Modern Communications
Chapter 5. Low-Density Parity-Check Codes

Husheng Li

Min Kao Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville

Spring, 2017
History of LDPC

- LDPC was invented by Prof. Gallager in his 1960 PhD dissertation.
- It resurrected in mid 1990s with the effort of MacKay, Luby, et al. Iterative decoding algorithm is found to implement LDPC.
- LDPC is the most powerful error corrected code, and is significant from traditional algebraic codes.
Representation of LDPC: Matrix

- We consider only binary LDPC.
- A LDPC is a linear block code given by the null space of an $m \times n$ parity check matrix $H$ that has a low density of 1s.
- Regular and irregular LDPCs.
- Low density: $\sim 1\%$
- Code rate:

\[
R \geq 1 - \frac{m}{n}.
\]
A LDPC can be represented by a Tanner graph, which is a bipartite graph.

There are two types of nodes: the variable nodes and the check nodes.
Cycles and Degree

- We are interested in cycles because short cycles will force the decoder to operate locally in some portions of the graph and thus prevent the global optimum.

- It is possible to more closely approach capacity limits with irregular LDPC codes than with regular LDPC codes. We can define the degree distribution polynomials:

\[ \lambda(X) = \sum_{d=1}^{d_0} \lambda_d X^{d-1}. \]
The original LDPC is random since its parity check matrix has little structure. Modern LDPC codes have much more structures.

Three structures:

- Cyclic: easy to encode; regular; complicated for decoding.
- Quasi-cyclic: tremendous structure, leading to simplify the encoder and decoder designs.
- Random but linear.
The key innovation behind LDPC is the low-density nature of the parity-check matrix, which facilitates iterative decoding.

The so-called sum-product algorithm (SPA) is a generic algorithm that provides near-optimal performance across a broad class of channels.

We can consider an LDPC to be a generalized concatenation of many single parity check codes (SPC). Then, the SPA make the individual decoders cooperate in a distributed manner.
Example of Decoding

(a) $\begin{array}{ccc} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{array}$

(b) $H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

(c) Row checks (row decoder)

Codeword (received word)

Column checks (column decoder)

(d) $\begin{array}{ccc} e & e & 1 \\ 1 & e & 0 \\ 0 & e & e \end{array}$

Received word

$\begin{array}{ccc} e & e & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & e & e \end{array}$

After row decoding

$\begin{array}{ccc} 1 & e & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & e & 1 \end{array}$

After column decoding

$\begin{array}{ccc} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{array}$

After row decoding
The message passing is optimal in cycle-free graphs, but sub-optimal in graphs with cycles.

This is illustrated by the soldier-counting problem.
The optimality criterion underlying the development of SPA decoder is the symbol-wise MAP.

The SPA is a distributed version of calculating the a posteriori probabilities.

LDPC can be considered as a collection of SPC codes concatenated through an interleaver to a collection of repetition codes.
We can consider the decoders of VN and CN working cooperatively.
The Gallager SPA Decoder

Algorithm 5.1 The Gallager Sum-Product Algorithm

1. **Initialization**: For all $j$, initialize $L_j$ according to (5.19) for the appropriate channel model. Then, for all $i, j$ for which $h_{ij} = 1$, set $L_{j-i} = L_j$.
2. **CN update**: Compute outgoing CN messages $L_{i-j}$ for each CN using (5.18),
   \[
   L_{i-j} = 2 \tanh^{-1} \left( \prod_{j' \in N(i)-\{j\}} \tanh \left( \frac{1}{2} L_{j'-i} \right) \right),
   \]
   and then transmit to the VNs. (This step is shown diagrammatically in Figure 5.8.)
3. **VN update**: Compute outgoing VN messages $L_{j-i}$ for each VN using Equation (5.11),
   \[
   L_{j-i} = L_j + \sum_{i \in N(j)-\{i\}} L_{i-j},
   \]
   and then transmit to the CNs. (This step is shown diagrammatically in Figure 5.7.)
4. **LLR total**: For $j = 0, 1, \ldots, n - 1$ compute
   \[
   L_j^{\text{total}} = L_j + \sum_{i \in N(j)} L_{i-j},
   \]
5. **Stopping criteria**: For $j = 0, 1, \ldots, n - 1$, set
   \[
   \hat{v}_j = \begin{cases} 
   1 & \text{if } L_j^{\text{total}} < 0, \\
   0 & \text{else,}
   \end{cases}
   \]
   to obtain $\hat{v}$. If $\hat{v}^T \mathbf{H} = 0$ or the number of iterations equals the maximum limit, stop; else, go to Step 2.

- The message passing forms the core of the Gallager SPA decoder.
- We also need an iteration-stopping criterion and initialization step.
- We apply the approach in the BEC, BSC and BI-AWGNC channels.
Implementation of the function $\phi(x)$ by use of a look-up table is insufficient for most hardware applications, due to dynamic range issues.

We can change the check-node processing using a slightly different perspective.
Reduce-Complexity SPA Approximation

- The MIN-SUM Decoder: We replace the sum with the maximum term that corresponds to the smallest $\beta_{ji}$.
- The attenuated and offset min-sum decoders: The min-sum decoder is too optimistic in assigning reliabilities.
- The min-sum-with-correction decoder: an approximation to the box-plus decoder.
- The approximate min* decoder: this can reduce the operations over the check nodes.
Decoders for BEC and BSC

- Iterative Erasure Filling: The iterative decoding algorithm for the BEC simply works to fill in the erased bits in such a way that all of the check equations contained in $H$ are eventually satisfied.

- ML Decoder for BEC.

- Gallager’s Algorithms A and B for BSC

- The Bit-Flipping Algorithm for the BSC