On the Construction and Performance of LDPC Codes

B.N. Sindhu Tejaswini, Rajendra Prasad Lal and V. Ch. Venkaiah

Abstract Low-Density-Parity-Check (LDPC) codes are excellent error correcting codes performing very close to the Shannon's limit, enabling efficient and reliable communication. Ever since their importance was known, a lot of research has gone into the construction/designing of efficient LDPC codes. Many different construction methods have been proposed so far. This paper explores some of these construction methods and includes their performance results on the Additive White Gaussian Noise (AWGN) channel. In particular, LDPC code construction using cage graphs and permutation matrices are investigated. Irregular LDPC codes have been constructed from regular LDPC codes using an expansion method, followed by their code rate comparison.

Keywords Channel coding · LDPC code · Parity-check matrix · Cage graph · Quasi-cyclic LDPC · Gray code · Code rate · SNR-BER plots

1 Introduction

Channel Coding, also called Error-Control Coding (ECC), is a mechanism of controlling errors in data transmission over unreliable communication channels [1]. The main aim of ECC is to help the receiver detect and correct errors introduced due to noise. The central idea here is that the message to be communicated is first 'encoded' by the sender, i.e. 'redundancy' is added to it to make it a codeword. This codeword is then sent through the channel and the received message is 'decoded' by the receiver into a message that resembles the original one. ECC has many

B.N. Sindhu Tejaswini (✉) · R.P. Lal · V.Ch.Venkaiah
SCIS, University of Hyderabad, Hyderabad, India
e-mail: sindhu.buddhavarapu@gmail.com

R.P. Lal
e-mail: rplcs@uohyd.ernet.in

V.Ch.Venkaiah
e-mail: vvcs@uohyd.ernet.in

applications in many real-world communication systems such as in storage devices, bar codes, satellite communication, mobile networks etc. This paper is about a class of error correcting codes called LDPC codes, which are one of the best error-correcting codes today. They have gained huge importance for their practical advantages over other codes.

This paper is organised as follows. In Sect. 2, a brief explanation of LDPC codes is given. Section 3 describes the construction of LDPC codes using cubic cages, and Sect. 4 describes a method to construct regular quasi-cyclic (QC) LDPC codes. In Sect. 5, we explain the Gray code method of constructing regular LDPC codes and give a comparison of the above three methods. Section 6 has experimental results and finally, we conclude the paper in Sect. 7.

2 LDPC Codes—A Brief Overview

LDPC codes were first introduced by Gallager in 1962 [2]. They are a class of linear block codes which are defined by a sparse parity-check matrix, H. They are highly efficient and can provide performance very close to the channel capacity. They also have linear time encoding/decoding algorithms in terms of code length.

Tanner Graph. It is a graphical representation of the parity-check matrix [3]. The Tanner graph has two sets of nodes—check nodes and variable nodes, which represent the rows and columns of H respectively. See Fig. 1 as example.

The LDPC decoding algorithm is an iterative message-passing algorithm which is described on the Tanner graph. Performance of this algorithm is affected by the presence of cycles, especially short cycles, in the graph. Short cycles leave a bad effect on the decoder, as they affect the independence of the information exchanged in the decoding process, and prevent the decoder from converging to the optimum result. Hence, short cycles have to be avoided. The length of the shortest cycle in a graph is called *girth*. If the girth is large enough, then the decoder can run a good number of iterations and decode correctly before it is affected by the cycle. This is why high-girth codes should be constructed.

$$\mathbf{H} = \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \left[\begin{array}{ccccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{array}$$

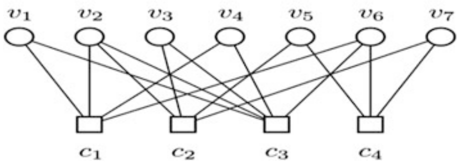

Fig. 1 LDPC code matrix and its tanner graph (*Source* [4])

Construction. Construction of a code is the definition of the pattern of connections among the rows and columns of the H matrix [5]. The main objectives of code construction are good decoding and easier hardware implementation [5].

Performance Evaluation. In order to evaluate the reliability of a digital channel, a plot showing Signal-to-Noise-Ratio (SNR) versus Bit Error Rate (BER) values is used. The SNR-BER plot gives the BER values of the channel at different SNR values. A lower BER value indicates fewer errors and thereby, a better error correction mechanism.

We now discuss some approaches of LDPC code construction in the next few sections, which were implemented and analyzed by us.

3 Construction of Column-Weight Two Codes Based on Cubic Cages

This method [6, 7] produces regular column-weight two LDPC codes. A *regular* code has a fixed row-weight and a fixed column-weight. A (k, g) cage graph is a k-regular graph of girth g having the least possible number of vertices. A cage graph of degree 3 is called a cubic cage. Cage graphs can also be used to represent the parity-check matrix of an LDPC code. A procedure to construct cubic cages is as follows [7].

1. Take a cubic tree T (tree in which each node has a degree of 1 or 3) with t vertices in it. It has r end or leaf vertices, where $r = t/2 + 1$.
2. Make n copies of that tree $T_1, T_2, T_3 \ldots, T_n$. The values t and n should be chosen carefully based on our girth requirement. Now label the vertices of the trees as $1, 2, \ldots, t$ such that the first r labelling correspond to the r end vertices in each tree. Labelling should be different for odd-indexed and even-indexed trees.
3. Next, choose r random positive integers $h_1, h_2, \ldots, h_r$ where $h_i < n/2$ for $1 \leq i \leq r$. Each h-value corresponds to one end vertex of the trees i.e., h_1 corresponds to the vertices numbered as 1, h_2 corresponds to those numbered as 2 and so on.
4. Map/connect the end vertices of these n trees based to the following rule.

   - If the value of any h is x, then its corresponding vertex in the first tree is connected to that in the tree that is after $x - 1$ trees from it. Similarly, its vertex in the second tree is connected to that in the tree that is after $x - 1$ trees from it and so on.
   - Connection should be done in a cyclic manner, wherein vertices of the last trees are connected to those in the first trees based on the above same rule. The graph obtained after this procedure is the final $(3, g)$ cage graph.

As an example, see Fig. 2, wherein $t = 6$, $n = 4$, and the vertices are labelled as shown. This figure illustrates how connections are made, wherein the vertices corresponding to two h-values, h_1 and h_2, are connected. Here, $h_1 = 1$ and $h_2 = 3$. Note that this figure is only for illustration purposes, and we do not claim that the graph shown in it is a cage graph. Also, the edges here are shown directed only for illustration.

Observation and Analysis. This method has been implemented by us and three cubic cages having girths 14, 15 and 16 respectively were constructed. To check for the possibility of obtaining higher girth cages, we tried out an experiment in which girth was calculated for all possible h-value sets. This however, did not give any higher girth. Other experiments can be carried out varying different parameters like —number of vertices in the cubic tree T (t), number of copies of the tree (n) etc., which may perhaps yield higher girth cages.

4 Quasi-cyclic LDPC Codes Using Quadratic Congruences

This method [8] constructs a *(j, k)* regular QC-LDPC code, where j is the column-weight and k is the row-weight of the code. A quasi-cyclic code with index s is a code in which the circular shift of any codeword by s positions gives another codeword. One procedure to construct QC-LDPC is as follows.

1. Select a prime $p > 2$. Choose the desired j and k values. Now construct two sequences $\{a_0, a_1, \ldots, a_{j-1}\}$ and $\{b_0, b_1, \ldots, b_{k-1}\}$ whose elements are randomly selected from $GF(p)$. Note that every element in a sequence should be unique. Then form a preliminary $j \times k$ matrix Y as follows [8].

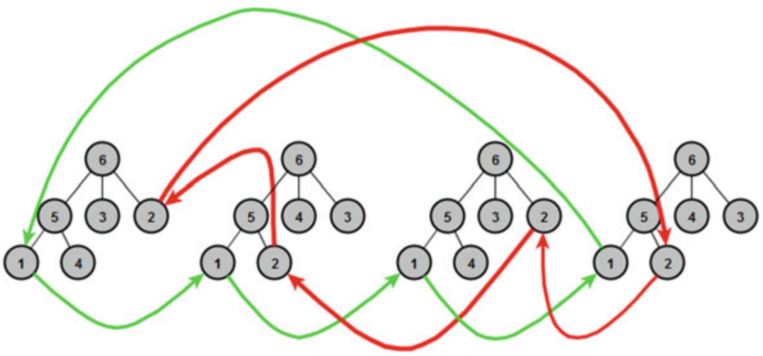

Fig. 2 Connection of trees based on two h-values ($h_1 = 1$, $h_2 = 3$)

$$Y = \begin{bmatrix} y_{0,\,0} & y_{0,\,1} & \cdots & y_{0,\,k-1} \\ y_{1,\,0} & y_{1,\,1} & \cdots & y_{1,\,k-1} \\ \vdots & & & \\ y_{j-1,\,0} & y_{j-1,\,1} & \cdots & y_{j-1,\,k-1} \end{bmatrix}$$

where the *(u, v)* element of Y $(0 \le u \le j-1$ and $0 \le v \le k-1)$ is calculated using the below equation for a fixed parameter *d*, $d \in \{1, 2,\ldots, p-1\}$.

$$y_{u,v} = \left[d(a_u + b_v)^2 + e_u + e_v\right](mod\,p)$$

and $e_u, e_v \in \{0, 1,\ldots, p-1\}$.

2. Now, the parity-check matrix H can be written as [8]

$$H = \begin{bmatrix} I(y_{0,\,0}) & I(y_{0,\,1}) & \cdots & I(y_{0,\,k-1}) \\ I(y_{0,\,0}) & I(y_{1,\,1}) & \cdots & I(y_{1,\,k-1}) \\ \vdots & & & \\ I(y_{j-1,\,0}) & I(y_{j-1,\,1}) & \cdots & I(y_{j-1,\,k-1}) \end{bmatrix}$$

where *I(x)* is a $p \times p$ identity matrix whose rows are cyclically shifted to the right by x positions. The final obtained H matrix will be a $jp \times kp$ matrix. Girths of the codes constructed from this method vary depending on the choice of p, j, k, d, e_u and e_v values.

Observation and Analysis. This method has been implemented by us and regular QC-LDPC codes of various sizes were constructed. Their BER simulations were run on AWGN channel with BPSK modulation, and were compared to those of the same-sized random codes. In all cases, QC-LDPC codes had lower error rates than the random ones. See Sect. 6.1 for results.

5 Gray Code Construction of LDPC Codes

It is a simple method [9] to construct regular column-weight two LDPC codes.

1. Let H be the parity-check matrix of the code, and ρ be the required row-weight. Now let X be the decimal point set of H (a set consisting of decimal numbers, whose elements form the parity-check matrix H), having $\rho + 1$ elements in it ($X = \{X_0, X_1,\ldots, X_\rho\}$), which satisfy one of the equations [9]

$$\begin{aligned} X_{i+1} &= 2X_i + 1 \qquad\qquad or \\ X_{i+1} &= 2^i + X_i \quad \textbf{for } i = 0,\ 1,\ 2,\ \ldots,\ \rho \end{aligned}$$

The first element is $X_0 = 0$ in both the cases.

2. H can now be constructed from X as follows. The elements of X form the first row of H. The subsequent rows are obtained by circularly shifting the previous row by one, until the first row repeats.
3. However, only codes with column-weight one are produced using this method. To obtain higher-weight codes, H is divided into sub-matrices as shown below. The number of sub-matrices is equal to the desired column-weight of the code.

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

$H_1, H_2, \ldots, H_{\Upsilon}$ are sub-matrices and Υ is the desired column-weight.

4. Construction of H_1 is same as the construction of H as described above. For constructing H_2, we fill the first row of H_1 in reverse order to form the first row of H_2. The subsequent rows are formed by cyclically shifting the previous row by one, either to the left or right, until the first row repeats. The same construction method is used for all other sub-matrices, in which the first row of H_2 is taken in reverse order to form the first row of H_3 and so on. These sub-matrices are then appended together to form the final H (as shown above) [9].
5. Elements of H are now represented in their Gray code form to obtain the final binary H matrix of order $m \times n$. The number of bits to be used for Gray code representation is up to the designer, but must be sufficient enough to represent the largest element of H. This way, there is flexibility offered in terms of code length.

The Table 1 gives a quick comparison of the above three methods of construction.

Table 1 Comparison of LDPC construction methods discussed

Parameter	Cage graph LDPC	QC-LDPC	Gray code LDPC
Input	Cubic tree T, h-values	Prime p, required w_c and w_r, a and b sequences, e_u, e_v, d	Required w_c and w_r, and no. of bits for gray code representation
Output	Column-weight two LDPC code	Regular QC-LDPC code	Regular LDPC code
Suitability	For constructing high-girth codes, and also in cases where the channel is highly error-prone	When the encoding complexity has to be less. Also in long distance communications like NEC, and in high data rate applications	Generally suitable for simpler applications

6 Experimental Results

6.1 QC-LDPC Versus Randomly Constructed LDPC

SNR-BER simulations of two QC and random LDPC codes for 25 decoder iterations over AWGN channel are given below (Fig. 3). QC-LDPC outperformed random LDPC for all code lengths tested.

6.2 Comparing Code Rate with and Without Expansion

In many cases, irregular codes give better BER results than regular ones. But the construction of irregular codes is more complex. Instead, in order to construct an irregular code, we first construct a regular code and expand it to make it irregular. The expansion method for this is given in [10]. Here in our experiment, constructed regular codes were expanded by two levels to make them irregular. The results given in Table 2 show that expansion decreases the code rate. Lesser code rate indicates lesser message bits and higher parity bits in the codeword. This implies better protection. Hence, codes having lesser rates have better error correction ability. However, very low rate codes offer less channel throughput and are not desirable.

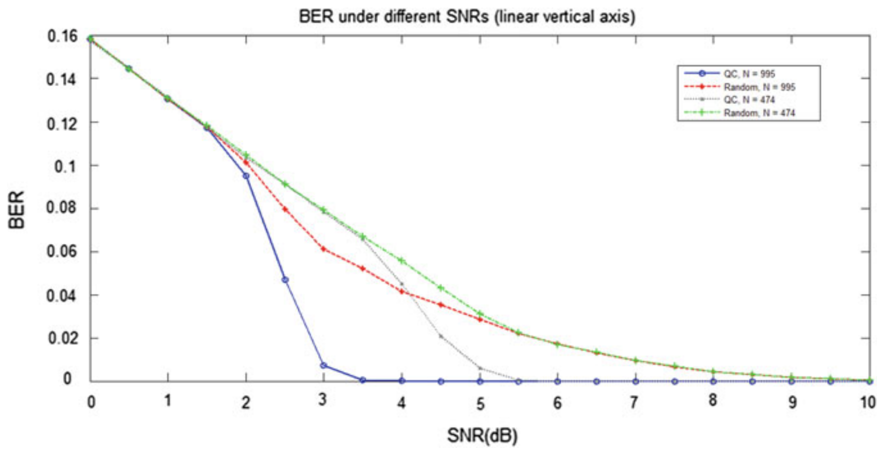

Fig. 3 237 × 474 QC versus random LDPC, and 597 × 995 QC versus random LDPC

Table 2 Comparison of code rates with and without expansion (Gray code construction method)

Row weight	Column weight	Code length (before)	Code length (after)	Code rate (before)	Code rate (after)
3	2	36	612	0.77	0.52
3	2	44	732	0.81	0.54
3	2	20	372	0.60	0.48
3	2	80	1272	0.90	0.56

7 Conclusion

This paper focused on some of the construction methods of LDPC codes and included their BER results. Comparison and suitability of these methods was briefly discussed. Any of these methods can be used to produce both regular and irregular codes (after expansion). It maybe possible to obtain more efficient and higher girth codes with any of these methods on further experimentation.

References

1. Wikipedia: https://en.wikipedia.org/wiki/Error_detection_and_correction.
2. R.G. Gallager: *Low-Density Parity-Check Codes*. MIT Press, Cambridge, 1963.
3. Sarah Johnson: *Introducing Low-Density Parity-Check Codes*. v 1.1, ACoRN Spring School.
4. http://ita.ucsd.edu/wiki/index.php?title=File:Tanner_graph_example.png.
5. Gabofetswe Alafang Malema: *Low-Density Parity-Check Codes: Construction and Implementation*. The University of Adelaide, November 2007.
6. Gabofetswe Malema, Michael Liebelt: *High Girth Column-Weight-Two LDPC Codes Based on Distance Graphs*. EURASIP Journal on Wireless Communications and Networking, Volume 2007, Article ID 48158.
7. Geoffrey Exoo: *A Simple Method for Constructing Small Cubic Graphs of Girths 14, 15, and 16*. The Electronic Journal of Combinatorics 3 (1996), #R30.
8. Chun-Ming Huang, Jen-Fa Huang, Chao-Chin Yang: *Construction of Quasi-Cyclic LDPC Codes from Quadratic Congruences*. IEEE Communications Letters, Vol. 12, No. 4, April 2008.
9. Mrs. Vibha Kulkarni, Dr. K. Jaya Sankar: *Design of Structured Irregular LDPC Codes from Structured Regular LDPC Codes*. 978-1-4799-4445-3, 2015, IEEE.
10. Rakesh Sharma, Ashish Goswami: *A Robust Approach for Construction of Irregular LDPC Codes*. Proc. of the International Conference on Future Trends in Electronics and Electrical Engineering - FTEE 2013, ISBN: 978-981-07-7021-1, doi:10.3850/ 978-981-07-7021-1_69.