Construction and Simulation Analysis of Low-Density Parity-Check Codes

Teodor B. Iliev[1]

***Abstract* – Low – Density Parity Check codes were invented in 1960 by R. Gallager. They were largely ignored until the discovery of turbo codes in 1993. Since then, LDPC codes have experienced a renaissance and are now one of the most intensely studied areas in coding. In this article we review the basic structure of LDPC codes and the iterative algorithms that are used to decode them.**

***Keywords* – Parity-check matrix, low–density parity check code**

I. INTRODUCTION

Low-Density parity-check (LDPC) codes, discovered by Gallager in 1962 [1], were rediscovered and shown to form a class of Shannon-limit-approaching codes in the late 1990 [2, 3]. These codes, decoded with iterative decoding based on belief propagation, such as the sum-product algorithm (SPA) [3, 4, 7], achieve amazingly good error performance. Ever since their rediscovery, design, construction, decoding, efficient encoding, performance analysis, and applications of these codes in digital communication and storage systems have become the focal points of research. Various methods for constructing LDPC codes have been proposed. Based on the methods of construction, LDPC codes can be classified into two general categories: 1) random (or random-like) codes generated by computer search based on certain design guidelines and required structural properties of their Tanner graphs 6], such as the girth and degree distributions [3, 5]; and 2) structured codes constructed based on algebraic and combinatorial methods, such as those given in [8, 9], and many others.

II. GRAPHICAL REPRESENTATION OF LDPC CODES

A linear block code C of rate $R = k/n$ can be defined in terms of a $(n-k)\times n$ parity-check matrix $H = [h_1, h_2,\ldots,h_n]$. Each entry h_{ij} of H is an element of a finite field GF(p). We will only consider binary codes in our work, so each entry is either a '0' or a '1' and all operations are modulo 2. The code C is the set of all vectors x that lie in the (right) nullspace of H, that is, H_x=0. Given a parity-check matrix H, we can find a corresponding $k\times n$ generator matrix G such that GH^T=0. The generator matrix can be used as an encoder according to $x^T = u^TG$.

In its simplest guise, an LDPC code is a linear block code with a parity-check matrix that is "sparse"; that is, it has a small number of nonzero entries. Gallager proposed constructing LDPC codes by randomly placing 1's and 0's in a $m\times n$ parity-check matrix H subject to the constraint that each row of H had the same number d_c of 1's and each column of H had the same number d_v of 1's.[9] On Fig. 1 is shown the m=15×n=20 parity-check matrix with d_v=3 and d_c=4 and defines an LDPC code with length n=20. Codes of this form are referred to as regular (d_v, d_c) LDPC codes of length n. In a (d_v, d_c) LDPC code each information bit is involved in d_v parity checks and each parity-check bit involves d_c information bits. The fraction of 1's in the parity-check matrix of a regular LDPC code is $\frac{md_c}{mn} = \frac{d_c}{n}$, which approaches zero as the block length gets large and leads to the name low-density parity-check codes.

$$H_1 = \left[\begin{array}{cccccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&1\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{array}\right]$$

Fig.1 Gallager – type low density parity check H_1

We must determine the rate of the code defined by H_1. Since the parity check matrix is randomly constructed, there is no guarantee that the rows are linearly independent and the matrix is full rank. Indeed, the rank of H_1 is 13<m=15, and this parity-check matrix actually defines a code with rate R=7/20. In general, such randomly constructed parity-check matrices will not be full rank and m≠n−k. We could eliminate linearly dependent rows to find a $(n-k)\times n$ parity-check matrix, but the new matrix would no longer be regular. For LDPC codes with large n, it is convenient to retain the original parity-check matrix even if it is not full rank, and it is also convenient to refer to $1-\frac{m}{n} = 1-\frac{d_v}{d_c}$ as the designed rate of the code.

[1]Teodor B. Iliev is with the Department of Communication Technique and Technologies, 8 Studentska Str., 7017 Rousse, Bulgaria, E-mail: tiliev@ecs.ru.acad.bg

Having defined a (d_v, d_c) regular LDPC code, we are now left to construct a particular instance of a code. To do this requires the choice of d_v, d_c, n, and k, which are constrained by the relationship that, for a regular code, $md_c = nd_v$ and that d_v must be less than d_c in order to have a rate less than R=1. Assuming that the block length n and the code rate are determined by the application, it remains to determine appropriate values for d_v and d_c. In [1] Gallager showed that the minimum distance of a typical regular LDPC code increased linearly with n, provided that $d_v \geq 3$. Therefore, most regular LDPC codes are constructed with d_v and d_c on the order of 3 or 4, subject to the above constraints. For large block lengths, the random placement of 1's in H such that each row has exactly d_c 1's and each column has exactly d_v 1's requires some effort, and systematic methods for doing this have been developed [3].

Tanner graph

An important advance in the theory of LDPC codes occurred when Tanner [6] used bipartite graphs to provide a graphical representation of the parity-check matrix. The bipartite graph is a graph in which the nodes may be partitioned into two subsets such that there are no edges connecting nodes within a subset. In the context of LDPC codes, the two subsets of nodes are referred to as variable nodes and check nodes. There is one variable node for each of the n bits in the code, and there is one check node for each of the m rows of H. An edge exists between the ith variable node and the jth check node if and only if h_{ij} =1. The bipartite graph corresponding to the parity-check matrix H_1 is shown in Fig. 9. In a graph, the number of edges incident upon a node is called the degree of the node. Thus, the bipartite graph of a (d_v, d_c) LDPC code contains n variable nodes of degree d_v and m check nodes of degree d_c.

It is clear that the parity-check matrix can be deduced from the bipartite graph, and thus the bipartite graph can be used to define the code C. We can therefore start talking about codes as defined by a set of variable nodes, a set of check nodes, and set of edges. This is the current approach to addressing LDPC codes. Note that the pair (d_v, d_c), together with the code length n, specifies an ensemble of codes, rather than any particular code. This ensemble is denoted by $C^n(d_v, d_c)$. Once the degrees of the nodes are chosen, we are still free to choose which particular connections are made in the graph.

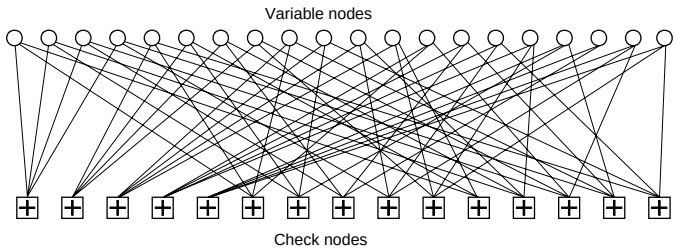

Fig.2 Bipartite graph for a (3,4) regular LDPC code with parity check matrix H_1

A socket refers to a point on a node to which an edge may be attached. A variable node has d_v sockets, meaning d_v edges may be attached to that node. There will be a total of nd_v sockets on variable nodes and md_c sockets on parity-check nodes. Clearly the number of variable-node sockets must be equal to the number of check-node sockets and a particular pattern of edge connections can be described as a permutation π from variable-node sockets to check-node sockets. An individual edge is specified by the pair (i, $\pi(i)$), which indicates that the ith variable node socket is connected to the $\pi(i)^{th}$ check-node socket.

Selecting a random code from the ensemble $C^n(d_v, d_c)$ therefore amounts to randomly selecting a permutation on nd_v elements. Many permutations will result in a graph that contains parallel edges—that is, in which more than one edge join the same variable and parity-check nodes. Note that in the parity-check matrix, an even number of parallel edges will cancel. If they are deleted from the graph, then the degrees of some nodes will be changed and the code will cease to be a regular LDPC code. If they are not deleted, their presence renders the iterative decoding algorithms ineffective. We must therefore make the restriction that permutations leading to parallel edges are disallowed.

An irregular LDPC code cannot be defined in terms of the degree parameters d_v and d_c. We must instead use degree distributions to describe the variety of node degrees in the graph. A degree distribution $\gamma(x)$ is a polynomial:

$$\gamma(x) = \sum_i \gamma_i x^{i-1} \tag{1}$$

such that $\gamma(1)=1$. The coefficients γ_i denote the fraction of edges in the graph which are connected to a node of degree i. We will also use the notation $\int \gamma$ to denote the inverse average node degree, given by

$$\int \gamma = \int_0^1 \gamma(x)dx = \sum_i \frac{\gamma_i}{i} \tag{2}$$

Let d be the average node degree corresponding to the degree distribution γ. To show that $\int \gamma = \frac{1}{d}$, let n be the total number of nodes, and let n_i be the number of nodes of degree i. The total number of edges in the graph is $d \cdot n$. Because γ_i is the fraction of edges connected to a node of degree i, we conclude that $\frac{\gamma_i}{i} = \frac{n_i}{d \cdot n}$. Completing the sum of Eq. (2), we arrive at $\int \gamma = \frac{1}{d}$.

The code length n and two degree distributions - $\lambda(x)$ and $\rho(x)$ for the variable and check nodes, respectively – are sufficient to define an ensemble $C^n(\lambda, \rho)$ of irregular LDPC codes. A graph G from this ensemble will have n variable nodes. The number of check nodes m, is given by

$$m = n\frac{\int \rho}{\int \lambda} \tag{3}$$

The number of degree-i variable nodes in G is

$$n\tilde{\lambda}_i = n\frac{\lambda_i}{i\int\lambda} \tag{4}$$

where $\tilde{\lambda}_i$ denotes the fraction of variable nodes of degree i. Similarly, the number of degree-i check nodes in G is

$$m\tilde{\rho}_i = m\frac{\rho_i}{i\int\rho} \tag{5}$$

where $\tilde{\rho}_i$ denotes the fraction of check nodes of degree i. And the design rate of the code represented by G is

$$r = \frac{n-m}{n} \tag{6}$$

We can enumerate the variable-node and check-node sockets of the irregular code graph. Selection of a code from the ensemble is a selection of a permutation on N_E elements, where N_E is the number of edges in the graph, given by

$$N_E = \frac{n}{\int\lambda} \tag{7}$$

III. MESSAGE PASSING DECODING ALGORITHMS

It was stated in the introduction that a principal advantage of low-density parity check codes is that they can be decoded using an iterative decoding algorithm whose complexity grows linearly with the block length of the code. Belief propagation is one instance of a broad class of message passing algorithms on graphs as discussed in [7, 10]. Similar decoding algorithms for the binary erasure channel and the binary symmetric channel are discussed in [11]. All message passing algorithms must, however, respect the following rule, which was introduced with turbo codes.

Rule (Extrinsic Information Principle) *A message sent from a node n along an edge e cannot depend on any message previously received on edge e.*

Before stating the algorithm, it is necessary to formulate the decoding problem. An LDPC code is constructed in terms of its parity-check matrix H which is used for decoding. To encode the information sequence u, it is necessary to derive a generator matrix G such that $GH^T=0$. Finding a suitable G is greatly simplified if we first convert H into the equivalent systematic parity-check matrix $H_S = [A|I_{n-k}]$. The systematic generator matrix is now given by

$$G_s = \left[I_k \middle| A^T\right]$$

and the information sequence is encoded as $x^T=u^TG_s$. It is worth noting that encoding by matrix multiplication has complexity $O(n^2)$ and that, in general, LDPC codes have linear decoding complexity, but quadratic encoding complexity.

The codeword x is transmitted across the additive white Gaussian noise (AWGN) channel, using BPSK modulation resulting in the received sequence $r=(2x-1)+n$. The optimal decoder for the AWGN channel is the maximum a posteriori (MAP) decoder that computes the log-likelihood ratio (LLR):

$$\Lambda(x_r) = \log\left(\frac{P(x_r = 1|y)}{P(x_r = 0|y)}\right)$$

and makes a decision by comparing this LLR to the threshold zero. The belief propagation on a graph with cycles can closely approximate the MAP algorithm, and we can state the decoding algorithm for LDPC codes on the AWGN channel using these results.

Let A={−1, +1} denote the message alphabet, let $r_i \in R$ denote the received symbol at variable node i, and let $\Lambda_i \in R$ denote the decision at variable node i. A message from variable node i to check node j is represented by $\mu_{i\to j} \in R$, and a message from check node j to variable node i is $\beta_{j\to i} \in R$. Let $C_{j\backslash i}$ be the set of variable nodes which connect to check node j, excluding variable node i. Similarly, let $V_{i\backslash j}$ be the set of check nodes which connect to variable node i, excluding check node j. The decoding algorithm is then as follows:

Step 1: Initialize $\Lambda_i = \frac{2}{\sigma^2} r_i$ for each variable node. ($\sigma^2=N_0/2$).

Step 2: Variable nodes send $\mu_{i\to j}=\lambda_i$ to each check node $j \in V_i$.

Step 3: Check nodes connected to variable node i send

$$\beta_{j\to i} = 2\tanh^{-1}\left(\prod_{l\in C_{j/i}} \tanh\left(\frac{\Lambda_l}{2}\right)\right) \tag{8}$$

Step 4: Variable nodes connected to check nodes j send

$$\mu_{i\to j} = \sum_{l\in V_{i/j}} \beta_{l\to i} \tag{9}$$

Step 5: When a fixed number of iterations have been completed or the estimated code word $\hat{x}$ satisfies the syndrome constraint $H\hat{x} = 0$, stop. Otherwise return to Step 3.

The check node's rule (8) is fairly complex. But for quantized messages it is possible to map LLRs to messages in such a way that the check-node rule can be implemented with some extra combinational logic and an adder. Such decoders provide very good performance with only a few bits of precision. If we want to keep the complexity even simpler, we can use the max-Log approximation, in which we can replace the above rule with:

$$\beta_{j\to i} \approx \min_{l\in C_{j/i}}\left(\{|\lambda_l|\}\right)\prod_{l\in C_{j/i}} sign(\Lambda_l) \tag{10}$$

Some performance loss will result from this approximation, but it may be justified by the overall savings in decoder complexity.

IV. SIMULATION RESULT

We perform simulation on an optimized regular (3,6) LDPC code with 128x256 parity check matrix over an additive white Gaussian noise channel (AWGN) with binary phase – shift keying (BPSK) modulation. In our simulation, we stop iteration as soon as a codeword is detected or when a

maximum number of iteration is reached. The maximum number of iteration is set to 10.

Fig. 3 shows the frame error rate as a function of the energy per bit (E_b) to the spectral noise density (N_0) – E_b/N_0 and 30 codewords errors. Fig. 4 shows the bit error rate as a function of E_b/N_0.

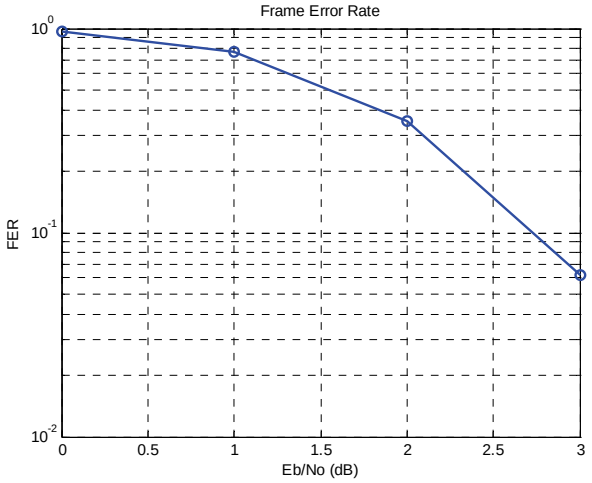

Fig.3 Frame error rate (FER) for LDPC codes over the additive white Gaussian noise channel

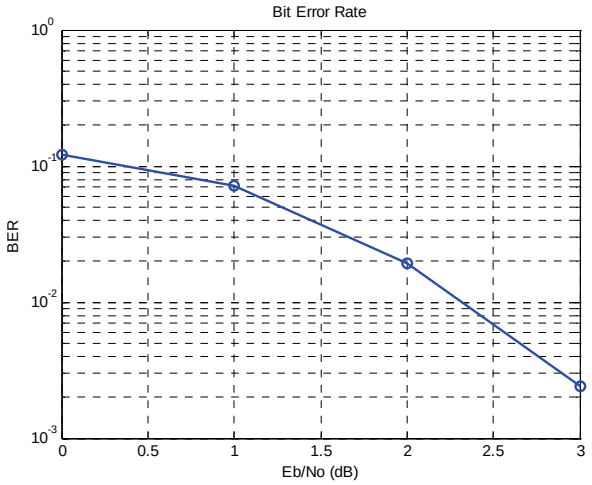

Fig.4 Bit error rate (BER) for LDPC codes over the additive white Gaussian noise channel

V. Conclusion

Analysis techniques formulated for LDPC codes are generic and can be used for other coding schemes based on iterative decoding methods. Each LDPC code is characterized by a number of fixed parameters that include the following: check node degree, variable node degree and blocksize N. These parameters are used to determine the nodes in the LDPC factor graph and a collection of permissible edges between the nodes. Given these parameters, there exists a finite number of possible ways in which edges can be connected between nodes in a factor graph. By varying the parameters of the graphs to increase their girth one can improve the distance properties of the code without increasing the number of parity checks in which each bit is involved. However the size of the graph and hence size of the code, required grows rapidly. Some modifications to standard decoding algorithms for LDPC codes have also been presented.

Finally, the goal of this paper was to present the main constructive principle of low–density parity check codes, as well as a design methodology, that gives good performance for many communication systems like latest DVB satellite communications standard.

Acknowledgement

The author is grateful to Georgi Petkov for his helpful comments.

References

[1] R. G. Gallager, “Low density parity check codes,” IRE Trans. Inf. Theory, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.
[2] D. J. C. MacKay and R. M. Neal, “Near-Shannon-limit performance of low density parity check codes,” Electron. Lett., vol. 32, pp. 1645–1646, Aug. 1996.
[3] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” IEEE Trans. Inf. Theory, vol. 45, no. 3, pp. 399–432, Mar. 1999.
[4] T. Richardson and R. Urbanke, “The capacity of low-density parity check codes under message-passing decoding,” IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 599–618, Feb. 2001.
[5] T. Richardson, A. Shokrollahi, and R. Urbanke, “Design of capacity approaching low density parity check codes,” IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[6] R. M. Tanner, “A recursive approach to low complexity codes,” IEEE Trans. Inf. Theory, vol. IT-27, no. 9, pp. 533–547, Sep. 1981.
[7] F. R. Kschischang, B. J. Frey, and H. -A. Loeliger, “Factor graphs and the sum-product algorithm,” IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[8] R. M. Tanner, “Spectral graphs for quasi-cyclic LDPC codes,” in Proc. IEEE Int. Symp. Inf. Theory, Washington, DC, Jun. 2001, p. 226.
[9] S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2004.
[10] G. D. Forney, “Codes on graphs: Normal realizations,” IEEE Trans. Inform. Theory, pp. 520–548, Feb. 2001.
[11] C. Schlegel, L. Perez, Trellis and turbo coding, IEEE Press, 2004