An Introduction to LDPC Codes*

William E. Ryan
Department of Electrical and Computer Engineering
The University of Arizona
Box 210104
Tucson, AZ 85721

August 19, 2003

1. Introduction

Low-density parity-check (LDPC) codes are a class of linear block codes which provide near-capacity performance on a large collection of data transmission and storage channels while simultaneously admitting implementable decoders. LDPC codes were first proposed by Gallager in his 1960 doctoral dissertation [1] and was scarcely considered in the 35 years that followed. One notable exception is the important work of Tanner in 1981 [2] in which Tanner generalized LDPC codes and introduced a graphical representation of LDPC codes, now called Tanner graphs. The study of LDPC codes was resurrected in the mid-1990's with the work of MacKay, Luby, and others [3], [4], [5] who noticed, apparently independently of the work of Gallager, the advantages of linear block codes which possess sparse (low-density) parity-check matrices.

This tutorial chapter provides the foundations for the study and practice of LDPC codes. We will start with the fundamental representations of LDPC codes via parity-check matrices and Tanner graphs. Classification of LDPC ensembles via Tanner graph degree distributions will be introduced, but we will only superficially cover the design of LDPC codes with optimal degree distributions via constrained pseudo-random matrix construction. We will also review some of the other LDPC code construction techniques which have appeared in the literature. The encoding problem for such LDPC codes will be presented and certain special classes of LDPC codes which resolve the encoding problem will be introduced. Finally, the iterative message-passing decoding algorithm (and certain simplifications) which provides near-optimal performance will be presented.

*This work was funded by NSF grants CCR-9814472 and CCR-9979310, NASA grant NAG5-10643, and a gift from the INSIC EHDR program.

2. Representations of LPDC Codes

2.1. Matrix Representation

Although LDPC codes can be generalized to non-binary alphabets, we shall consider only binary LDPC codes for the sake of simpicity. Because LDPC codes form a class of linear block codes, they may be described as a certain k-dimensional subspace $\mathcal{C}$ of the vector space $\mathbb{F}_2^n$ of binary n-tuples over the binary field $\mathbb{F}_2$. Given this, we may find a basis $B = \{\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_{k-1}\}$ which spans $\mathcal{C}$ so that each $\mathbf{c} \in \mathcal{C}$ may be written as $\mathbf{c} = u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + ... + u_{k-1}\mathbf{g}_{k-1}$ for some $\{u_i\}$; more compactly, $\mathbf{c} = \mathbf{u}G$ where $\mathbf{u} = [u_0\ u_1\ ...\ u_{k-1}]$ and G is the so-called $k \times n$ generator matrix whose rows are the vectors $\{\mathbf{g}_i\}$ (as is conventional in coding, all vectors are row vectors). The $(n-k)$-dimensional null space $\mathcal{C}^\perp$ of G comprises all vectors $\mathbf{x} \in \mathbb{F}_2^n$ for which $\mathbf{x}G^T = \mathbf{0}$ and is spanned by the basis $B^\perp = \{\mathbf{h}_0, \mathbf{h}_1, ..., \mathbf{h}_{n-k-1}\}$. Thus, for each $\mathbf{c} \in \mathcal{C}$, $\mathbf{c}\mathbf{h}_i^T = 0$ for all i or, more compactly, $\mathbf{c}H^T = \mathbf{0}$ where H is the so-called $(n-k) \times n$ parity-check matrix whose rows are the vectors $\{\mathbf{h}_i\}$, and is the generator matrix for the null space $\mathcal{C}^\perp$. The parity-check matrix H is so named because it performs $m = n - k$ separate parity checks on a received word.

A *low-density parity-check code* is a linear block code for which the parity-check matrix H has a low density of 1's. A *regular LDPC code* is a linear block code whose parity-check matrix H contains exactly w_c 1's in each column and exactly $w_r = w_c(n/m)$ 1's in each row, where $w_c << m$ (equivalently, $w_c << m$). The code rate $R = k/n$ is related to these parameters via $R = 1 - w_c/w_r$ (this assumes H is full rank). If H is low density, but the number of 1's in each column or row is not constant, then the code is an *irregular LDPC code.* It is easiest to see the sense in which an LDPC code is regular or irregular through its graphical representation.

2.2. Graphical Representation

Tanner considered LDPC codes (and a generalization) and showed how they may be represented effectively by a so-called bipartite graph, now call a Tanner graph [2]. The Tanner graph of an LDPC code is analogous to the trellis of a convolutional code in that it provides a complete representation of the code and it aids in the description of the decoding algorithm. A *bipartite graph* is a graph (nodes connected by edges) whose nodes may be separated into two types, and edges may only connect two nodes of different types. The two types of nodes in a Tanner graph are the *variable nodes* and the *check nodes* (which we shall call v-nodes and c-nodes, respectively).[1] The Tanner graph of a code is drawn according to the following rule: check node j is connected to variable node i whenever element h_{ji} in H is a 1. One may deduce from this that there are $m = n - k$ check nodes, one for each check equation, and n variable nodes, one for each code bit c_i. Further, the m rows of H specify the m c-node connections, and the n columns of H specify the n v-node connections.

Example. Consider a (10, 5) linear block code with $w_c = 2$ and $w_r = w_c(n/m) = 4$ with the

[1]The nomenclature varies in the literature: variable nodes are also called bit or symbol nodes and check nodes are also called function nodes.

following H matrix:

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

The Tanner graph corresponding to H is depicted in Fig. 1. Observe that v-nodes c_0, c_1, c_2, and c_3 are connected to c-node f_0 in accordance with the fact that, in the zeroth row of H, $h_{00} = h_{01} = h_{02} = h_{03} = 1$ (all others are zero). Observe that analogous situations holds for c-nodes f_1, f_2, f_3, and f_4 which corresponds to rows 1, 2, 3, and 4 of H, respectively. Note, as follows from the fact that $\mathbf{c}H^T = \mathbf{0}$, the bit values connected to the same check node must sum to zero. We may also proceed along columns to construct the Tanner graph. For example, note that v-node c_0 is connected to c-nodes f_0 and f_1 in accordance with the fact that, in the zeroth column of H, $h_{00} = h_{10} = 1$. ■

Note that the Tanner graph in this example is regular: each v-node has two edge connections and each c-node has four edge connections (that is, the *degree* of each v-node is 2 and the degree of each c-node is 4). This is in accordance with the fact that $w_c = 2$ and $w_r = 4$. Is is also clear from this example that $mw_r = nw_c$.

For irregular LDPC codes, the parameters w_c and w_r are functions of the column and row numbers and so such notation is not generally adopted in this case. Instead, it is usual in the literature (following [7]) to specify the v-node and c-node *degree distribution polynomials*, denoted by $\lambda(x)$ and $\rho(x)$, respectively. In the polynomial

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1},$$

λ_d denotes the fraction of all edges connected to degree-d v-nodes and d_v denotes the maximum v-node degree. Similarly, in the polynomial

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1},$$

ρ_d denotes the fraction of all edges connected to degree-d c-nodes and d_c denotes the maximum c-node degree. Note for the regular code above, for which $w_c = d_v = 2$ and $w_r = d_c = 4$, we have $\lambda(x) = x$ and $\rho(x) = x^3$.

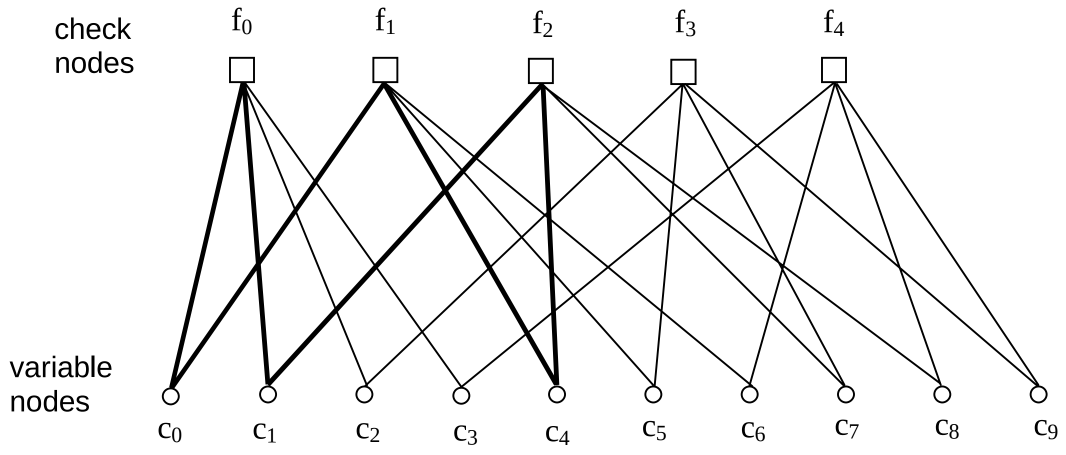

Fig. 1. Tanner graph for example code.

A *cycle* (or *loop*) of length ν in a Tanner graph is a path comprising ν edges which closes back on itself. The Tanner graph in the above example possesses a length-6 cycle as exemplified by the six bold edges in the figure. The *girth* γ of a Tanner graph is the minimum cycle length of the graph. The shortest possible cycle in a bipartite graph is clearly a length-4 cycle, and such cycles manifest themselves in the H matrix as four 1's that lie on the corners of a submatrix of H. We are interested in cycles, particularly short cycles, because they degrade the performance of the iterative decoding algorithm used for LDPC codes. This fact will be made evident in the discussion of the iterative decoding algorithm.

3. LDPC Code Design Approaches

Clearly, the most obvious path to the construction of an LDPC code is via the construction of a low-density parity-check matrix with prescribed properties. A large number of design techniques exist in the literature, and we introduce some of the more prominent ones in this section, albeit at a superficial level. The design approaches target different design criteria, including efficient encoding and decoding, near-capacity performance, or low-error rate floors. (Like turbo codes, LPDC codes often suffer from low-error rate floors, owing both to poor distance spectra and weaknesses in the iterative decoding algorithm.)

3.1. Gallager Codes

The original LDPC codes due to Gallager [1] are regular LDPC codes with an H matrix of the form

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_{w_c} \end{bmatrix}$$

where the submatrices H_d have the following structure. For any integers μ and w_r than greater than 1, each submatrix H_d is $\mu \times \mu w_r$ with row weight w_r and column weight 1. The submatrix

H_1 has the following specific form: for $i = 1, 2, ..., \mu$, the i-th row contains all of its w_r 1's in columns $(i-1)w_r + 1$ to iw_r. The other submatrices are simply column permutations of H_1. It is evident that H is regular, has dimension $\mu w_c \times \mu w_r$, and has row and column weights w_r and w_c, respectively. The absence of length-4 cycles in H is not guaranteed, but they can be avoided via computer design of H. Gallager showed that the ensemble of such codes has excellent distance properties provided $w_c \geq 3$ and $w_r > w_c$. Further, such codes have low-complexity encoders since parity bits can be solved for as a function of the user bits via the parity-check matrix [1].

Gallager codes were generalized by Tanner in 1981 [2] and were studied for application to code-division multiple-access communication channel in [9]. Gallager codes were extended by MacKay and others [3], [4].

3.2. MacKay Codes

MacKay had independently discovered the benefits of designing binary codes with sparse H matrices and was the first to show the ability of these codes to perform near capacity limits [3], [4]. MacKay has archived on a web page [10] a large number of LPDC codes he has designed for application to data communication and storage, most of which are regular. He provided in [4] algorithms to semi-randomly generate sparse H matrices. A few of these are listed below in order of increasing algorithm complexity (but not necessarily improved performance).

1. H is created by randomly generating weight-w_c columns and (as near as possible) uniform row weight.

2. H is created by randomly generating weight-w_c columns, while ensuring weight-w_r rows, and no two columns having overlap greater than one.

3. H is generated as in 2, plus short cycles are avoided.

4. H is generated as in 3, plus $H = [H_1 \ H_2]$ is constrained so that H_2 is invertible (or at least H is full rank).

One drawback of MacKay codes is that they lack sufficient structure to enable low-complexity encoding. Encoding is performed by putting H in the form $[P^T \ I]$ via Gauss-Jordan elimination, from which the generator matrix can be put in the systematic form $G = [I \ P]$. The problem with encoding via G is that the submatrix P is generally not sparse so that, for codes of length $n = 1000$ or more, encoding complexity is high. An efficient encoding technique employing only the H matrix was proposed in [6].

3.3. Irregular LDPC Codes

Richardson *et al.* [7] and Luby *et al.* [8] defined ensembles of irregular LDPC codes parameterized by the degree distribution polynomials $\lambda(x)$ and $\rho(x)$ and showed how to optimize these polynomials for a variety of channels. Optimality is in the sense that, assuming message-passing decoding (described below), a typical code in the ensemble was capable of reliable communication in worse channel conditions than codes outside the ensemble are. The worst-case channel condition is called the *decoding threshold* and the optimization of $\lambda(x)$ and $\rho(x)$ is found by a combination of a so-called *density evolution* algorithm and an optimization algorithm. Density evolution refers to the evolution of the probability density functions (pdf's) of the various quantities passed around the decoder's Tanner graph. The decoding threshold for a given $\lambda(x)$-$\rho(x)$ pair is determined by evaluation the pdf's of computed log-likelihood ratios (see the next section) of the code bits. The separate optimization algorithm optimizes over the $\lambda(x)$-$\rho(x)$ pairs.

Using this approach an irregular LDPC code has been designed whose simulated performance was within 0.045 dB of the capacity limit for a binary-input AWGN channel [11]. This code had length $n = 10^7$ and rate $R = 1/2$. It is generally true that designs via density evolution are best applied to codes whose rate is not too high ($R \lesssim 3/4$) and whose length is not too short ($n \gtrsim 5000$). The reason is that the density evolution design algorithm assumes $n \to \infty$ (hence, $m \to \infty$), and so $\lambda(x)$-$\rho(x)$ pairs which are optimal for very long codes, will not be optimal for medium-length and short codes. As discussed in [12], [13], [14], [15], such $\lambda(x)$-$\rho(x)$ pairs applied to medium-length and short codes gives rise to a high error-rate floor.

Finally, we remark that, as for the MacKay codes, these irregular codes do not intrinsically lend themselves to efficient encoding. However, as mentioned above, Richardson and Urbanke [6] have proposed algorithms for achieving linear-time encoding for these codes.

3.4. Finite Geometry Codes

In [16], [17], regular LDPC codes are designed using old-fashioned textbook [18] techniques based on finite geometries. The resulting LDPC codes fall into the cyclic and quasi-cyclic classes of block codes and lend themselves to simple encoder implementation via shift-register circuits. The cyclic finite geometry codes tend to have relatively large minimum distances, but the quasi-cyclic codes tend to have somewhat small minimum distances. Also, short LDPC codes (n on the order of 200 bits) designed using these techniques are generally better than short LDPC codes designed using pseudo-random H matrices.

The cyclic finite geometry codes have the drawback that the parity-check matrix used in decoding is $n \times n$ instead of $(n-k) \times n$. (It is possible to choose an $(n-k) \times n$ submatrix of the $n \times n$ matrix to decode, but the loss in performance is generally non-negligible.) The $n \times n$ matrix is circulant, with its first row equal to a certain *incidence vector* [16]. Another drawback is that the values of w_r and w_c are relatively large which is undesirable since the complexity of the iterative message-passing decoder is proportional to these values. One final drawback is that there is not a flexibility in the choice of length and rate, although this issue can be dealt with by code shortening and puncturing.

3.5. RA, IRA, and eIRA Codes

A type of code, called a *repeat-accumulate* (RA) code, which has the characteristics of both serial turbo codes and LDPC codes, was proposed in [20]. The encoder for an RA code is shown in Fig. 2 where it is seen that user bits are repeated (2 or 3 times is typical), permuted, and then sent through an accumulator (differential encoder). These codes have been shown to be capable of operation near capacity limits, but they have the drawback that they are naturally low rate (rate 1/2 or lower).

The RA codes were generalized in such a way that some bits were repeated more than others yielding *irregular repeat-accumulate* (IRA) codes [21]. As shown in Fig. 2, the IRA encoder comprises a low-density generator matrix, a permuter, and an accumulator. Such codes are capable of operation even closer to theoretical limits than RA codes, and they permit higher code rates. A drawback to IRA codes is that they are nominally non-systematic codes, although they be put in a systematic form, but it is at the expense of greatly lowering the rate as depicted in Fig. 2.

Yang and Ryan [13], [14], [15] have proposed a class of efficiently encodable irregular LDPC codes which might be called extended IRA (eIRA) codes. (These codes were independently proposed in [22].) The eIRA encoder is shown in Fig. 2. Note that the eIRA encoder is systematic and permits both low and high rates. Further, encoding can be efficiently performed directly from the H matrix which possesses an $m \times m$ submatrix which facilitates computation of the parity bits from the user bits [22], [15].

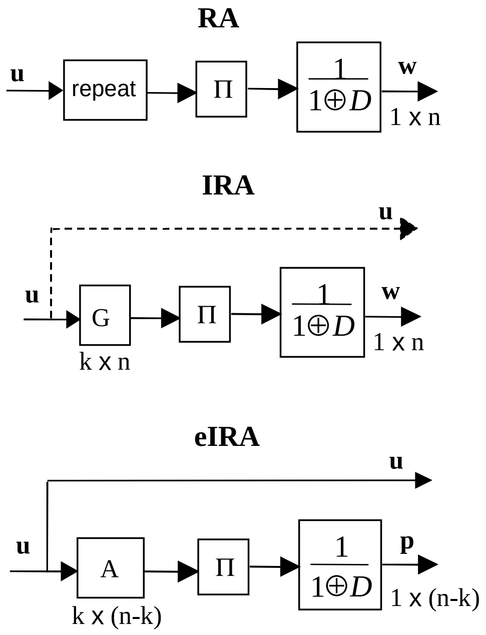

Fig. 2. Encoders for the repeat-acccumulate (RA), the irregular RA (IRA) code, and the extended IRA code (eIRA).

3.6. Array Codes

Fan has shown that a certain class of codes called array codes can be viewed as LDPC codes and thus can be decoded with a message passing algorithm [23], [24]. Subsequent to Fan's work, Eleftheriou and Ölçer [25] proposed a modified array code employing the following H matrix format

$$H = \begin{bmatrix} I & I & I & \cdots & I & I & \cdots & I \\ 0 & I & \alpha & \cdots & \alpha^{j-2} & \alpha^{j-1} & \cdots & \alpha^{k-2} \\ 0 & 0 & I & \cdots & \alpha^{2(j-3)} & \alpha^{2(j-2)} & \cdots & \alpha^{2(k-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & I & \alpha^{j-1} & \cdots & \alpha^{(j-1)(k-j)} \end{bmatrix} \tag{3.1}$$

where k and j are two integers such that $k, j \leq p$ where p denotes a prime number. I is the $p \times p$ identity matrix, O the $p \times p$ null matrix, and α a $p \times p$ permutation matrix representing a single left- or right-cyclic shift. The upper triangular nature of H guarantees encoding linear in the codeword length (encoding is essentially the same as for eIRA codes).

These modified array codes have very low error rate floors, and both low- and high-rate codes may be designed. However, as is clear from the description of H above, only selected code lengths and rates are available.

3.7. Combinatorial LDPC Codes

In view of the fact that LDPC codes may be design by constrained random construction of H matrices, it is not difficult to imagine that good LDPC codes may be designed via the application of combinatorial mathematics. That is, design constraints (such as no cycles of length four) applied to an H matrix of size $(n-k) \times n$ may be cast as a problem in combinatorics. Several researchers have successfully approached this problem via such combinatorial objects as Steiner systems, Kirkman systems, and balanced incomplete block designs [26], [27], [28], [29], [16].

4. Iterative Decoding Algorithms

4.1. Overview

In addition to introducing LDPC codes in his seminal work in 1960 [1], Gallager also provided a decoding algorithm that is typically near optimal. Since that time, other researchers have independently discovered that algorithm and related algorithms, albeit sometimes for different applications [4], [30]. The algorithm iteratively computes the distributions of variables in graph-based models and comes under different names, depending on the context. These names include: the sum-product algorithm (SPA), the belief propagation algorithm (BPA), and the message passing algorithm (MPA). The term "message passing" usually refers to all such iterative algorithms, including the SPA (BPA) and its approximations.

Much like optimal (maximum *a posteriori*, MAP) symbol-by-symbol decoding of trellis codes, we are interested in computing the *a posteriori* probability (APP) that a given bit in the transmitted codeword $\mathbf{c} = [c_0 \; c_1 \; ... \; c_{n-1}]$ equals 1, given the received word $\mathbf{y} = [y_0 \; y_1 \; ... \; y_{n-1}]$. Without loss of generality, let us focus on the decoding of bit c_i so that we are interested in computing the APP

$$\Pr(c_i = 1 \mid \mathbf{y})$$

or the APP ratio (also called the likelihood ratio, LR)

$$l(c_i) \triangleq \frac{\Pr(c_i = 0 \mid \mathbf{y})}{\Pr(c_i = 1 \mid \mathbf{y})}.$$

Later we will extend this to the more numerically stable computation of the log-APP ratio, also called the log-likelihood ratio (LLR):

$$L(c_i) \triangleq \log\left(\frac{\Pr(c_i = 0 \mid \mathbf{y})}{\Pr(c_i = 1 \mid \mathbf{y})}\right)$$

where here and in the sequel the natural logarithm is assumed.

The MPA for the computation of $\Pr(c_i = 1 \mid \mathbf{y})$, $l(c_i)$, or $L(c_i)$ is an iterative algorithm which is based on the code's Tanner graph. Specifically, we imagine that the v-nodes represent processors of one type, c-nodes represent processors of another type, and the edges represent message paths. In one half iteration, each v-node processes its input messages and passes its resulting output messages *up* to neighboring c-nodes (two nodes are said to be *neighbors* if they are connected by an edge). This is depicted in Fig. 3 for the message $m_{\uparrow 02}$ from v-node c_0 to c-node f_2 (the subscripted arrow indicates the direction of the message, keeping in mind that our Tanner graph convention places c-nodes above v-nodes). The information passed concerns $\Pr(c_0 = b \mid \text{input messages})$, $b \in \{0, 1\}$, the ratio of such probabilities, or the logarithm of the ratio of such probabilities. Note in the figure that the information passed to c-node f_2 is all the information available to v-node c_0 from the channel and through its neighbors, excluding c-node f_2; that is, only *extrinsic information* is passed. Such extrinsic information $m_{\uparrow ij}$ is computed for each connected v-node/c-node pair c_i/f_j at each half-iteration.

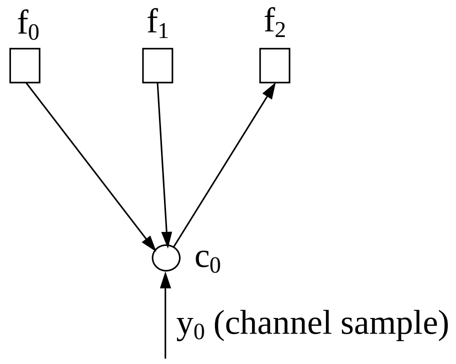

Fig. 3. Subgraph of a Tanner graph corresponding to an H matrix whose zeroth column is $[1\ 1\ 1\ 0\ 0\ ...\ 0]^T$. The arrows indicate message passing from node c_0 to node f_2.

In the other half iteration, each c-node processes its input messages and passes its resulting output messages *down* to its neighboring v-nodes. This is depicted in Fig. 4 for the message $m_{\downarrow 04}$ from c-node f_0 down to v-node c_4. The information passed concerns Pr(check equation f_0 is satisfied | input messages), $b \in \{0, 1\}$, the ratio of such probabilities, or the logarithm of the ratio of such probabilities. Note, as for the previous case, only extrinsic information is passed to v-node c_4. Such extrinsic information $m_{\downarrow ji}$ is computed for each connected c-node/v-node pair f_j/c_i at each half-iteration.

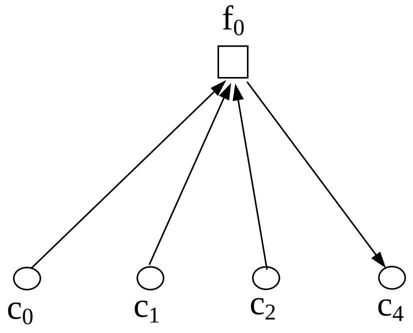

Fig. 4. Subgraph of a Tanner graph corresponding to an H matrix whose zeroth row is $[1\ 1\ 1\ 0\ 1\ 0\ 0\ ...\ 0]^T$. The arrows indicate message passing from node f_0 to node c_4.

After a prescribed maximum number of iterations or after some stopping criterion has been met, the decoder computes the APP, the LR, or the LLR from which decisions on the bits c_i are made. One example stopping criterion is to stop iterating when $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$, where $\hat{\mathbf{c}}$ is a tentatively decoded codeword.

The MPA assumes that the messages passed are statistically independent throughout the decoding process. When the y_i are independent, this independence assumption would hold true if the Tanner graph possessed no cycles. Further, the MPA would yield exact APPs (or LRs or LLRs, depending on the version of the algorithm) in this case [30]. However, for a graph of girth γ, the independence assumption is only true up to the $\gamma/2$-th iteration, after which messages start to loop back on themselves in the graph's various cycles. Still, simulations have shown that the message passing algorithm is generally very effective provided length-four cycles are avoided. Lin *et al.* [19] showed that some configurations of length-four cycles are not harmful. It was shown in [31] how the message-passing *schedule* described above and below (the so-called *flooding schedule*) may be modified to mitigate the negative effects of short cycles.

In the remainder of this section we present the "probability domain" version of the SPA (which computes APPs) and its log-domain version, the log-SPA (which computes LLRs), as well as certain approximations. Our treatment considers the special cases of the binary erasure channel (BEC), the binary symmetric channel (BSC), and the binary-input AWGN channel (BI-AWGNC).

4.2. Probability-Domain SPA Decoder

We start by introducing the following notation:

- $V_j = \{$v-nodes connected to c-node $f_j\}$
- $V_j \backslash i = \{$v-nodes connected to c-node $f_j\} \backslash \{$v-node $c_i\}$
- $C_i = \{$c-nodes connected to v-node $c_i\}$
- $C_i \backslash j = \{$c-nodes connected to v-node $c_i\} \backslash \{$c-node $f_j\}$
- $M_v(\sim i) = \{$messages from all v-nodes except node $c_i\}$
- $M_c(\sim j) = \{$messages from all c-nodes except node $f_j\}$
- $P_i = \Pr\left(c_i = 1 \mid y_i\right)$
- $S_i =$ event that the check equations involving c_i are satisfied
- $q_{ij}(b) = \Pr\left(c_i = b \mid S_i, y_i, M_c(\sim j)\right)$, where $b \in \{0, 1\}$. For the APP algorithm presently under consideration, $m_{\uparrow ij} = q_{ij}(b)$; for the LR algorithm, $m_{\uparrow ij} = q_{ij}(0)/q_{ij}(1)$; and for the LLR algorithm, $m_{\uparrow ij} = \log\left[q_{ij}(0)/q_{ij}(1)\right]$.
- $r_{ji}(b) = \Pr\left(\text{check equation } f_j \text{ is satisfied} \mid c_i = b, M_v(\sim i)\right)$, where $b \in \{0, 1\}$. For the APP algorithm presently under consideration, $m_{\downarrow ji} = r_{ji}(b)$; for the LR algorithm, $m_{\downarrow ji} = r_{ji}(0)/r_{ji}(1)$; and for the LLR algorithm, $m_{\downarrow ji} = \log\left[r_{ji}(0)/r_{ji}(1)\right]$.

Note that the messages $q_{ij}(b)$, while interpreted as probabilities here, are random variables (rv's) as they are functions of the rv's y_i and other messages which are themselves rv's. Similarly, by virtue of the message passing algorithm, the messages $r_{ji}(b)$ are rv's.

Consider now the form of $q_{ij}(0)$ which, given our new notation and the independence assumption, we may express as (see Fig. 5)

$$\begin{aligned} q_{ij}(0) &= \Pr\left(c_i = 0 \mid y_i, S_i, M_c(\sim j)\right) \\ &= (1 - P_i)\Pr\left(S_i \mid c_i = 0, y_i, M_c(\sim j)\right)/\Pr\left(S_i\right) \\ &= K_{ij}\,(1 - P_i)\prod_{j' \in C_i \backslash j} r_{j'i}(0) \end{aligned} \tag{4.1}$$

where we used Bayes' rule twice to obtain the second line and the independence assumption to obtain the third line. Similarly,

$$q_{ij}(1) = K_{ij}P_i \prod_{j' \in C_i \backslash j} r_{j'i}(1). \tag{4.2}$$

The constants K_{ij} are chosen to ensure that $q_{ij}(0) + q_{ij}(1) = 1$.

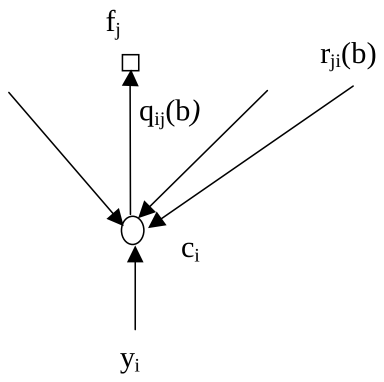

Fig. 5. Illustration of message passing half-iteration for the computation of $q_{ij}(b)$.

To develop an expression for the $r_{ji}(b)$, we need the following result.

Result 1. (Gallager [1]) Consider a sequence of M independent binary digits a_i for which $\Pr(a_i = 1) = p_i$. Then the probability that $\{a_i\}_{i=1}^{M}$ contains an *even* number of 1's is

$$\frac{1}{2} + \frac{1}{2}\prod_{l=i}^{M}(1 - 2p_i)\,. \tag{4.3}$$

Proof: Induction on M. ■

In view of this result, together with the correspondence $p_i \leftrightarrow q_{ij}(1)$, we have (see Fig. 6)

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2}\prod_{i' \in V_j \backslash i}(1 - 2q_{i'j}(1)) \tag{4.4}$$

since, when $c_i = 0$, the bits $\{c_{i'} : i' \in V_j \backslash i\}$ must contain an even number of 1's in order for check equation f_j to be satisfied. Clearly,

$$r_{ji}(1) = 1 - r_{ji}(0). \tag{4.5}$$

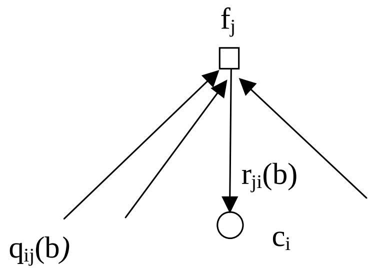

Fig. 6. Illustration of message passing half-iteration for the computation of $r_{ji}(b)$.

The MPA for the computation of the APP's is initialized by setting $q_{ij}(b) = \Pr(c_i = b \mid y_i)$ for all i, j for which $h_{ij} = 1$ (that is, $q_{ij}(1) = P_i$ and $q_{ij}(0) = 1 - P_i$). Here, y_i represents channel symbol that was actually received (i.e., it is a not variable here). We consider the following special cases.

BEC. In this case, $y_i \in \{0, 1, E\}$ where E is the erasure symbol, and we define $\delta = \Pr(y_i = E \mid c_i = b)$ to be the erasure probability. Then it is easy to see that

$$\Pr(c_i = b \mid y_i) = \begin{cases} 1 & \text{when } y_i = b \\ 0 & \text{when } y_i = b^c \\ 1/2 & \text{when } y_i = E \end{cases}$$

where b^c represents the complement of b.

BSC. In this case, $y_i \in \{0, 1\}$ and we define $\varepsilon = \Pr(y_i = b^c \mid c_i = b)$ to be the error probability. Then it is obvious that

$$\Pr(c_i = b \mid y_i) = \begin{cases} 1 - \varepsilon & \text{when } y_i = b \\ \varepsilon & \text{when } y_i = b^c \end{cases} .$$

BI-AWGNC. We first let $x_i = 1 - 2c_i$ be the i-th transmitted binary value; note $x_i = +1(-1)$ when $c_i = 0(1)$. We shall use x_i and c_i interchangeably hereafter. Then the i-th received sample is $y_i = x_i + n_i$ where the n_i are independent and normally distributed as $\eta(0, \sigma^2)$. Then it is easy to show that

$$\Pr(x_i = x \mid y_i) = \left[1 + \exp\left(-2yx/\sigma^2\right)\right]^{-1}$$

where $x \in \{\pm 1\}$.

Summary of the Probability-Domain SPA Decoder

1. For $i = 0, 1, ..., n-1$, set $P_i = \Pr(c_i = 1 \mid y_i)$ where y_i is the i-th received channel symbol. Then set $q_{ij}(0) = 1 - P_i$ and $q_{ij}(1) = P_i$ for all i, j for which $h_{ij} = 1$.

2. Update $\{r_{ji}(b)\}$ using equations (4.4) and (4.5).

3. Update $\{q_{ji}(b)\}$ using equations (4.1) and (4.2). Solve for the constants K_{ij}.

4. For $i = 0, 1, ..., n-1$, compute

$$Q_i(0) = K_i (1 - P_i) \prod_{j \in C_i} r_{ji}(0) \tag{4.6}$$

and

$$Q_i(1) = K_i P_i \prod_{j \in C_i} r_{ji}(1). \tag{4.7}$$

where the constants K_i are chosen to ensure that $Q_i(0) + Q_i(1) = 1$.

5. For $i = 0, 1, ..., n-1$, set

$$\hat{c}_i = \begin{cases} 1 \text{ if } Q_i(1) > Q_i(0) \\ 0 \text{ else} \end{cases}.$$

If $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$ or the number of iterations equals the maximum limit, stop; else, go to Step 2.

Remarks. This algorithm has been presented for pedagogical clarity, but may be adjusted to optimize the number of computations. For example, Step 4 may be computed before Step 3 and Step 3 may be replaced with the simple division $q_{ij}(b) = K_{ij} Q_i(b)/r_{ji}(b)$. We note also that, for good codes, this algorithm is able to detect an uncorrected codeword with near-unity probability (Step 5), unlike turbo codes [4].

4.3. Log-Domain SPA Decoder

As with the probability-domain Viterbi and BCJR algorithms, the probability-domain SPA suffers because multiplications are involved (additions are less costly to implement) and many multiplications of probabilities are involved which could become numerically unstable. Thus, as with the Viterbi and BCJR algorithms, a log-domain version of the SPA is to be preferred. To do so, we first define the following LLRs:

$$\begin{aligned}
L(c_i) &= \log\left(\frac{\Pr(c_i = 0 \mid y_i)}{\Pr(c_i = 1 \mid y_i)}\right) \\
L(r_{ji}) &= \log\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right) \\
L(q_{ij}) &= \log\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right) \\
L(Q_i) &= \log\left(\frac{Q_i(0)}{Q_i(1)}\right)
\end{aligned}$$

The initialization steps for the three channels under consideration thus become:

$$\begin{aligned}
L(q_{ij}) &= L(c_i) = \begin{cases} +\infty, \ y_i = 0 \\ -\infty, \ y_i = 1 \quad \text{(BEC)} \\ 0, \ y_i = E \end{cases} \qquad (4.8) \\
L(q_{ij}) &= L(c_i) = (-1)^{y_i} \log\left(\frac{1-\varepsilon}{\varepsilon}\right) \quad \text{(BSC)} \\
L(q_{ij}) &= L(c_i) = 2y_i/\sigma^2 \quad \text{(BI-AWGNC)}
\end{aligned}$$

For Step 1, we first replace $r_{ji}(0)$ with $1 - r_{ji}(1)$ in (4.5) and rearrange it to obtain

$$1 - 2r_{ji}(1) = \prod_{i' \in V_j \backslash i} (1 - 2q_{i'j}(1)) .$$

Now using the fact that $\tanh\left[\frac{1}{2}\log(p_0/p_1)\right] = p_0 - p_1 = 1 - 2p_1$, we may rewrite the equation above as

$$\tanh\left(\frac{1}{2}L(r_{ji})\right) = \prod_{i' \in V_j \backslash i} \tanh\left(\frac{1}{2}L(q_{i'j})\right) \tag{4.9}$$

The problem with these expressions is that we are still left with a product and the complex tanh function. We can remedy this as follows [1]. First, factor $L(q_{ij})$ into its sign and magnitude:

$$\begin{aligned} L(q_{ij}) &= \alpha_{ij}\beta_{ij} \\ \alpha_{ij} &= sign\left[L(q_{ij})\right] \\ \beta_{ij} &= |L(q_{ij})| \end{aligned}$$

so that (4.9) may be rewritten as

$$\tanh\left(\frac{1}{2}L(r_{ji})\right) = \prod_{i' \in V_j \backslash i} \alpha_{i'j} \cdot \prod_{i' \in V_j \backslash i} \tanh\left(\frac{1}{2}\beta_{i'j}\right) .$$

We then have

$$\begin{aligned} L(r_{ji}) &= \prod_{i'} \alpha_{i'j} \cdot 2\tanh^{-1}\left(\prod_{i'} \tanh\left(\frac{1}{2}\beta_{i'j}\right)\right) \\ &= \prod_{i'} \alpha_{i'j} \cdot 2\tanh^{-1}\log^{-1}\log\left(\prod_{i'} \tanh\left(\frac{1}{2}\beta_{i'j}\right)\right) \\ &= \prod_{i'} \alpha_{i'j} \cdot 2\tanh^{-1}\log^{-1}\sum_{i'}\log\left(\tanh\left(\frac{1}{2}\beta_{i'j}\right)\right) \\ &= \prod_{i' \in V_j \backslash i} \alpha_{i'j} \cdot \phi\left(\sum_{i' \in V_j \backslash i} \phi\left(\beta_{i'j}\right)\right) \end{aligned} \tag{4.10}$$

where we have defined

$$\phi(x) = -\log\left[\tanh(x/2)\right] = \log\left(\frac{e^x + 1}{e^x - 1}\right)$$

and used the fact that $\phi^{-1}(x) = \phi(x)$ when $x > 0$. The function is fairly well behaved, as shown in Fig. 7, and so may be implemented by a look-up table.

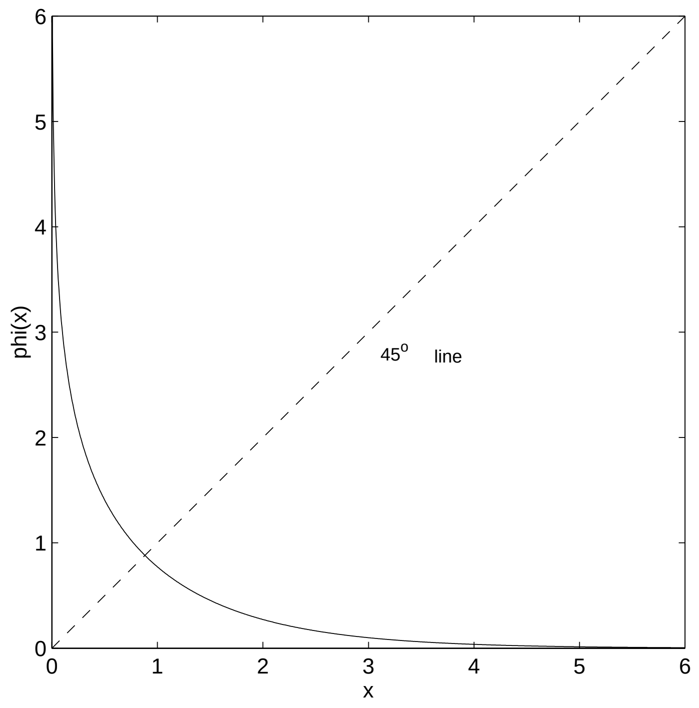

Fig. 7. Plot of the $\phi(x)$ function.

For Step 2, we simply divide equation (4.1) by (4.2) and take the logarithm of both sides to obtain

$$L(q_{ij}) = L(c_i) + \sum_{j' \in C_i \backslash j} L(r_{j'i}). \tag{4.11}$$

Step 3 is similarly modified so that

$$L(Q_i) = L(c_i) + \sum_{j \in C_i} L(r_{ji}). \tag{4.12}$$

Summary of the Log-Domain SPA Decoder

1. For $i = 0, 1, ..., n-1$, initialize $L(q_{ij})$ according to (4.8) for all i, j for which $h_{ij} = 1$.
2. Update $\{L(r_{ji})\}$ using equation (4.10).
3. Update $\{L(q_{ij})\}$ using equation (4.11).
4. Update $\{L(Q_i)\}$ using equation (4.12).
5. For $i = 0, 1, ..., n-1$, set

$$\hat{c}_i = \begin{cases} 1 \text{ if } L(Q_i) < 0 \\ 0 \text{ else} \end{cases}.$$

   If $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$ or the number of iterations equals the maximum limit, stop; else, go to Step 2.

Remark. This algorithm can be simplified further for the BEC and BSC channels since the initial LLRs (see (4.8)) are ternary in the first case and binary in the second case. See the discussion of the min-sum decoder below.

4.4. Reduced Complexity Decoders

It should be clear from the above that the log-domain SPA algorithm has lower complexity and is more numerically stable than the probability-domain SPA algorithm. We now present decoders of lower complexity which often suffer only a little in terms of performance. The degradation is typically on the order of 0.5, but is a function of the code and the channel as demonstrated in the example below.

4.4.1. The Min-Sum Decoder [32]

Consider the update equation (4.10) for $L(r_{ji})$ in the log-domain decoder. Note from the shape of $\phi(x)$ that the term corresponding to the smallest β_{ij} in the summation dominates, so that

$$\begin{aligned}\phi\left(\sum_{i'}\phi\left(\beta_{i'j}\right)\right) &\simeq \phi\left(\phi\left(\min_{i'}\beta_{i'j}\right)\right)\\ &= \min_{i'\in V_j\backslash i}\beta_{i'j}\ .\end{aligned}$$

Thus, the min-sum algorithm is simply the log-domain SPA with Step 1 replaced by

$$L(r_{ji}) = \prod_{i'\in V_j\backslash i}\alpha_{i'j}\cdot\min_{i'\in V_j\backslash i}\beta_{i'j}.$$

It can also be shown that, in the BI-AWGNC case, the initialization $L(q_{ij}) = 2y_i/\sigma^2$ may be replaced by $L(q_{ij}) = y_i$ when the min-sum algorithm is employed. The advantage, of course, is that knowledge of the noise power σ^2 is unnecessary in this case.

4.4.2. The Min-Sum-Plus-Correction-Factor Decoder [34]

Note that we can write

$$r_{ji}(b) = \Pr\left(\sum_{i'\in V_j\backslash i} c_{i'} = b \pmod 2 \mid M_v(\tilde{}i)\right)$$

so that $L(r_{ji})$ corresponds to the (conditional) LLR of a sum of binary rv's. Now consider the following general result.

Result 2. (Hagenauer *et al.* [33]) Consider two independent binary rv's a_1 and a_2 with probabilities $\Pr(a_i = b) = p_{ib}$, $b \in \{0,1\}$, and LLR's $L_i = L(a_i) = \log(p_{i0}/p_{i1})$. The LLR of the binary sum $A_2 = a_1 \oplus a_2$, defined as

$$L(A_2) = \log\left(\frac{\Pr(A_2 = 0)}{\Pr(A_2 = 1)}\right),$$

is given by

$$L(A_2) = \log\left(\frac{1 + e^{L_1+L_2}}{e^{L_1} + e^{L_2}}\right). \tag{4.13}$$

Proof.

$$\begin{aligned}
L(A_2) &= \log\left(\frac{\Pr(a_1 \oplus a_2 = 0)}{\Pr(a_1 \oplus a_2 = 1)}\right) \\
&= \log\left(\frac{p_{10}p_{20} + p_{11}p_{21}}{p_{10}p_{21} + p_{11}p_{20})}\right) \\
&= \log\left(\frac{1 + \frac{p_{10}}{p_{11}}\frac{p_{20}}{p_{21}}}{\frac{p_{10}}{p_{11}} + \frac{p_{20}}{p_{21}}}\right) \\
&= \log\left(\frac{1 + e^{L_1 + L_2}}{e^{L_1} + e^{L_2}}\right). \blacksquare
\end{aligned}$$

If more than two independent binary rv's are involved (as is the case for $r_{ji}(b)$), then the LLR of the sum of these rv's may be computed by repeated application of this result. For example, the LLR of $A_3 = a_1 \oplus a_2 \oplus a_3$ may be computed via $A_3 = A_2 \oplus a_3$ and

$$L(A_3) = \log\left(\frac{1 + e^{L(A_2) + L_3}}{e^{L(A_2)} + e^{L_3}}\right)$$

As a shorthand [33], we will write $L_1 \boxplus L_2$ to denote the computation of $L(A_2) = L(a_1 \oplus a_2)$ from L_1 and L_2; and $L_1 \boxplus L_2 \boxplus L_3$ to denote the computation of $L(A_3) = L(a_1 \oplus a_2 \oplus a_3)$ from L_1, L_2, and L_3; and so on for more variables.

We now define, for any pair of real numbers x, y,

$$\max{}^{*}(x,\ y) = \log\left(e^x + e^y\right). \tag{4.14}$$

which may be shown [35] to be

$$\max{}^{*}(x, y) = \max(x, y) + \log\left(1 + e^{-|x-y|}\right). \tag{4.15}$$

Observe from (4.13) and (4.14) that we may write

$$L_1 \boxplus L_2 = \max{}^{*}(0, L_1 + L_2) - \max{}^{*}(L_1, L_2), \tag{4.16}$$

so that

$$L_1 \boxplus L_2 = \max(0, L_1 + L_2) - \max(L_1, L_2) + s(L_1, L_2) \tag{4.17}$$

where $s(x, y)$ is a so-called *correction term* given by

$$s(x, y) = \log\left(1 + e^{-|x+y|}\right) - \log\left(1 + e^{-|x-y|}\right).$$

It can be shown [34] that

$$\max(0, L_1 + L_2) - \max(L_1, L_2) = \operatorname{sign}(L_1)\operatorname{sign}(L_2)\min(|L_1|, |L_2|)$$

so that

$$L_1 \boxplus L_2 = \operatorname{sign}(L_1)\operatorname{sign}(L_2)\min(|L_1|, |L_2|) + s(L_1, L_2) \tag{4.18}$$

which may be approximated as

$$L_1 \boxplus L_2 \simeq \operatorname{sign}(L_1)\operatorname{sign}(L_2)\min(|L_1|, |L_2|) \tag{4.19}$$

since $|s(x,y)| \le 0.693$.

Returning to the computation of $L(r_{ji})$ which we said corresponds to the LLR of a sum of binary rv's, under the independence assumption, we may write

$$L(r_{ji}) = L(q_{1j}) \boxplus ...L(q_{i-1,j}) \boxplus L(q_{i+1,j}) \boxplus ...L(q_{nj}) \ .$$

This expression may be computed via repeated application of Result 2 together with (4.17) (see [34] for an efficient way of doing this). Observe that, if the approximation (4.19) is employed, we have the min-sum algorithm. At the cost of slightly greater complexity, performance can be enhanced by using a slightly tighter approximation, by substituting $\tilde{s}(x,y)$ for $s(x,y)$ in (4.18) where [34]

$$\tilde{s}(x,y) = \begin{cases} c & \text{if } |x+y| < 2 \text{ and } |x-y| > 2\,|x+y| \\ -c & \text{if } |x-y| < 2 \text{ and } |x+y| > 2\,|x-y| \\ 0 & \text{otherwise} \end{cases}$$

and where c on the order of 0.5 is typical.

Example. We consider two regular Euclidean geometry (EG) LDPC codes and their performance with the three decoders discussed above: the (log-)SPA, the min sum, and the min sum with a correction factor (which we denote by min-sum-c, with c set to 0.5). The first code is a cyclic rate-0.82 (4095, 3367) EG LPDC code with minimum distance bound $d_{min} \ge 65$. Because the code is cyclic, it may be implemented using a shift-register circuit. The H matrix for this code is a 4095×4095 circulant matrix with row and column weight 64. The second code is a (generalized) quasi-cyclic rate-0.875 (8176, 7156) EG LDPC code. Because it is quasi-cyclic, encoding may be performed using several shift-register circuits. The H matrix for this code is 1022×8176 and has column weight 4 and row weight 32. It comprises eight 511×2044 circulant submatrices, each with column weight 2 and row weight 8. These codes are being considered for CCSDS standardization for application to satellite communications [37] (see also [16], [17], and [19]).

The performance of these codes for the three decoders on a BI-AWGNC is presented in Figs. 8 and 9. We make the following observations (all measurements are with respect to a BER of 10^{-5}).

- The length-4095 code is 1.6 dB away from the rate-0.82 BI-AWGNC capacity limit. The longer code, that is the length-8176 code, is closer to its capacity limit, only 0.9 dB away. Regular LDPC codes of these lengths might be designed which are a bit closer to their respective capacity limits, but one would have to resort to irregular LDPC codes to realize substantial gains. Of course, an irregular LDPC code would in general require a more complex encoder.

- For the length-4095 code, the loss relative to the SPA decoder suffered by the min-sum decoder is 1.1 dB and the loss suffered by the min-sum-c decoder is 0.5 dB. For the length-8176 code, these losses are 0.3 dB and 0.01 dB, respectively. We attribute the large losses for the length-4095 code to the fact that its decoder relies on an H matrix with weight-64 rows. Thus, a minimum among 64 small non-negative numbers is taken at each check node in the code's Tanner graph, so that a value near zero is usually produced and passed to a neighboring variable node. ■

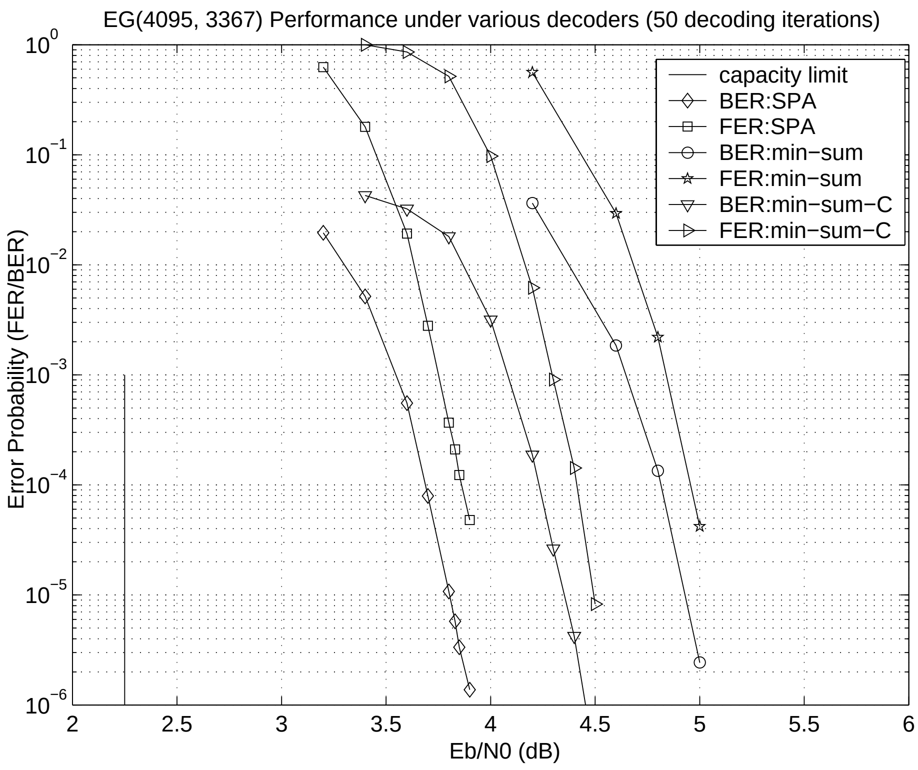

Fig. 8. Performance of a cyclic EG(4095, 3367) LDPC code on a binary-input AWGN channel and three decoding algorithms.

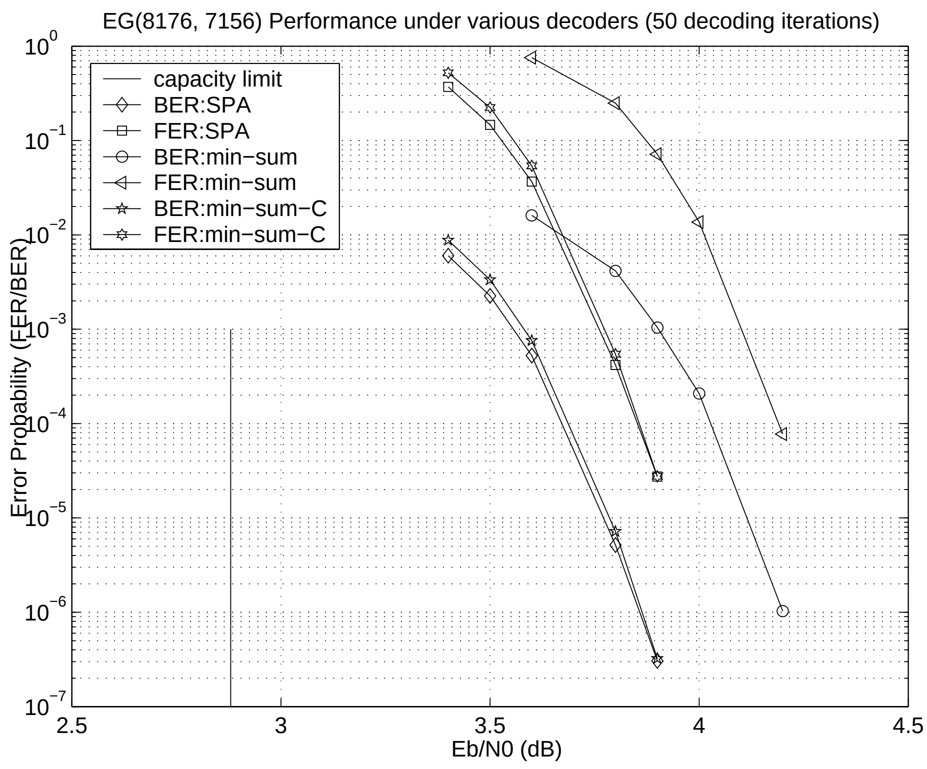

Fig. 9. Performance of a quasi-cyclic EG(8176, 7156) LDPC code on a binary-input AWGN channel and three decoding algorithms.

5. Concluding Remarks

Low-density parity-check codes are being studied for a large variety of applications, much as turbo codes, trellis codes, and other codes were when they were first introduced to the coding community. As indicated above, LDPC codes are capable of near-capacity performance while admitting an implementable decoder. Among the advantages LDPC codes have over turbo codes are: (1) they allow a parallelizable decoder, (2) they are more amenable to high code rates, (3) they generally possess a lower error-rate floor (for the same length and rate), (4) they possess superior performance in bursts (due to interference, fading, and so on), (5) they require no interleavers in the encoder and decoder, and (6) a single LDPC code can be universally good over a collection of channels [36]. Among their disadvantages are: (1) most LDPC codes have somewhat complex encoders, (2) the connectivity among the decoder component processors can be large and unwieldy, and (3) turbo codes can often perform better when the code length is short. It is easy to find in the literature many of the applications being explored for LPDC codes, including application to deep space and satellite communications, wireless (single and multi-antenna) communications, magnetic storage, and internet packet transmission.

Acknowledgments

The author would like to thank the authors of [19] and [31] for kindly providing preprints. He would also like to thank Yifei Zhang of the University of Arizona for producing Figs. 7 and 8.

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Trans. Information Theory*, pp. 21-28, Jan. 1962.

[2] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Information Theory*, pp. 533-547, Sept. 1981.

[3] D. MacKay and R. Neal, "Good codes based on very sparse matrices," in *Cryptography and Coding, 5th IMA Conf.*, C. Boyd, Ed., *Lecture Notes in Computer Science*, pp. 100-111, Berlin, Germany: Springer, 1995.

[4] D. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Information Theory*, pp. 399-431, March 1999.

[5] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Trans. Inf. Theory*, pp. 1732-1736, Nov. 1996.

[6] T. J. Richardson, and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Information Theory*, vol. 47, pp. 638-656, Feb. 2001.

[7] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Information Theory*, vol. 47, pp. 619-637, Feb. 2001.

[8] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Improved low-density parity check codes using irregular graphs," *IEEE Trans. Inf. Theory*, pp. 585-598, Feb. 2001.

[9] V. Sorokine, F. R. Kschischang, and S. Pasupathy, "Gallager codes for CDMA applications: Part I," *IEEE Trans. Comm.*, pp. 1660-1668, Oct. 2000 and "Gallager codes for CDMA applications: Part II," *IEEE Trans. Comm.*, pp. 1818-1828, Nov. 2000.

[10] D.J.C. Mackay, http://wol.ra.phy. cam.ac.uk/mackay.

[11] S. Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, " On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol 5, pp. 58 -60, Feb 2001.

[12] M. Chiani, and A. Ventura, "Design and performance evaluation of some high-rate irregular low-density parity-check codes," *Proc. 20001 IEEE GlobeCom Conf.*, pp. 990-994, Nov. 2001.

[13] M. Yang, Y. Li, and W. E. Ryan, "Design of efficiently-encodable moderate-length high-rate irregular LDPC codes," *Proc. 40th Annual Allerton Conference on Communication, Control, and Computing, Champaign, IL.*, pp. 1415-1424, Oct. 2002.

[14] M. Yang and W. E. Ryan, "Lowering the error rate floors of moderate-length high-rate LDPC codes," *Proc. 2003 Int. Symp. on Inf. Theory*, June-July, 2003.

[15] M. Yang, W. E. Ryan, and Y. Li, “Design of efficiently encodable moderate-length high-rate LDPC codes,” accepted, *IEEE Trans. Comm.*, 2003.

[16] Y. Kou, S. Lin, and M. Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” *IEEE Trans. Information Theory*, vol. 47, pp. 2711-2736, Nov. 2001.

[17] R. Lucas, M. Fossorier, Y. Kou, and S. Lin, “Iterative decoding of one-step majority-logic decodable codes based on belief propagation,” *IEEE Trans. Comm.*, pp. 931-937, June 2000.

[18] S. Lin and D. Costello, *Error-Control Coding: Fundamentals and Applications*, Prentice-Hall, 1983.

[19] H. Tang, J. Xu, S. Lin, and K. Abdel-Ghaffar, “Codes on finite geometries,” submitted to *IEEE Trans. Inf. Theory*, 2002.

[20] D. Divsalar, H. Jin, and R. McEliece, *Proc. 36th Annual Allerton Conf. on Comm., Control, and Computing*ce, “Coding theorems for turbo-like codes,” , pp. 201-210, Sept. 1998.

[21] H. Jin, A. Khandekar, and R. McEliece, “Irregular repeat-accumulate codes,” *Proc. 2nd. Int. Symp. on Turbo Codes and Related Topics*, Brest, France, pp. 1-8, Sept. 4, 2000.

[22] R. Narayanaswami, *Coded Modulation with Low-Density Parity-Check Codes*, M.S. thesis, Texas A&M University, 2001, Chapter 7.

[23] J. Fan, *Constrained coding and soft iterative decoding for storage*, Ph.D. dissertation, Stanford University, December 1999. (See also Fan’s Kluwer monograph by the same title.)

[24] J. Fan, “Array codes as low-density parity-check codes,” *2nd Int. Symposium on Turbo Codes and Related Topics*, Brest, France, pp. 543-546, Sept. 2000.

[25] E. Eleftheriou and S. Ölçer, “Low-density parity-check codes for digital subscriber lines,” *Proc. 2002 Int. Conf. on Comm.*, pp.1752-1757., April-May, 2002.

[26] D. MacKay and M. Davey, “Evaluation of Gallager codes for short block length and high rate applications,” in *Codes, Systems, and Graphical Models: Volume 123 of IMA Volumes in Mathematics and its Applications*, pp. 113-130, Spring-Verlag, New York, 2000.

[27] B. Vasic, “Structured iteratively decodable codes based on Steiner systems and their application to magnetic recording,” *Proc. 2001 IEEE GlobeCom Conf.*, pp. 2954-2958, Nov. 2001.

[28] B. Vasic, “Combinatorial constructions of low-density parity-check codes for iterative decoding,” *Proc. 2002 IEEE Int. Symp. Inf. Theory*, p. 312, June/July 2002.

[29] S. Johnson and S. Weller, “Construction of low-density parity-check codes from Kirkman triple systems,”*Proc. 2001 IEEE GlobeCom Conf.*, pp. 970-974, Nov. 2001.

[30] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, San Mateo, CA: Morgan Kaufmann, 1988.

[31] H. Xiao and A. Banihashemi, “Message-passing schedules for decoding LDPC codes,” submitted to *IEEE Trans. Comm.*, 2003.

[32] N. Wiberg, *Codes and Decoding on General Graphs*, Ph.D. dissertation, U. Linköping, Sweden, 1996.

[33] J. Hagenauer, E. Offer, and L. Papke, ”Iterative decoding of binary block and convolutional codes,” *IEEE Trans. Inf Theory*, pp. 429-445, March 1996.

[34] X-Y. Hu, E. Eleftherious, D-M. Arnold, and A. Dholakia, “Efficient implementation of the sum-product algorithm for decoding LDPC codes,” *Proc. 2001 IEEE GlobeCom Conf.*, pp. 1036-1036E, Nov. 2001.

[35] A. Viterbi, “An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes,” *IEEE JSAC*, pp. 260-264, Feb. 1998.

[36] C. Jones, A. Matache, T. Tian, J. Villasenor, and R. Wesel, “The universality of LDPC codes on wireless channels,” to appear, *Proc. 2003 IEEE Milcom conf.*, Oct. 2003.

[37] W. Fong, “White paper for LDPC codes for CCSDS channel Blue Book,” NASA GSFC White Paper submitted to the Panel 1B CCSDS Standards Committee, Oct. 2002.