Study of LDPC Codes

Reetu Singh¹, Aparna P²
¹,²Electronics and Communication, MVJ College of engineering, Bangalore

ABSTRACT
This paper is a brief study of a specific type of forward error correction code known as low density parity check code. We also study its decoding technique which makes it suitable for transmitting a message over a noisy transmission channel.

Keywords: LDPC code, Hard decision decoding, Soft decision decoding.

I. INTRODUCTION
Low density parity check (LDPC) codes form a class of Shannon limit or capacity-approaching error-correcting codes, these codes were first proposed by Gallager in 1962 in his doctoral thesis [3]. Unfortunately Gallager remarkable discovery was forgotten for almost 20 years due to computational complexity. LDPC codes were rediscovered by Mackay when codes on graph and iterative decoding come in picture.

II. REPRESENTATION OF LDPC CODE
Low Density Parity Check codes are a class of linear block error correcting codes. They can be specified by either a generator matrix G or Parity check matrix H. In terms of parity check matrix H, the code is simply the null space of parity check matrix H. An n-tuple v=(v₁,v₂,.........,vₙ) over GF(2) is a codeword if and only if vHᵀ=0. This states that the codeword in c must satisfy a set of parity-check equations specified by the rows of H[3][4].

Parity Check Matrix H has the following structural properties:
- Each row consists of dᵣ one’s (1’s)
- Each column consists of dᵥ one’s (1’s)
- Both dᵥ and dᵣ are small compared to the length of the code and the number of rows in parity check matrix H, therefore matrix H has a small density of 1’s, hence H is said to be low density parity check matrix.

Density of the parity check matrix for a (n,k) can be given as

\[ r=d_v/m=d_c/n \text{ (where } m=n-k) \]

Rate R of the code is represented as follows for LDPC code

\[ R=k/n \text{ (for a (n,k)} \]

III. GRAPHICAL REPRESENTATION
A convenient way to describe an LDPC code is in terms of its factor graph. This graph is also known as Tanner graph. This is a natural bipartite graph defined as follows. On the left side of the graph there are n vertices, called variable nodes, one for each codeword position. On the right side of the graph there are \( m = n - k \) vertices, called check nodes, one for each parity check (row of the parity check matrix). Each codeword bit corresponds to a variable node, labeled \( x_i \), and each parity bit corresponds to a check node, labeled \( c_j \). Each entry \( h_{ij} \) of 1 in the parity check matrix H corresponds directly to an edge (or connection) between variable node \( i \) and check node \( j \) in its graphical representation [8]. The H matrix of a regular LDPC code contains a fixed number of 1’s in each column, denoted by \( d_v \), and a fixed number of 1’s in each row, denoted by \( d_c \).

![Fig. 1. The setup of a classical encrypted communication channel](image)

A. Regular LDPC code
For a regular LDPC code, the number of edges connected to a check node and the number of edges connected to a variable node are constant.

B. Irregular LDPC code
An irregular LDPC code, on the other hand, has nodes of varying degrees defined by a degree distribution.
IV. LDPC CODE CONSTRUCTION

Several different algorithms exist to construct suitable LDPC codes. Gallager himself introduced one. Furthermore MacKay proposed one MacKay to semi-randomly generate sparse parity check matrices. This is quite interesting since it indicates that constructing good performing LDPC codes is not a hard problem. In fact, completely randomly chosen codes are good with a high probability. The problem that will arise is that the encoding complexity of such codes is usually rather high.

V. DECODING OF LDPC CODE

There are various algorithms to decode LDPC code such as majority logic decoding (MLG), bit flipping decoding (BF), weighted bit flipping decoding, a posteriori probability (APP) decoding and iterative decoding based on belief propagation. Among them iterative decoding based on belief propagation provide best error performance [2].

The main reason for this name is that messages are passed from check nodes to variable nodes, and from variable nodes back to check nodes. An important aspect is that the message that is sent from a variable node to a check node c must not involve the message sent in the previous step from c to v. (means in iteration a node should pass only the extrinsic messages). This is true for messages passed from check nodes c to variable node v.

The messages from the variable node to check node and check node to variable node represent probabilities or beliefs. The algorithm is also known as belief propagation and the LDPC codes can be decoded using iterative belief propagation (BP). In detail, the message passed from check node to variable node is the probability or belief that variable node has certain information (values) given all the messages passed to check node in the previous step from variable nodes other than v (intrinsic information at the variable node). On the other hand, the message passed from variable node to check node is the probability or belief that check node has certain information given all the messages passed to variable node in the previous step from variable nodes other than c (intrinsic information at the check node). It is easy to work with likelihoods, or even log-likelihoods, instead of using probabilities to represent messages. In BP, likelihood functions are recursively computed by each node in the graph, and a message containing this information is transmitted along each edge [7].

To explain this algorithm with a simple example, a very simple variant which works with hard decision, will be introduced first. Later on, the algorithm will be extended to work with soft decision which generally leads to better decoding results. For channel we consider a Binary symmetric channel.

VI. HARD-DECISION DECODING

The algorithm will be explained for the (8,4) code, whose parity check matrix and Tanner graph is represented in figure: 1. Let assume an error free received codeword would be e.g. \( c = [1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1] \). Let’s suppose that we have a BSC channel and the received codeword with one error bit – say bit x1 flipped to 0 [5].

A. First Step

In the first step (also known as zeroth iteration) all variable-nodes xi send a “message” to their check-nodes c j containing the bit they believe to be the correct one for them. At this stage the only information variable-node xi has, is the corresponding received i-th bit of received message x. That means for example, that variable node x1 sends a message containing 0 to check node c1 and c3, similarly variable node x2 sends received messages bit to check node c1 and c2, in similar fashion all the variable nodes sends the received message to check nodes whom they are connected through the edges, in the zeroth iteration.

| TABLE 1. RECEIVED AND SENT MESSAGES AT CHECK NODES |
|------------|------------|-----------|-----------|
| Check-node | Received   | Sent      |
| C1         | x2=0       | x4=1      | x5=0      | x8=1 |
|            | 1→x2       | 1→x4      | 0→x5      | 1→x8 |
| C2         | x1=0       | x3=0      | x5=0      | x7=1 |
|            | 1→x1       | 1→x3      | 1→x5      | 0→x6 |
| C3         | x3=0       | x5=1      | x7=0      | x8=1 |
|            | 0→x3       | 1→x5      | 0→x7      | 1→x8 |
| C4         | x1=0       | x5=1      | x7=0      | x8=0 |
|            | 1→x1       | 0→x5      | 1→x7      | 1→x8 |

B. Second Step

In the second step every check node c j calculates a response to every connected variable node. The response message contains the bit that c j believes to be the correct one for this variable-node xi assuming that the other v-nodes connected to c j are correct. In other words: If you look at the example, every check-node c j is connected to 4 variable-nodes. So a check-node c j looks at the message received from three variable-nodes and calculate the bit that the fourth variable-node should have in order to fulfill the parity check equation. Table 2 illustrate this.
TABLE II. MESSAGES RECEIVED AT VARIABLE NODE

<table>
<thead>
<tr>
<th>Variable Node</th>
<th>message received</th>
<th>Message from check nodes</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>0</td>
<td>c₂→1, c₃→1</td>
<td>1</td>
</tr>
<tr>
<td>x₂</td>
<td>0</td>
<td>c₁→1, c₂→1</td>
<td>0</td>
</tr>
<tr>
<td>x₃</td>
<td>0</td>
<td>c₁→1, c₃→1</td>
<td>0</td>
</tr>
<tr>
<td>x₄</td>
<td>1</td>
<td>c₁→1, c₂→0</td>
<td>0</td>
</tr>
<tr>
<td>x₅</td>
<td>0</td>
<td>c₁→0, c₂→1</td>
<td>1</td>
</tr>
<tr>
<td>x₆</td>
<td>1</td>
<td>c₁→0, c₃→1</td>
<td>1</td>
</tr>
<tr>
<td>x₇</td>
<td>0</td>
<td>c₂→0, c₃→1</td>
<td>0</td>
</tr>
<tr>
<td>x₈</td>
<td>1</td>
<td>c₁→1, c₃→1</td>
<td>1</td>
</tr>
</tbody>
</table>

Important is, that this might also be the point at which the de-coding algorithm terminates. This will be the case if all check equations are fulfilled. We will later see that the whole algorithm contains a loop, so another possibility to stop would be a threshold for the amount of loops.

C. Third Step
The variable-nodes receive the messages from the check nodes and use this additional information to decide if their originally received bit is OK. A simple way to do this is a majority vote. When coming back to our example that means, that each variable-node has three sources of information concerning its bit. The original bit received and two suggestions from the check nodes, now the variable node can send another message with their (hard) decision for the correct value to the check nodes.

D. Forth Step
Go to step 2.

In our example, the second execution of step 2 would terminate the decoding process since x₁ has voted for 0 in the last step. This corrects the transmission error and all check equations are now satisfied.

VII. SOFT DECISION DECODING

In soft decision decoding messages are expressed in probabilities, LLRs form LDPC codes can be iteratively decoded using the iterative decoding with belief propagation also known as Sum-Product Algorithm (SPA), which is a general-purpose algorithm for probabilistic inference based on factor graphs. The variable-node and check-node operations of SPA can be generalized in Equation (1) and (2), where \( L_i^0 \) represents the initial received LLR at variable node i, \( \lambda_{i→j} \) represents an LLR from variable node i to check node j, \( L_{j→i} \) represents an LLR from check node j to variable node i, dc denotes degree of the current check node, and dv denotes the degree of the current node variable [1].

\[
\lambda_{i→j} = L_i^0 + \sum_{k=1}^{dv} L_{k→i} 
\]

The equations 1: To update variable node value

The equations 2: To update check node value

The equations 3: variable node value after nth iteration

\[
F_{j→i} = 2 \left( \frac{1}{2} \right) \left( \frac{\lambda_{k→j}}{2} \right)
\]

\[
\lambda_{i→j}^n = L_i^0 + \sum_{k=1}^{dv} L_{k→i}
\]

The message passing decoding procedure can be described as the following steps. Here also the extrinsic principle of excluding self message is applied at both the check-node and variable node operations:

1) Set \( n = 0 \). Initialize LLR \( L_i^0 \)
2) Variable node LLRs are passed along all edges to the connected check nodes as inputs
3) At the check node, the outgoing LLR sent from the current check node j along the edge to each connected variable node i is computed using equation 2
4) At the variable node, the outgoing LLR sent from the current variable node i to each connected check node j is computed using equation 1
5) The result for the current decoding iteration n is \( \lambda_i^n \), the sum of the result of the variable node computations, computed using the equation 3
6) If \( \lambda_i^n \) satisfies the constraint \( H_i \), \( H_i^T = 0 \) or a fixed number of iterations are reached, terminate decoding and output the current result. If neither condition is met, set \( n = n+1 \) and return to step 2 to start the next iteration.

Although this algorithm is able to achieve excellent bit-error-rate performance, but the implementation of this is complex and requires more hardware resources.

VIII. APPLICATIONS

LDPC codes have already been adopted in satellite-based digital video broadcasting and long-haul optical communication standards. In 2003, an LDPC code beat six turbo codes to become the error correcting code in the new DVB-S2 standard for the satellite transmission of digital television. In 2008, LDPC beat convolutional turbo codes as the forward error correction (FEC) system for the ITU-T G.hn standard. The G.hn specifications
define networking over power lines, phone lines and coaxial cables with data rates up to 1 Gbit/s[6].

LDPC are highly likely to be adopted in the IEEE wireless local area network standard, and are under consideration for the long-term evolution of third generation mobile telephony. As of 2009, LDPC codes are also part of the Wi-Fi 802.11 standard as an optional part of 802.11n and 802.11ac, in the High Throughput (HT) PHY specification. LDPC is also used for 10GBase-T Ethernet, which sends data at 10 gigabits per second over twisted-pair cables.

IX. CONCLUSION

Iterative decoding by Belief Propagation is very efficient algorithm to decode both binary and non-binary LDPC code but it takes much iteration to compute the code, which can be improved by modifying the method to update variable and check node messages. Variable nodes and check nodes exchange messages according to a predetermined schedule. A scheme of determining the update order of extrinsic messages (edge messages) is called a scheduling scheme. This affects the convergence speed based on the iterations of the decoder. There are two scheduling schemes, which are the standard message passing schedule and the layered decoding schedule. Layered decoding converges faster than the standard message passing (Flooding Schedule) algorithm [7].

REFERENCES