Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery and New Results[1]

Yu Kou and Shu Lin
Department of Electrical and Computer Engineering
University of California
Davis, CA 95616

Marc P.C. Fossorier
Department of Electrical Engineering
University of Hawaii at Manoa
Honolulu, HI 96822

Abstract

This paper presents a geometric approach to the construction of low density parity check (LDPC) codes. Four classes of LDPC codes are constructed based on the lines and points of Euclidean and projective geometries over finite fields. Codes of these four classes have good minimum distances and their Tanner graphs have girth 6. Finite geometry LDPC codes can be decoded in various ways, ranging from low to high decoding complexity and from reasonably good to very good performance. They perform very well with iterative decoding. Furthermore, they can be put in either cyclic or quasi-cyclic form. Consequently, their encoding can be achieved in linear time and implemented with simple feedback shift registers. This advantage is not shared by other LDPC codes in general and is important in practice. Finite geometry LDPC codes can be extended and shortened in various ways to obtain other good LDPC codes. Several techniques of extension and shortening are presented. Long extended finite geometry LDPC codes have been constructed and they achieve a performance only a few tenths of a dB away from the Shannon theoretical limit with iterative decoding.

Key Words: Low density parity check codes, Euclidean geometry, projective geometry, cyclic code, quasi-cyclic code, column splitting, row-splitting, shortening, iterative decoding, bit flipping decoding.

[1]This research was supported by NSF under Grants CCR-0096191, CCR-0098029 and NASA under Grants NAG 5-9025 and NAG 5-10480.

1. Introduction

Low density parity check (LDPC) codes were first discovered by Gallager [1, 2] in the early 1960's and have recently been rediscovered and generalized [3–14]. It has been shown that these codes achieve a remarkable performance with iterative decoding that is very close to the Shannon limit [4, 9–14]. Consequently, these codes have become strong competitors to turbo codes [23–26] for error control in many communication and digital storage systems where high reliability is required.

An LDPC code is defined as the null space of a parity check matrix **H** with the following structural properties: (1) each row consists of ρ "ones"; (2) each column consists of γ "ones"; (3) the number of "ones" in common between any two columns, denoted λ, is no greater than 1; (4) both ρ and γ are small compared to the length of the code and the number of rows in **H** [1, 2]. Since ρ and γ are small, **H** has a small density of "ones" and hence is a sparse matrix. For this reason, the code specified by **H** is called an LDPC code. The LDPC code defined above is known as a regular LDPC code. If not all the columns or all the rows of the parity check matrix **H** have the same number of "ones" (or weights), an LDPC code is said to be irregular.

Although LDPC codes have been shown to achieve outstanding performance, no analytic (algebraic or geometric) method has been found for constructing these codes. Gallager only provided a class of pseudo-random LDPC codes [1, 2]. Good LDPC codes that have been found are largely computer generated, especially long codes. Encoding of these long computer generated LDPC codes is quite complex due to the lack of code structure such as cyclic or quasi-cyclic structure. Furthermore, their minimum distances are either poor or hard to determine.

In this paper, we investigate the construction of LDPC codes from a geometric approach. The construction is based on the lines and points of a finite geometry. Well known finite geometries are Euclidean and projective geometries over finite fields. Based on these two families of finite geometries, four classes of LDPC codes are constructed. Codes of these four classes are either cyclic or quasi-cyclic, and therefore their encoding can be implemented with linear feedback shift registers based on their generator (or characterization) polynomials [27, 28]. This linear time encoding is very important in practice and is not shared by other LDPC codes in general. We call codes of these four classes finite geometry LDPC codes.

Finite geometry LDPC codes have relatively good minimum distances and their Tanner graphs do not contain cycles of length 4. They can be decoded with various decoding methods, ranging from low to high complexity and from reasonably good to very good performance. These decoding methods include: one-step majority-logic (MLG) decoding [28, 31], Gallager's bit flipping (BF) decoding [2], weighted MLG decoding [49], weighted BF decoding, a posteriori probability (APP) decoding [2], and iterative decoding based on belief propagation (commonly known as sum-product algorithm (SPA)) [10, 11, 15, 20–22]. Finite geometry LDPC codes, especially high rate codes, perform very well with the iterative SPA decoding.

A finite geometry LDPC code can be extended by splitting each column of its parity check matrix **H** into multiple columns. This column splitting results in a new sparse matrix and hence a new LDPC code of longer length. If column splitting is done properly, the extended code performs amazingly well using the SPA decoding. An error performance only a few tenths of a dB away from the Shannon limit can be achieved. New LDPC codes can also be constructed by splitting each row of the parity check matrix of a finite geometry LDPC code into multiple rows. Combining column and row splittings of the parity check matrices of finite geometry LDPC codes, we can obtain a large class of LDPC codes with a wide range of code lengths and rates. A finite geometry LDPC code can also be shortened by puncturing the columns of its parity check matrix that correspond to the points on a set of lines or a sub-geometry of the geometry based on which the code is constructed. Shortened finite geometry LDPC codes also perform well with the SPA decoding.

The paper is organized as follows. Section 2 presents a construction method of LDPC codes based on the lines and points of a finite geometry. Two types of codes are constructed and their minimum distances are lower bounded. Section 3 gives the construction and characterization of LDPC codes based on Euclidean and projective geometries. Various decoding methods for finite geometry LDPC codes are discussed in Section 4. A simple weighted BF decoding algorithm and a two-stage hybrid soft/hard decoding scheme are presented. Section 5 presents simulation results of error performance of some finite geometry LDPC codes using various decoding methods. Techniques for extending and shortening finite geometry LDPC codes are given in sections 6 and 7, respectively. Section 8 discusses the possible combinations of finite geometry LDPC codes and turbo codes in concatenation form. Finally, Section 9 concludes this paper with some remarks and suggestions of further research work.

2. Finite Geometry LDPC Codes and Their General Structure

This section presents a simple construction of LDPC codes based on the lines and points of finite geometries. Two types of codes are constructed and their general structural properties are investigated.

Lower bounds on their minimum distances are derived.

Let **G** be a finite geometry with n points and J lines which has the following fundamental structural properties: (1) every line consists of ρ points; (2) any two points are connected by one and only one line; (3) every point is intersected by γ lines (i.e., every point lies on γ lines); and (4) two lines are either parallel (i.e., they have no point in common) or they intersect at one and only one point. There are two families of finite geometries which have the above fundamental structural properties, namely Euclidean and projective geometries over finite fields.

Form a $J \times n$ matrix $\mathbf{H}_{\mathbf{G}}^{(1)} = [h_{i,j}]$ over GF(2) whose rows and columns correspond to the lines and points of the finite geometry **G** , respectively, where $h_{i,j} = 1$ if and only if the i-th line of **G** contains the j-th point of **G** and $h_{i,j} = 0$, otherwise. A row in $\mathbf{H}_{\mathbf{G}}^{(1)}$ simply displays the points on a specific line of **G** and has weight ρ. A column in $\mathbf{H}_{\mathbf{G}}^{(1)}$ simply displays the lines that intersect at a specific point in **G** and has weight γ. The rows of $\mathbf{H}_{\mathbf{G}}^{(1)}$ are called the incidence vectors of the lines in **G**, and the columns are called the intersecting vectors of the points in **G**. Therefore, $\mathbf{H}_{\mathbf{G}}^{(1)}$ is the incidence matrix of the lines in **G** over the points in **G**. It follows from the second structural property of **G** that every two columns have exactly one "1-component" in common, and it follows from the fourth structural property of **G** that any two rows have at most one "1-component" in common. The density of this matrix, denoted r, is defined as the ratio of the total number of "ones" in **H** to the total number of entries in **H**. Then we readily see that $r = \rho/n = \gamma/J$. If ρ and γ are small compared to n and J, then $\mathbf{H}_{\mathbf{G}}^{(1)}$ is a low density matrix which has all the structural properties defined in Section 1.

The null space over GF(2) of $\mathbf{H}_{\mathbf{G}}^{(1)}$ gives a binary LDPC code of length n. Such a code is called the type-I geometry-**G** LDPC code, denoted $\mathbf{C}_{\mathbf{G}}^{(1)}$. The rows of $\mathbf{H}_{\mathbf{G}}^{(1)}$ are not necessarily linearly independent. Let $\mathcal{R}$ be the rank of $\mathbf{H}_{\mathbf{G}}^{(1)}$. Then $\mathbf{C}_{\mathbf{G}}^{(1)}$ is a binary $(n, n - \mathcal{R})$ linear code with $\mathbf{H}_{\mathbf{G}}^{(1)}$ as its parity check matrix.

Let $\mathbf{H}_{\mathbf{G}}^{(2)}$ be the transpose of $\mathbf{H}_{\mathbf{G}}^{(1)}$, i.e., $\mathbf{H}_{\mathbf{G}}^{(2)} = [\mathbf{H}_{\mathbf{G}}^{(1)}]^T$. Then $\mathbf{H}_{\mathbf{G}}^{(2)}$ is also a low density matrix with row weight γ and column weight ρ. The null space over GF(2) of $\mathbf{H}_{\mathbf{G}}^{(2)}$ gives a binary LDPC code of length J, denoted $\mathbf{C}_{\mathbf{G}}^{(2)}$. Since $\mathbf{H}_{\mathbf{G}}^{(1)}$ and $\mathbf{H}_{\mathbf{G}}^{(2)}$ have the same rank $\mathcal{R}$, $\mathbf{C}_{\mathbf{G}}^{(2)}$ is a binary $(J, J - \mathcal{R})$ linear code. This code is called the type-II geometry-**G** LDPC code. $\mathbf{C}_{\mathbf{G}}^{(1)}$ and $\mathbf{C}_{\mathbf{G}}^{(2)}$ are called companion codes and have the same number of parity check symbols.

Let $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_J$ be the rows of $\mathbf{H}_{\mathbf{G}}^{(1)}$ where $\mathbf{h}_j = (h_{j,1}, h_{j,2}, \cdots, h_{j,n})$ for $1 \leq j \leq J$. An n-tuple $\mathbf{v} = (v_1, v_2, \cdots, v_n)$ is a codeword of the type-I geometry-**G** LDPC code $\mathbf{C}_{\mathbf{G}}^{(1)}$ specified by

$\mathbf{H}_{\mathbf{G}}^{(1)}$ if and only if the inner product

$$s_j = \mathbf{v} \cdot \mathbf{h}_j = \sum_{l=1}^{n} v_l h_{j,l} = 0, \tag{1}$$

for $1 \leq j \leq J$. The sum given by (1) is called a parity-check sum (or simply check sum), which is simply a linear sum of a subset of code bits. A code bit v_l is said to be checked by the sum $s_j = \mathbf{v} \cdot \mathbf{h}_j$ (or the row $\mathbf{h}_j$) if $h_{j,l} = 1$. Let A_l be the set of rows in $\mathbf{H}_{\mathbf{G}}^{(1)}$ that check on the code bit v_l. Let S_l denote the set of check sums formed by the rows in A_l. It follows from the structural properties of $\mathbf{H}_{\mathbf{G}}^{(1)}$ that the code bit v_l is contained in every check sum in S_l and any of the other $n-1$ code bits is contained in at most one check sum in S_l. The check sums in S_l (or the rows in A_l) are said to be orthogonal on the code bit v_l [28, 31]. The check sums in S_l are called the orthogonal check sums on code bit v_l and the rows in A_l are called the orthogonal vectors on v_l. For $1 \leq l \leq n$, each code bit v_l is checked by exactly γ orthogonal check sums. These orthogonal check sums can be used for majority-logic decoding of the code [28, 31]. The code is capable of correcting any error pattern with $\lfloor \gamma/2 \rfloor$ or fewer errors using one-step majority-logic decoding [28, 31]. As a result, the minimum distance $d_{\mathbf{G}}^{(1)}$ of the type-I geometry-**G** LDPC code $\mathbf{C}_{\mathbf{G}}^{(1)}$ is at least γ+1.

Similarly, it can be shown that there are ρ check sums orthogonal on each code bit of a codeword in the type-II geometry-**G** code $\mathbf{C}_{\mathbf{G}}^{(2)}$. Therefore, $\mathbf{C}_{\mathbf{G}}^{(2)}$ is also one-step majority-logic decodable and has a minimum distance $d_{\mathbf{G}}^{(2)}$ at least ρ+1.

For a linear block code of length n specified by a parity check matrix of J rows, a graph can be constructed to display the relationship between its code bits and the check sums that check on them. This graph consists of two levels of vertices. The first level consists of n vertices which represent the n code bits of the code. These vertices, denoted $v_1, v_2, \cdots, v_n$, are called the code bit (or variable) vertices. The second level consists of J vertices which represent the J check sums, $s_1, s_2, \cdots, s_J$, that the code bits must satisfy. These vertices are called the check sum vertices. A code bit vertex v_l is connected to a check sum vertex s_j by an edge, denoted (v_l, s_j), if and only if the code bit v_l is contained in the check sum s_j. No two code bit vertices are connected and no two check sum vertices are connected. This graph is a bipartite graph [32] which was first proposed by Tanner [3] to study the structure and iterative decoding of LDPC codes, and hence it is called the Tanner graph. The number of edges that are connected to (or incident at) a code bit vertex v_l , called the degree of v_l, is simply the number of check sums that contain v_l. The number of edges that are incident at the check sum vertex s_j, called the degree of s_j, is simply the number of code bits that are checked by the check sum s_j.

For a regular LDPC code, the degrees of all the code bit vertices are the same and the degrees of all the check sum vertices are the same. Such a Tanner graph is said to be regular.

A cycle in a graph of vertices and edges is defined as a sequence of connected edges which starts from a vertex and ends at the same vertex, and satisfies the condition that no vertex (except the initial and the final vertex) appears more than once [32]. The number of edges on a cycle is called the length of the cycle. The length of the shortest cycle in a graph is called the girth of the graph. The Tanner graph of a linear block code contains no cycles of length 2 and no cycles of odd lengths. Therefore, the girth of the Tanner graph of a linear block code is at least 4.

In decoding a linear block code with the SPA decoding, the performance very much depends on cycles of short lengths in its Tanner graph. These short cycles, especially cycles of length 4, make successive decoding iterations highly correlated and hence severely limit the decoding performance [3, 10, 11, 20, 33–35]. Therefore, to use the SPA for decoding, it is important to design codes without short cycles in their Tanner graphs, especially cycles of length 4.

Both types of geometry-**G** LDPC codes are regular and hence their Tanner graphs are regular. Since the row and column weights of $\mathbf{H}_{\mathbf{G}}^{(1)}$ are ρ and γ, respectively, the degrees of each check sum vertex and each code bit vertex in the Tanner graph of the type-I geometry-**G** LDPC code $\mathbf{C}_{\mathbf{G}}^{(1)}$ are ρ and γ, respectively. Since $\mathbf{H}_{\mathbf{G}}^{(2)}$ is the transpose of $\mathbf{H}_{\mathbf{G}}^{(1)}$, the degrees of each check sum vertex and each code bit vertex in the Tanner graph of the type-II geometry-**G** code $\mathbf{C}_{\mathbf{G}}^{(2)}$ are γ and ρ, respectively. In fact, the Tanner graphs of the type-I and type-II geometry-**G** LDPC codes are dual graphs, i.e., the code bit vertices of one graph become the check sum vertices of the other graph and the check sum vertices of one graph become the code bit vertices of the other graph.

It follows from the structural properties of the parity check matrices $\mathbf{H}_{\mathbf{G}}^{(1)}$ and $\mathbf{H}_{\mathbf{G}}^{(2)}$ that no two code bits are checked simultaneously by two check sums. This implies that the Tanner graphs of both types of geometry-**G** LDPC codes do not contain cycles of length 4. However, they do contain cycles of length 6. To show this, we use the fundamental property of a finite geometry that any two points are connected by a line. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be any two points in the finite geometry **G**. Then there is a line $\mathcal{L}_1$ connecting $\mathbf{p}_1$ and $\mathbf{p}_2$. Let $\mathbf{p}_3$ be a third point in **G** but not on $\mathcal{L}_1$. Then there is a line $\mathcal{L}_2$ connecting $\mathbf{p}_1$ and $\mathbf{p}_3$ and a line $\mathcal{L}_3$ connecting $\mathbf{p}_2$ and $\mathbf{p}_3$. These three lines enclose a triangle with $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ as the vertices. In the Tanner graph of the type-I geometry-**G** LDPC code, these three lines $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ correspond to three check sum vertices, say s_1, s_2 and s_3, and the three points $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ correspond to three code bit vertices, say v_1, v_2 and v_3. Each of these three check sum vertices, s_1, s_2

and s_3 is connected to only two of the three code bit vertices v_1, v_2 and v_3. Since no two check sum vertices are connected to the same pair of code bit vertices, the edges connecting s_1, s_2 and s_3 to v_1, v_2 and v_3 form a cycle of length 6 in the Tanner graph of the type-I geometry-**G** code $\mathbf{C}_{\mathbf{G}}^{(1)}$. The number of cycles of length 6 is equal to the number of triangles in **G** which can be enumerated and is

$$N_6 = \frac{1}{6}n(n-1)(n-\rho). \tag{2}$$

Since the Tanner graphs of type-I and type-II geometry-**G** codes are dual, they have the same girth and the same cycle distribution. The above analysis shows that the girth of the Tanner graph of an LDPC code constructed based on the lines and points of a finite geometry is 6.

3. Euclidean and Projective Geometry LDPC Codes

Euclidean and projective geometries over finite fields form two large families of finite geometries. The structures of these two families of finite geometries have been well studied and can be found in any major text in combinatorics or groups of finite order. References [36–38] give a good exposition of this subject. A simple discussion of these two types of finite geometries can also be found in [28]. To make this paper self contained, the fundamental structural properties of lines and points of these two types of geometries are briefly described before the code construction.

Based on the lines and points of Euclidean and projective geometries, four classes of finite geometry LDPC codes can be constructed. They are: (1) type-I Euclidean geometry (EG)-LDPC codes; (2) type-II EG-LDPC codes; (3) type-I projective geometry (PG)-LDPC codes; and (4) type-II PG-LDPC codes. Among these four classes of codes, two are cyclic and two are quasi-cyclic.

3.1 Type-I EG-LDPC Codes

Let EG($m, 2^s$) be an m-dimensional Euclidean geometry over the Galois field GF(2^s) where m and s are two positive integers. This geometry consists of 2^{ms} points, each point is simply an m-tuple over GF(2^s). The all-zero m-tuple $\mathbf{0} = (0, 0, \cdots, 0)$ is called the origin. The 2^{ms} m-tuples over GF(2^s) that represent the points of EG($m, 2^s$) form an m-dimensional vector space over GF(2^s). Therefore, EG($m, 2^s$) is simply the m-dimensional vector space of all the 2^{ms} m-tuples over GF(2^s). A line in EG($m, 2^s$) is either a one-dimensional subspace of EG($m, 2^s$) or a coset of a one-dimensional subspace. Therefore, a line in EG($m, 2^s$) consists of 2^s points. There are

$$2^{(m-1)s}(2^{ms}-1)/(2^s-1) \tag{3}$$

lines in EG($m, 2^s$). Every line has $2^{(m-1)s} - 1$ lines parallel to it. For any point in EG($m, 2^s$), there are

$$(2^{ms} - 1)/(2^s - 1) \tag{4}$$

lines intersecting at this point.

Let GF(2^{ms}) be the extension field of GF(2^s). Each element in GF(2^{ms}) can be represented as an m-tuple over GF(2^s). Therefore, the 2^{ms} elements in GF(2^{ms}) may be regarded as the 2^{ms} points in EG($m, 2^s$) and hence GF(2^{ms}) may be regarded as the Euclidean geometry EG($m, 2^s$). Let α be a primitive element of GF(2^{ms}). Then $0 = \alpha^\infty, \alpha^0, \alpha^1, \alpha^2, ..., \alpha^{2^{ms}-2}$ form the 2^{ms} points of EG($m, 2^s$), where $0 = \alpha^\infty$ is the origin. Let α^j be a nonorigin point in EG($m, 2^s$). Then the 2^s points, $\{\beta\alpha^j\} \triangleq \{\beta\alpha^j : \beta \in GF(2^s)\}$, form a line in EG($m, 2^s$). Since for $\beta = 0, 0 \cdot \alpha^j = 0$, the line contains the origin α^∞ as a point. We say that $\{\beta\alpha^j\}$ passes through the origin. Let α^i and α^j be two linearly independent points in EG($m, 2^s$). Then the collection of the following points,

$$\{\alpha^i + \beta\alpha^j\} \triangleq \{\alpha^i + \beta\alpha^j : \beta \in GF(2^s)\},$$

form a line in EG($m, 2^s$) that passes through the point α^i. Lines $\{\beta\alpha^j\}$ and $\{\alpha^i + \beta\alpha^j\}$ do not have any point in common and hence they are parallel. Let α^k be a point which is linearly independent of α^i and α^j. Then lines $\{\alpha^i + \beta\alpha^j\}$ and $\{\alpha^i + \beta\alpha^k\}$ intersect at the point α^i.

Let $\mathbf{H}_{EG}^{(1)}(m, s)$ be a matrix over GF(2) whose rows are the incidence vectors of all the lines in EG($m, 2^s$) that do not pass through the origin and whose columns correspond to the $2^{ms} - 1$ nonorigin points in EG($m, 2^s$). The columns are arranged in the order of $\alpha^0, \alpha^1, \alpha^2, ..., \alpha^{2^{ms}-2}$, i.e., the i-th column corresponds to the point α^i. Then $\mathbf{H}_{EG}^{(1)}(m, s)$ consists of $n = 2^{ms} - 1$ columns and

$$J = (2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1) \tag{5}$$

rows. $\mathbf{H}_{EG}^{(1)}(m, s)$ has the following structures: (1) each row has weight $\rho = 2^s$; (2) each column has weight $\gamma = (2^{ms} - 1)/(2^s - 1) - 1$; (3) any two columns have at most one "1-component" in common , i.e., $\lambda = 1$; (4) any two rows have at most one "1-component" in common. The density of $\mathbf{H}_{EG}^{(1)}(m, s)$ is $r = 2^s/(2^{ms} - 1)$ which is small for $m \geq 2$ and $s \geq 2$. Therefore $\mathbf{H}_{EG}^{(1)}(m, s)$ is a low density matrix.

Let $\mathbf{C}_{EG}^{(1)}(m, s)$ be the null space of $\mathbf{H}_{EG}^{(1)}(m, s)$. Then $\mathbf{C}_{EG}^{(1)}(m, s)$ is a regular LDPC code of length $n = 2^{ms} - 1$. We call this code the type-I m-dimensional EG-LDPC code. Since the column weight of $\mathbf{H}_{EG}^{(1)}(m, s)$ is $\gamma = (2^{ms} - 1)/(2^s - 1) - 1$, the minimum distance of $\mathbf{C}_{EG}^{(1)}(m, s)$ is at least $(2^{ms} - 1)/(2^s - 1)$. It turns out that this EG-LDPC code is the one-step majority-logic decodable

(0,s)th order EG code constructed based on EG($m, 2^s$) [28,39,40] and is the dual code of a polynomial code [40–43]. Therefore, it is cyclic and its generator polynomial is completely characterized by its roots in GF(2^{ms}).

Let h be a nonnegative integer less than 2^{ms}. Then h can be expressed in radix-2^s form as follows:

$$h = \delta_0 + \delta_1 2^s + \cdots + \delta_{m-1} 2^{(m-1)s}$$

where $0 \leq \delta_i < 2^s$ for $0 \leq i < m$. The 2^s**-weight** of h, denoted $W_{2^s}(h)$, is defined as the following sum,

$$W_{2^s}(h) \triangleq \delta_0 + \delta_1 + \cdots + \delta_{m-1}. \tag{6}$$

For a nonnegative integer l, let $h^{(l)}$ be the remainder resulting from dividing $2^l h$ by $2^{ms} - 1$. Then $0 \leq h^{(l)} < 2^{ms} - 1$. Let $\mathbf{g}_{EG}^{(1)}(X)$ be the generator polynomial of the type-I m-dimensional EG-LDPC code. Let α be a primitive element of GF(2^{ms}). Then α^h is a root of $\mathbf{g}_{EG}^{(1)}(X)$ if and only if [28,39,40]

$$0 < \max_{0 \leq l < s} W_{2^s}(h^{(l)}) \leq (m-1)(2^s - 1). \tag{7}$$

From the above characterization of the roots of $\mathbf{g}_{EG}^{(1)}(X)$, it has been shown [40] that $\mathbf{g}_{EG}^{(1)}(X)$ has the following sequence of consecutive powers of α,

$$\alpha, \alpha^2, \cdots, \alpha^{(2^{ms}-1)/(2^s-1)-1},$$

as roots. It follows from the BCH-bound [27–30] that the minimum distance of the type-I m-dimensional EG-LDPC code is lower bounded as follows:

$$d_{EG}^{(1)}(m, s) \geq (2^{ms} - 1)/(2^s - 1). \tag{8}$$

This bound is exactly the same as the bound given above based on majority-logic decoding.

The number of parity check symbols of the type-I m-dimensional EG-LDPC code is of course equal to the degree of its generator polynomial $\mathbf{g}_{EG}^{(1)}(X)$. However, a combinatorial expression for this number has been derived by Lin [42].

A special subclass of EG-LDPC codes is the class of type-I two-dimensional EG-LDPC codes ($m = 2$). For any positive integer $s \geq 2$, the type-I two-dimensional EG-LDPC code has the following parameters [28, 40]:

$$\begin{array}{ll}
\text{Length} & n = 2^{2s} - 1, \\
\text{Number of parity bits} & n - k = 3^s - 1, \\
\text{Minimum distance} & d_{EG}^{(1)}(2, s) = 2^s + 1, \\
\text{Row weight of the parity check matrix} & \rho = 2^s, \\
\text{Column weight of the parity check matrix} & \gamma = 2^s.
\end{array} \qquad (9)$$

For this special case, the geometry EG($2, 2^s$) contains $2^{2s} - 1$ lines that do not pass through the origin. Therefore, the parity check matrix $\mathbf{H}_{EG}^{(1)}(2, s)$ of the type-I two-dimensional EG-LDPC code is a $(2^{2s} - 1) \times (2^{2s} - 1)$ square matrix. Actually, $\mathbf{H}_{EG}^{(1)}(2, s)$ can be constructed easily by taking the incidence vector $\mathbf{v}_{\mathcal{L}}$ of a line $\mathcal{L}$ in EG($2, 2^s$) that does not pass through the origin and then cyclically shifting this incidence vector $2^{2s} - 2$ times. This results in $2^{2s} - 1$ incidence vectors for the $2^{2s} - 1$ distinct lines in EG($2, 2^s$) that do not pass through the origin. The incidence vector $\mathbf{v}_{\mathcal{L}}$ and its $2^{2s} - 2$ cyclic shifts form the rows of the parity check matrix $\mathbf{H}_{EG}^{(1)}(2, s)$. Therefore, $\mathbf{H}_{EG}^{(1)}(2, s)$ is a square circulant matrix. A list of type-I two-dimensional EG-LDPC codes is given in Table 1.

Example 1. *Consider the 2-dimensional Euclidean geometry EG($2, 2^2$). Let α be a primitive element of GF($2^{2\times 2}$). The incidence vector for the line $\mathcal{L} = \{\alpha^7, \alpha^8, \alpha^{10}, \alpha^{14}\}$ is $(0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,1)$. This vector and its 14 cyclic shifts form the parity check matrix $\mathbf{H}_{EG}^{(1)}(2, 2)$. The null space of this matrix is the (15,7) type-I two-dimensional EG-LDPC code, the first code given in Table 1.*

It follows from the analysis of finite geometry LDPC codes given in Section 2 that the Tanner graph of the type-I m-dimensional Euclidean geometry code does not contain cycles of length 4. With some modifications to (2) (due to the exclusion of the origin), we find that the number of cycles of length 6 is

$$N_{6,EG}^{(1)}(m, s) \;=\; \frac{1}{6} \cdot 2^s \cdot (2^{(m-1)s} - 1)(2^{ms} - 3 \cdot 2^s + 3)(2^{ms} - 1). \qquad (10)$$

We see that the Tanner graph of the code contains many cycles of length 6.

3.2 Type-II EG-LDPC Codes

Let $\mathbf{H}_{EG}^{(2)}(m, s) = [\mathbf{H}_{EG}^{(1)}(m, s)]^T$. Then $\mathbf{H}_{EG}^{(2)}(m, s)$ is a matrix with $2^{ms} - 1$ rows and

$$J = (2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1)$$

columns. The rows of this matrix correspond to the nonorigin points of EG($m, 2^s$) and the columns correspond to the lines in EG($m, 2^s$) that do not pass through the origin. Its column and row weights are $\gamma = 2^s$ and $\rho = (2^{ms} - 1)/(2^s - 1) - 1$, respectively. Any two rows of this matrix have exactly one "1-component" in common, and any two columns have at most one "1-component" in common.

The null space of $\mathbf{H}_{EG}^{(2)}(m, s)$ gives an LDPC code of length J. This code is called the type-II m-dimensional EG-LDPC code. This code is also one-step majority-logic decodable and has minimum distance $d_{EG}^{(2)}(m, s)$ at least $2^s + 1$. Since $\mathbf{H}_{EG}^{(1)}(m, s)$ and $\mathbf{H}_{EG}^{(2)}(m, s)$ have the same rank, $\mathbf{C}_{EG}^{(1)}(m, s)$ and $\mathbf{C}_{EG}^{(2)}(m, s)$ have the same number of parity check symbols. Since the Tanner graphs of $\mathbf{C}_{EG}^{(1)}(m, s)$ and $\mathbf{C}_{EG}^{(2)}(m, s)$ are dual, they have the same cycle distribution. For $m = 2$, since the parity check matrix $\mathbf{H}_{EG}^{(1)}(2, s)$ of the type-I two-dimensional EG-LDPC code $\mathbf{C}_{EG}^{(1)}(2, s)$ is a square matrix whose rows are the cyclic shifts of the first row, the rows of $\mathbf{H}_{EG}^{(2)}(2, s)$ are simply permutations of the rows of $\mathbf{H}_{EG}^{(1)}(2, s)$. Therefore, $\mathbf{C}_{EG}^{(1)}(2, s)$ and $\mathbf{C}_{EG}^{(2)}(2, s)$ are identical.

In general, for $m \neq 2$, $\mathbf{C}_{EG}^{(2)}(m, s)$ is not cyclic but it can be put in quasi-cyclic form. To see this, consider the $(2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1)$ lines in EG($m, 2^s$) that do not pass through the origin. The incidence vectors of these lines can be partitioned into

$$K = (2^{(m-1)s} - 1)/(2^s - 1) \tag{11}$$

cyclic classes. Each of these K cyclic classes contains $2^{ms} - 1$ incidence vectors of lines which are obtained by cyclically shifting any incidence vector in the class $2^{ms} - 1$ times. For each cyclic class of incidence vectors of lines, we can choose a representative and the rest of the incidence vectors are generated by cyclically shifting this representative. Now we construct a $(2^{ms} - 1) \times K$ matrix $\mathbf{H}_0$, whose K columns are the K representative incidence vectors of the cyclic classes. For $1 \leq i \leq 2^{ms} - 2$, let $\mathbf{H}_i$ be a $(2^{ms} - 1) \times K$ matrix whose columns are the i-th (downward) cyclic shifts of the columns of $\mathbf{H}_0$. Form the following matrix

$$\mathbf{H}_{EG}^{(2)}(m, s) = [\mathbf{H}_0, \mathbf{H}_1, ..., \mathbf{H}_{2^{ms}-2}]. \tag{12}$$

Then the null space of $\mathbf{H}_{EG}^{(2)}(m, s)$ gives a quasi-cyclic type-II m-dimensional EG-LDPC code $\mathbf{C}_{EG}^{(2)}(m, s)$. Every K cyclic shifts of a codeword in $\mathbf{C}_{EG}^{(2)}(m, s)$ is also a codeword in $\mathbf{C}_{EG}^{(2)}(m, s)$. Encoding of quasi-cyclic codes can also be achieved with linear feedback shift registers [27].

3.3 Type-I PG-LDPC Codes

The construction of PG-LDPC codes for both types is based on the lines and points of projective geometries over finite fields. For the purpose of code construction, a brief description of this family of finite geometries is given here.

Let GF($2^{(m+1)s}$) be the extension field of GF(2^s). Let α be a primitive element of GF($2^{(m+1)s}$).

Let

$$n = (2^{(m+1)s} - 1)/(2^s - 1) \tag{13}$$

and $\beta = \alpha^n$. Then the order of β is $2^s - 1$. The 2^s elements 0,1, $\beta, \beta^2, \cdots, \beta^{2^s-2}$ form all the elements of GF(2^s). Consider the first n powers of α, $\Lambda = \{\alpha^0, \alpha^1, \alpha^2, \cdots, \alpha^{n-1}\}$. Partition the nonzero elements of GF($2^{(m+1)s}$) into n disjoint subsets as follows:

$$\{\alpha^i, \beta\alpha^i, \beta^2\alpha^i, \cdots, \beta^{2^s-2}\alpha^i\}, \tag{14}$$

for $0 \leq i < n$. Each set consists of $2^s - 1$ elements and each element is a multiple of the first element in the set. Represent each set by its first element as follows: $(\alpha^i) \triangleq \{\alpha^i, \beta\alpha^i, \cdots, \beta^{2^s-2}\alpha^i\}$ with $0 \leq i < n$. For any $\alpha^j \in$ GF($2^{(m+1)s}$), if $\alpha^j = \beta^l \cdot \alpha^i$ with $0 \leq i < n$, then α^j is in (α^i) and represented by (α^i).

If we represent each element in GF($2^{(m+1)s}$) as an $(m+1)$-tuple over GF(2^s), then (α^i) consists of $2^s - 1$ $(m+1)$-tuples over GF(2^s). The $(m+1)$-tuple for α^i represents the $2^s - 1$ $(m+1)$-tuples in (α^i). The $(m+1)$-tuple over GF(2^s) that represents (α^i) may be regarded as a point in a finite geometry over GF(2^s). Then the points, $(\alpha^0), (\alpha^1), (\alpha^2), \cdots, (\alpha^{n-1})$, form an m-dimensional projective geometry over GF(2^s), denoted PG($m, 2^s$) [28, 36–38]. Note that the $2^s - 1$ elements in $\{\alpha^i, \beta\alpha^i, \cdots, \beta^{2^s-2}\alpha^i\}$ are considered to be the same point in PG($m, 2^s$) and a projective geometry does not have an origin.

Let (α^i) and (α^j) be any two distinct points in PG($m, 2^s$). Then the line passing through (or connecting) (α^i) and (α^j) consists of points of the following form: $(\eta_1\alpha^i + \eta_2\alpha^j)$, where η_1 and η_2 are from GF(2^s) and are not both equal to zero. Since $(\eta_1\alpha^i + \eta_2\alpha^j)$ and $(\beta^l\eta_1\alpha^i + \beta^l\eta_2\alpha^j)$ are the same point, therefore, each line in PG$(2^{(m+1)s})$ consists of

$$((2^s)^2 - 1)/(2^s - 1) = 2^s + 1 \tag{15}$$

points.

Let (α^l) be a point not on the line $\{(\eta_1\alpha^i + \eta_2\alpha^j)\}$. Then the line $\{(\eta_1\alpha^i + \eta_2\alpha^j)\}$ and the line $\{(\eta_1\alpha^l + \eta_2\alpha^j)\}$ have (α^j) as a common point (the only common point). We say that they intersect at (α^j). The number of lines in PG($m, 2^s$) that intersect at a given point is

$$(2^{ms} - 1)/(2^s - 1). \tag{16}$$

There are

$$J = (2^{ms} + \cdots + 2^s + 1)(2^{(m-1)s} + \cdots + 2^s + 1)/(2^s + 1) \tag{17}$$

lines in PG($m, 2^s$).

Form a matrix $\mathbf{H}_{PG}^{(1)}(m, s)$ whose rows are the incidence vectors of the lines in PG($m, 2^s$) and whose columns correspond to the points of PG($m, 2^s$). The columns are arranged in the order, $(\alpha^0), (\alpha^1), \cdots, (\alpha^{n-1})$. $\mathbf{H}_{PG}^{(1)}(m, s)$ has J rows and n columns. It follows from the structural properties of lines and points described above that $\mathbf{H}_{PG}^{(1)}(m, s)$ has the following structural properties: (1) each row has weight $\rho = 2^s + 1$; (2) each column has weight $\gamma = (2^{ms} - 1)/(2^s - 1)$; (3) any two columns have exactly one "1-component" in common; and (4) any two rows have at most one "1-component" in common. The density of $\mathbf{H}_{PG}^{(1)}(m, s)$ is $r = (2^{2s} - 1)/(2^{(m+1)s} - 1)$. For $m \geq 2$, r is relatively small. Therefore $\mathbf{H}_{PG}^{(1)}(m, s)$ is a sparse matrix.

Let $\mathbf{C}_{PG}^{(1)}(m, s)$ be the null space of $\mathbf{H}_{PG}^{(1)}(m, s)$. Then $\mathbf{C}_{PG}^{(1)}(m, s)$ is a regular LDPC code, called the type-I m-dimensional PG-LDPC code. Since the column weight of $\mathbf{H}_{PG}^{(1)}(m, s)$ is $\gamma = (2^{ms} - 1)/(2^s - 1)$, the minimum distance $d_{PG}^{(1)}(m, s)$ of $\mathbf{C}_{PG}^{(1)}(m, s)$ is at least $(2^{ms} - 1)/(2^s - 1) + 1$. This regular LDPC code turns out to be the one-step majority-logic decodable (1,s)th order PG code constructed based on the lines and points of PG($m, 2^s$) discovered in the late 1960's [28, 44, 45] and is the dual of a nonprimitive polynomial code [40–43]. It is cyclic and therefore can be encoded with a linear feedback shift register based on its generator polynomial.

Let h be a nonnegative integer less than $2^{(m+1)s} - 1$. For a nonnegative integer l, let $h^{(l)}$ be the remainder resulting from dividing $2^l h$ by $2^{(m+1)s} - 1$. The 2^s-weight of h, $W_{2^s}(h)$, is defined by (6). Let $\mathbf{g}_{PG}^{(1)}(X)$ be the generator polynomial of the type-I m-dimensional PG-LDPC code constructed based on PG($m, 2^s$). Let α be a primitive element of GF($2^{(m+1)s}$). Then $\mathbf{g}_{PG}^{(1)}(X)$ has α^h as a root if and only if h is divisible by $2^s - 1$ and

$$0 \leq \max_{0 \leq l < s} W_{2^s}(h^{(l)}) = j(2^s - 1) \tag{18}$$

with $0 \leq j \leq m - 1$ [28, 40, 44]. Let $\xi = \alpha^{2^s - 1}$. The order of ξ is then $n = (2^{(m+1)s} - 1)/(2^s - 1)$. From the characterization of the roots of $\mathbf{g}_{PG}^{(1)}(X)$ given by (18), it can be shown [39, 40] that $\mathbf{g}_{PG}^{(1)}(X)$ has the following consecutive powers of ξ,

$$\xi^0, \xi^1, \xi^2, \cdots, \xi^{(2^{ms}-1)/(2^s-1)} \tag{19}$$

as roots. Therefore, it follows from the BCH bound that the minimum distance of the type-I m-dimensional PG-LDPC code is lower bounded as follows:

$$d_{PG}^{(1)}(m, s) \geq (2^{ms} - 1)/(2^s - 1) + 1. \tag{20}$$

This bound is exactly the bound derived based on one-step majority-logic decoding.

The number of parity-check symbols of the type-I m-dimensional PG-LDPC for a given s can be enumerated by determining the roots of its generator polynomial. A combinatorial expression for this number can be found in [42].

A special subclass of PG-LDPC codes is the class of type-I two-dimensional PG-LDPC codes constructed based on PG$(2, 2^s)$ for various s. For any positive integer $s \geq 2$, the type-I two-dimensional PG-LDPC code has the following parameters [28, 42, 46, 47]:

$$\begin{array}{ll} \text{Length} & n = 2^{2s} + 2^s + 1, \\ \text{Number of parity bits} & n - k = 3^s + 1, \\ \text{Minimum distance} & d_{PG}^{(1)}(2, s) = 2^s + 2, \\ \text{Row weight of the parity check matrix} & \rho = 2^s + 1, \\ \text{Column weight of the parity check matrix} & \gamma = 2^s + 1. \end{array} \tag{21}$$

It is a difference-set code [28,46]. The parity check matrix $\mathbf{H}_{PG}^{(1)}(2, s)$ of this code is a $(2^{2s} + 2^s + 1) \times (2^{2s} + 2^s + 1)$ square matrix, which can be formed by taking the incidence vector of a line in PG$(2, 2^s)$ and its $2^{2s} + 2^s$ cyclic shifts as rows. A list of type-I two-dimensional PG-LDPC codes is given in Table 2.

3.4 Type-II PG-LDPC Codes

Let $\mathbf{H}_{PG}^{(2)}(m, s)$ be the transpose of $\mathbf{H}_{PG}^{(1)}(m, s)$. Then the rows and columns of $\mathbf{H}_{PG}^{(2)}(m, s)$ correspond to the points and lines of PG$(m, 2^s)$, respectively. $\mathbf{H}_{PG}^{(2)}(m, s)$ is also a low density matrix with row weight $\rho = (2^{ms} - 1)/(2^s - 1)$ and column weight $\gamma = 2^s + 1$. The null space of $\mathbf{H}_{PG}^{(2)}(m, s)$ gives a regular LDPC code, called the type-II m-dimensional PG-LDPC code, denoted $\mathbf{C}_{PG}^{(2)}(m, s)$. This code is also one-step majority-logic decodable with minimum distance $d_{PG}^{(2)}(m, s)$ at least $2^s + 2$. For $m = 2$, $\mathbf{H}_{PG}^{(1)}(2, s)$ is a square matrix whose rows are the n cyclic shifts of the first row, and the rows of $\mathbf{H}_{PG}^{(2)}(2, s)$ are simply permutations of the rows of $\mathbf{H}_{PG}^{(1)}(2, s)$. As a result, $\mathbf{C}_{PG}^{(1)}(2, s)$ and $\mathbf{C}_{PG}^{(2)}(2, s)$ are identical. In general, for $m \neq 2$, $\mathbf{C}_{PG}^{(2)}(m, s)$ is not cyclic but it can be put in quasi-cyclic form in a similar manner as for the type-II m-dimensional EG-LDPC code.

Example 2. *Let $m = 3$ and $s = 2$. The three-dimensional projective geometry PG($3, 2^2$) has 85 points and 357 lines. To construct the type-I three-dimensional PG-LDPC code $\mathbf{C}_{PG}^{(1)}(3, 2)$, we form the parity-check matrix $\mathbf{H}_{PG}^{(1)}(3, 2)$ whose rows are the incidence vectors of all the 357 lines in PG($3, 2^2$) and whose columns correspond to all the 85 points in PG($3, 2^2$). The matrix $\mathbf{H}_{PG}^{(1)}(3, 2)$ can be put in* *the following form*

$$\mathbf{H}_{PG}^{(1)}(3,2) = \begin{bmatrix} \mathbf{I}_{17}\,\mathbf{I}_{17}\,\mathbf{I}_{17}\,\mathbf{I}_{17}\,\mathbf{I}_{17} \\ \mathbf{H}_1 \\ \mathbf{H}_2 \\ \mathbf{H}_3 \\ \mathbf{H}_4 \end{bmatrix},$$

where $\mathbf{I}_{17}$ is the 17×17 identity matrix and each $\mathbf{H}_i$ is an 85×85 circulant matrix. The circulant matrices $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3, \mathbf{H}_4$ have the following vectors (in polynomial form) as their first rows, respectively: $\mathbf{h}_1(X) = 1 + X^{24} + X^{40} + X^{71} + X^{84}$, $\mathbf{h}_2(X) = X^1 + X^{49} + X^{58} + X^{81} + X^{84}$, $\mathbf{h}_3(X) = X^3 + X^{14} + X^{32} + X^{78} + X^{84}$, $\mathbf{h}_4(X) = X^{16} + X^{33} + X^{50} + X^{67} + X^{84}$. The matrix $\mathbf{H}_{PG}^{(1)}(3,2)$ has row weight $\rho = 5$ and column weight $\gamma = 21$. The null space of $\mathbf{H}_{PG}^{(1)}(3,2)$ gives a (85,24) type-I three-dimensional PG-LDPC code $\mathbf{C}_{PG}^{(1)}(3,2)$. The companion code of this code is the null space of the parity-check matrix $\mathbf{H}_{PG}^{(2)}(3,2) = [\mathbf{H}_{PG}^{(1)}(3,2)]^T$. The matrix $\mathbf{H}_{PG}^{(2)}(3,2)$ has row weight $\rho = 21$ and column weight $\gamma = 5$. $\mathbf{C}_{PG}^{(2)}(3,2)$ has the same number of parity check bits as $\mathbf{C}_{PG}^{(1)}(3,2)$. Hence, it is a (357, 296) PG-LDPC code with minimum distance at least 6.

It follows from (2), (13) and (15) that the number of cycles of length 6 in the Tanner graph of an m-dimensional (type-I or II) PG-LDPC code is

$$N_{6,PG} = \frac{1}{6}(2^{(m+1)s} - 1)(2^{ms} - 1)(2^{(m-1)s} - 1)\left(\frac{2^s}{2^s - 1}\right)^3. \tag{22}$$

4. Decoding of Finite Geometry LDPC Codes

Finite geometry LDPC codes can be decoded in various ways, namely one-step MLG decoding [28,31], BF decoding [1, 2], weighted MLG decoding, weighted BF decoding, APP decoding [2, 31] and SPA decoding [10, 11, 15, 20, 22]. These decoding methods range from low to high decoding complexity and from reasonably good to very good error performance. They provide a wide range of trade-offs among decoding complexity, decoding speed and error performance. MLG and BF decodings are hard-decision decoding and they can be easily implemented. Since finite geometry LDPC codes have relatively good minimum distances, they provide relatively large coding gains over the uncoded system. MLG decoding has the least decoding delay and very high decoding speed can be achieved. APP and the SPA decodings are soft-decision decoding schemes. They require extensive decoding computation but they provide the best error performance. Weighted MLG and BF decodings are between hard-

and soft-decision decodings. They improve the error performance of the MLG and BF decodings with some additional computational complexity. They offer a good trade-off between error performance and decoding complexity. The SPA decoding gives the best error performance among the six decoding methods for finite geometry LDPC codes and yet is practically implementable.

The first MLG decoding algorithm was devised by Reed [48] for decoding Reed-Muller codes [27]. Later Reed's algorithm was reformulated and generalized by Massey for decoding both block and convolutional codes [31]. A thorough discussion of various types and implementation of MLG decoding can be found in [28]. Therefore, we will not describe this decoding method here. APP decoding also gives minimum error performance, however it is computationally intractable and hence it will not be discussed here for decoding finite geometry LDPC codes. A good presentation of APP decoding can be found in [1, 2].

Suppose a finite geometry (EG- or PG-) LDPC code **C** is used for error control over an AWGN channel with zero mean and power spectral density $N_0/2$. Assume BPSK signaling with unit energy. A codeword $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ is mapped into a bipolar sequence $\mathbf{x} = (x_0, x_1, \cdots, x_{n-1})$ before its transmission where $x_l = (2v_l - 1) = +1$ for $v_l = 1$ and $x_l = -1$ for $v_l = 0$ with $0 \le l \le n-1$. Let $\mathbf{y} = (y_0, y_1, \cdots, y_{n-1})$ be the soft-decision received sequence at the output of the receiver matched filter. For $0 \le l \le n-1, y_l = \pm 1 + n_l$ where n_l is a Gaussian random variable with zero mean and variance $N_0/2$. Let $\mathbf{z} = (z_0, z_1, \cdots, z_{n-1})$ be the binary hard decision received sequence obtained from **y** as follows: $z_l = 1$ for $y_l > 0$ and $z_l = 0$ for $y_l \le 0$.

Let **H** be the parity check matrix of the finite geometry LDPC code **C** with J rows and n columns. Let $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_J$, denote the rows of **H**, where $\mathbf{h}_j = (h_{j,0}, h_{j,1}, \cdots, h_{j,n-1})$ for $1 \le j \le J$. Then

$$\mathbf{s} = (s_1, s_2, \cdots, s_J) = \mathbf{z} \cdot \mathbf{H}^T. \tag{23}$$

gives the syndrome of the received sequence **z**, where the j-th syndrome component s_j is given by the check-sum,

$$s_j = \mathbf{z} \cdot \mathbf{h}_j = \sum_{l=0}^{n-1} z_l h_{j,l}. \tag{24}$$

The received vector **z** is a codeword if and only if $\mathbf{s} = \mathbf{0}$. If $\mathbf{s} \neq \mathbf{0}$, errors in **z** are detected. A nonzero syndrome component s_j indicates a parity failure. The total number of parity failures is equal to the number of nonzero syndrome components in $\mathbf{s}$. Let

$$\mathbf{e} = (e_0, e_1, \cdots, e_{n-1}) = (v_0, v_1, \cdots, v_{n-1}) + (z_0, z_1, \cdots, z_{n-1}) \tag{25}$$

Then **e** is the error pattern in **z**. This error pattern **e** and the syndrome **s** satisfy the condition,

$$\mathbf{s} = (s_1, s_2, \cdots, s_J) = \mathbf{e} \cdot \mathbf{H}^T \tag{26}$$

where

$$s_j = \mathbf{e} \cdot \mathbf{h}_j = \sum_{l=0}^{n-1} e_l h_{j,l} \tag{27}$$

for $1 \leq j \leq J$.

4.1 BF Decoding

BF decoding of LDPC codes was devised by Gallager in the early 1960's [1,2]. When detectable errors occur during the transmission, there will be parity failures in the syndrome $\mathbf{s} = (s_1, s_2, \cdots, s_J)$ and some of the syndrome bits are equal to 1. BF decoding is based on the change of the number of parity failures in $\{\mathbf{z} \cdot \mathbf{h}_j : 1 \leq j \leq J\}$ when a bit in the received sequence **z** is changed.

First the decoder computes all the parity check sums based on (24) and then changes any bit in the received vector $\mathbf{z}$ that is contained in more than some fixed number δ of unsatisfied parity check equations. Using these new values, the parity check sums are recomputed, and the process is repeated until the parity check equations are all satisfied. This decoding is an iterative decoding algorithm. The parameter δ, called **threshold**, is a design parameter which should be chosen to optimize the error performance while minimizing the number of computations of parity check sums. The value of δ depends on the code parameters $\rho, \gamma, d_{min}(C)$ and the signal-to-noise ratio (SNR).

If decoding fails for a given value of δ, then the value of δ can be reduced to allow further decoding iterations. For error patterns with number of errors less than or equal to the error correcting capability of the code, the decoding will be completed in one or a few iterations. Otherwise, more decoding iterations are needed. Therefore, the number of decoding iterations is a random variable and is a function of the channel SNR. A limit may be set on the number of iterations. When this limit is reached, the decoding process is terminated to avoid excessive computations. Due to the nature of low density parity checks, the above decoding algorithm corrects many error patterns with number of errors exceeding the error correcting capability of the code.

A very simple BF decoding algorithm is given below:

Step 1 Compute the parity check sums (syndrome bits). If all the parity check equations are satisfied (i.e., all the syndrome bits are zero), stop the decoding.

Step 2 Find the number of unsatisfied parity check equations for each code bit position, denoted f_i,

$i = 0, 1, ..., n-1$.

Step 3 Identify the set Ω of bits for which f_i is the largest.

Step 4 Flip the bits in set Ω.

Step 5 Repeat steps 1 to 4 until all the parity check equations are satisfied (for this case, we stop the iteration in step 1) or a predefined maximum number of iterations is reached.

BF decoding requires only logical operations. The number of logical operations N_{BF} performed for each decoding iteration is linearly proportional to $J\rho$ (or $n\gamma$), say $N_{BF} = K_{BF}J\rho$, where the constant K_{BF} depends on the implementation of the BF decoding algorithm. Typically, K_{BF} is less than three. The simple BF decoding algorithm can be improved by using adaptive thresholds δ's. Of course, this improvement is achieved at the expense of more computations. EG- and PG-LDPC codes perform well with the BF decoding due to the large number of check sums orthogonal on each code bit.

4.2 Weighted MLG and BF Decodings

The simple hard-decision MLG and BF decodings can be improved to achieve better error performance by including some kind of reliability information (or measure) of the received symbols in their decoding decisions. Of course, additional decoding complexity is required for such performance improvement.

Consider the soft-decision received sequence $\mathbf{y} = (y_0, y_1, \cdots, y_{n-1})$. For the AWGN channel, a simple measure of the reliability of a received symbol y_l is its magnitude, $|y_l|$. The larger the magnitude $|y_l|$ is, the larger the reliability of the hard-decision digit z_l is. Many algorithms for decoding linear block codes based on this reliability measure have been devised. In the following, this reliability measure is used to modify the one-step majority logic decoding and the BF decoding.

Again consider a finite geometry LDPC code specified by a parity check matrix **H** with J rows, $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_J$. For $0 \leq l \leq n-1$ and $1 \leq j \leq J$, define

$$|y_j|_{min}^{(l)} \triangleq \{\min\{|y_i|\} : 0 \leq i \leq n-1, h_{j,i} = 1\} \tag{28}$$

and

$$E_l \triangleq \sum_{s_j^{(l)} \in S_l} (2s_j^{(l)} - 1)|y_j|_{min}^{(l)}, \tag{29}$$

where S_l is the set of check sums orthogonal on bit-position l. The value E_l is simply a weighted check sum that is orthogonal on the code bit position l. Let $\mathbf{e} = (e_0, e_1, \cdots, e_{n-1})$ be the error pattern to be estimated. Then the one-step MLG decoding can be modified based on the weighted check sum

E_l as follows:

$$e_l = \begin{cases} 1, & \text{for } E_l > 0, \\ 0, & \text{for } E_l \leq 0, \end{cases} \tag{30}$$

for $0 \leq l \leq n-1$. The above decoding algorithm is called weighted MLG decoding and was first proposed by Kolesnik in 1971 [49] for decoding majority logic decodable codes.

The decision rule given by (30) can be used in BF decoding. In this case the decoding is carried out as follows:

Step 1. Compute the check sums. If all the parity check equations are satisfied, stop the decoding.

Step 2. Compute E_l based on (29), for $0 \leq l \leq n-1$.

Step 3. Find the bit position l for which E_l is the largest.

Step 4. Flip the bit z_l.

Step 5. Repeat Step 1 to 4. This process of bit flipping continues until all the parity check equations are satisfied or a preset maximum number of iterations is reached.

This modified BF algorithm is called weighted BF decoding algorithm.

The above weighted decoding algorithms are in a way soft-decision decoding algorithms and require real addition operations to compute the weighted check sums, E_l's, to make decisions. Since a real addition operation is much more complex than a logical operation, the computational complexities of both weighted MLG and BF decodings are dominated by the total number of real additions needed to decode a received sequence. From (29), we can readily see that for weighted MLG decoding, the number of real additions required for decoding a received sequence is $K_{MLG}(J\rho + n\gamma)$ where K_{MLG} is a constant. However for weighted BF decoding, the number of real additions needed for each decoding iteration is $K_{MLG}(J\rho + n\gamma)$. Since $J\rho$ (or $n\gamma$) is the total number of 1-entries in the parity check matrix **H** of the code, the computational complexities of both weighted MLG and BF decodings are linearly proportional to the total number of 1-entries in **H**.

4.3 The Sum-Product Algorithm

The sum-product algorithm (SPA) [17–20, 33] is an iterative decoding algorithm based on belief propagation [10, 11, 20–22] which is extremely efficient for decoding LDPC codes. Like MAP (maximum a posteriori probability) decoding algorithm [50], it is a symbol-by-symbol soft-in/soft-out decoding algorithm. It processes the received symbols iteratively to improve the reliability of each decoded code symbol based on the parity check sums computed from the hard-decisions of the received symbols and the parity check matrix **H** that specifies the code. The reliability of a decoded symbol can be measured by its marginal posteriori probability, its log-likelihood ratio (LLR) or the value of its corresponding received symbol. The computed reliability measures of code symbols at the end of each decoding iteration are used as inputs for the next iteration. The decoding iteration process continues until a certain stopping condition is satisfied. Then based on the computed reliability measures of code symbols, hard decisions are made.

Again we consider a finite geometry LDPC code **C** of length n specified by a parity check matrix **H** with J rows, $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_J$. For $1 \leq j \leq J$, define the following index set for $\mathbf{h}_j$:

$$B(\mathbf{h}_j) = \{l : h_{j,l} = 1, 0 \leq l < n\}, \tag{31}$$

which is called the support of $\mathbf{h}_j$.

The implementation of the SPA decoding [10] is based on the computation of marginal a posteriori probabilities, $P(v_l|\mathbf{y})'$s, for $0 \leq l < n$. Then the LLR for each code bit is given by

$$L(v_l) = \log \frac{P(v_l = 1|\mathbf{y})}{P(v_l = 0|\mathbf{y})}. \tag{32}$$

Let $p_l^0 = P(v_l = 0)$ and $p_l^1 = P(v_l = 1)$ be the prior probabilities of $v_l = 0$ and $v_l = 1$, respectively.

For $0 \leq l < n, 1 \leq j \leq J$ and each $\mathbf{h}_j \in A_l$, let $q_{j,l}^{x,(i)}$ be the conditional probability that the transmitted code bit v_l has value x, given the check-sums computed based on the check vectors in $A_l \backslash \mathbf{h}_j$ at the i-th decoding iteration. For $0 \leq l < n, 1 \leq j \leq J$ and $\mathbf{h}_j \in A_l$, let $\sigma_{j,l}^{x,(i)}$ be the conditional probability that the check-sum s_j is satisfied, given $v_l = x$ (0 or 1) and the other code bits in $B(\mathbf{h}_j)$ have a separable distribution $\{q_{j,t}^{v_t,(i)} : t \in B(\mathbf{h}_j) \backslash l\}$, i.e.,

$$\sigma_{j,l}^{x,(i)} = \sum_{\{v_t : t \in B(\mathbf{h}_j) \backslash l\}} P(s_j | v_l = x, \{v_t : t \in B(\mathbf{h}_j) \backslash l\}) \cdot \prod_{t \in B(\mathbf{h}_j) \backslash l} q_{j,t}^{v_t,(i)}. \tag{33}$$

The computed values of $\sigma_{j,l}^{x,(i)}$ are then used to update the values of $q_{j,l}^{x,(i+1)}$ as follows:

$$q_{j,l}^{x,(i+1)} = \alpha_{j,l}^{(i+1)} p_l^x \prod_{\mathbf{h}_t \in A_l \backslash \mathbf{h}_j} \sigma_{t,l}^{x,(i)} \tag{34}$$

where $\alpha_{j,l}^{(i+1)}$ is chosen such that $q_{j,l}^{0,(i+1)} + q_{j,l}^{1,(i+1)} = 1$.

At the i-th iteration step, the pseudo-posterior probabilities are given by

$$P^{(i)}(v_l = x|\mathbf{y}) = \alpha_l^{(i)} p_l^x \prod_{\mathbf{h}_j \in A_l} \sigma_{j,l}^{x,(i-1)}, \tag{35}$$

where α_l^i is chosen such that $P^{(i)}(v_l = 0|\mathbf{y}) + P^{(i)}(v_l = 1|\mathbf{y}) = 1$. Based on these probabilities, we can form the vector $\mathbf{z}^{(i)} = (z_0^{(i)}, z_1^{(i)}, \cdots, z_{n-1}^{(i)})$ as the decoded candidate with

$$z_l^{(i)} = \begin{cases} 1, & \text{for } P^{(i)}(v_l = 1|\mathbf{y}) > 0.5 \\ 0, & \text{otherwise.} \end{cases}$$

Then compute $\mathbf{z}^{(i)} \cdot \mathbf{H}^T$. If $\mathbf{z}^{(i)} \cdot \mathbf{H}^T = \mathbf{0}$, stop decoding iteration process and output $\mathbf{z}^{(i)}$ as the decoded codeword.

The SPA decoding in terms of probability consists of the following steps:

Initialization: Set $i = 0$, maximum number of iterations to I_{max}. For every pair (j, l) such that $h_{j,l} = 1$ with $1 \leq j \leq J$ and $0 \leq l < n$, set $q_{j,l}^{0,(0)} = p_l^0$ and $q_{j,l}^{1,(0)} = p_l^1$.

Step 1: For $0 \leq l < n, 1 \leq j \leq J$ and each $\mathbf{h}_j \in A_l$, compute the probabilities, $\sigma_{j,l}^{0,(i)}$ and $\sigma_{j,l}^{1,(i)}$. Go to Step 2.

Step 2: For $0 \leq l < n, 1 \leq j \leq J$ and each $\mathbf{h}_j \in A_l$, compute the values of $q_{j,l}^{0,(i+1)}$ and $q_{j,l}^{1,(i+1)}$ and the values of $P^{(i+1)}(v_l = 0|\mathbf{y})$ and $P^{(i+1)}(v_l = 1|\mathbf{y})$. Form $\mathbf{z}^{(i+1)}$ and test $\mathbf{z}^{(i+1)} \cdot \mathbf{H}^T$. If $\mathbf{z}^{(i+1)} \cdot \mathbf{H}^T = \mathbf{0}$ or the maximum iteration number I_{max} is reached, go to Step 3. Otherwise, set $i := i + 1$ and go to Step 1.

Step 3: Output $\mathbf{z}^{(i+1)}$ as the decoded codeword and stop the decoding process.

In the above SPA decoding, real number addition, subtraction, multiplication, division, exponential and logarithm operations are needed. In implementation, the last four types of operations are more complex than addition and subtraction. For this reason, we simply ignore the number of additions and subtractions in analyzing the computational complexity. From (33) to (35), we find that the number of multiplications and divisions needed in each iteration of the SPA decoding is of the order $O(2J\rho + 4n\gamma)$ and the number of exponential and logarithm operations needed for each iteration of decoding is of the order $O(n)$. A detail exposition of the SPA can be found in [10, 17–20, 33].

4.4 Two-Stage Hybrid Decoding

The SPA decoding is computationally expensive. Each decoding iteration requires many real number computations. If decoding of a code with the SPA converges slowly, a large number of iterations is needed to achieve the desired performance. A large number of iterations results in a large number of computations and long decoding delay which is not desirable in high speed communications. However for finite geometry LDPC codes, this difficulty can be overcome by using a two-stage hybrid soft/hard decoding scheme. At the first stage, a code is decoded with the SPA with a small fixed number of iterations, say I. At the completion of the I-th iteration, hard decisions of decoded symbols are made based on their LLR's. This results in a binary sequence **z** of estimated code bits. This sequence **z** is then decoded with the simple one-step MLG decoding. This two-stage hybrid decoding works well for finite geometry LDPC codes because they have large minimum distances and SPA decoding of these codes converges very fast. Simulation results for many codes show that the performance gap between 5 iterations and 100 iterations is within 0.2 dB. Therefore, at the first stage, we may set the number of iterations for the SPA decoding to 5 or less (in many cases, 2 iterations are enough). The resulting estimated code sequence **z** may still contain a small number of errors. These errors will be corrected by the one-step MLG decoding at the second stage due to the large majority-logic error correcting capability of the finite geometry LDPC codes.

The two-stage hybrid soft/hard decoding scheme offers a good trade-off between error performance and decoding complexity. Furthermore, it reduces decoding delay.

5. Performance of Finite Geometry LDPC Codes

To demonstrate the error performance of finite geometry LDPC codes, we select several EG- and PG-LDPC codes of various lengths and decode them with various decoding methods. Figures 1-8 show the error probabilities of these codes.

Figure 1 gives the bit error performance of the type-I two-dimensional (255,175) EG-LDPC code and the type-I two-dimensional (273,191) PG-LDPC code given in Tables 1 and 2, respectively. These two codes are equivalent in terms of geometries based on which they are constructed. They have about the same rate and minimum distance. The EG-LDPC code is decoded with various decoding methods but the PG-LDPC code is only decoded with the SPA decoding. From Figure 1, we see that these two codes have almost the same error performance with the SPA decoding. We also see that the SPA decoding gives the best error performance at the expense of computational complexity. The hard-decision BF decoding achieves relatively good error performance with much less computational complexity. It outperforms the simple one-step MLG decoding by 0.45 dB at the BER of 10^{-5}. With some additional computational complexity, the weighted BF decoding achieves 0.75 dB and 1.20 dB coding gains over the hard-decision BF and MLG decodings at the BER of 10^{-5}, respectively, and it is only 1.2 dB away from the performance of the SPA decoding. It requires much less computational complexity than that of the SPA decoding. Therefore, weighted BF decoding provides a very good trade-off between the error performance of the SPA decoding and the complexity of the simple one-step MLG decoding. Figure 2 gives a comparison of the error performance of the two finite geometry LDPC codes and that of two best computer generated (273,191) Gallager's LDPC codes [10] with γ equals to 3 and 4, respectively. All codes are decoded with the SPA decoding. For the two finite geometry LDPC codes, the maximum number of decoding iterations is set to 50, however for the Gallager's codes, the maximum number of decoding iterations is set to 200. We see that both finite geometry LDPC codes outperform their corresponding computer generated Gallager's codes. The Gallager's code with $\gamma = 3$ also shows an error floor. This indicates that the code has poor minimum distance.

Figure 3 shows the bit error performance of the type-I two-dimensional (1023, 781) EG-LDPC code and the type-I two-dimensional (1057,813) PG-LDPC code given in Tables 1 and 2, respectively. These two codes are equivalent in terms of the code construction geometries and they have about the same rate and minimum distance. Again, the EG-LDPC code is decoded with various decoding methods and the PG-LDPC code is only decoded with the SPA decoding. The two codes perform almost the same with the SPA decoding. At the BER of 10^{-5}, the performance of both codes is only 1.7 dB away from the Shannon limit (with binary-input constraint computed based on the rate of the (1023, 781) code). For codes of length 1000 and rate 0.77, this performance is amazingly good. Again, we see that the weighted BF performs very well and provides a good trade-off between the error performance of the SPA decoding and the decoding complexity of the simple one-step MLG decoding. The block error performance of both codes with the SPA decoding is also shown in Figure 3. They both perform well. Figure 4 gives a comparison of the error performance of the two finite geometry LDPC codes and that of two best computer generated (1057,813) Gallager's LDPC codes with γ equals to 3 and 4, respectively. All codes are decoded with the SPA decoding. We see that the two finite geometry LDPC codes slightly outperform their corresponding Gallager's codes.

The next two codes being evaluated are the type-I two-dimensional (4095,3367) EG-LDPC code and the type-I two-dimensional (4161,3431) PG-LDPC code, the fifth codes given in Tables 1 and 2, respectively. Both codes have rates about 0.83. Their error performances with various types of decoding are shown in Figure 5. With the SPA decoding, they perform 1.5 dB from the Shannon limit at the BER of 10^{-5}.

For $m = s = 3$, the type-I three-dimensional EG-LDPC code $\mathbf{C}_{EG}^{(1)}(3,3)$ is a (511,139) code with minimum distance at least 73. Its parity check matrix $\mathbf{H}_{EG}^{(1)}(3,3)$ is a 4599×511 matrix with row weight $\rho = 8$ and column weight $\gamma = 72$. Then, $\mathbf{H}_{EG}^{(2)}(3,3) = [\mathbf{H}_{EG}^{(1)}(3,3)]^T$ is a 511×4599 matrix with row weight 72 and column weight 8. The null space of $\mathbf{H}_{EG}^{(2)}(3,3)$ gives the type-II three-dimensional EG-LDPC code which is a (4599,4227) code with minimum distance at least 9 and rate

0.919. The type-I code is a low rate code but the type-II code is a high rate code. Both codes have 372 parity check bits. The bit and block error performances of both codes with the SPA decoding are shown in Figure 6. We see that the (4599, 4227) type-II EG-LDPC code performs very well. At BER of 10^{-5}, its performance is only 1 dB away from the Shannon limit.

For $m = 5$ and $s = 2$, the type-II 5-dimensional EG-LDPC code $\mathbf{C}_{EG}^{(2)}(5, 2)$ constructed based on the lines and points of EG$(5, 2^2)$ is an (86955,85963) code with rate 0.9886 and minimum distance at least 5. With the SPA decoding, this code performs only 0.4 dB away from the Shannon limit at the BER of 10^{-5} as shown in Figure 7. Its block error performance is also very good.

In decoding the finite geometry LDPC codes with the SPA decoding, we set the maximum number I_{max} of decoding iterations to 50. Many codes have been simulated. Simulation results of all these codes show that the SPA decoding converges very fast. For example, consider the type-I two-dimensional (4095,3367) EG-LDPC code, the fifth code given in Table 1. Figure 8 shows the convergence of the SPA decoding for this code with $I_{max} = 100$. We see that at BER of 10^{-4}, the performance gap between 5 and 100 iterations is less than 0.2 dB, and the performance between 10 and 100 iterations is less than 0.05 dB. This fast convergence of the SPA decoding for finite geometry LDPC codes is not shared by the computer generated Gallager's codes whose parity check matrices have small column weights, 3 or 4.

To demonstrate the effectiveness of the two-stage hybrid soft/hard decoding scheme for finite geometry LDPC codes, we consider the decoding of the type-I two-dimensional (4095,3367) EG-LDPC code. Figure 8 shows that decoding this code with the SPA, the performance gap between 2 iterations and 100 iterations is about 0.5 dB at the BER of 10^{-5}. Therefore, in two-stage hybrid decoding, we may set the first stage SPA decoding to two iterations and then carry out the second stage with the one-stage MLG decoding. The code is capable of correcting 32 or fewer errors with one-step MLG decoding. Figure 9 shows that the code performs very well with the two-stage hybrid decoding.

The parity check matrix of a type-I finite geometry LDPC code in general has more rows than columns. This is because the number of lines is larger than the number of points in either Euclidean geometry or projective geometry, except for the two-dimensional case. Therefore, the number of rows is larger than the rank of the matrix. In decoding a finite geometry LDPC code with the SPA (or BF decoding), all the rows of its parity check matrix are used for computing check sums to achieve good error performance. If we remove some redundant rows for the parity check matrix, simulation results show that the error performance of the code will be degraded. Therefore, finite geometry LDPC codes in general require more computations than their equivalent computer generated LDPC codes with small row and column weights (often column weight is 3 or 4 and the row weight is 6).

6. Code Construction by Column and Row Splitting of the Parity Check Matrices of Finite Geometry LDPC Codes

A finite geometry (type-I or type-II) LDPC code **C** of length n can be extended by splitting each column of its parity check matrix $\mathbf{H}$ into multiple columns. This results in a new parity matrix with smaller density and hence a new LDPC code. If the column splitting is done properly, very good extended finite geometry LDPC codes can be obtained. Some of the extended finite geometry LDPC codes constructed perform amazingly well with the SPA decoding. They achieve an error performance only a few tenths of a dB away from the Shannon limit. They are the first known algebraically constructed codes approaching the Shannon limit.

Let $\mathbf{g}_0, \mathbf{g}_1, \cdots, \mathbf{g}_{n-1}$ denote the columns of the parity check matrix **H**. First we consider splitting each column of **H** into the same number of columns. All the new columns have the same length as the original column. The weight (or "ones") of the original column is distributed among the new columns. A regular column weight distribution can be done as follows. Let q be a positive integer such that $2 \leq q \leq \gamma$. Dividing γ by q, we have $\gamma = q \times \gamma_{ext} + b$, where $0 \leq b < q$. Split each column $\mathbf{g}_i$ of **H** into q columns $\mathbf{g}_{i,1}, \mathbf{g}_{i,2}, \cdots, \mathbf{g}_{i,q}$ such that the first b columns, $\mathbf{g}_{i,1}, \mathbf{g}_{i,2}, \cdots, \mathbf{g}_{i,b}$, have weight $\gamma_{ext} + 1$ and the next $q - b$ columns, $\mathbf{g}_{i,b+1}, \mathbf{g}_{i,b+2}, \cdots, \mathbf{g}_{i,q}$, have weight γ_{ext}. The distribution of γ "ones" of $\mathbf{g}_i$ into $\mathbf{g}_{i,1}, \mathbf{g}_{i,2}, \cdots, \mathbf{g}_{i,q}$ is carried out in a rotating manner. In the first rotation, the first "1" of $\mathbf{g}_i$ is put in $\mathbf{g}_{i,1}$, the second "1" of $\mathbf{g}_i$ is put in $\mathbf{g}_{i,2}$, and so on. In the second rotation, the $(q+1)$-th "one" of $\mathbf{g}_i$ is put in $\mathbf{g}_{i,1}$, the $(q+2)$-th "one" of $\mathbf{g}_i$ is put in $\mathbf{g}_{i,2}$ and so on. This rotating distribution of the "ones" of $\mathbf{g}_i$ continues until all the "ones" of $\mathbf{g}_i$ have been distributed into the q new columns.

The above column splitting results in a new parity check matrix $\mathbf{H}_{ext}$ with qn columns which has the following structural properties: (1) each row has weight ρ; (2) each column either has weight γ_{ext} or has weight $\gamma_{ext} + 1$; (3) any two columns have at most one "1" in common. If the density of **H** is r, the density of $\mathbf{H}_{ext}$ is then r/q. Therefore, the above column splitting results in a new parity check matrix with smaller density. The null space of $\mathbf{H}_{ext}$ gives an extended finite geometry LDPC code $\mathbf{C}_{ext}$. If γ is not divisible by q, then the columns of $\mathbf{H}_{ext}$ have two different weights, γ_{ext} and $\gamma_{ext} + 1$. Therefore, a code bit of the extended code $\mathbf{C}_{ext}$ is either checked by γ_{ext} check sums or by

$\gamma_{ext} + 1$ check sums. In this case, the extended LDPC code $\mathbf{C}_{ext}$ is an irregular LDPC code.

Example 3. *For $m = 2$ and $s = 6$, the type-I two-dimensional EG-LDPC code $\mathbf{C}_{EG}^{(1)}(2, 6)$ is a (4095,3367) code with minimum distance 65, the fifth code given in Table 1. The parity check matrix of this code has row weight $\rho = 64$ and column weight $\gamma = 64$, respectively. Its error performance is shown in Figure 5. At the BER of 10^{-5}, the required SNR is 1.5 dB away from the Shannon limit. Suppose we split each column of the parity check matrix of this code into 16 columns with rotating column weight distribution. This column splitting results in a (65520,61425) extended type-I EG-LDPC code whose parity check matrix has row weight $\rho = 64$ and column weight $\gamma_{ext} = 4$. The rate of this new code is 0.937. This code decoded with the SPA decoding achieves an error performance which is only 0.42 dB away from the Shannon limit at the BER of 10^{-4} as shown in Figure 10. We see that it has a sharp waterfall error performance. In decoding, the maximum number of decoding iterations is set to 50, but the decoding converges very fast. The performance gap between 10 and 50 iterations is less than 0.1 dB.*

Given a base finite geometry LDPC code **C**, it can be extended into codes of many different lengths. All these extended codes have different rates and behave differently. Consider the type-I two-dimensional (4095,3367) EG-LDPC code discussed in Example 3. Suppose we split each column of its parity check matrix into various numbers of columns from 2 to 23. Table 3 shows the performances of all the extended codes in terms of SNR's required to achieve the BER=10^{-4} and the gaps between the required SNR's and their corresponding Shannon limits. We see that splitting each column of the parity check matrix of the base code into 16 or 17 columns gives the best performance in terms of the Shannon limit gap.

Example 4. *For $m = 2$ and $s = 7$, the type-I two-dimensional EG-LDPC code is a (16383, 14197) code with minimum distance 129, the sixth code in Table 1. The column and row weights of its parity check matrix are both 128. Suppose we split each column of the parity check matrix of this code into 32 columns. We obtain a (524256,507873) extended type-I EG-LDPC code with rate 0.9688. The bit error performances of this extended code and its base code are shown in Figure 11. At the BER of 10^{-5}, the performance of the extended code is 0.3 dB away from the Shannon limit.*

Example 5. *Let $m = s = 3$. The type-I three-dimensional EG-LDPC code constructed based on the lines and points of EG($3, 2^3$) is a (511,139) code with minimum distance at least 73 and rate 0.272. It* *is a low rate code. Its parity check matrix is a* 4599×511 *matrix with row weight* $\rho = 8$ *and column weight* $\gamma = 72$. *Suppose this code is extended by splitting each column of its parity check matrix into 24 columns. Then the extended code is a (12264,7665) LDPC code with rate 0.625. The bit error performances of this extended code and its base code are shown in Figure 12. The error performance of the extended code is only 1.1 dB away from the Shannon limit at the BER of* 10^{-5}.

Given a finite geometry LDPC code specified by a parity check matrix **H**, each column of **H** can be split in different manner and into different numbers of columns. Consequently, many extended finite geometry LDPC codes can be obtained by splitting columns of the parity check matrix **H**. If the columns are split differently, the resultant extended code is an irregular LDPC code.

Column splitting of the parity check matrix of a finite geometry LDPC code may result in an extended code which is neither cyclic nor quasi-cyclic. However, if we arrange the rows of the parity check matrix into circulant submatrices and then split each column into a fixed number of new columns with column weight distributed in a rotating and circular manner, the resultant extended code can be put in quasi-cyclic form. To see this, we consider a type-I EG-LDPC code of length n. Let **H** be the parity check matrix of this code with J rows and n columns. The rows of **H** can be grouped into K $n \times n$ circulant submatrices, $\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_K$, where $K = J/n$. Each circulant submatrix $\mathbf{H}_i$ is obtained by cyclically shifting the incidence vector of a line n times. Therefore, $\mathbf{H}$ can be put in the following form:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_K \end{bmatrix}. \tag{36}$$

Now we split each column of **H** into q columns in a similar manner as that described earlier in this section. However, the 1-component's in a column of **H** must be labeled in a specific circular order. For $0 \leq j < n$, let $\mathbf{g}_j^{(i)}$ be the j-th column of the i-th circulant matrix $\mathbf{H}_i$. Then the j-th column $\mathbf{g}_i$ of $\mathbf{H}$ is obtained by cascading $\mathbf{g}_j^{(1)}, \mathbf{g}_j^{(2)}, \cdots, \mathbf{g}_j^{(K)}$ with one on top the other. We label the 1-component's of the j-th column $\mathbf{g}_j$ of **H** as follows. The first 1-component of $\mathbf{g}_j^{(1)}$ on or below the main diagonal line of circulant $\mathbf{H}_1$ and inside $\mathbf{H}_1$ is labeled as the first 1-component of the j-th column $\mathbf{g}_j$ of **H**. The first 1-component of $\mathbf{g}_j^{(2)}$ on or below the main diagonal line of circulant $\mathbf{H}_2$ and inside $\mathbf{H}_2$ is labeled as the second 1-component of $\mathbf{g}_j$. Continue this labeling process until we label the first 1-component of $\mathbf{g}_j^{(K)}$ on or below the main diagonal line of circular $\mathbf{H}_K$ and inside $\mathbf{H}_K$ as the K-th 1-component of column

$\mathbf{g}_j$. Then we come back to circulant $\mathbf{H}_1$ and start the second round of the labeling progress. The second 1-component of $\mathbf{g}_j^{(1)}$ below the main diagonal line of $\mathbf{H}_1$ and inside $\mathbf{H}_1$ is labeled as the $(K+1)$-th 1-component of $\mathbf{g}_j$. The second 1-component of $\mathbf{g}_j^{(2)}$ below the main diagonal line of circulant $\mathbf{H}_2$ is labeled as the $(K+2)$-th 1-component of $\mathbf{g}_j$. Continue the second round labeling process until we reach to the K-th circulant $\mathbf{H}_K$ again. Then we loop back to circulant $\mathbf{H}_1$ and continue the labeling process. During the labeling process, whenever we reach down to the bottom of a circulant matrix $\mathbf{H}_i$, we wrap around to the top of the same column $\mathbf{g}_j^{(i)}$ of $\mathbf{H}_i$. The above labeling process continues until all the 1-components of $\mathbf{g}_j$ are labeled. Once the labeling of 1-component's of $\mathbf{g}_j$ is completed, we distribute the 1-component's of $\mathbf{g}_j$ into the q new columns in the same rotating manner as described earlier in this section. So the weight of each column of **H** is distributed into new columns in a doubly circular and rotating manner. Clearly the labeling and weight distribution can be carried out at the same time. Let $\mathbf{H}_{ext}$ be the new matrix resulting from the above column splitting. Then $\mathbf{H}_{ext}$ consists of K $n \times nq$ submatrices, $\mathbf{H}_{ext,1}, \mathbf{H}_{ext,2}, \cdots, \mathbf{H}_{ext,K}$. For $0 \leq i < K$, the rows of $\mathbf{H}_{ext,i}$ are cyclic shifts of the first row q bits at a time. As a result, the null space of $\mathbf{H}_{ext}$ gives an extended finite geometry LDPC code in quasi-cyclic form. Type-II EG-LDPC codes can be extended and put in quasi-cyclic form in a similar manner.

For PG-LDPC codes, J may be not be divisible by n. In this case, not all the submatrices of the parity check matrix **H** of a type-I PG-LDPC code can be arranged as $n \times n$ square circulant matrices. Some of them are non-square circulant matrices as shown in Example 2. The rows of such a matrix are still cyclic shifts of the first row and the number of rows divides n. In regular column splitting, the labeling and distribution of 1-components of a column in a non-square circulant submatrix still follow the 45° diagonal and wrap back to the top order. When we reach the last row, move back to the first row and start to move down from the next column. After column splitting, each extended submatrix is still a circulant matrix and the extended code is in quasi-cyclic form. The columns of the parity check matrix of a type-II PG-LDPC code can be split in a similar manner.

The last three examples show that splitting each column of the parity check matrix **H** of a finite geometry LDPC code **C** into multiple columns properly results in an extended LDPC code $\mathbf{C}_{ext}$ which performs very close to the Shannon limit with the SPA decoding. A reason for this is that column splitting reduces the degree of each code bit vertex in the Tanner graph **G** of the base code and hence reduces the number of cycles in the graph. Splitting a column of **H** into q columns results in splitting a code bit vertex of the Tanner graph **G** of the base code into q code bit vertices in the Tanner graph $\mathbf{G}_{ext}$

of the extended code $\mathbf{C}_{ext}$. Each code bit vertex in $\mathbf{G}_{ext}$ is connected to a smaller number of check sum vertices than in $\mathbf{G}$. Figure 13(a) shows that splitting a column in **H** into two columns results in splitting a code bit vertex in the Tanner graph **G** into two code bit vertices in the Tanner graph $\mathbf{G}_{ext}$. The original code bit vertex has a degree of 4 but each code bit after splitting has a degree of 2. This code bit splitting breaks some cycles that exist in the Tanner graph $\mathbf{G}$ of the base code **C**. Figures 14(a) and 15 show the breaking of cycles of lengths 4 and 6. Therefore, column splitting of a base finite geometry LDPC code breaks many cycles of its Tanner graph and results in an extended LDPC code whose Tanner graph has many fewer cycles. This reduction in cycles in the Tanner graph improves the performance of the code with the SPA decoding. In fact, breaking cycles with column splitting of the parity check matrix can be applied to any linear block code. This may result in good LDPC codes.

LDPC codes can also be obtained by splitting each row of the parity check matrix **H** of a base finite geometry LDPC code into multiple rows. The resultant code has the same length as the base code but has a lower code rate. Furthermore, proper row splitting also preserves the cyclic or quasi-cyclic structure of the code. Clearly, LDPC codes can be obtained by splitting both columns and rows of the parity check matrix of a base finite geometry code.

Splitting a row in the **H** matrix is equivalent to splitting a check sum vertex in the Tanner graph of the code and hence reduces the degree of the vertex as shown in Figure 13(b). Therefore, row splitting of the parity check matrix of a base code can also break many cycles in the Tanner graph of the base code. An example of cycle breaking by check sum vertex splitting is shown in Figure 14(b). Clearly a combination of column and row splitting will break many cycles in the Tanner graph of the base code. This may result in a very good LDPC code.

Example 6. *Consider the (255,175) type-I EG-LDPC two-dimensional code given in Table 1. Its performance is shown in Figure 1. The column and row weights of the parity check matrix* **H** *are both 16. If each column of* **H** *is split into 5 columns and each row of* **H** *is split into 2 rows, we obtain a parity check matrix* $\mathbf{H}'$ *whose columns have two weights, 3 and 4, and whose rows have weight 8. The null space of* $\mathbf{H}'$ *gives a (1275,765) LDPC code whose error performance is shown in Figure 16.*

Example 7. *Again we consider the (4095,3367) type-I two-dimensional EG-LDPC code* $\mathbf{C}_{EG}^{(1)}(2,6)$ *given in Table 1. If we split each column of the parity check matrix* **H** *of this code into 16 columns and each row of* **H** *into 3 rows, we obtain a new parity check matrix* $\mathbf{H}'$ *with column weight 4 and row weights 21 and 22. The null space of* $\mathbf{H}'$ *gives a (65520,53235) extended LDPC code. This extended* *code and its base code have about the same rate. Its error performance is shown in Figure 17, and it is 0.7 dB away from the Shannon limit at the BER of* 10^{-5}*. However, the performance of its base code is 1.5 dB away from the Shannon limit. This example shows that by a proper combination of column and row splittings of the parity check matrix of a base finite geometry LDPC code, we may obtain a new LDPC code which has about the same rate but better error performance.*

7. Shortened Finite Geometry LDPC Codes

Both types of finite geometry LDPC codes can be shortened to obtain good LDPC codes. This is achieved by deleting properly selected columns from their parity check matrices. For a type-I code, the columns to be deleted correspond to a properly chosen set of points in the finite geometry based on which the code is constructed. For a type-II code, the columns to be deleted correspond to a properly chosen set of lines in the finite geometry. In this section, several shortening techniques are presented.

First we consider shortening type-I finite geometry LDPC codes. We use a type-I EG-LDPC code to explain the shortening techniques. The same techniques can be used to shorten a type-I PG-LDPC code. Consider the type-I EG-LDPC code $\mathbf{C}_{EG}^{(1)}(m,s)$ constructed based on the m-dimensional Euclidean geometry EG($m,2^s$). Let EG($m-1,2^s$) be an $(m-1)$-dimensional subspace (also called an $(m-1)$-flat) of EG($m,2^s$) [28, 36–38]. If the points in EG($m-1,2^s$) are removed from EG($m,2^s$), we obtain a system **S**, denoted EG($m,2^s$)$\backslash$EG($m-1,2^s$), that contains $2^{ms}-2^{(m-1)s}$ points. Every line (or 1-flat) contained in EG($m-1,2^s$) is deleted from EG($m,2^s$). Every line that is completely outside of EG($m-1,2^s$) remains in **S** and still contains 2^s points. Every line not completely contained in **S** contains only 2^s-1 points, since by deleting an EG($m-1,2^s$) from EG($m,2^s$) we also delete a point in EG($m-1,2^s$) from each such line. The columns of $\mathbf{H}_{EG}^{(1)}(m,s)$ that correspond to the points in the chosen $(m-1)$-flat EG($m-1,2^s$) are deleted, the rows in $\mathbf{H}_{EG}^{(1)}(m,s)$ that correspond to the lines contained in EG($m-1,2^s$) become rows of zeros in the punctured matrix, the rows of $\mathbf{H}_{EG}^{(1)}(m,s)$ that correspond to the lines contained in **S** become rows in the punctured matrix with weight 2^s, and the rows of $\mathbf{H}_{EG}^{(1)}(m,s)$ that correspond to lines not completely contained in **S** become rows in the punctured matrix with weight 2^s-1. Removing the rows of zeros from the punctured matrix, we obtain a new matrix $\mathbf{H}_{EG,S}^{(1)}(m,s)$ that has

$$\frac{2^{(m-1)s}(2^{ms}-1)-2^{(m-2)s}(2^{(m-1)s}-1)}{2^s-1} \tag{37}$$

rows and $2^{ms}-2^{(m-1)s}$ columns. Every column of $\mathbf{H}_{EG,S}^{(1)}(m,s)$ still has weight 2^s, but the rows of $\mathbf{H}_{EG,S}^{(1)}(m,s)$ have two different weights, 2^s-1 and 2^s. The matrix $\mathbf{H}_{EG,S}^{(1)}(m,s)$ still has low density of "ones" and the null space of $\mathbf{H}_{EG,S}^{(1)}(m,s)$ gives a shortened EG-LDPC code whose minimum distance is at least the same as that of the original EG-LDPC code.

Consider the EG-LDPC code constructed based on the two-dimensional Euclidean geometry EG$(2,2^s)$. Its parity check matrix $\mathbf{H}_{EG}^{(1)}(2,s)$ is a $(2^{2s}-1)\times(2^{2s}-1)$ matrix whose rows are the incidence vectors of the lines in EG$(2,2^s)$ that do not pass through the origin. The weight of each column of $\mathbf{H}_{EG}^{(1)}(2,s)$ is $\gamma = 2^s$ and the weight of each row of $\mathbf{H}_{EG}^{(1)}(2,s)$ is $\rho = 2^s$. Let $\mathcal{L}$ be a line in EG$(2,2^s)$ that does not pass through the origin. Delete the columns in $\mathbf{H}_{EG}^{(1)}(2,s)$ that correspond to the 2^s points on $\mathcal{L}$. This results in a matrix $\mathbf{H}'$ with $2^{2s}-2^s-1$ columns. The row in $\mathbf{H}_{EG}^{(1)}(2,s)$ that corresponds to the line $\mathcal{L}$ becomes a row of zeros in $\mathbf{H}'$. Removing this zero row from $\mathbf{H}'$, we obtain a $(2^{2s}-2)\times(2^{2s}-2^s-1)$ matrix $\mathbf{H}_{EG,S}^{(1)}(2,s)$. Each column of $\mathbf{H}_{EG,S}^{(1)}(2,s)$ still has weight $\gamma = 2^s$. Removing a column of $\mathbf{H}_{EG}^{(1)}(2,s)$ that correspond a point **p** on $\mathcal{L}$ will delete a "one" from 2^s-1 rows in $\mathbf{H}_{EG}^{(1)}(2,s)$ which are the incidence vectors of the lines that intersect with line $\mathcal{L}$ at the point **p**. Therefore, there are $2^s(2^s-1)$ rows in $\mathbf{H}_{EG,S}^{(1)}(2,s)$ with weight $\rho_1 = 2^s-1$. There are 2^s-2 lines in EG$(2,2^s)$ not passing through the origin of EG$(2,2^s)$ that are parallel to $\mathcal{L}$. Deleting the columns of the $\mathbf{H}_{EG}^{(1)}(2,s)$ that correspond to the points on $\mathcal{L}$ does not change the weights of the rows that are the incidence vectors of the 2^s-2 lines parallel to $\mathcal{L}$. Therefore, there are 2^s-2 rows in $\mathbf{H}_{EG,S}^{(1)}(2,s)$ with weight $\rho_2 = 2^s$. Any two columns in $\mathbf{H}_{EG,S}^{(1)}(2,s)$ still have at most one "1" in common. The density of $\mathbf{H}_{EG,S}^{(1)}(2,s)$ is $2^s/(2^{2s}-2)$. Therefore, $\mathbf{H}_{EG,S}^{(1)}(2,s)$ is still a low density matrix. The null space of $\mathbf{H}_{EG,S}^{(1)}(2,s)$ is a shortened EG-LDPC code with minimum distance at least 2^s+1.

Example 8. *Consider the type-I two-dimensional (255, 175) EG-LDPC code constructed based on EG(2, 2^4). The code has rate 0.6863. A line in EG(2,2^4) has 16 points. Puncturing this EG-LDPC code based on a line in EG(2,2^4) not passing through the origin results in a (239,160) LDPC code with rate 0.667. Note that the puncturing removes 15 information bits and one parity check bit from the (255,175) EG-LDPC code. Figure 18 shows that the error performance of this punctured code is slightly better than that of the original code.*

Puncturing can also be achieved with combination of removing columns and rows of the low density parity check matrix $\mathbf{H}_{EG}^{(1)}(m,s)$. For example, let **Q** be a set of l lines in EG(m,2^s) not passing through the origin that intersect at a common point α^i, where $1 \le l \le (2^{ms}-1)/(2^s-1)$. Let **P** be the set of lines in EG$(m,2^s)$ that are parallel to the lines in **Q**. Suppose we puncture **H** as follows: (1) remove all the rows in $\mathbf{H}_{EG}^{(1)}(m,s)$ that are the incidence vectors of the lines in **Q** and **P**; and (2) remove the columns that correspond to the points on the lines in **Q**. The total number of distinct points on the lines in **Q** is $l\cdot(2^s-1)+1$. The total number of lines in **Q** and **P** is $l\cdot(2^{(m-1)s}-1)$. Therefore, the puncturing results in a matrix $\mathbf{H}_{EG,S}^{(1)}(m,s)$ with $(2^{(m-1)s}-1)(\frac{2^{ms}-1}{2^s-1}-l)$ rows and $2^{ms}-l(2^s-1)-2$ columns.

Example 9. *Consider puncturing the (255,175) EG-LDPC code. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two lines in EG(2,2^s) not passing through the origin that intersect at the point α^i. There are 28 lines not passing through the origin parallel to either $\mathcal{L}_1$ or $\mathcal{L}_2$. Puncturing the parity check matrix $\mathbf{H}_{EG}^{(1)}(2,4)$ of the (255,175) EG-LDPC code based on $\mathcal{L}_1$, $\mathcal{L}_2$ and their parallel lines results in a 225×224 matrix $\mathbf{H}_{EG,S}^{(1)}(2,4)$. The LDPC code generated by $\mathbf{H}_{EG,S}^{(1)}(2,4)$ is a (224,146) code with minimum distance at least 15. Its error performance is shown in Figure 18.*

Clearly, shortening of a type-I finite geometry LDPC code can be achieved by deleting columns from its parity check matrix **H** that correspond to the points in a set of q parallel $(m-1)$-flats. Zero rows resulting from the column deletion are removed. This results in a shortened LDPC code of length of $2^{ms}-q2^{(m-1)s}$ or $2^{ms}-q2^{(m-1)s}-1$ depending whether the $(m-1)$-flat that contains the origin is included in the deletion.

To shorten a type-II m-dimensional EG-LDPC code, we first put its parity check matrix $\mathbf{H}_{EG}^{(2)}(m,s)$ in circulant form,

$$\mathbf{H}_{EG}^{(2)}(m,s)=[\mathbf{H}_1,\mathbf{H}_2,\cdots,\mathbf{H}_K], \tag{38}$$

where $K=(2^{(m-1)s}-1)/(2^s-1)$ and $\mathbf{H}_i$ is a $(2^{ms}-1)\times(2^{ms}-1)$ circulant matrix whose columns are cyclic shifts of the incidence vector of a line. For any integer l with $0<l<K$, we select l circulant submatrices from $\mathbf{H}_{EG}^{(2)}(m,s)$ and delete them. This deletion results in a new matrix $\mathbf{H}_{EG,S}^{(2)}(m,s)$ with $2^{ms}-1$ rows and $(K-l)(2^{ms}-1)$ columns. The column and row weights of this matrix are 2^s and $(K-l)2^s$. Its null space gives a shortened type-II EG-LDPC code which is still quasi-cyclic. This shortened code has minimum distance at least 2^s+1. A type-II PG-LDPC code can be shortened in the same manner.

Example 10. *For $m=s=3$, the type-II EG-LDPC code constructed based on EG($3,2^3$) is a (4599,4227) code with minimum distance 9 whose error performance is shown in Figure 6 (this code was discussed in Section 5). The parity check matrix $\mathbf{H}_{EG}^{(2)}(3,3)$ of this code is a 511×4599 matrix. In circulant form, this matrix consists of nine 511×511 circulant submatrices. Suppose we delete one circulant submatrix (any one) from this matrix. The null space of the resultant shortened matrix gives* *a (4088, 3716) LDPC code with minimum distance at least 9 and rate 0.909. The error performance of this shortened code is shown in Figure 19. At the BER of 10^{-5}, its error performance is 1.1 dB away from the Shannon limit. If we remove any 3 circulant submatrices from $\mathbf{H}_{EG}^{(2)}(3,3)$, we obtain a (3066,2694) LDPC code with rate 0.878. Its error performance is also shown in Figure 19. If we delete any six circulant submatrices from $\mathbf{H}_{EG}^{(2)}(3,3)$, we obtain a (1533,1161) LDPC code with rate 0.757. Its error performance is 1.9 dB away from the Shannon limit at the BER of 10^{-5}. For comparison, the error performance of the original (4599,4227) base code is also included in Figure 19.*

8. A Marriage of LDPC Codes and Turbo Codes

Turbo codes with properly designed interleaver achieve an error performance very close to the Shannon limit [23–26]. These codes perform extremely well for BER's above 10^{-4} (waterfall performance), however they have a significant weakened performance at BER's below 10^{-5} due to the fact that the component codes have relatively poor minimum distances, which manifests itself at very low BER's. The fact that these codes do not have large minimum distances causes the BER curve to flatten out at BER's below 10^{-5}. This phenomenon is known as error floor. Because of the error floor, turbo codes are not suitable for applications requiring extremely low BER's, such as some scientific or command and control applications. Furthermore in turbo decoding, only information bits are decoded and they can not be used for error detection. The poor minimum distance and lack of error detection capability make these codes perform badly in terms of block error probability. Poor block error performance also makes these codes not suitable for many communication applications. On the contrary, finite geometry LDPC codes do not have all the above disadvantages of turbo codes, except that they may not perform as well as the turbo codes for BER's above 10^{-4}.

The advantage of extremely good error performance of turbo codes for BER's above 10^{-4} and the advantages of finite geometry LDPC codes such as no error floor, possessing error detection capability after decoding and good block error performance, can be combined to form a coding system that performs well for all ranges of SNR's. One such system is the concatenation of a turbo inner code and a finite geometry LDPC outer code. To illustrate this, we form a turbo code that uses the (64,57) distance-4 Hamming code as the two component codes. The bit and block error performances of this turbo code are shown in Figure 20 from which we see the error floor and poor block error performance. Suppose this turbo code is used as the inner code in concatenation with the extended (65520,61425) EG-LDPC code given in Example 3 as the outer code. The overall rate of this concatenated LDPC-turbo system is 0.75. It achieves both good waterfall bit and block error performances as shown in Figure 20. At the BER of 10^{-5}, its performance is 0.7 dB away from the Shannon limit. This concatenated system performs better than a concatenated system in which a Reed-Solomon (RS) code, say the NASA standard (255,223) RS code over GF(2^8), is used as the outer code and decoded algebraically or decoded based on a reliability based decoding algorithm.

Another form of the marriage of turbo coding and a finite geometry code is to use finite geometry codes as component codes in a turbo coding setup.

9. Conclusion and Suggestions for Further Work

In this paper, a geometric approach to the construction of LDPC codes has been presented. Four classes of LDPC codes have been constructed based on the lines and points of the well known Euclidean and projective geometries over finite fields. These codes have been shown to have relatively good minimum distances and their Tanner graphs have girth 6. They can be decoded with various decoding methods, ranging from low to high decoding complexity, from reasonably good to very good error performance. A very important property of these four classes of finite geometry LDPC codes is that they are either cyclic or quasi-cyclic. Encoding of cyclic and quasi-cyclic codes is a linear time process and can be achieved with simple feedback shift registers. This linear time encoding is very important in practice. This advantage is not shared by other LDPC codes in general, especially the randomly computer generated LDPC codes and irregular LDPC codes.

The finite geometry LDPC codes can be extended or shortened in various ways to form many other good LDPC codes of various lengths and rates. Extension by column splitting of the parity check matrix of a finite geometry LDPC code is a powerful method to construct long powerful LDPC codes. Some long extended finite geometry LDPC codes have been constructed and they achieve a performance that is only a few tenths of a dB away from the Shannon limit. Techniques for column splitting and deletion have been proposed so that both the extended and shortened finite geometry LDPC codes can be put in quasi-cyclic form.

In this paper, it has been shown that finite geometry is a powerful tool for constructing good LDPC codes. Finite geometry is a branch in combinatorial mathematics, there are other important branches in combinatorial mathematics which may also be useful in constructing LDPC codes. One such branch is balanced incomplete block design (BIBD) [37, 38, 52, 53]. Let $\mathbf{X} = \{x_1, x_2, \cdots, x_n\}$ be a set of n objects. A BIBD of $\mathbf{X}$ is a collection of b ρ-subsets of $\mathbf{X}$, denoted by $B_1, B_2, \cdots, B_b$ and called the blocks, such that the following conditions are satisfied: (1) each object appears in exactly γ of the b blocks; and (2) every two objects appear simultaneously in exactly λ of the blocks. Such a BIBD can be described by its incidence matrix **Q**, which is a $b \times n$ matrix with 0's and 1's as entries. The columns and rows of the matrix **Q** correspond to the objects and the blocks of $\mathbf{X}$, respectively. The entry at the i-th row and j-th column of **Q** is "1" if the object x_j is contained in the block B_i and is 0 otherwise. If $\lambda = 1$ and both γ and ρ are small, then **Q** and its transpose $\mathbf{Q}^T$ are sparse matrices and they can be used as the parity check matrices to generate LDPC codes whose Tanner graphs does not contain cycles of length 4. Over the years, many such BIBD's have been constructed. For example, for any positive integer t such that $4t+1$ is a power of a prime, there exists a BIBD with $n = 20t+5, b = (5t+1)(4t+1), \gamma = 5t+1, \rho = 5$ and $\lambda = 1$. The set of integers t for which $4t+1$ is a power of a prime is $\{1, 2, 3, 4, 6, 7, 9, 10, 12, 13, 15, 18, 20, \cdots\}$ which is infinite. For this class of BIBD's, the incidence matrix **Q** is a $(5t+1)(4t+1) \times (20t+5)$ matrix with density $5/(20t+5)$, a sparse matrix. Then **Q** and $\mathbf{Q}^T$ generate two LDPC codes. Of course, column and row splitting techniques can be applied to **Q** and $\mathbf{Q}^T$ to generate other LDPC codes. The above construction based on BIBD's may yield good LDPC codes. In fact, one such code of length 1044 and dimension 899 has been constructed, which performs very well, 2 dB away from the Shannon limit. This construction approach should be a direction for further research.

Acknowledgements

The authors wish to thank the referees for their valuable and constructive comments that improved the presentation of this paper.

REFERENCES

[1] R. G. Gallager,“Low Density Parity Check Codes,” *IRE Transactions on Information Theory*, IT-8, pp. 21-28, January 1962.

[2] R. G. Gallager, *Low Density Parity Check Codes*, MIT Press, Cambridge, Mass., 1963.

[3] R. M. Tanner,“A Recursive Approach to Low Complexity Codes,”*IEEE Transactions on Information Theory*, IT-27, pp. 533-547, September 1981.

[4] D. J. C. MacKay and R. M. Neal,“Near Shannon Limit Performance of Low Density Parity Check Codes,” *Electronics Letters* 32 (18): 1645-1646, 1996.

[5] M. Sipser and D. Spielman, “Expander Codes,” *IEEE Transactions on Information Theory*, Vol. 42, No. 6, pp. 1710-1722, November 1996.

[6] D. Spielman, “Linear-time Encodable Error-correcting Codes,” *IEEE Transactions on Information Theory*, Vol. 42, No. 6, pp. 1723-1731, November 1996.

[7] M. C. Davey and D. J. C. MacKay,“Low Density Parity Check Codes over GF(q),” *IEEE Communications Letters*, June 1998.

[8] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, “Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation,” *Proceedings of 1998 IEEE International Symposium on Information Theory*, pp. 171, Cambridge, Mass., August 16-21, 1998.

[9] D. J. C. MacKay, “Gallager Codes that are Better Than Turbo Codes,” *Proc. 36th Allerton Conf. Communication, Control, and Computing,* Monticello, IL., September 1998.

[10] D. J. C. MacKay,“Good Error-Correcting Codes Based on Very Sparse Matrices,” *IEEE Transactions on Information Theory*, IT-45, pp.399-432, March 1999.

[11] T. Richardson, A. Shokrollahi, and R. Urbanke, “Design of Capacity-Approaching Irregular Codes,” *IEEE Transactions on Information Theory*, Vol. 47, No.2, pp.619-637, February 2001.

[12] T. Richardson and R. Urbanke, “The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding,” *IEEE Transactions on Information Theory,* Vol.47, pp. 599-618, February 2001.

[13] Y. Kou, S. Lin and M. Fossorier, “Low Density Parity Check Codes based on Finite Geometries: A Rediscovery,” *Proc. IEEE International Symposium on Information Theory,* Sorrento, Italy, June 25-30, 2000.

[14] ————, “Construction of Low Density Parity Check Codes: A Geometric Approach,” *Proc. 2nd Int. Symp. on Turbo Codes and Related Topics*, pp. 137-140, Brest, France, Sept. 4-7, 2000.

[15] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Morgan Kaufmann, San Mateo, 1988.

[16] S. L. Lauritzen and D. J. Spiegelhalter, “Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems,” *Journal of the Royal Statistical Society B,* 50, pp 157-224, 1988.

[17] N. Wiberg, H. -A. Loeliger, and R. Kötter, “Codes and Iterative Decoding on General Graphs,” *European Transactions on Telecommunications,* Vol. 6, pp. 513-526, 1955.

[18] R. J. McEliece, D. J. C. MacKay, and J. -F. Cheng,"Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm," *IEEE Journal on Selected Areas*, Vol. 16, pp. 140-152, February 1998.

[19] F. R. Kschischang and B. J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in General Models," *IEEE Jour. Selected Areas in Communications,* Vol. 16, No. 12, pp. 219-230, February 1998.

[20] F. R. Kschischang, B. J. Frey and H. -A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, Vol. 47, pp. 498-519, February 2001.

[21] M. Fossorier, M. Mihaljevic, and H. Imai,"Reduced Complexity Iterative Decoding of Low Density Parity Check Codes," *IEEE Transactions on Communications*, Vol. 47, pp. 673-680, May 1999.

[22] R. Lucas, M. Fossorier, Y. Kou, and S. Lin,"Iterative Decoding of One-Step Majority Logic Decodable Codes Based on Belief Propagation," *IEEE Transactions on Communications*, Vol. 48, pp. 931-937, June 2000.

[23] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," *Proc. 1993 IEEE International Conference on Communications,* Geneva, Switzerland, pp. 1064-1070, May 1993.

[24] C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes," *IEEE Transactions on Communications*, Vol. 44, pp. 1261-1271, October 1996.

[25] S. Benedetto and G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Transactions on Information Theory*, Vol. 42. No. 2, pp. 409-428, March 1996.

[26] J. Hagenauer, E. Offer and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, Vol. 42, pp. 429-445, March 1996.

[27] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed., MIT Press, Cambridge, MA., 1972.

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

[29] R. C. Bose and D. J. Ray-Chaudhuri, "On a Class of Error Correcting Binary Group Codes," *Information and Control*, No. 3, pp 68-79, March 1960.

[30] E. R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, NY, 1968.

[31] J. L. Massey, *Threshold Decoding*, MIT Press, Cambridge, Mass. 1963.

[32] N. Deo, *Graph Theory with Applications to Engineering and Computer Science,* Prentice Hall, Englewood Cliffs, NJ, 1974.

[33] N. Wiberg, "Codes and Decoding on General Graphs," *Ph.D Dissertation*, Department of Electrical Engineering, University of Linköping, Linköping, Sweden, April 1996

[34] T. Etzion, A. Trachtenberg, and A. Vardy, "Which Codes Have Cycle-Free Tanner Graphs," *IEEE Transactions on Information Theory*, Vol. 45. No. 6, pp. 2173-2181, September 1999.

[35] G. D. Forney, Jr., "Codes on Graphs: Normal Realizations," IEEE Transactions on Information Theory, Vol. 47, pp.520-548, February 2001.

[36] R. D. Carmichael, *Introduction to the Theory of Groups of Finite Order*, Dover Publications, Inc., New York, NY., 1956.

[37] A. P. Street and D. J. Street, *Combinatorics of Experimental Design*, Oxford Science Publications, Clarendon Press, Oxford, 1987.

[38] H. B. Mann, *Analysis and Design of Experiments*, Dover Publications, New York, NY, 1949.

[39] E. J. Weldon, Jr., "Euclidean Geometry Cyclic Codes," *Proc. Symp. Combinatorial Math.*, University of North Carolina, Chapel Hill, N.C., April 1967.

[40] T. Kasami, S. Lin, and W. W. Peterson, "Polynomial Codes," *IEEE Transactions on Information Theory*, Vol. 14, pp. 807-814, 1968.

[41] S. Lin, "On a Class of Cyclic Codes," *Error Correcting Codes*, (Edited by H. B. Mann), John Wiley & Sons, Inc., New York, 1968.

[42] —–, "On the Number of Information Symbols in Polynomial Codes," *IEEE Transactions on Information Theory*, IT-18, pp.785-794, November 1972.

[43] T. Kasami and S. Lin, "On Majority-Logic Decoding for Duals of Primitive Polynomial Codes," *IEEE Transactions on Information Theory*, IT-17(3), pp. 322-331, May 1971.

[44] E. J. Weldon, Jr., "New Generations of the Reed-Muller Codes, Part II: Non-primitive Codes," *IEEE Transactions on Information Theory*, IT-14, pp. 199-205, May 1968.

[45] L. D. Rudolph, "A Class of Majority Logic Decodable Codes," *IEEE Transactions on Information Theory*, IT-13, pp. 305-307, April 1967.

[46] E. J. Weldon, Jr.,"Difference-Set Cyclic Codes," *Bell System Technical Journal*, 45, pp. 1045-1055, September 1966.

[47] F. L. Graham and J. MacWilliams, "On the Number of Parity Checks in Difference-Set Cyclic Codes," *Bell System Technical Journal*, 45, pp. 1056-1070, September 1966.

[48] I. S. Reed, "A Class of Multiple-Error-Correcting Codes and the Decoding Scheme," *IRE Trans.*, IT-4, pp. 38-49, September 1954.

[49] V. D. Kolesnik, "Probability Decoding of Majority Codes," *Prob. Peredachi Inform.*, Vol. 7, pp. 3-12, July 1971.

[50] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, Vol. 20, pp. 284-287, March 1974.

[51] R. Lucas, M. Bossert, and M. Breitbach, "On Iterative Soft-Decision Decoding of Linear Block Codes and Product Codes," *IEEE Journal on Selected Areas in Communications*, Vol. 16, pp. 276-296, February 1998.

[52] C.J. Colbourn and J.H. Dinitz, *The CRC Handbook of Combinatorial Designs*, CRC Press, Inc., New York, 1996.

[53] H.J. Ryser, *Combinatorial Mathematics*, John Wiley, 1963

Table 1: A list of type-I two-dimensional EG-LDPC codes

s	n	k	d_{min}	ρ	γ
2	15	7	5	4	4
3	63	37	9	8	8
4	255	175	17	16	16
5	1023	781	33	32	32
6	4095	3367	65	64	64
7	16383	14197	129	128	128

Table 2: A list of type-I two-dimensional PG-LDPC codes

s	n	k	d_{min}	ρ	γ
2	21	11	6	5	5
3	73	45	10	9	9
4	273	191	18	17	17
5	1057	813	34	33	33
6	4161	3431	66	65	65
7	16513	14326	130	129	129

Table 3: Extended codes constructed from the type-I two-dimensional (4095,3367) EG-LDPC code

q	n	k	ρ	γ	rate	Shannon Limit(dB)	Drop Point Crossing $BER = 10^{-4}$(dB)	The Gap Between Drop Point and Shannon Limit
1	4095	3367	64	64	0.822	2.25	3.7	1.45
2	8190	4095	64	32	0.5	0.18	6	5.82
3	12285	8190	64	21 or 22	0.666	1.06	4.9	3.84
4	16380	12285	64	16	0.75	1.63	4.5	2.87
5	20475	16380	64	12 or 13	0.8	2.04	4.35	2.31
6	24570	20475	64	10 or 11	0.833	2.36	4.25	1.89
7	28665	24570	64	9 or 10	0.857	2.62	4.22	1.63
8	32760	28665	64	8	0.875	2.84	4.1	1.26
9	36855	32760	64	7 or 8	0.888	3.05	4.11	1.11
10	40950	36855	64	6 or 7	0.9	3.2	4.12	0.92
11	45045	40950	64	5 or 6	0.909	3.34	4.16	0.82
12	49140	45045	64	5 or 6	0.916	3.47	4.2	0.73
13	53235	49140	64	4 or 5	0.923	3.59	4.23	0.64
14	57330	53235	64	4 or 5	0.928	3.7	4.27	0.57
15	61425	57330	64	4 or 5	0.933	3.8	4.32	0.52
16	65520	61425	64	4	0.937	3.91	4.33	0.42
17	69615	65520	64	3 or 4	0.941	3.98	4.4	0.4
18	73710	69615	64	3 or 4	0.944	4.05	4.5	0.45
19	77805	73710	64	3 or 4	0.947	4.1	4.54	0.44
20	81900	77805	64	3 or 4	0.95	4.2	4.65	0.45
21	85995	81900	64	3 or 4	0.952	4.26	4.7	0.44
22	90090	85995	64	2 or 3	0.954	4.3	4.79	0.49
23	94185	90090	64	2 or 3	0.956	4.38	4.9	0.52

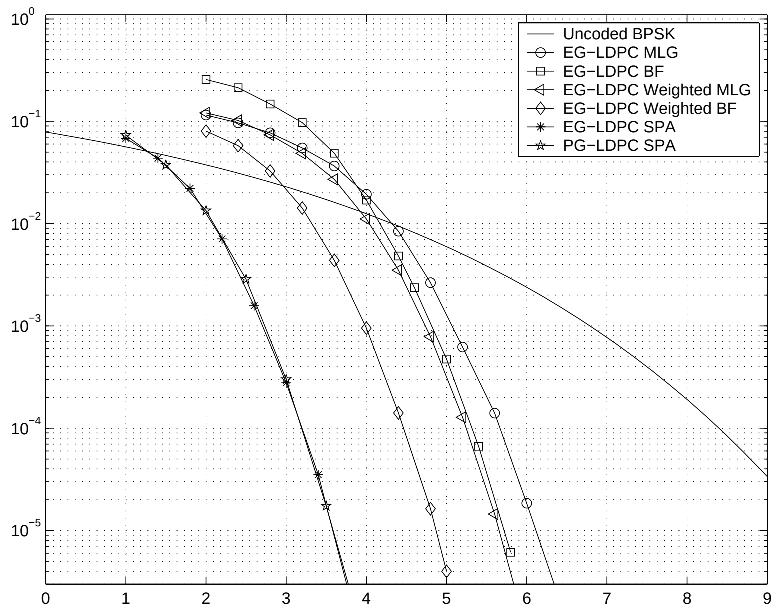


Figure 1: Bit-error probabilities of the type-I two-dimensional (255, 175) EG-LDPC code and (273,191) PG-LDPC code based on different decoding algorithms.

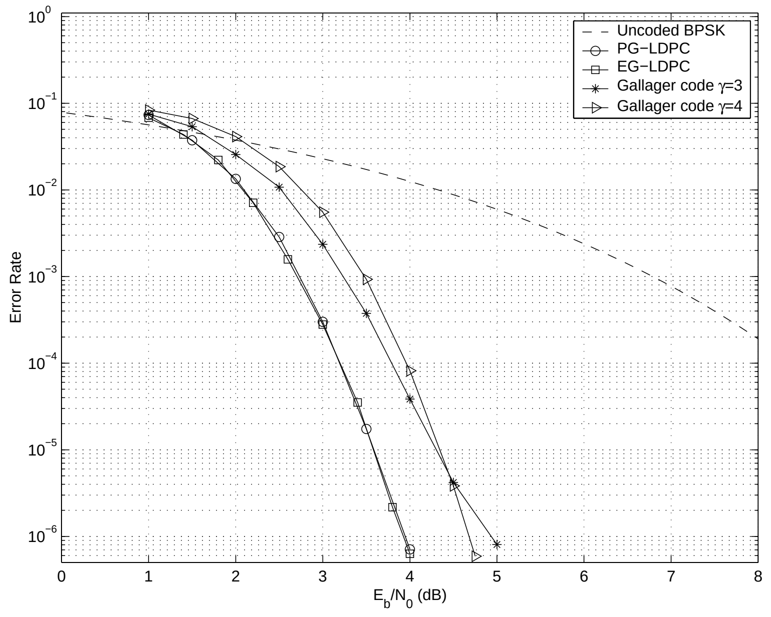


Figure 2: Bit-error probabilities of the (255, 175) EG-LDPC code, (273,191) PG-LDPC code and two computer generated (273,191) Gallager codes with the SPA decoding.

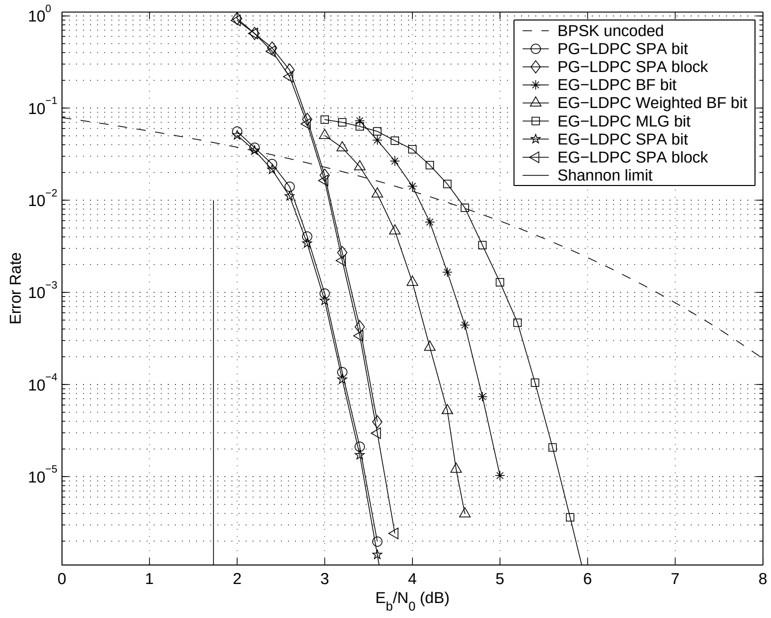


Figure 3: Bit- and block-error probabilities of the type-I two-dimensional (1023, 781) EG-LDPC code and (1057, 813) PG-LDPC code based on different decoding algorithms.

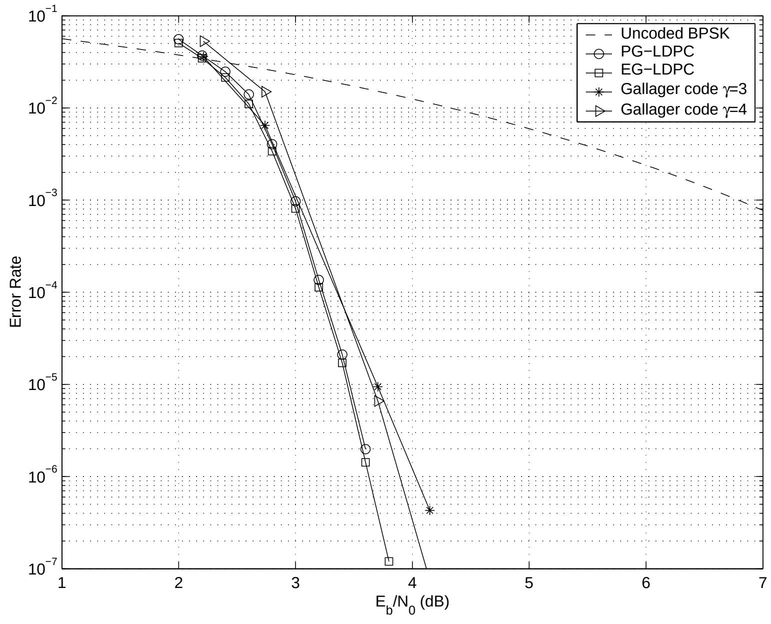


Figure 4: Bit-error probabilities of the (1023, 781) EG-LDPC code, (1057, 813) PG-LDPC code and two computer generated (1057, 813) Gallager codes with the SPA decoding.

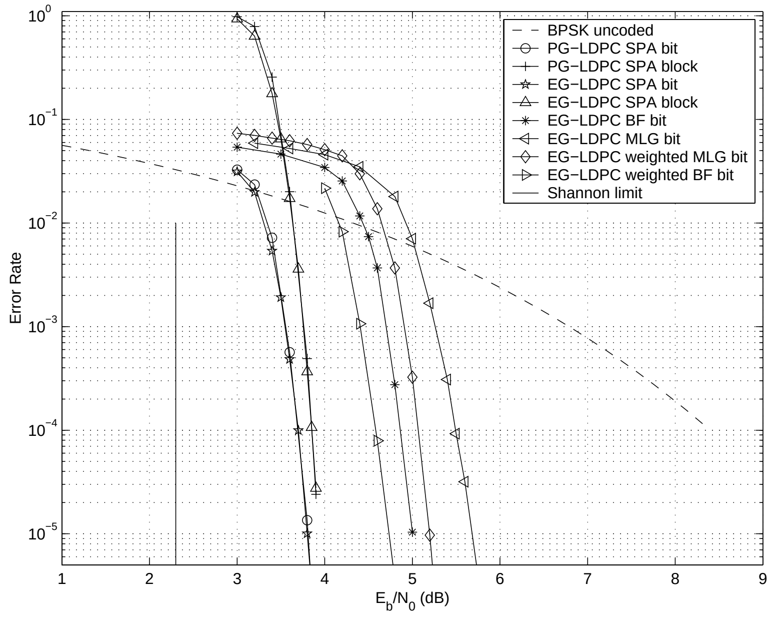


Figure 5: Bit- and block-error probabilities of the type-I two-dimensional (4095, 3367) EG-LDPC code and (4161, 3431) PG-LDPC code based on different decoding algorithms.

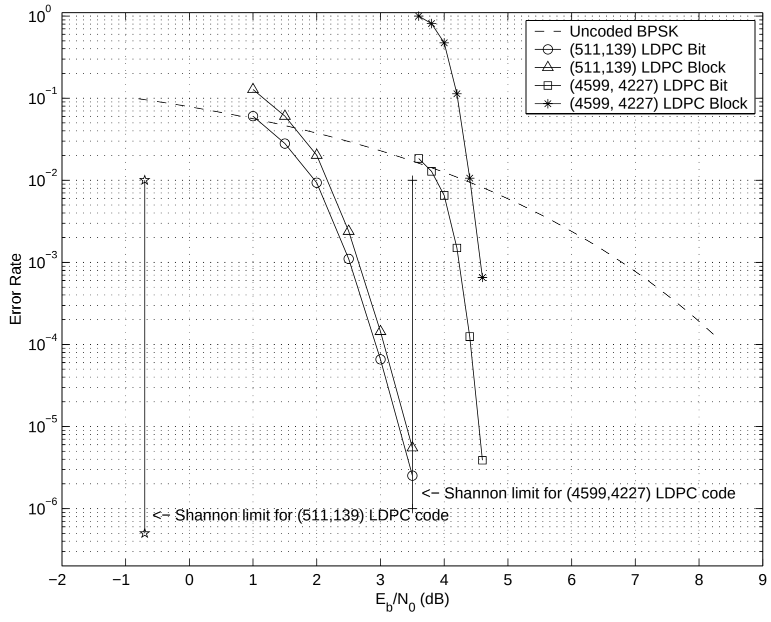


Figure 6: Error performances of the type-I three-dimensional (511, 139) EG-LDPC code and the type-II three-dimensional (4599, 4227) EG-LDPC code with the SPA decoding.

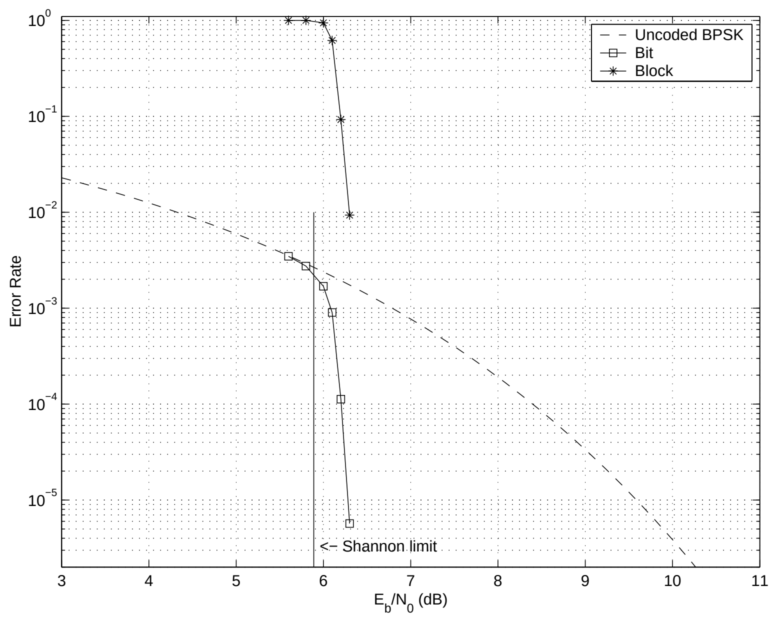


Figure 7: Error performance of the type-II five-dimensional (86955, 85963) EG-LDPC code with the SPA decoding.

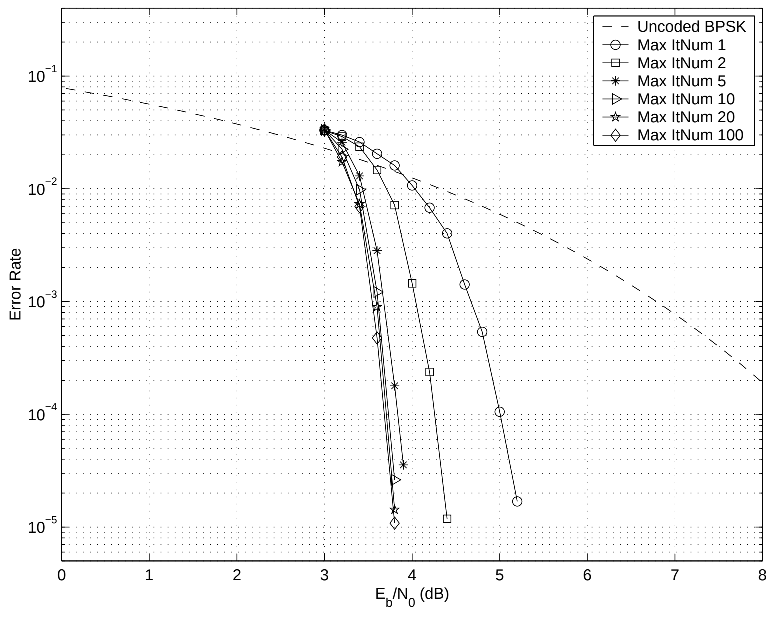


Figure 8: Convergence of the SPA decoding for the type-I two-dimensional (4095,3367) EG-LDPC code.

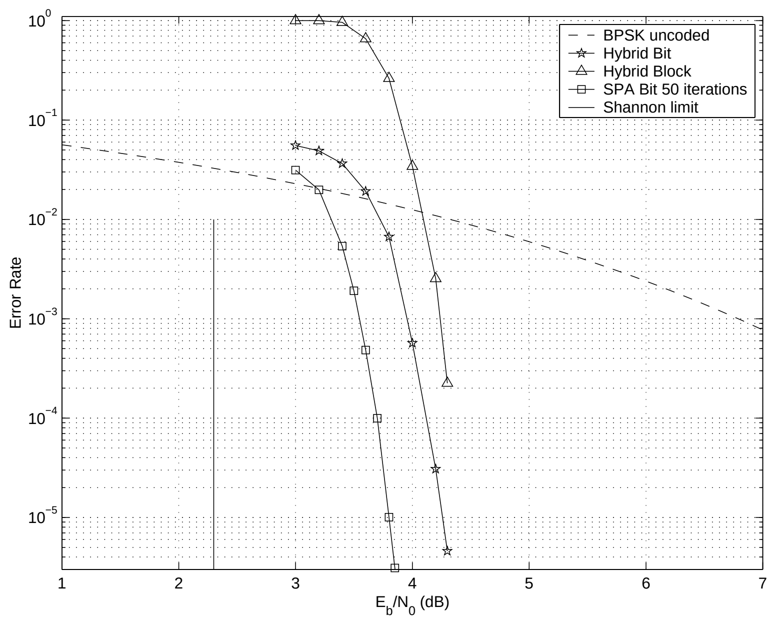


Figure 9: Bit error probabilities of the type-I two-dimensional (4095, 3367) EG-LDPC code based on two-stage hybrid decoding.

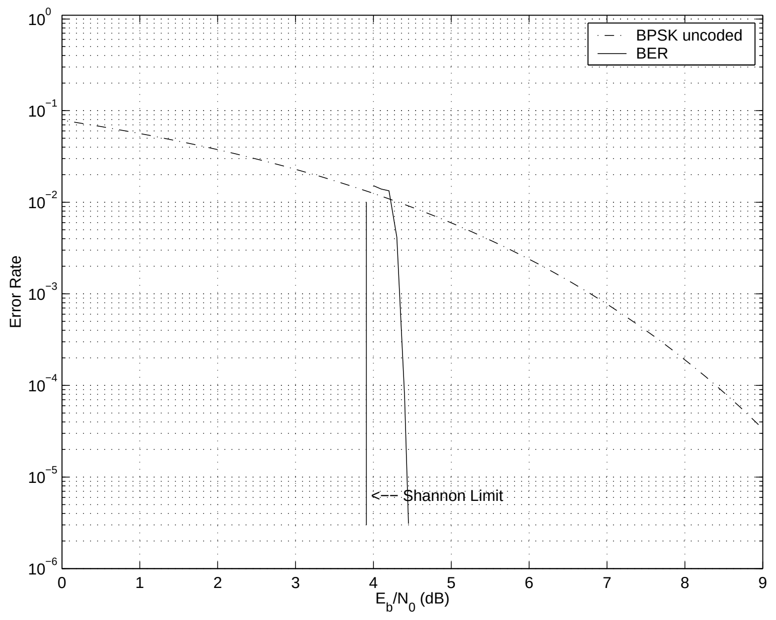


Figure 10: Bit- and block-error probabilities of the extended (65520,61425) EG-LDPC code with the SPA decoding.

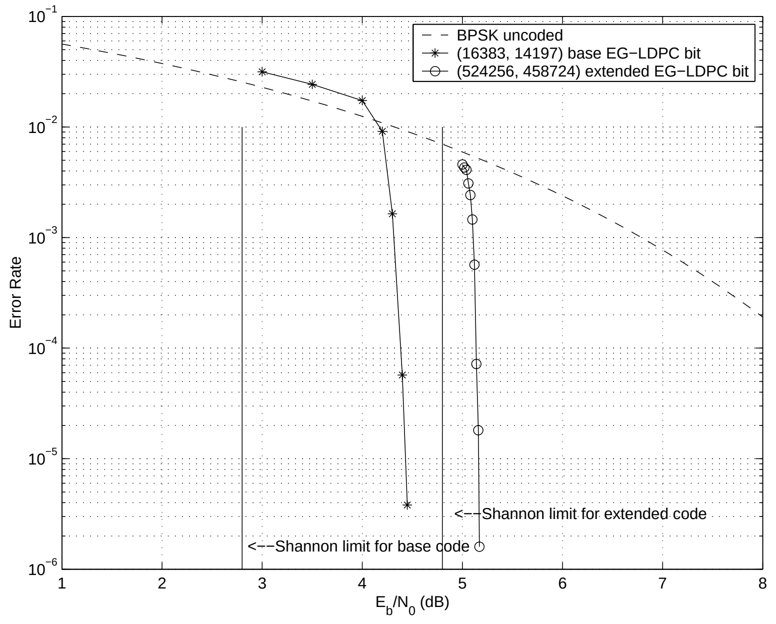


Figure 11: Error performances of the type-I two-dimensional (16383,14197) EG-LDPC code and the extended (524256,507873) EG-LDPC code with the SPA decoding.

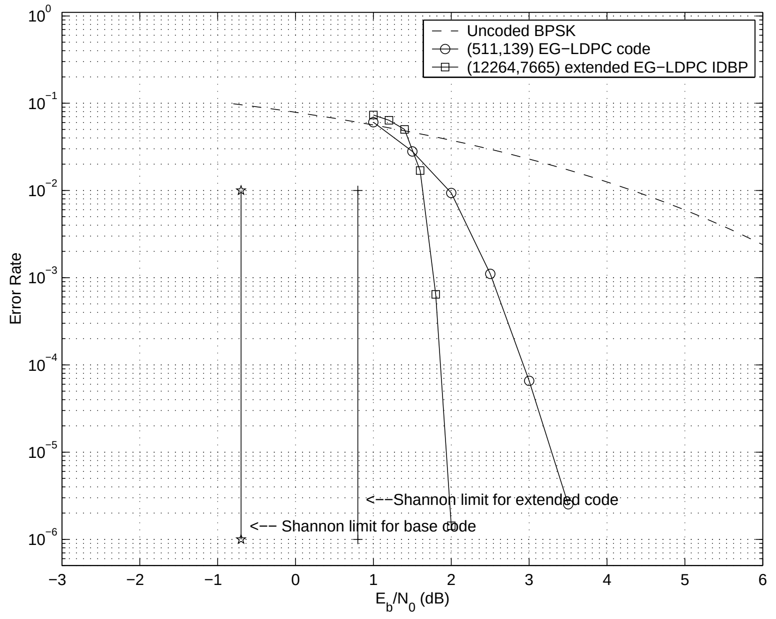
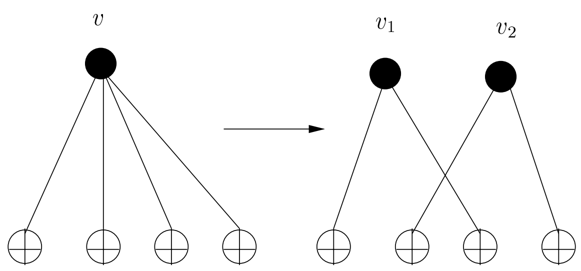
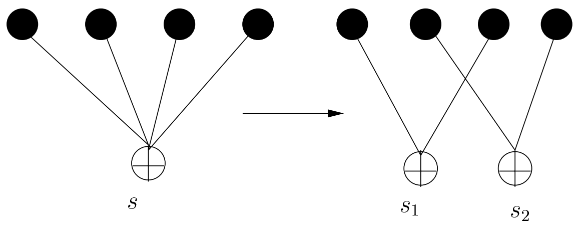


Figure 12: Error Performance of the type-I three-dimensional (511,139) EG-LDPC code and the extended (12264,7665) EG-LDPC code with the SPA decoding.

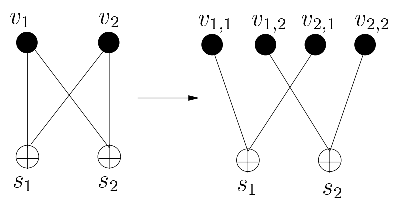


(a) Column splitting

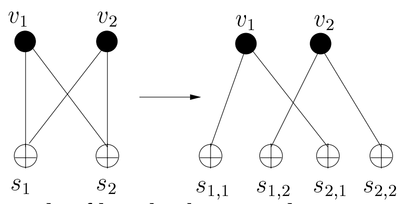


(b) Row splitting

Figure 13: Graph decomposition by column/row splitting.



(a) Breaking a cycle of length 4 by column splitting operation.



(b) Breaking a cycle of length 4 by row splitting operation.

Figure 14: Cycle decomposition.

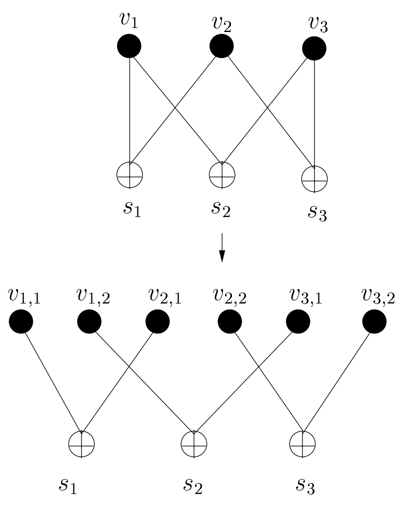


Figure 15: Decomposition of a cycle of length 6 by column splitting.

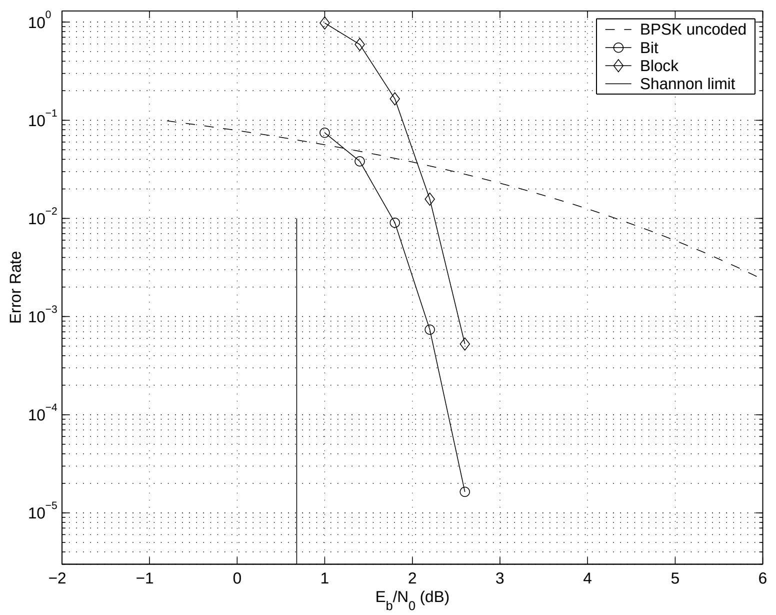


Figure 16: Bit- and block-error probabilities of the extended (1275,765) LDPC code with the SPA decoding.

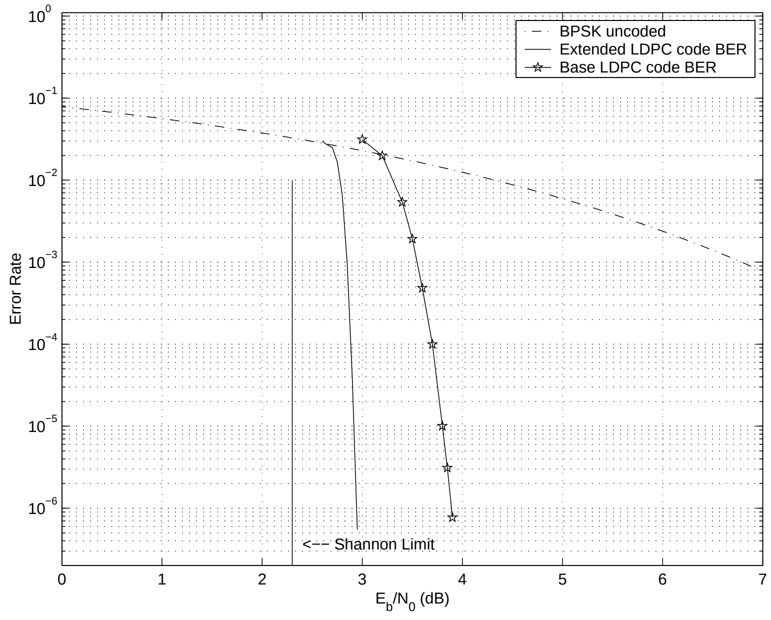


Figure 17: Bit-error probabilities of the extended (65520,53235) EG-LDPC code and the type-I two-dimensional (4095, 3367) EG-LDPC code with the SPA decoding.

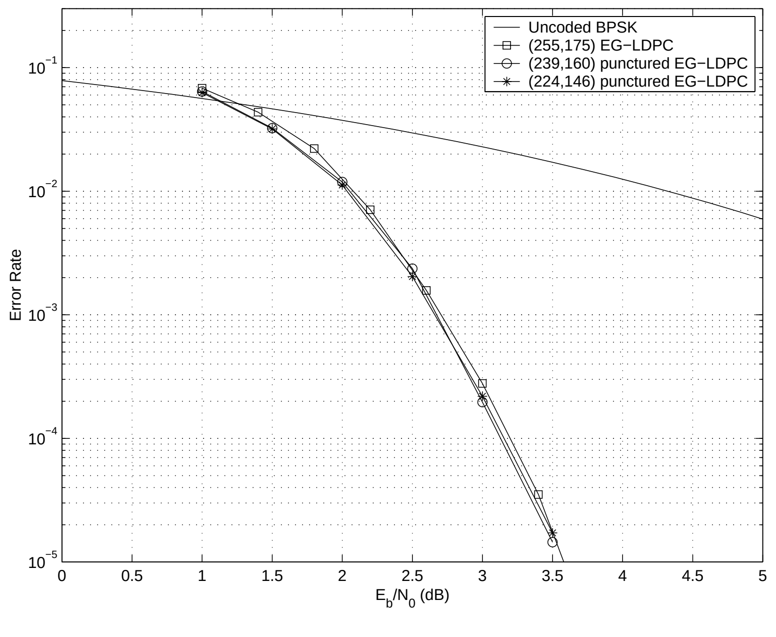


Figure 18: Bit-error probabilities of the (255,175) EG-LDPC code, the (239,160) and (224, 146) shortened EG-LDPC codes with the SPA decoding.

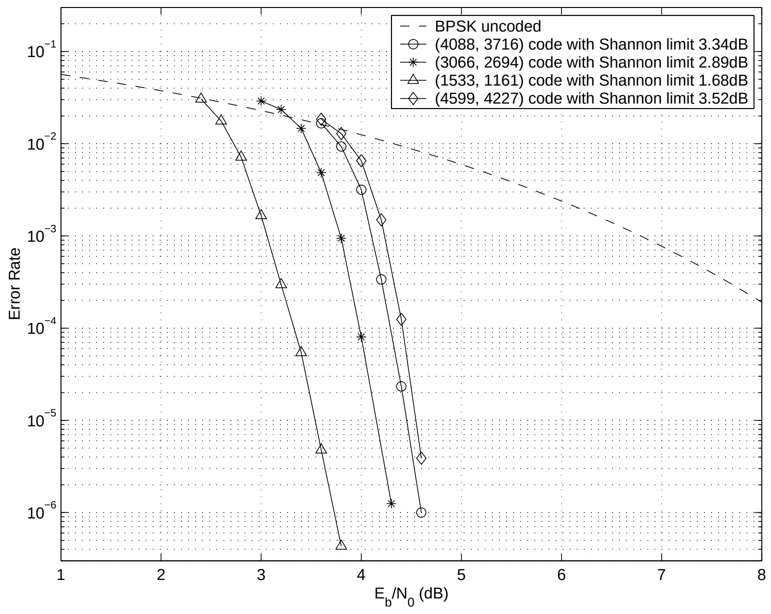


Figure 19: Bit-error probabilities of the (4088,3716), (3066,2694) and (1533, 1161) shortened EG-LDPC codes and the type-II 3-dimensional EG-LDPC code with the SPA decoding.

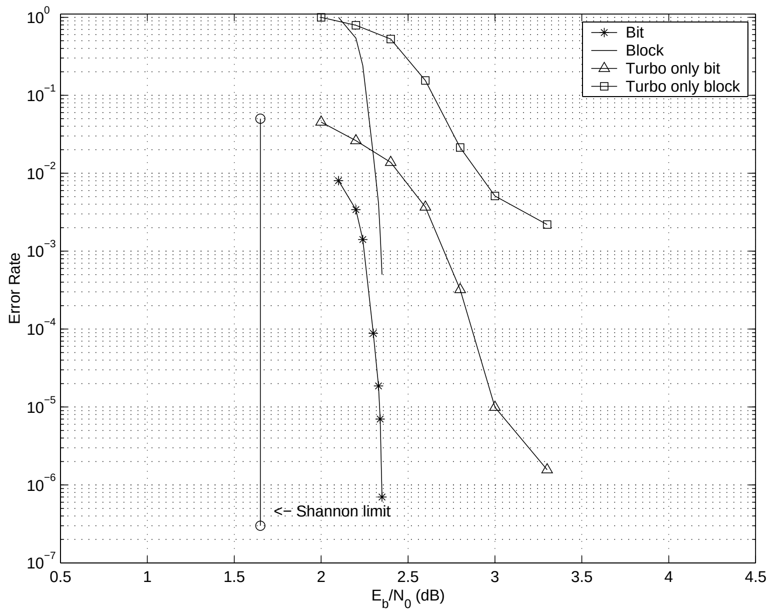


Figure 20: Bit and block error performance of a concatenated LDPC-turbo coding system with a turbo inner code and an extended EG-LDPC outer code.

List of Tables

List of Figures