Error Correcting Codes: Combinatorics, Algorithms and Applications **(Fall 2007)**

Lecture 33: LDPC Codes

November 12, 2007

Lecturer: Atri Rudra *Scribe: Michel Kulhandjian*

Low Density Parity Check (LDPC) codes were first introduced by Gallager [1] in his thesis. After the invention, these codes were largely forgotten and there was "no activity" till mid-late 90s when there was a renewed spurt of research activity on LDPC codes. Available resource for the state-of-the-art in LDPC codes can be found in Richardson-Urbanke book [4]. For a quick overview, see the survey by Guruswami [2]. In this lecture, we are going mainly to follow the survey by Guruswami. Unlike other classes of codes that we have seen, LDPC codes are accompanied by very fast encoding and decoding algorithms. For example, we have seen that the capacity of the BSC_p can be achieved with decoding time complexity $\text{poly}(n)2^{\text{poly}(\frac{1}{\varepsilon})}$. We want the decoding time ideally to be $n\text{poly}(\frac{1}{\varepsilon})$. LDPC codes provably achieve this though for BEC_α (Luby *et al* [3]). This makes LDPC codes not only attractive from the theoretical point of view, but also for practical applications. However, we will not discuss LDPC codes of Luby *et al* in this lecture but go through some of Gallager's results and some brief idea how his ideas extended. The name of the code suggest that it has to do with parity check codes.

1 Low Density Parity Check Codes

Recall that any $[n, k]_2$ code has an $(n-k) \times n$ parity check matrix H. Every parity check matrix H can be associated with a *factor graph* $\mathcal{G}$. In particular $\mathcal{G}$ is a bipartite graph that has n left vertices, called *variable* nodes, one for each codeword position. On the right are $n-k$ vertices, called *check* nodes, one for each parity check (or row in H). A check node is adjacent to all variable nodes whose corresponding codeword symbols appear in the corresponding row in H. In other words, H is precisely the adjacency matrix of $\mathcal{G}$.

Let us fix some notation that we will use later. Recall that a code C with parity check matrix H is defined as follows: for every $\mathbf{c} \in C$, $H\mathbf{c}^T = \mathbf{0}$. In other words there are $(n-k)$ constraints on $(c_1, \ldots, c_n)$ variables, where c_i denotes the ith bit in a codeword. Let the jth $(1 \leq j \leq n-k)$ parity check (or equivalently $H_j(c_1, \ldots, c_n)^T$) be denoted by p_j. There is an edge (c_i, p_j) in $\mathcal{G}$ if c_i occurs in p_j or equivalently $H_{ji} = 1$, where H_{ji} is the entry corresponding to row j and column i in H. For example $[7, 4, 3]_2$ Hamming code has a 3×7 parity check matrix H:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

and the corresponding factor graph of H is:

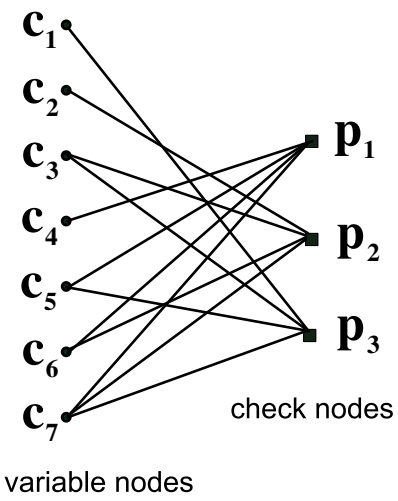


Figure 1: Factor graph of the Hamming code

The number of edges equals the number of ones in the parity check matrix. According to definition of sparse bipartite graph, the number of ones in parity check matrix H is bounded by $O(n)$. Equivalently the number of edges in the factor graph is bounded by $O(n)$.

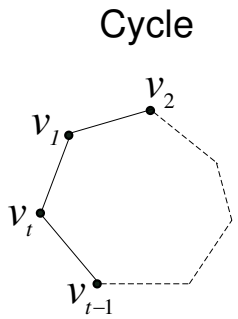
Definition 1.1. *A special class of LDPC codes are regular LDPC codes where the factor graph has left vertexes with degree exactly d_v and every right vertexes with degree exactly d_c for integers $d_v, d_c \geq 1$.*

This regular LDPC codes were originally studied by Gallager. We claim that the rate R of (d_v, d_c)-regular LDPC code is equal to

$$R = 1 - \frac{d_v}{d_c}.$$

This is because the number of edges of factor graph is equal to $d_v n = d_c(n-k)$. Note that this implies $d_c > d_v$ for positive R. We will consider families of regular LDPC codes that are obtained by constructing a family of (d_v, d_c)-regular factor graphs with n variable nodes with girth greater than $\Omega(\log n)$. We will also need the following graph-theoretic notion.

Definition 1.2. *Girth of a graph is the size of the smallest cycle.*



In this graph $v_1, \ldots, v_t$ is a cycle.

Gallager proved in the following results [1]:

(i) With high probability, for large enough d_v and d_c, a random (d_v, d_c)-regular LDPC code achieves the GV bound (for large enough n).

(ii) Random (d_v, d_c)-regular LDPC codes (along with MLD) get "close" to BSC_p capacity.

Since only exponential time solutions to the MLD function are known, Gallager also developed simple, iterative decoding algorithm for LDPC codes. We are not going to look (i) or (ii) and going to study the performance of regular LDPC codes over the BEC_α.

1.1 Decoding on the BEC_α

Although Gallager did not explicitly study BEC_α, his methods do apply to it. For the BEC_α we will study an iterative message-passing decoding algorithm. General idea of such a decoding algorithm is to pass messages in round. Each round has two phases. In the first phase messages are passed from variable nodes to check nodes (of the factor graph). In the second phase, messages are passed back from the check nodes to the variable nodes. Let the received word be $\mathbf{y} = (y_1, \ldots, y_n) \in \{0, 1, ?\}^n$ and for each round messages are sent as follows:

(i) variable-to-check message phase: If (c_i, p_j) is an edge then c_i sends p_j a message. In particular, if $y_i \neq ?$ then c_i passes along y_i to all its neighboring check nodes.

(ii) check-to-variable message phase: If (p_j, c_i) is an edge, the p_j sends c_i a message. In the particular, if the check node p_j knows the correct value for c_i, it passes the value to c_i.

In particular, the messages will have the following semantics:

1. If variable c_i knows its correct value then the message is the value, else the message is an erasure.

2. If check node p_j knows the value of c_i then passes that value, else it passes an erasure.

At end (hopefully) every c_i knows its value with high probability. The message-passing decoding algorithm has the following *extrinsic* property: the message that is sent from, c_i to p_j does not depend on message sent by p_j in the previous round (a similar property holds for messages sent from the variable nodes to the parity check nodes). Formally the message maps are given as follows, message $c_i \to p_j$ in round t is:

$$\Psi_{c_i}^{t,p_j}(y_i, m_1^{t-1}, \ldots, m_{d_v-1}^{t-1}) \to \{0, 1, ?\},$$

where $m_1^{t-1}, \ldots, m_{d_v-1}^{t-1}$ are the $d_v - 1$ messages received by c_i from all its neighboring check nodes other than p_j, in the previous round. Generally the parameter p_j is dropped because the mapping is the same for every p_j. The mapping of message $p_j \to c_i$ in round t is:

$$\Psi_{p_j}^{t,c_i}(m_1^t, \ldots, m_{d_c-1}^t) \to \{0, 1, ?\},$$

where again $m_1^t, \ldots, m_{d_c-1}^t$ are the $d_c - 1$ messages received from all the variable nodes other than c_i in the current round (recall that the check node to variable nodes phase takes place after variable-to-check node phase in the same round). Similarly for check nodes the parameter c_i is dropped because generally it is the same for every c_i.

We now define the message functions. In round $t = 1$, message $c_i \to p_j$ is computed as follows:

$$\Psi_{c_i}^{0,p_j}(y_i, m_1^0, \ldots, m_{d_v-1}^0) = y_i$$

and for $t \geq 2$ we have:

$$\Psi_{c_i}^{t,p_j}(y_i, m_1^{t-1}, \ldots, m_{d_v-1}^{t-1}) = \begin{cases} b & \text{, if at least one of } \{y_i, m_1^{t-1}, \ldots, m_{d_v-1}^{t-1}\} \text{ is b } \in \{0,1\} \\ ? & \text{, if } y_i = m_1^{t-1} = \ldots = m_{d_v-1}^{t-1} = ? \end{cases}$$

and message $p_j \to c_i$ when $t \geq 1$ we have:

$$\Psi_{p_j}^{t,c_i}(m_1^t, \ldots, m_{d_v-1}^t) = \begin{cases} ? & \text{, if any one of } m_i^t = ? \\ m_1^t \oplus \ldots \oplus m_{d_c-1}^t & \text{, otherwise} \end{cases}$$

Recall that the codewords are those vectors $(c_1, \ldots, c_n)$ such that for all check nodes the sum of the neighboring positions among the message nodes is zero, i.e.

$$c_i \oplus m_1^{t-1} \oplus \ldots \oplus m_{d_c-1}^{t-1} = 0.$$

In other words if $\Psi_{c_i}^{t,p_j}(\cdot)$ only transmits correct values in $\{0, 1\}$, then $\Psi_{p_j}^{t,c_i}(\cdot)$ computes the correct bit for c_i if all its inputs are in $\{0, 1\}$.

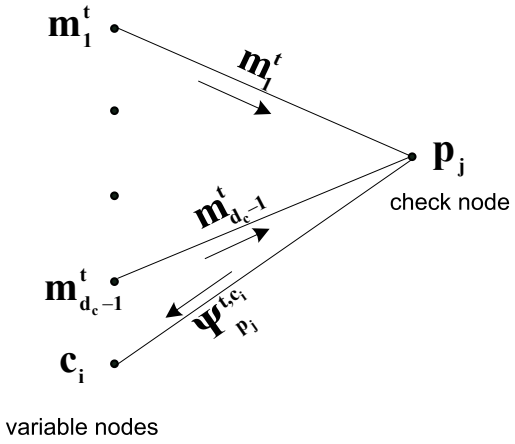


Figure 2: Message computation for c_i

References

[1] R.G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, Cambridge, MA, 1963.

[2] V. Guruswami. Iterative decoding of low-density parity check codes (a survey). *Bulletin of the EATCS*, September 2006.

[3] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.

[4] T. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2007.