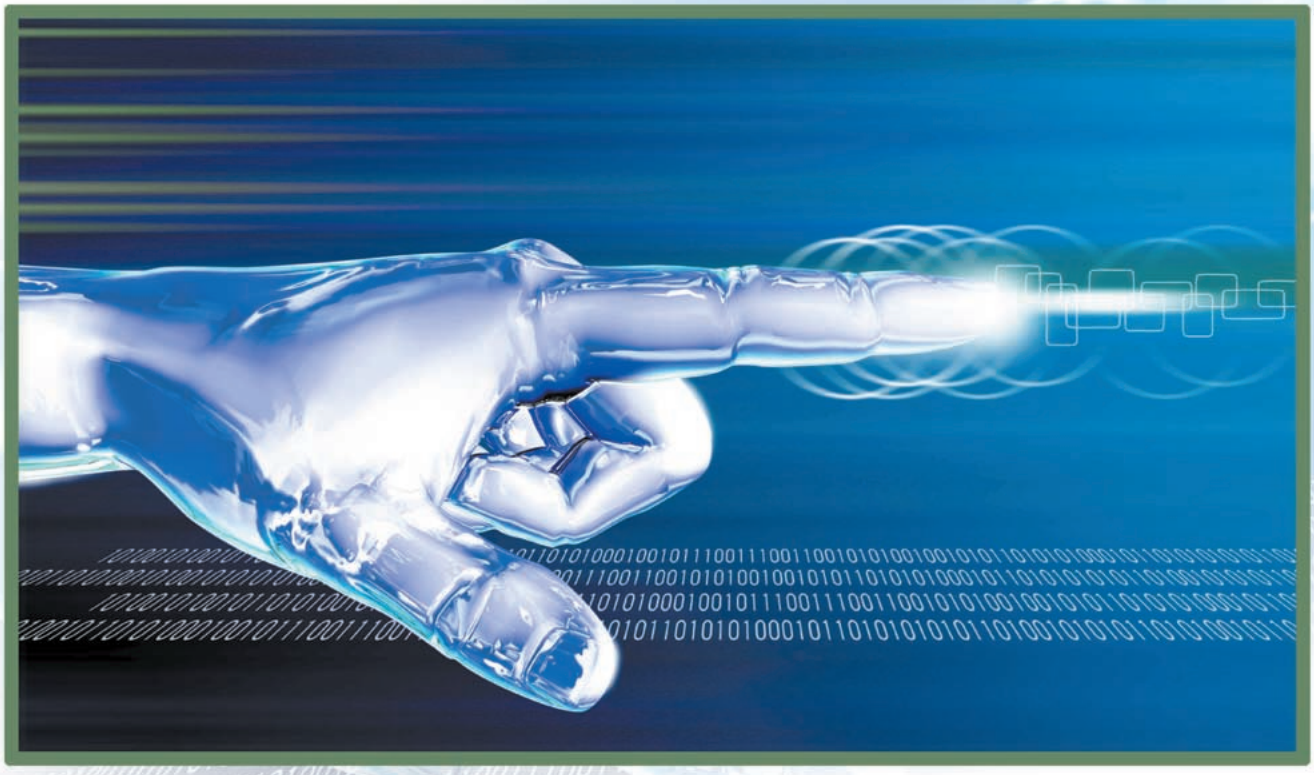
Structured Low-Density Parity-Check Codes

Methods to design regular LDPC codes with large girth.



José M.F. Moura, Jin Lu, and Haotian Zhang

We consider the problem of designing unoriented bipartite graphs with large girth. These graphs are the Tanner graphs associated with the parity-check matrix **H** of low density parity-check (LDPC) codes or Gallager codes. Larger girth improves the computational and bit error rate (BER) performance of these codes. The article overviews several existing methods in the literature and then describes two new constructions for LDPC codes with large girth—geometry based and turbo structured LDPC codes. Simulation results show the potential improvement in the error floor that such codes provide over randomly generated LDPC codes.

The Origin of LDPC Codes

LDPC codes were originally introduced by Gallager in his doctoral thesis; see [1]. Since the discovery of turbo codes in 1993 by Berrou

1053-5888/04/$20.00©2004IEEE

et al. [2] and the rediscovery of LDPC codes by Mackay and Neal in 1995 (see [3]), there has been renewed interest in turbo codes and LDPC codes because their bit error rate performance approaches asymptotically the Shannon limit [4]. Much research is devoted to characterizing the performance of LDPC codes and designing codes that have good performance. Commonly, a graph, the Tanner graph, is associated with the code and an important parameter affecting the performance of the code is the girth of its Tanner graph. In this article we consider the design of structured regular LDPC codes whose Tanner graphs have large girth. The regularity and structure of LDPC codes utilize memory more efficiently and simplify the implementation of LDPC coders. The Tanner graph is a special type of graph, a bipartite graph, where the nodes divide into two disjoint classes with edges only between nodes in the two different classes. The problem we consider then is a generic problem in graph theory, namely, that of designing bipartite graphs with large girth. We actually consider a more special class of this generic problem, in particular, the design of undirected regular bipartite graphs with large girth.

Graphs and Codes

In this section we review a few needed concepts in linear codes, low density parity-check codes, and graphs.

Linear Block Codes

In communications, a dataword is a sequence of p user bits. An (n, p) linear block code $\mathcal{C}$ with dataword length p and codeword (block) length n over the binary field $\mathcal{F}_2$ can be regarded as a p-dimensional subspace of the n-dimensional vector space $V_n(\mathcal{F}_2)$ over the binary field $\mathcal{F}_2$, [5]. There are 2^p datawords $\bar{u} = [u_0, u_1, \ldots, u_{p-1}]$ in the code $\mathcal{C}$, and each of them corresponds to a unique codeword $\bar{c} = [c_0, c_1, \ldots, c_{n-1}]$. Note that $\bar{u}$ and $\bar{c}$ are row vectors, not column vectors. Since $\mathcal{C}$ is a subspace of dimension p, there exist p linearly independent vectors $\bar{g}_0, \bar{g}_1, \ldots, \bar{g}_{p-1}$ that span $\mathcal{C}$. Therefore, each codeword can be written as a linear combination of these independent vectors: $\bar{c} = \bar{u}\,\mathbf{G}$, where $\mathbf{G} = [\bar{g}_0\ \bar{g}_1\ \ldots\ \bar{g}_{p-1}]^T$ is the $p \times n$ generator matrix [6]. The null space of the subspace $\mathcal{C}$, i.e., the orthogonal subspace of $\mathcal{C}$ with dimension $m = n - p$, can be spanned by m linearly independent vectors $\bar{h}_0, \bar{h}_1, \ldots, \bar{h}_{m-1}$. Each codeword $\bar{c}$ satisfies the parity-check equations $\forall i : \bar{c}\bar{h}_i^T = 0$. The $m \times n$ parity-check matrix $\mathbf{H}$ collects these $\bar{h}_i$ in matrix format as $\mathbf{H} = [\bar{h}_0\ \bar{h}_1\ \ldots\ \bar{h}_{m-1}]^T$, [6].

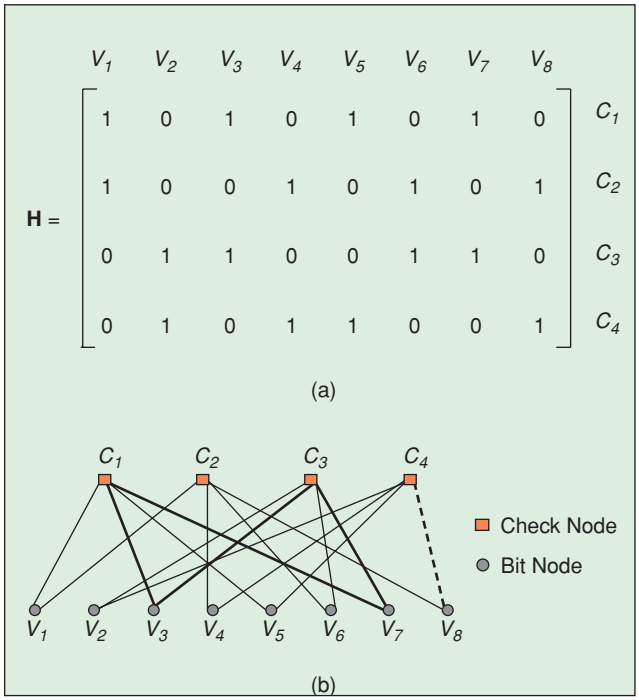
LDPC Codes

An LDPC code is a special class of linear block codes whose parity-check matrix $\mathbf{H}$ has a low density of ones, i.e., is sparse. This sparsity renders low complexity decoding and leads to simple implementations. We will emphasize regular (n, j, k) LDPC codes, [7], where the n columns and m rows of its parity-check matrix $\mathbf{H}$ all have the same number of ones, j and k, respectively, i.e., the columns have uniform weight j and the rows have uniform weight k, where $k \ll n$ and $j \ll m = n - p$. By counting the number of ones in $\mathbf{H}$, it follows that $nj = mk$. The code rate of $\mathbf{H}$ is $r = p/n = (n - m)/n = (k - j)/k = 1 - j/k$. Note that the equation is true only if $\mathbf{H}$ is full rank. If $\mathbf{H}$ is not full rank, the code rate is $r = (n - \text{rank}(\mathbf{H}))/n$.

Tanner Graph

To the LDPC code we can associate a graph $\mathcal{G}$, referred to as its Tanner graph [8]. For the sake of completeness, we recall basic definitions from graph theory. Let $\mathcal{G} = \{(\mathcal{V}, \mathcal{E})\}$ be a graph, where $\mathcal{V}$ is a set of vertices or nodes V and $\mathcal{E}$ is a set of edges E connecting the vertices. The degree of a node V is the number of edges incident on V. In an undirected graph, a series of successive edges forming a continuous curve passing from one vertex to another is called a chain. A chain of nodes where the initial and the terminal nodes are the same and that does not use the same edge more than once is a cycle. The length of a cycle is the number of its edges; and the girth g of $\mathcal{G}$ is the length of the shortest cycle. The graph $\mathcal{G}$ is bipartite if the set of vertices $\mathcal{V}$ can be decomposed into two disjoint sets $\mathcal{V}_1$ and $\mathcal{V}_2$ such that no two vertices within either $\mathcal{V}_1$ or $\mathcal{V}_2$ are connected by an edge. It is well known from graph theory that a graph with at least two nodes is bipartite if and only if all its cycles are of even length.

Tanner graphs are bipartite where the two disjoint sets $\mathcal{V}_1$ or $\mathcal{V}_2$ collect the bit nodes and the check nodes: each bit of a codeword is assigned a bit node, and each parity-check equation is assigned a check node. We illustrate the relation between the $m \times n$ parity-check matrix $\mathbf{H}$ and its Tanner graph $\mathcal{G}$ with reference to Figure 1. The figure shows a 4×8 $\mathbf{H}$ matrix and its



▲ *1. (a) The parity-check matrix* ***H*** *of an LDPC code. (b) Corresponding Tanner graph* $\mathcal{G}$.

Tanner graph $\mathcal{G}$, respectively. The m rows of $\mathbf{H}$ correspond to the check nodes and the n columns to the bit nodes of the Tanner graph $\mathcal{G}$; they are represented by filled squares and filled circles in Figure 1, respectively. A 1 in row C_i and column V_j in $\mathbf{H}$ is represented by an edge in the Tanner graph connecting the associated check node C_i and the bit node V_j; for example, the 1 in row 4 and column 8 in $\mathbf{H}$ in Figure 1 is illustrated by the dashed line between C_4 and V_8 in Figure 1. The bold solid lines (C_1, V_3), (V_3, C_3), (C_3, V_7), and (V_7, C_1) depict a cycle in the Tanner graph; this turns out to be the shortest cycle in this graph, so that its girth is $g = 4$.

Decoding

The decoding of LDPC codes fits the general framework of Tanner graphs and the sum-product algorithm [9], which is an iterative message passing algorithm operating on the Tanner graph; see [10].

Cycles, Girth, and Performance

It is well known that with a cycle-free Tanner graph, the sum-product algorithm terminates naturally in a finite number of steps and yields optimal decoding in the sense that the symbol error probability is minimized [9], [11]. However, cycle-free Tanner graphs have poor BER performance: their minimum distance is two at code rates $r > 1/2$, and their error floors occur at unacceptable values of SNR [12]. When Tanner graphs have cycles, the resulting sum-product algorithm is suboptimal [9], [11]. Cycles, especially short cycles in the Tanner graph, lead to inefficient decoding and prevent the sum-product algorithm from converging to the optimal decoding result [13], [12], taxing the performance of the LDPC decoders. We discuss briefly why the code performance is affected by short cycles. Intuitively, the girth determines the smallest number of iterations for a message sent by a node (in the shortest cycle of the graph) to propagate back to the node itself. This causes loss of independence in the extrinsic information merged on a node in the iterative decoding through the successive iterations; Gallager [1] showed that the number of independent iterations M is proportional to the girth g of the Tanner graph, in particular, $M < g/4 \leq M + 1$. A second reason relates to the minimum distance of the code. Tanner [8] derives a lower bound on the minimum distance $d_{\min}$; this lower bound increases with the girth g of the code. Therefore, LDPC codes with large girth are to be preferred.

To increase the girth g and avoid short cycles, the parity-check matrix $\mathbf{H}$ should be sufficiently sparse. In turn, this means that the block length must be large enough. In fact, Gallager [1] provides a loose lower bound on the block length n given the girth g of a regular LDPC code. For a parity-check matrix $\mathbf{H}$ with column weight j and row weight k, the bound on the block length n is

▲ 1) for $g = 4m + 2$, $n \geq \sum_1^{m+1} S_i$ where $S_1 = 1$ and for $i \geq 2$, $S_i = j(j-1)^{(i-2)}(k-1)^{(i-1)}$

▲ 2) for $g = 4m$, $n \geq \sum_1^m L_i$, where $L_i = k(j-1)^{(i-1)}(k-1)^{(i-1)}, \forall i$.

For example, to construct an $(n, j = 3, k = 12)$ LDPC code with girth $g = 12$, its block length n should satisfy $n \geq 6,084$. Of course, this lower bound is just that; even if n exceeds the bound for a given g^* there might not exist a regular LDPC code with this block length n and the desired girth g^*.

Two alternative approaches to designing LDPC codes with girth $g > 4$ are described in later sections. Due to the overview and introductory nature of the article, we focus specifically on examples of codes with column weight $j = 2$ and $4 < g \leq 20$ and codes with $j = 3$ and $4 < g \leq 8$. Codes with $j = 2$ are usually of less interest in digital communications systems because, as Gallager proved in his original work [1], the corresponding $d_{\min}$ grows only logarithmically fast with the block size n of the code, while for codes with $j \geq 3$, $d_{\min}$ grows linearly with n.

Constructions for Large Girth LDPC Codes: Brief Overview of the Literature

Designing LDPC codes with large girth is a combinatorial design problem. Given four positive integers m, n, j, k such that $mk = nj$, we are interested in constructing an $m \times n$ parity-check matrix $\mathbf{H}$ with uniform column weight j and uniform row weight k such that the associated Tanner graph has large girth g.

Search Methods

A straightforward method for designing such $\mathbf{H}$ is brute force. For an $m \times n$ parity-check matrix $\mathbf{H}$ with uniform column weight j, there are $\left[\binom{m}{j}\right]^n$ possible choices, which makes an exhaustive search computationally infeasible for values of n of practical interest. Further, a large number of choices are actually isomorphic to each other and lead to identical LDPC codes. Campello, Modha, and Rajagopalan [14], [15] provide a heuristic method called "bit-filling" to search for LDPC codes with large girth. The computational complexity of bit-filling is $\mathcal{O}(k_{\max} m^3)$ where $k_{\max}$ is the maximum degree of any check node and m is the number of rows in $\mathbf{H}$. The polynomial complexity makes bit-filling feasible to implement. The $\mathbf{H}$ matrix generated by bit-filling has uniform column weight j but nonuniform row weight k. Hence the LDPC codes constructed are *not* regular codes in strict sense. Also, there is no guarantee that codes with the largest possible girth g are constructed for a given n. Another heuristic algorithm (see [16]) searches for a good LDPC code based on the average of the girth distribution of the code. The complexity of this algorithm is shown in [16] to be $O(n^2)$. The algorithm described in [16] is suitable for designing short (10,000 bits or shorter) codes.

Finite Geometries

Kou et al. [17], [18] present a geometric approach to the design of LDPC codes based on the lines and points of Euclidean and projective geometries over finite fields [19], [20]. Codes construted from these finite geometries have no cycles of length four. By casting the design problem in the context of finite geometries, they obtain four classes of (quasi-) cyclic LDPC codes with girth $g = 6$.

Euclidean and projective geometries [19], [21] with n points and J lines satisfy the following structural properties:

▲ 1) each line is composed of ρ points

▲ 2) there is one and only one line between any two points

▲ 3) each point lies on γ lines

▲ 4) any pair of lines has only one common point or no common point.

Figure 2 gives a simple example of a finite geometry with $n = 4$, $J = 6$, $\rho = 2$, and $\gamma = 3$.

Associated with a finite geometry G is the $J \times n$ incidence matrix $H_G^{(1)} = [h_{i,j}]$ where each row and column correspond to a line and a point, respectively, and $h_{i,j} = 1$ if and only if the jth point is on the ith line in G; otherwise, $h_{i,j} = 0$. The row and column weights ρ and γ correspond to the ρ points in the corresponding line and to the γ lines passing the corresponding point, respectively. If $\rho \ll n$ and $\gamma \ll J$, the matrix $H_G^{(1)}$ can be regarded as a low-density parity-check matrix. The codes corresponding to $H_G^{(1)}$ are called type I geometry-G LDPC codes whose block length is n. Interchanging the roles of the rows and columns in the incidence matrix, the codes corresponding to $H_G^{(2)} = H_G^{(1)T}$ are called type II geometry-G LDPC codes.

References [17] and [18] used these finite-geometries to construct the above two types of LDPC codes. From the fundamental properties 2) and 4) of finite-geometries, these codes are free of four-cycles; they have a wide range of block lengths and code rates and achieve good minimum distances. These codes can be encoded in linear time and by simple feedback shift registers, and they can be extended and shortened in various ways to obtain other good LDPC codes. Simulation results in [17] show that with iterative decoding the performance of long extended finite-geometry LDPC codes are only a few tenths of a dB away from the Shannon limit.

Balanced Incomplete Block Designs

Recently, structured four-cycle-free regular LDPC codes have been constructed [22]–[28], based on combinatorial designs known as balanced incomplete block designs (BIBD) [29], a concept thoroughly studied since the end of the 19th century.

A BIBD with parameters (v, k, λ, r, b) is an ordered pair (X, A) in which a set X of v points is partitioned into a collection A of b subsets (blocks) in such a way

Large girth speeds the convergence of iterative decoding and improves the performance at least in the high SNR range, by slowing down the onset of the error floor.

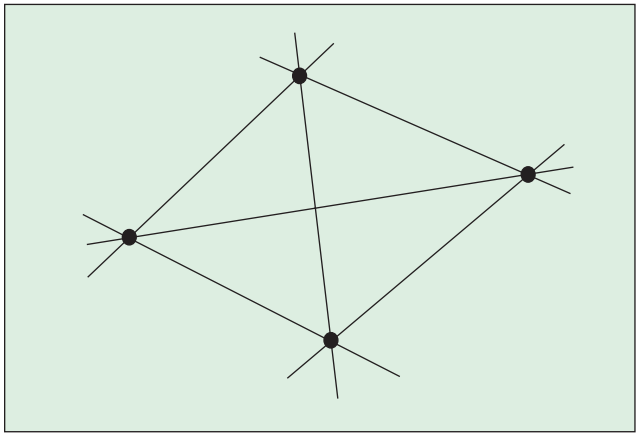
that any two points determine λ blocks with k points in each block and each point is contained in r different blocks [29]. Since $bk = vr$ and $\lambda(v-1) = r(k-1)$, only three of the five parameters are independent. Therefore, the notation (v, k, λ)-BIBD represents a BIBD on v points, block size k, and index λ.

Example 1: Let $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and A the collection of 12 three-element blocks: $A = \{(1,2,3), (4,5,6), (7,8,9), (1,4,7), (2,5,8), (3,6,9), (1,5,9), (2,6,7), (3,4,8), (1,6,8), (2,4,9), (3,5,7)\}$, the pair (X, A) is a $(9, 3, 1)$-BIBD [30].

The (v, k, λ)-BIBD with $k = 3$ and $\lambda = 1$ is called a Steiner triple system. The $(9, 3, 1)$-BIBD in the previous example is a Steiner triple system. Steiner triple systems exist for all $v = 1, 3 \bmod 6$. (We use both notations $x = y \bmod z$ and $x = (y, \bmod z)$.)

If a set of blocks partitions the point set X in a BIBD, the block set is a parallel class. A resolution of a BIBD is a partition of the family A of blocks into parallel classes. Clearly, a resolution is composed of exactly r parallel classes. A BIBD with at least one resolution is resolvable. A resolvable Steiner triple system is a Kirkman triple system. Kirkman triple systems exist for all $v = 3 \bmod 6$. For the $(9, 3, 1)$-BIBD in Example 1, which is also a Steiner triple system, the set A can be partitioned into four parallel classes as follows:

$\{(1, 2, 3), (4, 5, 6), (7, 8, 9)\}$, $\{(1, 4, 7), (2, 5, 8), (3, 6, 9)\}$, $\{(1, 5, 9), (2, 6, 7), (3, 4, 8)\}$, $\{(1, 6, 8), (2, 4, 9), (3, 5, 7)\}$



▲ 2. A finite geometry with $\rho = 2, \gamma = 3$.

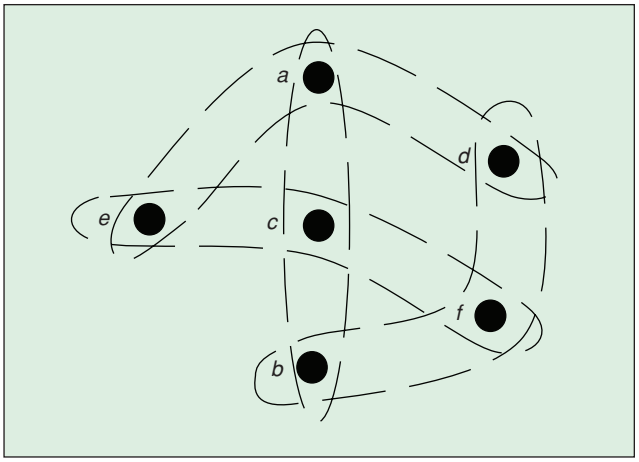
i.e., this Steiner triple system is resolvable, and so it is also a Kirkman triple system.

Following [25], define a point-block incidence matrix $H = [h_{ij}]_{v \times b}$, where $h_{ij} = 1$ if the ith-point in X occurs in the jth-block of A, and $h_{ij} = 0$ otherwise. For the (v, k, λ)-BIBD with $k \geq 2$ and $\lambda = 1$, if each point is regarded as a parity-check equation and each block is a bit in a linear block code, then $\mathbf{H}$ is a parity-check matrix of size $v \times b$ whose uniform row weight is r and the uniform column weight is k. If $r \ll v$ and $k \ll b$, $\mathbf{H}$ is the sparse parity-check matrix of an LDPC code. Figure 3 shows the parity-check matrix $\mathbf{H}$ corresponding to the $(9, 3, 1)$-BIBD in Example 1, where the rows from top to bottom correspond to the points $1, 2, \ldots, 9$ in X, and the columns from left to right correspond to the blocks in A.

In a parity-check matrix $\mathbf{H}$, two columns that have two 1's on the same rows correspond to a four-cycle in the associated Tanner graph. It is easy to see that for the point-block incidence matrix $\mathbf{H}$ arising from BIBDs with $\lambda = 1$ this cannot happen. Therefore their Tanner graphs are free of four-cycles.

$$\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

▲ *3. The **H** matrix corresponding to a (9, 3, 1)-BIBD.*



▲ *4. A Pasch configuration.*

Mackay and Davey [22] were the first to use BIBDs, in particular, Steiner triple systems, to design LDPC codes. Johnson and Weller [23], [24] present a family of $(n, 3, k)$ LDPC codes based on Kirkman triple systems. These codes are systematically constructed and are free of four-cycles. Vasic et al. [28] do further work on constructing four-cycle-free $(n, 3, k)$ LDPC codes based on Kirkman triple systems, present an efficient encoding algorithm for hardware implementation, and study their decoding performance by the sum-product algorithm in perpendicular magnetic read channels with different partial response targets and different types of noise. In [25], Vasic constructs structured four-cycle-free $(n, 3, k)$ LDPC codes based on Steiner triple systems and difference families of Abelian groups, proposes a hardware efficient encoding algorithm, and demonstrates that high-rate Steiner codes have a substantial performance gain over current schemes for magnetic recording systems. References [26] and [27] also construct four-cycle-free LDPC codes based on other BIBD techniques, such as mutually orthogonal Latin rectangles (MOLR) and anti-Pasch BIBDs. A Latin rectangle based on a set of n elements S is an $r \times s$ rectangular array, say A, with the property that each row of A is an s-permutation of elements of S and each column of A is an r-permutation of elements of S [31]. The Latin square is of order r if $r = s$. Two Latin rectangles are orthogonal, if no ordered pair occurs more than once when the rectangles are superimposed. A set of Latin rectangles is said to be mutually orthogonal if any two of the rectangles are orthogonal [32]. In [26], Vasic presents a family of LDPC codes based on MOLR and analyzes their decoding performance in perpendicular magnetic read channels.

An (n, k) configuration in a BIBD is a subset of k blocks of A whose union is an n-element subset of X. A Pasch configuration [33], or a quadrilateral, is a $(6, 4)$ configuration in a Steiner triple system. Figure 4 shows an example of a Pasch configuration, where a, b, c, d, e, f is a six-subset of points of X, and $a, b, c, a, d, e, b, d, f, c, e, f$ is a subset of four blocks in A. Reference [27] constructs LDPC codes with the minimum distance $d_{\min} \geq 6$ based on BIBDs without Pasch configurations, i.e., anti-Pasch BIBDs, to increase the minimum distance of the codes. For a linear block code, any $d_{\min} - 1$ columns of the associated parity-check matrix $\mathbf{H}$ are linearly independent. If there exist one or more Pasch configurations in a Steiner triple system, the four columns of $\mathbf{H}$ corresponding to four blocks in a Pasch configuration, respectively, must be linearly dependent, since each point occurs exactly twice in the configuration. Therefore, $d_{\min} = 4$. If we can avoid Pasch configurations in a Steiner triple system to get an anti-Pasch system, the associated parity-check matrix has $d_{\min} \geq 6$, since $d_{\min}$ must be even for $\mathbf{H}$ when the column weight is $j = 3$.

The construction of regular LDPC codes based on anti-Pasch resolvable designs was also independently

proposed by Johnson and Weller [23]. These well-structured codes are free of four-cycles, achieve very high code rates, and can have arbitrary column weight, although this reference focuses on codes with low column weights. These codes are quasi-cyclic so that their encoder can be realized by using shift registers.

All these BIBD based codes are well structured, free of four-cycles, i.e., girth $g = 6$, and achieve very high code rate; e.g., [28] gives a $(n, 3, k)$ LDPC code with block length $n = 2420$ and code rate $r = 0.95$. However, BIBD based codes have a great restriction, namely, they cannot avoid cycles of length greater than four; they exhibit a large number of six-cycles. New methods are needed to design LDPC codes with girth $g > 6$.

Before leaving this section, we note that there are many other constructions available in the literature that due to space limitations we cannot address. We refer briefly to Margulis' work [34] that designed LDPC codes with good girth properties based on Cayley graphs. Margulis presents a construction for $(n, 3, 6)$ regular LDPC codes whose girth increases linearly as $\log n$. Constructions for $1/2$ rate codes with large girth using Ramanujan graphs [35] are presented in [36]. Both constructions [34], [36] lead to large girth codes but are limited to $1/2$ rates. Tanner et al. [37] present a class of group-structured LDPC codes with girth $g \leq 12$ based on subgroups of the multiplicative group of a prime field $GF(p)$. Their parity-check matrices have fixed column weight $j = 3$ and fixed row weight $k = 5$. The design rate of these codes is only 0.4. Hu et al. propose in [38] a nonalgebraic method named progressive edge-growth (PEG). Their paper presents examples of irregular codes of girth eight by progressively establishing edges between bit and check nodes in an edge-by-edge manner. PEG optimizes the placement of a new edge on the Tanner graph with the goal of maximizing the local girth.

Geometry-Based Designs

Here we present geometry-based constructions for LDPC codes with girth $g \geq 6$. In [39] and [40], we present constructions for column weight $j = 2$ LDPC codes with girth up to 20. Here we illustrate briefly how to extend these results for column weight $j = 3$.

We first introduce an alternative graphical representation for $\mathbf{H}$. Let $\mathbf{H}$ be the parity-check matrix of an LDPC code with v parity-check equations, i.e., $\mathbf{H}$ is $v \times n$. We represent these parity-check equations by a set X of v points. We call X the point set of the LDPC $\mathbf{H}$ matrix. For LDPC codes with $j = 3$, each column of the $\mathbf{H}$ matrix is represented by a triangle composed of three points in X that correspond to the three nonzero elements in this column. We call the resulting graph the structure graph for the LDPC $\mathbf{H}$ matrix; we refer to it by $\mathcal{G}_H$ or simply $\mathcal{G}$.

Consider a point set $X = \{a_1, \ldots, a_8, b_1, \ldots, b_p\}$ and divide it into two subsets $X_1 = \{a_1, \ldots, a_p\}$ and $X_2 = \{b_1, \ldots, b_p\}$. The points in each of these subsets

There has been renewed interest in turbo codes and LDPC codes because their bit error rate performance approaches asymptotically the Shannon limit.

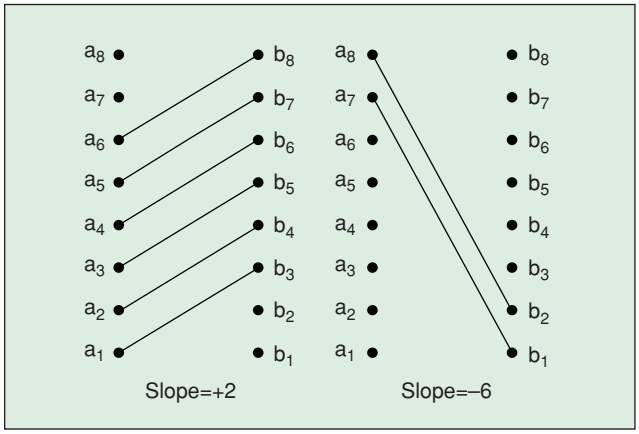
are positioned in a vertical line. We now label sequentially from bottom to top the points in each subset X_1 and X_2 by a_i and b_j, $1 \leq i, j \leq p = 8$, respectively, as shown in Figure 5. The reader should note that this figure repeats twice the set X. Each edge is between a point $a_i \in X_1$ and a point $b_j \in X_2$. Here we need to introduce two concepts: slope of an edge and admissible slope pair.

The slope s of an edge between points $a_i \in X_1$ and $b_j \in X_2$ is $s = j - i$. Slopes take values in the range $-(p-1) \leq s \leq (p-1)$, where p is the number of points in each subset. Hence, there are $(2p-1)$ possible slopes. Note that we have assigned the points in subset X_1 as reference points when calculating the slopes. The number of possible edges with slope s is $(p - |s|)$. In Figure 5, we give examples of edges with slope $+2$ and -6, when $p = 8$.

A slope pair (s, s') is an admissible slope pair (ASP) iff $s' = -\text{sgn}(s) \cdot (p - |s|)$, where the slopes s and s' will be referred to as mirror slopes. The two slopes $+2$ and -6, shown in Figure 5, compose an AS. Each ASP (s, s') allows us to introduce $(p - |s|) + (p - |s'|) = p$ edges, and it increases the degree of each point in X by one.

(n, 3, k) LDPC Codes with Girth Eight

We now consider the construction of codes with column weight $j = 3$ girth $g = 8$. In the structure graph for $(n, 3, k)$ LDPC codes, each column of the $\mathbf{H}$ matrix is represented by a triangle whose three vertices correspond



▲ *5. Edges with specific slopes.*

to the three nonzero elements in the column, the column triangles. As long as each edge belongs to a single column triangle, the corresponding LDPC code is free of four-cycles, since two distinct connection lines between two nodes in $\mathcal{G}_H$ stand for a four-cycle. However, six-cycles are also represented by triangles in the structure graph, the cycle triangles, as shown in Figure 6(a). A triangle is a cycle triangle only when its three edges belong to three different columns; otherwise, it is a column triangle. In other words, each edge in a cycle triangle is shared with a column triangle. Figure 6(b) shows several examples of triangles in the structure graph of an $(n, 3, k)$ LDPC code: {a, b, f}, {b, c, d}, and {d, e, f} are column triangles; and {b, d, f} is a cycle triangle, since its edges are all shared with the previous triangles.

To construct six-cycle-free $(n, 3, k)$ LDPC codes with high code rates, we introduce as many columns as possible for a given value of v. The selection of columns follows two basic rules: each edge is used at most once to construct column triangles; each new column should not introduce cycle triangles.

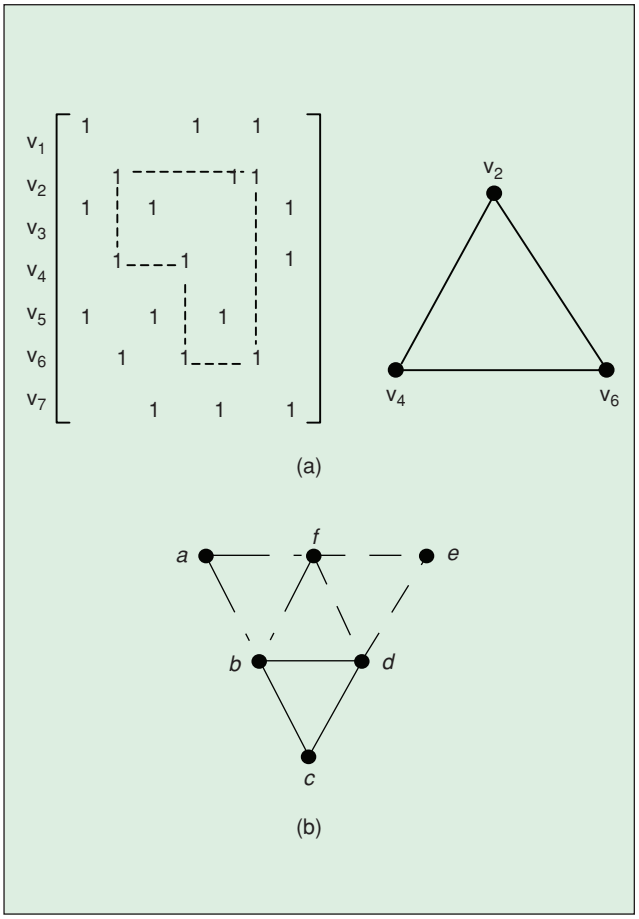
Assume $v = 3p$, with p an integer. We partition these points into three subsets X_0, X_1, and X_2 of equal size p. The points in each subset are aligned in a vertical line, as shown in Figure 7, for $p = 9$. Like before, an edge in the structure graph must connect points in different subsets. The section S_i represents all the edges between the two neighboring subsets X_i and $X_{\mathrm{mod}(i+1,3)}$; it corresponds to an ASP set $\mathcal{A}_i$. The slope of each edge in S_i is calculated using a point in X_i as the reference point. These ASP sets are of the same size. The task is now to find an admissible slope set $\mathcal{A}_i$ for each section X_i to obtain as many column triangles as possible without introducing cycle triangles. The following two facts underlie our construction of girth $g = 8$ $(n, 3, k)$ LDPC codes. The proofs are omitted here.

Fact 1: In a structure graph, any column triangle or cycle triangle must be composed of three edges in three different sections with slopes s_0, s_1, and s_2, respectively, satisfying $\mathrm{mod}\,(s_0 + s_1 + s_2, p) = 0$.

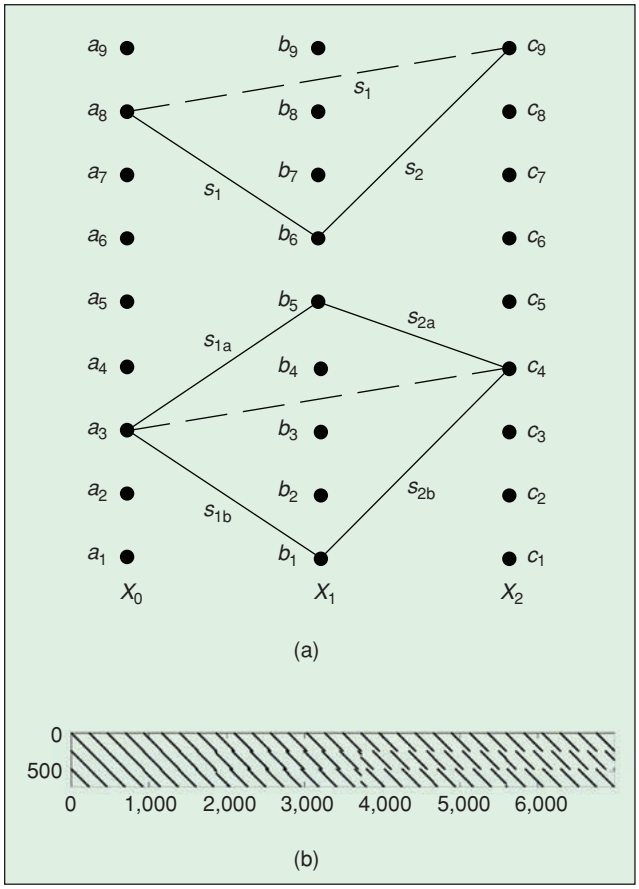
Fact 2: Assume the three ASPs (s_0, s_0'), (s_1, s_1'), and (s_2, s_2') belong to ASP sets $\mathcal{A}_0$, $\mathcal{A}_1$, and $\mathcal{A}_2$, respectively, and that $\mathrm{mod}\,(s_0 + s_1 + s_2, p) = 0$. Then, after introducing all the edges corresponding to all the three ASPs, p triangles occur, and no two of these have a common edge.

To construct $(n, 3, k)$ LDPC codes with girth eight, we need to find three sets $\mathcal{A}_i$ of admissible slope pairs one for each of the three sections S_i, $i = 0, 1, 2$. These sets satisfy the following conditions:

▲ *Condition 1:* The three ASP sets have the same cardinality N_s, i.e., $\mathcal{A}_i = \{(s_{i1}, s_{i1}'), \ldots, (s_{iN_s}, s_{iN_s}')\}$.



▲ *6. (a) A six-cycle in an* ***H*** *matrix and its structure graph. (b) Triangles in the structure graph of an (n, 3, k) LDPC code.*



▲ *7. (a) At the top, a triangle in the structure graph when p = 9; at the bottom, an example of an eight-cycle. (b) The* ***H*** *matrix of a Type-I code with n = 6990 and r = 0.9.*

▲ *Condition 2:* The slope pairs in each $\mathcal{A}_i$ must satisfy the following conditions:

—*Condition 2.1:* For $1 \le j \le N_s$, $\mod(s_{0j} + s_{1j} + s_{2j}, p) = 0$.

—*Condition 2.2:* If $\mod(s_{0i} + s_{1j} + s_{2k}, p) = 0$, then $i = j = k$.

By Fact 2, condition 2.1 guarantees that the three slope pairs (s_{0j}, s'_{0j}), (s_{1j}, s'_{1j}), and (s_{2j}, s'_{2j}), one from each ASP set, introduce p column triangles. Condition 2.2 guarantees that no cycle triangles are introduced.

We developed an algorithm to construct these three ASP sets, but, due to lack of space, we omit its details here. The resulting codes are referred to as type I codes. Figure 7(b) shows the structure of one parity-check matrix $\mathbf{H}$ when $v = 699$, $n = 6,990$, and $r = 0.9$. Reference [41] presents a class of LDPC codes with $j = 3$ and $g = 8$ based on permutation matrices that, for example, for a similar code rate $r = 8/9$ has block shorter length $n = 4,509$.

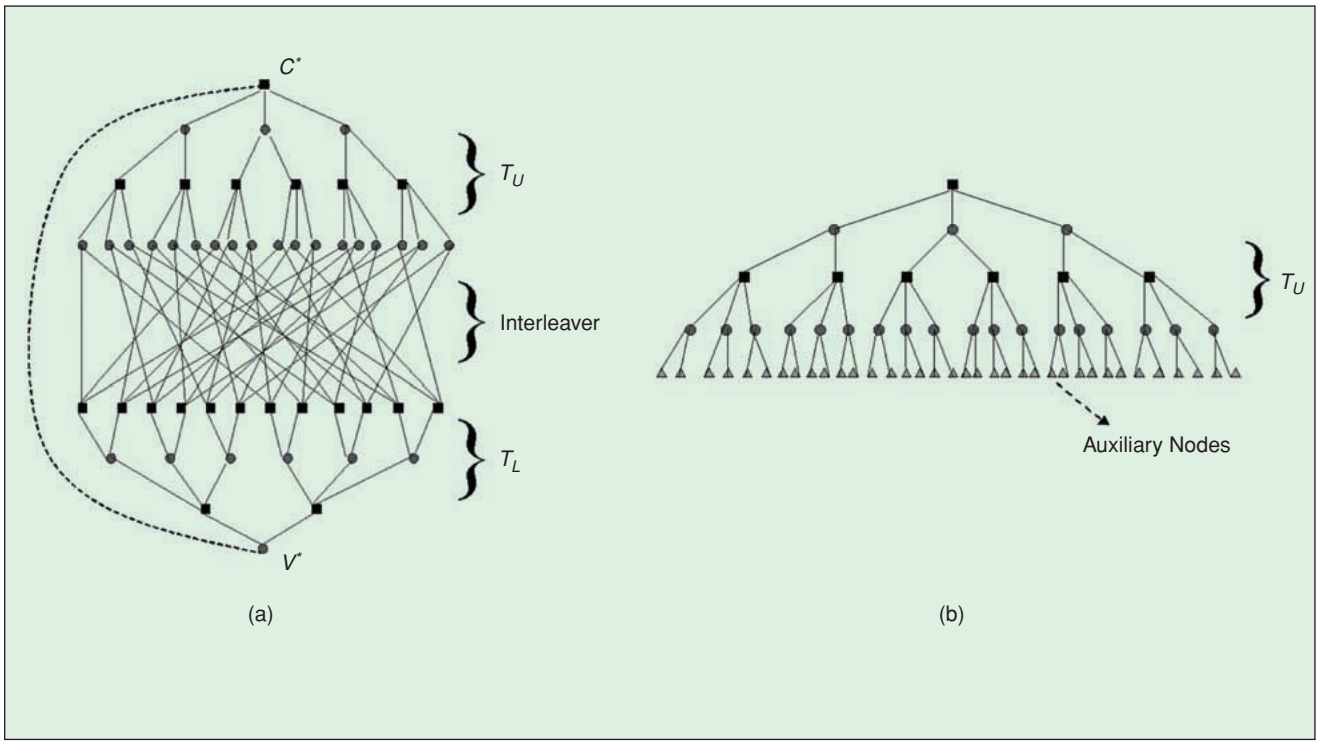
Though type I $(n, 3, k)$ LDPC codes with girth eight have no six-cycles, they will have in general many eight-cycles. We can eliminate some of these eight-cycles by introducing additional conditions on the ASPs. The net effect is to reduce the code rate for a given v. We refer to the resulting codes as type II $(n, 3, k)$ LDPC codes with girth $g = 8$.

The eight-cycles we delete have the form shown on the bottom of Figure 7(a). These cycles occur when the slopes selected s_{0a}, s_{0b}, s_{1a}, and s_{1b} satisfy $\mod(s_{0a} + s_{1a}, p) = \mod(s_{0b} + s_{1b}, p)$, where $(s_{0a}, s'_{0a}), (s_{0b}, s'_{0b}) \in \mathcal{A}_0$, and (s_{1a}, s'_{1a}), $(s_{1b}, s'_{1b}) \in \mathcal{A}_1$. We add this additional condition when searching for admissible slope sets to avoid these eight-cycles. The $\mathbf{H}$ matrix of a type II code is also well structured, similar to that of a type I code.

We expect type II codes to have better decoding performance than type I codes, since they have a smaller number of eight-cycles. Simulation results confirm this. This performance gain is at the cost of a lower code rate for the same parity-check number v, because there are more strict conditions in searching for the admissible slope sets.

Turbo-Structured LDPC Codes

In this section, we present LDPC codes whose Tanner graphs of the associated parity-check matrix $\mathbf{H}$ exhibit a specific architecture that is stimulated by the structure of turbo codes; see also [42] and [43]. The Tanner graph of these codes has three components as shown in Figure 8(a): two height-balanced trees—an upper-tree T_U, whose leaf-nodes are bit nodes (solid circles), and a lower-tree T_L, whose leaf-nodes are check nodes (solid squares)—and an interleaver that "couples" in a turbo-like manner the leaf-nodes of T_U to the leaf-nodes of T_L. We refer to such codes as turbo-structured LDPC (TS-LDPC) codes or turbo-like LDPC codes. The height of each tree, i.e., the number of layers in each tree, is h. The first layer of the upper tree T_U reduces to a single check node C^*, the root, as shown in Figure 8(a), while the root of T_L is a bit node V^*. We note that either subtree T_U or T_L, or both, can be missing. If both are absent, we refer to the codes as flattened TS-LDPC codes.



▲ *8. (a) Turbo-structured LDPC code: $h = 4$, $j = 3$, and $k = 4$. (b) Auxiliary nodes in the upper tree T_U.*

An LDPC code is a special class of linear block codes whose parity-check matrix has a low density of ones, i.e., is sparse.

The structure formed by the edges connecting the leaf-nodes of T_L and the leaf nodes of T_U is the interleaver $\mathcal{I}$. Because we are interested in regular codes, we let all the bit nodes have uniform degree j and all the check nodes have the same degree k; see, for example, the Tanner graph for a TS-LDPC code with $h = 4$, $j = 3$, and $k = 4$ in Figure 8(a). To achieve uniform degree, note that the roots of T_U and of T_L, C^*, and V^*, respectively, are connected to each other by the dashed line in the TS-LDPC code in Figure 8(a). The code rate for a TS-LDPC code is $r = 1 - (j/k)$. For example, if a code rate $r = 8/9$ and column-weight-six LDPC code is desired, simply let $j = 6$ and $k = 54$ in each of the component trees.

The design of TS-LDPC codes with large girth is facilitated by their specific architecture. By construction, each subtree T_U and T_L is, in isolation, free of cycles. The cycles are introduced by the interleaver. We discuss briefly below the methodology to design TS-LDPC codes with column weight $j = 2$ and girth $g = 20$ and codes with $j \geq 3$ and $g = 8$.

We comment that the codes we present here are turbo like from the decoder point of view, i.e., from the parity-check matrix $\mathbf{H}$ and its associated Tanner graph. This stands in sharp contrast and should not be confused with the codes presented in [44] and [45] that combine two LDPC codes as component codes in the encoder side. We also note that [41] designs graph based codes for column weight $j = 2$ and girth g up to 12 that, in the context of TS-LDPC codes, have no lower tree. The technique we discuss in this article can design TS-LDPC codes with arbitrary column weight j.

Interleaver Design

By construction, each leaf-node in the upper tree T_U is connected to $q = j - 1$ leaf-nodes in T_L. This means our TS-LDPC interleaver is a 1-to-q mapping while, usually, interleavers are a 1-1 mapping between elements of two sets with the same size. To get a standard interleaver, we introduce "auxiliary nodes" (solid triangles) as shown in Figure 8(b). For each leaf-node in T_U, we add $j - 1$ auxiliary nodes as its children. Similarly, each leaf-node in T_L has $k - 1$ auxiliary nodes as its descendants. Denote by A_U the set containing all the auxiliary nodes in T_U and by A_L the set containing all the auxiliary nodes in T_L. Then, designing the interleaver corresponds to finding an appropriate 1-1 mapping between elements of A_U and A_L. We observe that a cycle contains at least two auxiliary nodes in T_U and two auxiliary nodes in T_L. For convenience, and depending on how many auxiliary nodes a cycle involves, we group the cycles into two disjoint categories: type I and type II cycles. Type I cycles contain four and only four auxiliary nodes, two from T_U and two from T_L; type II cycles contain more than four auxiliary nodes. In the sequel, by the girth of a type I or of a type II cycle we mean the length of the shortest type I or type II cycle, respectively, present in the graph.

To design large-girth LDPC codes, we proceed as follows. First, we develop a method, the symbol-wise reversal and shift, that maximizes the girth of type I cycles as determined from the height h of the subtrees. Second, we present conditions to eliminate type II cycles of length four and six. We note that for flattened TS-LDPC codes only type II cycles are present.

($p-q$)-Alternate-Decimal Format

Let the subtrees T_U and T_L have height h, the bit nodes degree j, and the check nodes degree k. Let $p = k - 1$ and $q = j - 1$. Each subtree has $[(k-1)(j-1)]^{h/2}$ auxiliary nodes that are indexed from 0 to $[(k-1)(j-1)]^{h/2} - 1$. The nodes in the upper subtree T_U are indexed in a $p - q$-alternate-decimal format, labeled by X_{p-q}, that we introduce now. In this format, we need only h digits, where h is the height of the tree, to represent the label X_{p-q}. These digits x_i are numbered from one to h, starting from the rightmost digit; we refer to the position of the digit x_i as the ith-coordinate in the $(p - q)$-alternate-decimal format. The odd coordinates take values zero to $q - 1$, and the even coordinates take values zero to $p - 1$. Similarly, we index all the auxiliary nodes of T_L in a $(q - p)$-alternate-decimal; we label these digits from left to right and refer to the position of the digit x_i as the ith-coordinate of the $(q - p)$-alternate-decimal representation. We provide an example. With reference to Figure 8(b) where the upper tree has height $h = 4$, the index X_{p-q} of an auxiliary node in the upper tree T_U is

$$\begin{aligned} X_{p-q} &= \underbrace{x_4}_{p}\underbrace{x_3}_{q}\underbrace{x_2}_{p}\underbrace{x_1}_{q} \\ &= \left(x_4 \times p^1 q^2 + x_3 \times p^1 q^1 \right. \\ &\quad \left. + x_2 \times p^0 q^1 + x_1 \times p^0 q^0\right)_{10}, \end{aligned} \tag{1}$$

where $(\cdot)_{10}$ represents the decimal value of X_{p-q}.

Symbol-Wise Reversal and Shift Operators

The symbol-wise reversal operator $\pi_s(\cdot)$ applied to the label X_{p-q} represented in $(p - q)$-alternate-decimal form with h coordinates exchanges the digits at the ith- and $(h + 1 - i)$th-coordinates, i.e.,

$$\pi_s(X_{p-q}) = \underbrace{x_1}_{q}\underbrace{x_2}_{p}\underbrace{x_3}_{q}\underbrace{x_4}_{p}.$$

We now introduce symbol-wise shifts. We represent by $\dot{+}$ the symbol-wise addition. Using this addition, we add in $(q-p)$-alternate-decimal format the shift S_{q-p} to the index $\pi_s(X_{p-q})$ to obtain a new index. For example, let $\pi_s(X_{p-q}) = x_1x_2x_3x_4$ and $S_{q-p} = s_1s_2s_3s_4$, then

$$\pi_s\left(X_{p-q}\right) \dot{+} S_{q-p} = y_1y_2y_3y_4,$$
$$\text{where} \quad y_i = x_i \dot{+} s_i = \text{mod}\left(x_i + s_i, \text{div}_i\right).$$

where $\text{div}_i = p$ if i is even and $\text{div}_i = q$ if i is odd. Note that with symbol-wise addition there is no carry. Similarly, we define symbol-wise subtraction $\dot{-}$.

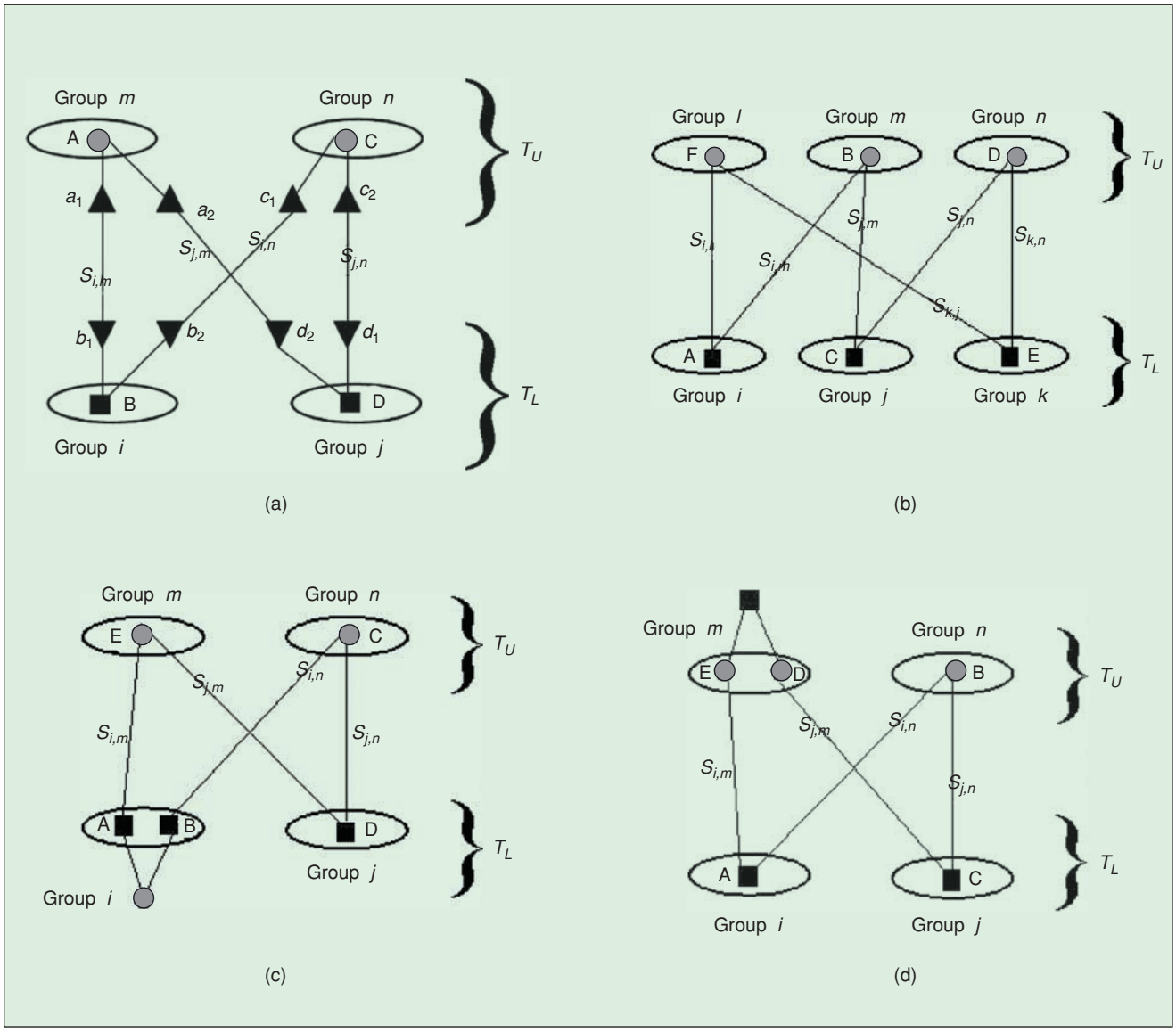
Type I Cycles

Type I cycles are avoided by choosing an arbitrary shift S in $(q-p)$-alternate-decimal notation and by simply connecting the auxiliary node X_{p-q} in T_U to the auxiliary node $\pi_s(X_{p-q}) \dot{+} S$ in T_L. It can be shown that this rule maximizes the girth of type I cycles.

Type II Cycles

To avoid type II cycles, we start by clustering the auxiliary nodes of T_U that have the same digit at the leftmost coordinate in their $(p-q)$-alternate-decimal index in $k-1$ groups. Likewise, we assemble the auxiliary nodes of T_L according to the values of their leftmost coordinates into $(j-1)$ groups. We now discuss the shift $S_{i,m}$ connecting (any of the) nodes in group i in T_L to (any of the) nodes in group m in T_U. For different i and m, the shifts $S_{i,m}$ may be the same or different. Hence, there are $(j-1)(k-1)$ shifts $S_{i,m}$ to adjust to prevent type II cycles of short length.

Fact 3: Consider two groups of nodes m and n in T_U and two groups i and j in T_L. If the corresponding shifts satisfy $S_{i,m} \dot{+} S_{j,n} \neq S_{i,n} \dot{+} S_{j,m}$, then no



▲ *9. (a) Type-II four-cycle. (b)–(d) three type-II six-cycles.*

The Tanner graph is a special type of graph, a bipartite graph, where the nodes divide into two disjoint classes with edges only between nodes in the two different classes.

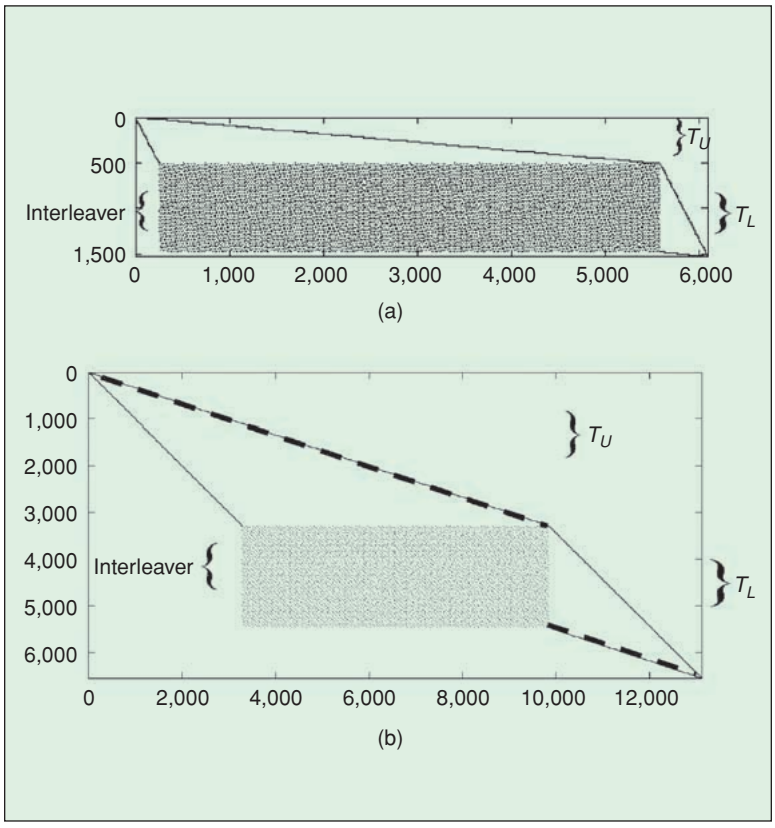
type II 4-cycle is formed among these four groups.

To prevent type II six-cycles, we first note that the only possible type II six-cycles are the ones displayed in Figure 9(b)–(d). Fact 3 is then extended such that these type II six-cycles are avoided. Due to lack of space, we present only the conditions on the shifts that prevent the type II six-cycles in Figure 9(c): $S_{i,n} \dot{-} S_{j,n} \dot{+} S_{j,m} \dot{-} S_{i,m} > pq = (k-1)(j-1)$. Similar conditions can be derived for the other type II six-cycles and, in general, for type II $2l$-cycles.

By choosing suitable shifts $S_{u,v}$ for $u = 1, \ldots, (k-1)$ and $v = 1, \ldots, (j-1)$, we can prevent type II cycles of length 4, 6, and so on up to $g-2$ where g is the desired girth of the code. Figure 10(a) illustrates the $1,521 \times 6,084$ matrix $\mathbf{H}$ designed by this method and that corresponds to a (6,084, 3, 12) TS-LDPC code with girth $g = 8$. Matrix $\mathbf{H}$ clearly exhibits the upper and lower trees T_U (thick dashed line) and T_L (thin solid line) and the interleaver (cloud of points). Along the solid (thin) lines in $\mathbf{H}$ there is a single one in each row, while along the dashed thicker diagonals there are 11 ones in each row, so that per row there are 12 ones. Each row in the cloud in $\mathbf{H}$ has 11 ones also.

Designing LDPC codes with column weight $j = 2$ is a much simpler task, and details are found in [43]; Figure 10(b) shows a (13,120, 2, 4) TS-LDPC with rate $r = 1/2$ and girth $g = 20$.

TS-LDPC codes provide efficient memory utilization. Generically, an (n, j, k)-LDPC code is represented by an $m \times n$ parity-check matrix $\mathbf{H}$. An efficient direct representation of $\mathbf{H}$ exploits its sparseness to record only the nonzero column indices in each row, hence, $n \times j$ indices are recorded. In contrast, with TS-LDPC codes, we only need to record the $(k-1) \times (j-1)$ matrix $\mathbf{S}$ of shifts. For example, for the $j = 3$ LDPC code with $\mathbf{H}$ shown in Figure 10(a), rather than storing the $6,084 \times 3 = 18,252$ column indices where the ones are located, we need to store only the $(12-1) \times (3-1) = 22$ shifts, which significantly reduces the memory required to store the parity-check matrix $\mathbf{H}$.



▲ *10. Parity-check matrix H for (n, j, k) TS-LDPC code with rate r and girth g. (a) (6,084, 3, 12), $r = 3/4$, $g = 8$ and (b) (13,120, 2, 4), $r = 1/2$, $g = 20$.*

Simulation Results

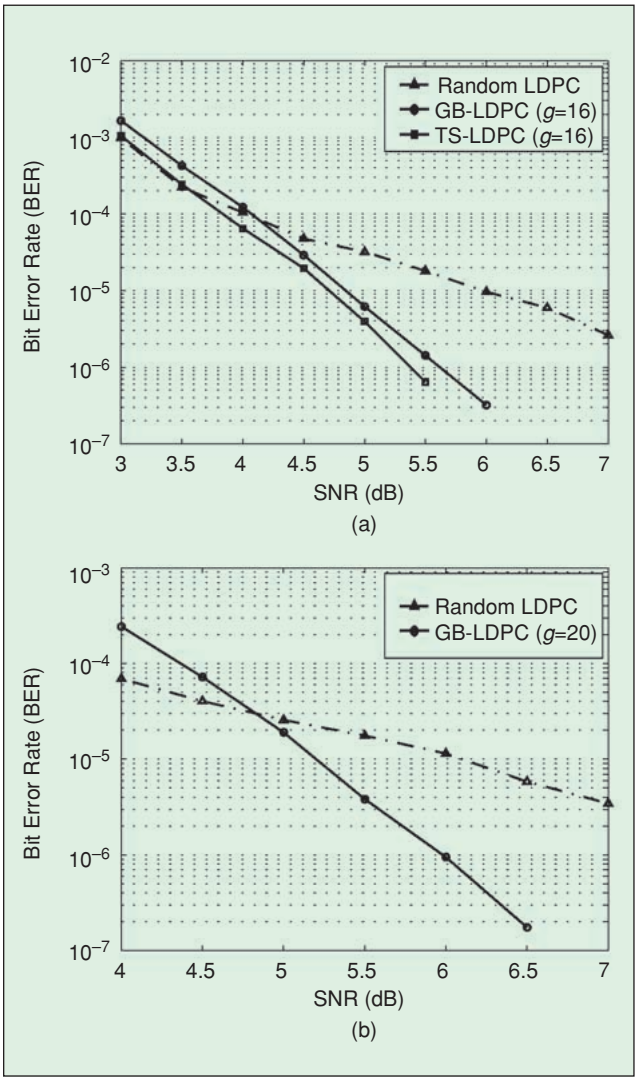
We compare by simulation the BER of the geometry-based designs of LDPC codes (GB-LDPC) and of the turbo-structured LDPC codes (TS-LDPC) with the BER of randomly constructed LDPC codes in additive white Gauss noise (AWGN) channels. The codes are decoded with the sum-product algorithm [9], and we adopt the rate-adjusted signal to noise ratio (SNR) defined in [46] $\mathrm{SNR}_1 = 10\log_{10}\left[E_b/\left(2r\sigma^2\right)\right]$, where $\mathrm{SNR}_2 = 10\log_{10}[E_b/2\sigma^2]$ is the usual SNR. The difference $\mathrm{SNR}_1 - \mathrm{SNR}_2 = -10\log r$ may be significant because $r < 1$.

Figure 11(a) compares the BER performance for three $(n = 4368, 2, k)$ LDPC codes with rate $r = 1/2$: a random, a GB-, and a TS-LDPC code. The GB- and the TS-LDPC codes both have girth $g = 16$. In the high SNR region, the GB- and the TS-LDPC codes outperform the random code: at $\mathrm{BER} = 10^{-5}$ this gain is 1.1 and 1.2 dB, respectively. In the low SNR regime, the GB-LDPC code exhibits a small performance loss, while the TS-LDPC code has performance similar to that of the random code. Figure 11(b) compares the BER performance for an $(n, 2, k)$ GB-LDPC code with girth 20 with a random $(n, 2, k)$ LDPC code. The GB-LDPC code outperforms the random LDPC code by 1.4 dB at $\mathrm{BER} = 10^{-5}$. These plots show that $(n, 2, k)$ LDPC codes with large girth outperform random codes in the high SNR

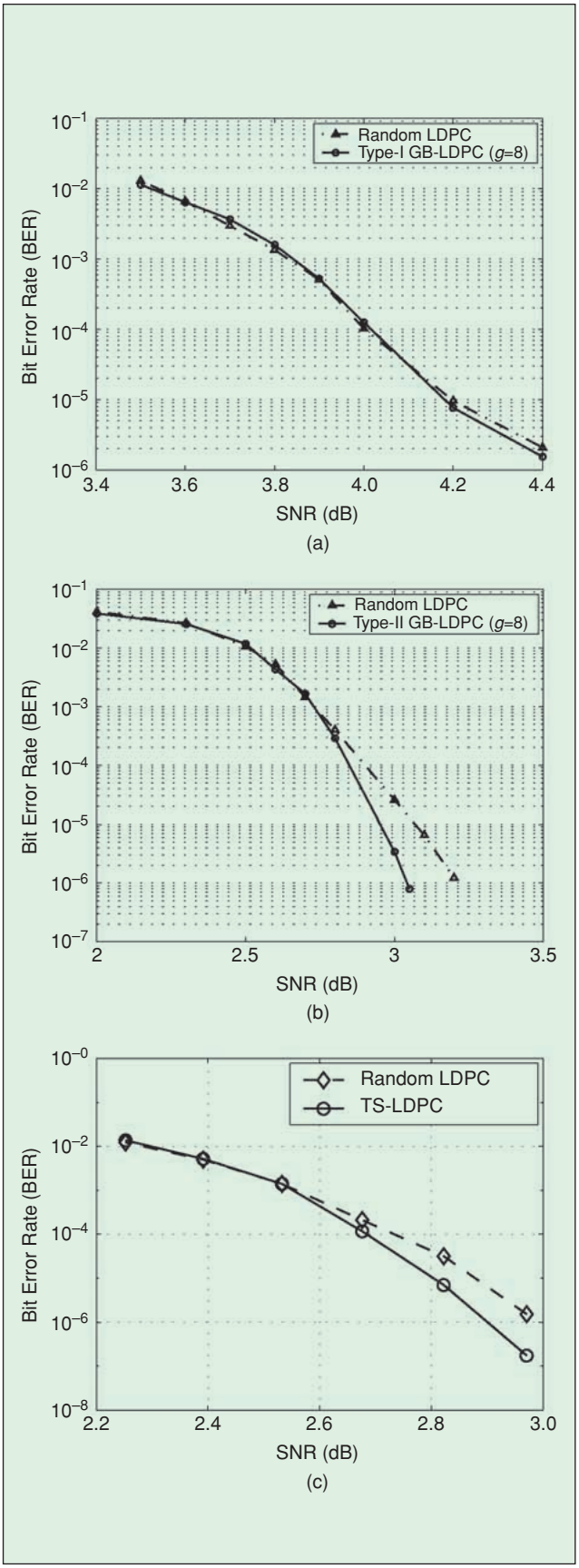
region. This is well explained by the fact that, for LDPC codes with column weight $j = 2$, the minimum distance $d_{\min} = g/2$, where g is the girth, and that, in the high SNR region, $d_{\min}$ is a dominant factor in determining the code BER performance.

Figure 12(a) compares the BER performance of two $(n = 6,990, 3, k)$ codes with rate $r = 9/10$: a GB-LDPC code with girth $g = 8$ and a random LDPC code. These codes show similar BER performance across the SNR region tested, though the GB-LDPC code outperforms the random LDPC code by 0.02 dB at BER $= 10^{-5}$. At higher SNR, we expect the GB-LDPC code to outperform the random code.

Figure 12(b) compares the BER performance for two $(n = 6,986, 3, k)$ codes with rate $r = 11/14$: a type II GB-LDPC code with girth $g = 8$ and a random LDPC code. The codes show similar BER performance in the low SNR region; however, in the high SNR region, the GB-LDPC code outperforms the random LDPC code by 0.15 dB at BER $= 10^{-5}$.



▲ *11. Comparison of BER performance for random code with (n, 2, k) GB- and TS-LDPC codes: (a) girth 16 and (b) girth 20 (GB-LDPC code only).*



▲ *12. BER performance. (a) Type I (n, 3, k) GB-LDPC code with g = 8 and random LDPC code, r = 9/10. (b), (c) Type II (n, 3, k) GB-LDPC code with g = 8, and random LDPC code, r = 11/14. (c) (n, 3, k) TS-LDPC code with g = 8 and random code, r = 3/4.*

Figure 12(a) and (b) confirms that the BER performance of LDPC codes depends not only on the girth but also on the cycle distribution. Both type I and type II $(n, 3, k)$ GB-LDPC codes have roughly the same block length ($n = 6,990$ and $n = 6,984$, respectively) and the same girth $g = 8$, but the type II code has a much smaller number of eight-cycles in their Tanner graph. A small price paid in the code rate (down from $r = 0.9$ for the type I code to $r = 11/14 = 0.786$ for the type II code) gained roughly 1.3 dB in BER performance as can be seen from Figure 12: for BER $= 10^{-6}$, the type I code requires SNR $= 4.4$ dB while the type II code requires SNR $= 3.1$ dB.

Figure 12(c) shows the BER performance for a column weight $j = 3$ TS-LDPC code with girth eight (solid line). For comparison, we also show the BER performance of a randomly constructed LDPC code with column weight $j = 3$ (dashed line). Both codes have the same block length 6084 and the same code rate 3/4. From Figure 12, it can be seen that the BER performance of the TS-LDPC code is 0.08 dB better than that of the random LDPC code at BER $= 10^{-6}$ while at low SNR both codes have comparable error-correcting performance.

Comparing the performance of codes with $j = 2$ with codes with $j = 3$, we see that the $j = 3$ codes exhibit at least a 3 dB gain over $j = 2$ codes. This is in line with the expected loss in performance for $j = 2$ codes.

Conclusion

This article addresses methods to design regular LDPC codes with large girth. In graph terms, this corresponds to designing bipartite undirected regular graphs with large girth. Large girth speeds the convergence of iterative decoding and improves the performance at least in the high SNR range, by slowing down the onsetting of the error floor. We reviewed several existing constructions from exhaustive search to highly structured designs based on Euclidean and projective finite geometries and combinatorial designs such as balanced incomplete block designs. We described GB and TS LDPC codes and compared the BER performance of GB- and TS-LDPC codes with large girth to the BER performance of random codes. These studies confirm that in the high SNR regime these codes with high girth exhibit better BER performance. The regularity of the codes provide additional advantages that we did not explore in this article like the simplicity of their hardware implementation and fast encoding.

Acknowledgments

This research was supported by the Data Storage Systems Center (DSSC) at Carnegie Mellon University and by NSF through grant ECS-0225449.

José M.F. Moura is a professor of electrical and computer engineering at Carnegie Mellon University. He holds an Sc.D. degree from MIT. His research interests are in statistical signal processing. He was the editor in chief for *IEEE Transactions on Signal Processing* and was vice-president for Publications for the IEEE Signal Processing Society and vice-president for Publications for the IEEE Sensors Council. He is currently on the board of *IEEE Proceedings*. He is a Fellow of the IEEE and a corresponding member of the Academy of Sciences of Portugal.

Jin Lu received the B.Sc. and M.Sc. degrees from the Department of Electronic Engineering, Tsinghua Univeristy, Beijing, China, in 1998 and 2000, respectively. He is currently pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania. His research interests include channel coding and statistical signal processing in data storage systems, and communication channels. He has published several technical papers.

Haotian Zhang is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, Pennsylvania. He obtained the B.Sc. and the M.Sc. degrees in electronic engineering from Tsinghua University, P.R. China, in 1997 and 2000, respectively. His research areas include channel coding in both data storage systems and communication systems, and wireless sensor networks.

References

[1] R.G. Gallager, *Low-Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.

[2] G. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding: Turbo codes," in *Proc. 1993 Int. Conf. Comm.,* Geneva, Switzerland, May 1993, pp. 1064–1070.

[3] D.J.C. Mackay and R.M. Neal, "Good codes based on very sparse matrices," in *Cryptography and Coding, 5th IMA Conference* (Lecture Notes in Computer Science), C. Boyd, Ed. 1995, vol. 1025, pp. 110–111.

[4] C.E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.,* pp. 372–423, 623–656, 1948.

[5] E.A. Lee and D.G. Messerschmitt, *Digital Communication*. Boston/Dordrecht/London: Kluwer, 1994.

[6] S.G. Wilson, *Digital Modulation and Coding*. Englewood Cliffs, NJ: Prentice Hall, 1996.

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

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

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

[10] H. Song and B.V.K.V. Kumar, "Low density parity check codes for partial response channels," *IEEE Signal Processing Mag.,* vol. 21, no. 1, pp. 56–66, Jan 2004.

[11] F.R. Kschischang and B.J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas*

Commun., vol. 16, no. 2, pp. 219–230, Feb. 1998.

[12] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?," *IEEE Trans. Inform. Theory,* vol. 45, no. 6, pp. 2173–2181, Sept. 1999.

[13] R.J. McEliece, D.J.C. Mackay, and J.F. Cheng, "Turbo decoding as an instance of Pearl's "belief propagation" algorithm," *IEEE J. Select. Areas Commun.,* vol. 16, no. 2, pp. 140–152, Feb. 1998.

[14] J. Campello, D.S. Modha, and S. Rajagopalan, "Designing LDPC codes using bit-filling," in *Proc. ICC 2001,* Helsinki, Finland, June 2001, vol. 1, pp. 55–59.

[15] J. Campello and D.S. Modha, "Extended bit-filling and LDPC code design," in *IEEE Globecom 2001,* San Antonio, TX, Nov. 2001, vol. 2, pp. 985–989.

[16] Y. Mao and A.H. Banihashemi, "A heuristic search for good LDPC codes at short block lengths," in *Proc. ICC 2001,* Helsinki, Finland, June 2001, vol. 1, pp. 41–44.

[17] Yu Kou, Shu Lin, and M.P.C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inform. Theory,* vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[18] Y. Kou, S. Lin, and M.P.C. Fossorier, "Low density parity check codes: Construction based on finite geometries," in *Proc. IEEE Globecom 2000,* San Francisco, CA, Nov. 2000, vol. 2, pp. 825–829.

[19] W.T. Fishback, *Projective and Euclidean Geometry.* New York/London/Sydney/Toronto: Wiley, 1969.

[20] S. Lin and D.J.Costello Jr., *Error Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice Hall, 1983.

[21] L.M. Batten, *Combinatorics of Finite Geometries.* Cambridge, UK: Cambridge Univ. Press, 1986.

[22] D.J.C. Mackay and M.C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems and Graphical Models; IMA Volumes in Mathematics and Its Applications,* B. Marcus and J. Rosenthal, Eds. New York: Springer-Verlag, 2000, vol. 123, pp. 113–130.

[23] S.J. Johnson and S.R. Weller, "Regular low-density parity-check codes from combinatorial design," in *Proc. Inf. Tech. Workshop (ITW) 2001,* Cairns, Australia, Sept. 2001.

[24] S.J. Johnson and S.R. Weller, "Construction of low-density parity-check codes from Kirkman triple systems," in *Proc. IEEE Globecom 2001,* San Antonio, TX, Nov. 2001, vol. 2, pp. 970–974.

[25] B. Vasic, "Structured iteratively decodable codes based on Steiner systems and their application in magnetic recording," in *Proc. IEEE Globecom 2001,* San Antonio, TX, Nov. 2001, vol. 5, pp. 2954–2960.

[26] B. Vasic, E.M. Kurtas, and A.V. Kuznetsov, "LDPC codes based on mutually orthogonal Latin rectangles and their application in perpendicular magnetic recording," *IEEE Trans. Magnetics,* vol. 38, no. 5, pp. 2346–2348, Sept. 2002.

[27] B. Vasic, "High-rate low-density parity check codes based on anti-Pasch affine geometries," in *Proc. ICC 2002*, Washington, DC, Apr. 28–May 2 2002, vol. 3, pp. 1332–1336.

[28] B. Vasic, E.M. Kurtas, and A.V. Kuznetsov, "Kirkman systems and their application in perpendicular magnetic recording," *IEEE Trans. Magn.,* vol. 38, no. 4, pp. 1705–1710, July 2002.

[29] J.H. Dinitz and D.R. Stinson, "A brief introduction to design theory," in *Contemporary Design Theory: A Collection of Surveys,* J.H. Dinitz and D.R. Stinson, Eds. New York: wiley, 1992, chap. 1, pp. 1–12.

[30] C.C. Lindner and C.A. Rodger, *Design Theory.* Boca Raton, London, New York, Washington, DC: CRC, 1997.

[31] H.J. Ryser, *Combinatorial Mathematics.* Washington, DC: Mathematical Assoc. America, 1963.

[32] D.R. Stinson, R. Wei, and L. Zhu, "New constructions for perfect hash families and related structures using combinatorial designs and codes," *J. Combinatorial Designs*, vol. 8, pp. 189–200, 2000.

[33] A.C. Ling, C.J. Colbourn, M.J. Grannell, and T.S. Griggs, "Construction techniques for anti-Pasch Steiner triple systems," *J. London Math. Society,* vol. 61, no. 43, pp. 641–657, June 2000.

[34] G.A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica,* vol. 2, no. 1, pp. 71–78, 1982.

[35] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs," *Combinatorica,* vol. 8, no. 3, pp. 261–277, 1988.

[36] J. Rosenthal and P.O. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. 38th Allerton Conf. Communication, Control, and Computing,* Monticello, IL, Oct. 2000, pp. 248–257.

[37] R.M. Tanner, D. Srikdhara, and T. Fuja, "A class of group-structured ldpc codes," in *The Proc. ISTA,* Ambleside, England, 2001.

[38] X.Y. Hu, E. Eleftheriou, and D.M. Arnold, "Progressive edge-growth tanner graphs," in *Proc. IEEE GLOBECOM,* San Antonio, TX, Nov. 2001.

[39] H. Zhang and J.M.F. Moura, "Large-girth LDPC codes based on graphical models," in *Proc. IEEE Signal Processing and Wireless Communications (SPAWC) Workshop,* Rome, Italy, June 2003.

[40] H. Zhang and J.M.F. Moura, "Structured regular LDPC codes with large girth," in *IEEE Globecom 2003,* San Francisco, CA, Dec 2003.

[41] H. Song, "Iterative soft detection and decoding for data storage channels," Ph.D. dissertation, Dept. of Elec. Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, 2003.

[42] J. Lu and J.M.F. Moura, "Turbo like decoding of LDPC codes," in *Proc. IEEE INTERMAG 2003,* Boston, MA, Apr. 2003.

[43] J. Lu and J.M.F. Moura, "Turbo design for LDPC codes with large girth," in *Proc. IEEE Signal Processing and Wireless Communications (SPAWC) Workshop*, Rome, Italy, June 2003.

[44] L. Ping and K.Y. Wu, "Concatenated tree codes: A low-complexity, high-performance approach," *IEEE Trans. Inform. Theory,* vol. 47, no. 2, pp. 791–799, Feb. 2001.

[45] H. Behairy and S.-C. Chang, "Parallel concatenated Gallager codes for CDMA applications," in *IEEE Globecom 2001,* San Antonio, TX, Nov. 2001, vol. 2, pp. 1002–1006.

[46] D.J.C. Mackay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory,* vol. 45, no. 2, pp. 399–431, Mar. 1999.