij}], \quad \text{where } b_{ij} = k a_{ij} \tag{5}$$

$$B = \max(A, k) = [b_{ij}], \quad \text{where } b_{ij} = \max(a_{ij}, k). \tag{6}$$

III. The Lollipop Recursion

This section presents a recursion that transforms the problem of counting paths and thus cycles in a bipartite graph into the problem of counting lollipop walks. Eight matrices that count the number of paths and lollipop walks in a bipartite graph $G(\mathcal{U} \cup \mathcal{W}, \mathcal{E})$ with edge matrix E must first be defined. Let $P^{\mathcal{U}}_{2k}$ be a $|\mathcal{U}| \times |\mathcal{U}|$ matrix where the (i,j)th element is the number of paths of length $2k$ from $u_i \in \mathcal{U}$ to $u_j \in \mathcal{U}$; similarly define $P^{\mathcal{W}}_{2k}$. The diagonal elements of $P^{\mathcal{U}}_{2k}$ and $P^{\mathcal{W}}_{2k}$ are necessarily zero. Let $P^{\mathcal{U}}_{2k+1}$ be a $|\mathcal{U}| \times |\mathcal{W}|$ matrix where the (i,j)th element is the number of paths of length $2k+1$ from $u_i \in \mathcal{U}$ to $w_j \in \mathcal{W}$; similarly define $P^{\mathcal{W}}_{2k+1}$. Let $L^{\mathcal{U}}_{(2k', 2k-2k')}$ be a $|\mathcal{U}| \times |\mathcal{U}|$ matrix where the (i,j)th element is the number of $(2k', 2k-2k')$-lollipop walks from $u_i \in \mathcal{U}$ to $u_j \in \mathcal{U}$; similarly define $L^{\mathcal{W}}_{(2k', 2k-2k')}$. Let $L^{\mathcal{U}}_{(2k'+1, 2k-2k')}$ be a $|\mathcal{U}| \times |\mathcal{W}|$ matrix where the (i,j)th element is the number of $(2k'+1, 2k-2k')$-lollipop walks from $u_i \in \mathcal{U}$ to $w_j \in \mathcal{W}$; similarly define $L^{\mathcal{W}}_{(2k'+1, 2k-2k')}$.

Based on the preceding definitions, there is a simple recursion for the path matrices with the initial conditions $P^{\mathcal{U}}_0 = I, P^{\mathcal{W}}_0 = I$ and $L^{\mathcal{U}}_{(0,0)} = L^{\mathcal{W}}_{(0,0)} = 0$

$$P^{\mathcal{U}}_{2k+1} = P^{\mathcal{U}}_{2k} E - \sum_{i=0}^{k-1} L^{\mathcal{U}}_{(2i+1, 2k-2i)} \tag{7}$$

$$P^{\mathcal{U}}_{2k} = P^{\mathcal{U}}_{2k-1} E^T - \sum_{i=0}^{k-1} L^{\mathcal{U}}_{(2i, 2k-2i)} \tag{8}$$

$$P^{\mathcal{W}}_{2k+1} = P^{\mathcal{W}}_{2k} E^T - \sum_{i=0}^{k-1} L^{\mathcal{W}}_{(2i+1, 2k-2i)} \tag{9}$$

$$P^{\mathcal{W}}_{2k} = P^{\mathcal{W}}_{2k-1} E - \sum_{i=0}^{k-1} L^{\mathcal{W}}_{(2i, 2k-2i)}. \tag{10}$$

To prove (7), note that a path of length $2k+1, \alpha_1, \alpha_2, \ldots, \alpha_{2k+2}$, is formed by augmenting a single edge to a path of length $2k, \alpha_1, \alpha_2, \ldots, \alpha_{2k+1}$, such that $\alpha_{2k+2} \neq \alpha_i \forall i \in [1, 2k+1]$. All walks $\alpha_1, \alpha_2, \ldots, \alpha_{2k+1}, \alpha_{2k+2}$ where the first $2k+1$ vertices form a path are counted by $P^{\mathcal{U}}_{2k} E$. The product $P^{\mathcal{U}}_{2k} E$ thus counts both length-$2k+1$ paths and all length-$2k+1$ lollipop walks, hence (7). Equations (8)–(10) are proved in a similar manner. The recursion defined by (7)–(10) transforms the problem of counting paths and cycles to that of counting lollipop walks and is thus denoted the *lollipop recursion*.

Recall that cycles of length $2k$ are $(0, 2k)$-lollipop walks. Since $(0, 2k)$-lollipop walks are paths of length $2k-1$ with an added edge such that $\alpha_{2k+1} = \alpha_1$, there is a simple expression for counting the number of cycles of length $2k$ passing through each vertex

$$L^{\mathcal{U}}_{(0,2k)} = \left(P^{\mathcal{U}}_{2k-1} E^T\right) \circ I \tag{11}$$

$$L^{\mathcal{W}}_{(0,2k)} = \left(P^{\mathcal{W}}_{2k-1} E\right) \circ I. \tag{12}$$

The total number of cycles of length $2k$ is

$$N_{2k} = \frac{1}{2k} \mathrm{Tr}\left(L^{\mathcal{U}}_{(0,2k)}\right) = \frac{1}{2k} \mathrm{Tr}\left(L^{\mathcal{W}}_{(0,2k)}\right). \tag{13}$$

It is clear from (11) and (12) that in order to count cycles of length $g, g+2$, and $g+4$ in a bipartite graph of girth g, the lollipop recursion must be solved up to and including $2k+1 = g+3$. In a bipartite graph of girth g, there are no $(m, n-m)$-lollipop walks for which $n - m < g$. Expanding (7)-(10) yields, for $0 < k < \frac{g}{2}$

$$P^{\mathcal{U}}_{2k} = P^{\mathcal{U}}_{2k-1} E^T - L^{\mathcal{U}}_{(2k-2,2)} \tag{14}$$

$$P^{\mathcal{W}}_{2k} = P^{\mathcal{W}}_{2k-1} E - L^{\mathcal{W}}_{(2k-2,2)} \tag{15}$$

$$P^{\mathcal{U}}_{2k+1} = P^{\mathcal{U}}_{2k} E - L^{\mathcal{U}}_{(2k-1,2)} \tag{16}$$

$$P^{\mathcal{W}}_{2k+1} = P^{\mathcal{W}}_{2k} E^T - L^{\mathcal{W}}_{(2k-1,2)} \tag{17}$$

and for $k \geq \frac{g}{2}$

$$P^{\mathcal{U}}_{g} = P^{\mathcal{U}}_{g-1} E^T - L^{\mathcal{U}}_{(0,g)} - L^{\mathcal{U}}_{(g-2,2)} \tag{18}$$

$$P^{\mathcal{W}}_{g} = P^{\mathcal{W}}_{g-1} E - L^{\mathcal{W}}_{(0,g)} - L^{\mathcal{W}}_{(g-2,2)} \tag{19}$$

$$P^{\mathcal{U}}_{g+1} = P^{\mathcal{U}}_{g} E - L^{\mathcal{U}}_{(1,g)} - L^{\mathcal{U}}_{(g-1,2)} \tag{20}$$

$$P^{\mathcal{W}}_{g+1} = P^{\mathcal{W}}_{g} E^T - L^{\mathcal{W}}_{(1,g)} - L^{\mathcal{W}}_{(g-1,2)} \tag{21}$$

$$P^{\mathcal{U}}_{g+2} = P^{\mathcal{U}}_{g+1} E^T - L^{\mathcal{U}}_{(0,g+2)} - L^{\mathcal{U}}_{(2,g)} - L^{\mathcal{U}}_{(g,2)} \tag{22}$$

$$P^{\mathcal{W}}_{g+2} = P^{\mathcal{W}}_{g+1} E - L^{\mathcal{W}}_{(0,g+2)} - L^{\mathcal{W}}_{(2,g)} - L^{\mathcal{W}}_{(g,2)} \tag{23}$$

$$P^{\mathcal{U}}_{g+3} = P^{\mathcal{U}}_{g+2} E - L^{\mathcal{U}}_{(1,g+2)} - L^{\mathcal{U}}_{(3,g)} - L^{\mathcal{U}}_{(g+1,2)} \tag{24}$$

$$P^{\mathcal{W}}_{g+3} = P^{\mathcal{W}}_{g+2} E^T - L^{\mathcal{W}}_{(1,g+2)} - L^{\mathcal{W}}_{(3,g)} - L^{\mathcal{W}}_{(g+1,2)}. \tag{25}$$

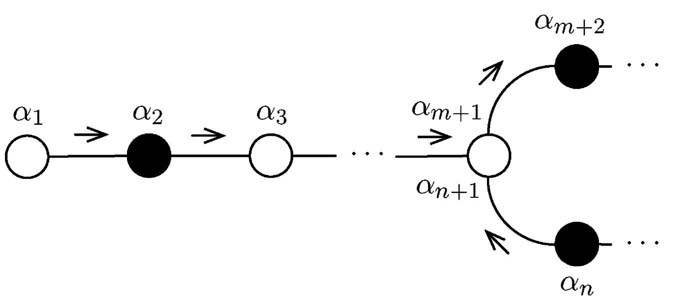

Fig. 3. A (m,n-m)-lollipop walk from α_1 to $\alpha_{n+1} = \alpha_{m+1}$ where m is even.

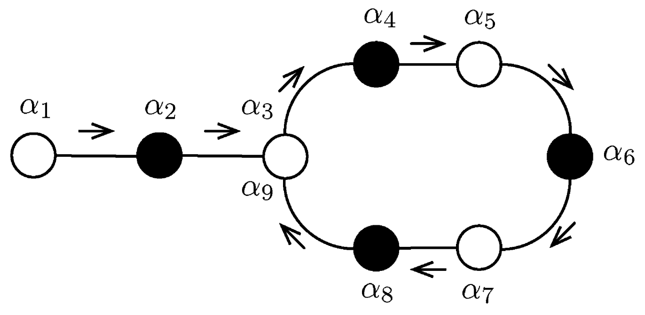

Fig. 4. A (2,6)-lollipop walk from α_1 to $\alpha_9 = \alpha_3$.

Expressions for the $(m, n-m)$-lollipop walk matrices in (14)–(25) for which $m > 0$ are given in Section V. In the following section, a general proof structure for the lollipop matrix equations is detailed.

IV. Lollipop Matrix Proof Structure

Fig. 3 illustrates an $(m, n-m)$-lollipop walk where m is even.[2] Throughout the remainder of this correspondence, lollipop walks are illustrated with vertices from class $\mathcal{U}$ represented by white circles and vertices from class $\mathcal{W}$ represented by black circles. The lollipop matrix $L^{\mathcal{U}}_{(m,n-m)}$ thus counts walks of the form shown in Fig. 3.

The product $EL^{\mathcal{W}}_{(m-1,n-m)}$ counts $(m, n-m)$-lollipop walks but also walks where $\alpha_1 = \alpha_i$ for $i \in [3, n-1]_o$. In order to count $(m, n-m)$-lollipop walks, the inclusion/exclusion principle that was used to counts paths in (7)–(10) is applied: expressions for walks where $\alpha_1 = \alpha_i$ for $i \in [3, n-1]_o$ are found and subtracted (or excluded) from the product $EL^{\mathcal{W}}_{(m-1,n-m)}$.

As an example, consider the case where $m = 2$ and $n = 8$ illustrated in Fig. 4. The product $EL^{\mathcal{W}}_{(1,6)}$ counts both $(2,6)$-lollipop walks and walks where $\alpha_1 = \alpha_3, \alpha_1 = \alpha_5$ and $\alpha_1 = \alpha_7$. In order to count $(2,6)$-lollipop walks, expressions for walks where $\alpha_1 = \alpha_3, \alpha_1 = \alpha_5$ and $\alpha_1 = \alpha_7$ are found and then subtracted (or excluded) from the product $EL^{\mathcal{W}}_{(1,6)}$.

As m and n get large, the number of terms that must be excluded from the product $EL^{\mathcal{W}}_{(m-1,n-m)}$ also gets large. However, many of these terms are zero as they violate the girth constraint. Returning to the $m = 2, n = 8$ example illustrated in Fig. 4, if $g = 6$ then neither walks where $\alpha_1 = \alpha_5$ nor $\alpha_1 = \alpha_7$ are possible since these cases require the existence of a length-4 cycle.

V. Lollipop Matrix Equations

In the following, only expressions for the $L^{\mathcal{U}}_{(m,n-m)}$ lollipop matrices are given explicitly. In order to obtain an expression for $L^{\mathcal{W}}_{(m,n-m)}$ from the corresponding $L^{\mathcal{U}}_{(m,n-m)}$ expression, exchange all $\mathcal{U}$'s and $\mathcal{W}$'s and replace E with E^T. For example

$$L^{\mathcal{U}}_{(1,2)} = E \max\left(L^{\mathcal{W}}_{(0,2)} - 1, 0\right) \tag{26}$$

becomes

$$L^{\mathcal{W}}_{(1,2)} = E^T \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) \tag{27}$$

and

$$L^{\mathcal{U}}_{(2,2)} = \mathcal{Z}\left[EL^{\mathcal{W}}_{(1,2)}\right] \tag{28}$$

becomes

$$L^{\mathcal{W}}_{(2,2)} = \mathcal{Z}\left[E^T L^{\mathcal{U}}_{(1,2)}\right]. \tag{29}$$

Equations (30) and (31) hold for $1 < k < \frac{g}{2}$

$$L^{\mathcal{U}}_{(2k-1,2)} = EL^{\mathcal{W}}_{(2k-2,2)} - \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) L^{\mathcal{U}}_{(2k-3,2)} \tag{30}$$

$$L^{\mathcal{U}}_{(2k,2)} = EL^{\mathcal{W}}_{(2k-1,2)} - \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) L^{\mathcal{U}}_{(2k-2,2)}. \tag{31}$$

Note that in (32)–(38), Kronecker delta notation, where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise, is employed in order to present the $g = 4, g = 6$, and $g > 6$ cases in a unified fashion

$$\begin{aligned} L^{\mathcal{U}}_{(g-1,2)} = {} & EL^{\mathcal{W}}_{(g-2,2)} - P^{\mathcal{U}}_{g-1} \circ E \\ & - \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) L^{\mathcal{U}}_{(g-3,2)} \end{aligned} \tag{32}$$

$$\begin{aligned} L^{\mathcal{U}}_{(g,2)} = {} & \mathcal{Z}\left[EL^{\mathcal{W}}_{(g-1,2)}\right] \\ & - \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) L^{\mathcal{U}}_{(g-2,2)} \\ & + P^{\mathcal{U}}_{g-2} \circ P^{\mathcal{U}}_2 - \delta_{g,4} P^{\mathcal{U}}_2 \end{aligned} \tag{33}$$

$$\begin{aligned} L^{\mathcal{U}}_{(g+1,2)} = {} & EL^{\mathcal{W}}_{(g,2)} - L^{\mathcal{U}}_{(0,g)} L^{\mathcal{U}}_{(1,2)} - P^{\mathcal{U}}_{g+1} \circ E \\ & - \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) L^{\mathcal{U}}_{(g-1,2)} \\ & + 2\left[P^{\mathcal{U}}_{g-1} \max\left(L^{\mathcal{W}}_{(0,2)} - 2, 0\right)\right] \circ E \\ & + L^{\mathcal{U}}_{(g-1,2)} \circ E + 2P^{\mathcal{U}}_{g-1} \circ E \\ & + 2\delta_{g,4}\left[\binom{P^{\mathcal{U}}_2}{2} E - P^{\mathcal{U}}_3\right] \circ E \end{aligned} \tag{34}$$

$$L^{\mathcal{U}}_{(1,g)} = EL^{\mathcal{W}}_{(0,g)} - 2P^{\mathcal{U}}_{g-1} \circ E \tag{35}$$

$$L^{\mathcal{U}}_{(2,g)} = \mathcal{Z}\left[EL^{\mathcal{W}}_{(1,g)}\right] - 6\delta_{g,4}\binom{P^{\mathcal{U}}_2}{3} \tag{36}$$

$$\begin{aligned} L^{\mathcal{U}}_{(3,g)} = {} & EL^{\mathcal{W}}_{(2,g)} - \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) L^{\mathcal{U}}_{(1,g)} \\ & - 6\delta_{g,6}\binom{P^{\mathcal{U}}_3}{3} - 4\delta_{g,4}\binom{P^{\mathcal{U}}_3}{2} \circ E \\ & + 6\delta_{g,4}\left[E\binom{P^{\mathcal{W}}_2}{2} - P^{\mathcal{U}}_3\right] \circ E \\ & + 4\delta_{g,4}\left[\binom{P^{\mathcal{U}}_2}{2} E - P^{\mathcal{U}}_3\right] \circ E \end{aligned} \tag{37}$$

$$\begin{aligned} L^{\mathcal{U}}_{(1,g+2)} = {} & EL^{\mathcal{W}}_{(0,g+2)} - 2P^{\mathcal{U}}_{g+1} \circ E \\ & - 2\delta_{g,4}\binom{P^{\mathcal{U}}_3}{2} \circ E \\ & + 2\delta_{g,4}\left[E\binom{P^{\mathcal{W}}_2}{2} - P^{\mathcal{U}}_3\right] \circ E \\ & + 2\delta_{g,4}\left[\binom{P^{\mathcal{U}}_2}{2} E - P^{\mathcal{U}}_3\right] \circ E. \end{aligned} \tag{38}$$

[2]The case where m is even is presented without loss of generality; the proposed proof technique also applies to the case where m is odd.

VI. Examples

In this section, the proposed cycle counting algorithm is used to count the short cycles in a number of Tanner graphs.

A. A Golay Code Tanner Graph

Consider the $(23,12)$ Golay code with systematic parity check matrix shown in (39) at the bottom of the page. The cycle counting algorithm computes $N_4 = 452, N_6 = 10\,319$, and $N_8 = 199\,994$ in the correpsonding Tanner graph (which has incidence matrix $E = H_G$).

B. A Small LDPC Tanner Graph

Consider the small LDPC code due to [19] with parity check matrix shown in (40) at the bottom of the page. This code was designed to have a Tanner graph with girth 12. Accordingly, the cycle counting algorithm computes $N_4 = N_6 = N_8 = N_{10} = N_{14} = 0, N_{12} = 28$, and $N_{16} = 21$ in the corresponding Tanner graph.

C. LDPC Tanner Graphs

In this example, the relationship between cycle structure and performance for two sets of LDPCs is examined. First consider two rate-$1/2$ Gallager codes with block length 816. Definitions for both codes are due to MacKay [20]. Code A has column weight 3 and corresponds to MacKay's code $816.3.174$ while Code B has column weight 5 and corresponds to MacKay's code $816.55.178$. The short cycle structure of the Tanner graphs corresponding to these codes is tabulated in Table I. Codes A and B have the same girth $g = 6$ but Code B has many more short cycles. The effect of these cycles on code performance is dramatic: Code A outperforms Code B by approximately 0.75 dB at a bit-error rate of 10^{-5} [20] (on the additive white Gaussian noise (AWGN) channel with binary antipodal signaling).

TABLE I
SHORT CYCLE STRUCTURE OF THE CODE A AND B TANNER GRAPHS

	Code A	Code B
N_6	132	7921
N_8	1494	210740
N_{10}	9278	6054731

TABLE II
SHORT CYCLE STRUCTURE OF THE CODE C AND D TANNER GRAPHS

	Code C	Code D
N_6	-	13244
N_8	802	420609
N_{10}	11279	13567791
N_{12}	86791	-

Next consider two rate-$1/2$, block length 504 LDPCs constructed using the progressive-egde growth method due to Hu *et al.* [21]. Codes C and D correspond to MacKay's codes $PEGReg252x504$ and $PEGirReg252x504$, respectively. The short cycle structure of the Tanner graphs corresponding to these codes is tabulated in Table II. Given that the Tanner graph corresponding to Code C has larger girth, one may initially expect it to outperform Code D. However, Code D outperforms Code C by approximately 0.5 dB at a bit-error rate of 10^{-5} [21] (on the AWGN channel with binary antipodal signaling). This seeming contradiction is explained by examining the regularity of the respective Tanner graph cycle structures. Table III tabulates the mean μ_k and standard deviation σ_k of the number of cycles of length k incident on each code bit vertex in the Tanner graphs corresponding to Codes C and D. The small σ_k values for Code C indicate an extremely regular cycle distribution whereas the σ_k values are very large for Code D indicating a much more irregular, or random, cycle distribution. In this example, the negative impact of cycle regularity

$$H_G = \begin{bmatrix}
1&0&0&1&1&1&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&0&0&0&0&0\\
1&0&1&1&0&1&1&0&1&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\
1&0&1&1&1&0&1&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&1&0&0&1&1&1&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&1&0&1&0&1&1&1&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0\\
1&1&0&1&1&0&0&1&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
1&1&1&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&1&1&0&1&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&1&0&0\\
1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix} \quad (39)$$

$$H_L = \begin{bmatrix}
1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0
\end{bmatrix} \quad (40)$$

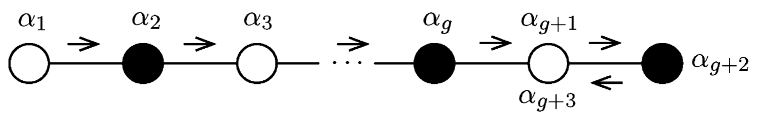

Fig. 5. A $(g, 2)$-lollipop walk from α_1 to $\alpha_{g+3} = \alpha_{g+1}$.

TABLE III
SHORT CYCLE STATISTICS OF THE CODE C AND D TANNER GRAPHS

	Code C		Code D	
	μ_k	σ_k	μ_k	σ_k
$k = 6$	-	-	26.3	55.3
$k = 8$	1.6	0.7	834.5	1767.2
$k = 10$	22.4	1.8	26920.2	56369.6
$k = 12$	172.2	8.3	-	-

on the performance of Code C is greater than the positive impact of large girth.

It is clear from the above examples that girth alone is not enough to judge the quality of a LDPC design. The *number* and *statistics* of the short cycles are both important metrics for quality.

On a 2.5-GHz Apple Power Macintosh G5 Computer (with 2 GB of RAM), the short cycles in Codes A, B, C, and D were counted in 5.03, 4.72, 1.94, and 1.43 s, respectively. Note that the Altivec-enabled BLAS library was used for fast matrix multiplication [22].

VII. COMPLEXITY

The time complexity of the proposed algorithm is dominated by matrix multiplication and may be estimated by counting the number of matrix multiplications required to compute the number of cycles of length $g, g+2$, and $g+4$ in a bipartite graph with girth $g > 6$. Carefully counting the total number of matrix multiplications yields an expression that is linear in g

$$6g + 37.$$

The time complexity of each matrix multiplication grows as $\mathcal{O}(n^3)$ where $n = \max(|\mathcal{U}|, |\mathcal{W}|)$. The time complexity of counting short cycles thus grows as $\mathcal{O}(gn^3)$.

The space complexity of the proposed algorithm is dominated by matrix storage which grows as $\mathcal{O}(n^2)$. More specifically, computing the number of cycles of length $g, g+2$, and $g+4$ in a bipartite graph with girth g and with vertex class sizes $|\mathcal{U}|$ and $|\mathcal{W}|$ requires at most

$$11|\mathcal{U}|^2 + 11|\mathcal{W}|^2 + 21|\mathcal{U}||\mathcal{W}|$$

storage locations. Note that high bit-width storage locations (e.g., 64-bit integer or double-precision floating point) must be used in order to accommodate large dynamic ranges in the matrix entries. Also note that the space complexity of the proposed algorithm does not grow with girth.

VIII. NOTES

A technical note containing detailed proofs for (26)–(38) and a C++ implementation of the proposed algorithm are available at: http://csi.usc.edu/chugg/tools.

APPENDIX

A. *Proof of (33)*

$$L^{\mathcal{U}}_{(g,2)} = \mathcal{Z}\left[EL^{\mathcal{W}}_{(g-1,2)}\right] - \max\left(L^{\mathcal{U}}_{(0,2)} - 1, 0\right) L^{\mathcal{U}}_{(g-2,2)} + P^{\mathcal{U}}_{g-2} \circ P^{\mathcal{U}}_2 - \delta_{g,4} P^{\mathcal{U}}_2.$$

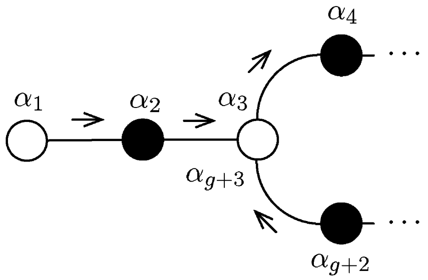

Fig. 6. A $(2, g)$-lollipop walk from α_1 to $\alpha_{g+3} = \alpha_3$.

Walks of the form shown in Fig. 5 from α_1 to $\alpha_{g+3} = \alpha_{g+1}$ are counted by $L^{\mathcal{U}}_{(g,2)}$. The product $EL^{\mathcal{W}}_{(g-1,2)}$ counts $(g,2)$-lollipop walks, walks where $\alpha_1 = \alpha_3$, and walks where $\alpha_1 = \alpha_{g+1}$. Walks where $\alpha_1 = \alpha_i$ for $i \in [5, g-1]_o$ require the existence of a cycle of length less than g and thus violate the girth constraint.

1) Walks Where $\alpha_1 = \alpha_3$: The product $L^{\mathcal{U}}_{(0,2)} L^{\mathcal{U}}_{(g-2,2)}$ counts walks where $\alpha_1 = \alpha_3$ along with walks where $\alpha_1 = \alpha_3$ and $\alpha_2 = \alpha_4$ and walks where $\alpha_1 = \alpha_3$ and $\alpha_2 = \alpha_{g+2}$. Note that walks where $\alpha_1 = \alpha_3$ and $\alpha_2 = \alpha_j$ for $j \in [6, g]_e$ violate the girth constraint. Walks where $\alpha_2 = \alpha_4$ are excluded by instead considering the product $\max(L^{\mathcal{U}}_{(0,2)} - 1, 0) L^{\mathcal{U}}_{(g-2,2)}$. A walk where $\alpha_2 = \alpha_{g+2}$ requires that α_1 and α_{g+1} be connected by both a path of length $g-2$ and a path of length 2. The direct product $P^{\mathcal{U}}_{g-2} \circ P^{\mathcal{U}}_2$ counts such walks.

2) Walks Where $\alpha_1 = \alpha_{g+1}$: Eliminating the diagonal elements of $EL^{\mathcal{W}}_{(g-1,2)}$ with the $\mathcal{Z}[\cdot]$ operator removes the $\alpha_1 = \alpha_{g+1}$ walks.

3) The Special Case of $g = 4$: When $g = 4$, the direct product $P^{\mathcal{U}}_2 \circ P^{\mathcal{U}}_2$ includes pairs of identical length–2 paths. This double counting is remedied by replacing $P^{\mathcal{U}}_2 \circ P^{\mathcal{U}}_2$ with $2\binom{P^{\mathcal{U}}_2}{2}$ or, equivalently, $P^{\mathcal{U}}_2 \circ P^{\mathcal{U}}_2 - P^{\mathcal{U}}_2$.

B. *Proof of (36)*

$$L^{\mathcal{U}}_{(2,g)} = \mathcal{Z}\left[EL^{\mathcal{W}}_{(1,g)}\right] - 6\delta_{g,4}\binom{P^{\mathcal{U}}_2}{3}.$$

Walks of the form shown in Fig. 6 from α_1 to $\alpha_{g+3} = \alpha_3$ are counted by $L^{\mathcal{U}}_{(2,g)}$. When $g > 4$, the product $EL^{\mathcal{W}}_{(1,g)}$ counts $(2,g)$-lollipop walks and walks where $\alpha_1 = \alpha_3$. Note that walks where $\alpha_1 = \alpha_i$ for $i \in [5, g+1]_o$ violate the girth constraint when $g > 4$. Eliminating the diagonal elements of $EL^{\mathcal{W}}_{(g-1,2)}$ with the $\mathcal{Z}[\cdot]$ operator removes the $\alpha_1 = \alpha_3$ walks.

When $g = 4$, walks where $\alpha_1 = \alpha_5$ do not violate the girth constraint and must also be excluded from the product $EL^{\mathcal{W}}_{(1,g)}$. A walk where $\alpha_1 = \alpha_5$ requires that α_1 and α_3 be connected by three distinct length2 paths. The expression $6\binom{P^{\mathcal{U}}_2}{3}$ counts such walks.

REFERENCES

[1] T. R. Halford and K. M. Chugg, “Enumerating and Counting Cycles in Bipartite Graphs,” Communication Sciences Institute, USC, Los Angeles, CA, Tech. Rep. CSI-04-05-02, May 2004. Presented at the 2004 Communication Theory Workshop, Capri, Italy.

[2] R. G. Gallager, "Low density parity check codes," *IEEE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.

[3] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1660–1686, Nov. 1996.

[4] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEE Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.

[5] N. Wiberg, "Codes and Decoding on General Graphs," Ph.D. dissertation, Linköping University, Linköping, Sweden, 1996.

[6] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," in *Proc. IEEE Symp. Information Theory*, Adelaide, Australia, Sep. 1995, p. 468.

[7] H. Jin and R. J. McEliece, "General coding theorems for turbo-like codes," in *Proc. IEEE Symp. Information Theory*, Sorrento, Italy, Jun./Jul. 2000, p. 120.

[8] Y. Mao and A. H. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," in *Proc. Int. Conf. Communications*, vol. 1, Helsinki, Finland, Jun. 2001, pp. 41–44.

[9] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Communun.*, vol. 44, no. 5, pp. 591–600, May 1996.

[10] K. M. Chugg, A. Anastasopoulos, and X. Chen, *Iterative Detection: Adaptivity, Complexity Reduction, and Applications*. Norwell, MA: Kluwer, 2001.

[11] S. Crozier, "New high-spread high-distance interleavers for turbo-codes," in *Proc. 20th Biennial Symp. Communications*, Kingston, ON, Canada, May 2000, pp. 3–7.

[12] T. Richardson, M. A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity check codes," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[13] J. Flum and M. Grohe, "The parameterized complexity of counting problems," in *Proc. IEEE Symp. Foundations of Computer Science*, Vancouver, BC, Canada, Nov. 2002, pp. 538–547.

[14] N. Alon, R. Yuster, and U. Zwick, "Finding and counting given length cycles," *Algorithmica*, vol. 17, no. 3, pp. 209–223, 1997.

[15] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[16] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[17] R. Diestel, *Graph Theory*, 2nd ed. New York: Springer-Verlag, 2000.

[18] A. G. Thomason, "Hamiltonian cycles and uniquely edge-colorable graphs," *Ann. Discr. Math.*, vol. 3, pp. 259–268, 1978.

[19] H. Zhang and J. M. F. Moura, "The design of structured regular LDPC codes with large girth," in *Proc. GLOBECOM Conf.*, Taipei, Taiwan, Dec. 2003, pp. 4022–4027.

[20] D. J. C. MacKay. Encyclopedia of Sparse Graph Codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[21] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[22] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley, "An updated set of basic linear algebra subprograms (BLAS)," *ACM Trans. Math. Software*, vol. 28, no. 2, pp. 135–151, June 2002.

Performance of Low-Density Parity-Check Codes With Linear Minimum Distance

Hossein Pishro-Nik, *Member, IEEE,* and
Faramarz Fekri, *Senior Member, IEEE*

***Abstract*—This correspondence studies the performance of the iterative decoding of low-density parity-check (LDPC) code ensembles that have linear typical minimum distance and stopping set size. We first obtain a lower bound on the achievable rates of these ensembles over memoryless binary-input output-symmetric channels. We improve this bound for the binary erasure channel. We also introduce a method to construct the codes meeting the lower bound for the binary erasure channel. Then, we give upper bounds on the rate of LDPC codes with linear minimum distance when their right degree distribution is fixed. We compare these bounds to the previously derived upper bounds on the rate when there is no restriction on the code ensemble.**

***Index Terms*—Bipartite graphs, erasure channel, error floor, iterative decoding, low-density parity-check (LDPC) codes, minimum distance, performance bound.**

I. Introduction

In some applications, it is necessary to design codes that do not suffer from the error floor problem at the desired bit error rates (BERs), while their rates are close to the channel capacity. For example, in some page-oriented memories, low-density parity-check (LDPC) codes can result in very efficient coding schemes [1]. In these memory systems, we can use large block lengths and thus we get performance close to the Shannon limit. However, BER's less than 10^{-12} are required. Since the storage capacity of the system is directly proportional to the code rate, it is very important that the code rate be close to the capacity of the channel. Thus, we need to design LDPC codes that do not show error floor for the BERs higher than 10^{-12}, and at the same time, have a threshold near the Shannon limit.

One method to solve the error floor problem is to use an outer code. In this method we use the outer code to reduce the BER. This method slightly increases the complexity of the system. This is specifically undesirable in page-oriented memories where simple and fast decoding algorithms are required. Moreover, using an outer code results in a rate loss; however, the rate loss is usually small. There are also methods for decreasing the error-floor effect for the capacity-approaching codes [2]; however, these methods are sometimes not effective for the BERs required by the storage systems. Depending on the application, the above methods may or may not be suitable. As it will be described in more detail, an alternative option is to use codes with linear minimum distance. These codes also have some desirable properties other than good error floor performance. Thus, in this paper our aim is to study codes with linear minimum distance and to find bounds on their achievable rates.

Manuscript received February 12, 2004; revised June 27, 2005. This work was supported in part by the Air Force Office of Scientific Research under Grant F49620-02-1-0053.

H. Pishro-Nik is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003-9292 USA (e-mail: pishro@ecs.umass.edu).

F. Fekri is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: fekri@ece.gatech.edu).

Communicated by A. E. Ashikhmin, Associate Editor of Coding Theory.
Digital Object Identifier 10.1109/TIT.2005.860439