LDPC Codes – Part 2

- Repeat codes
- LDPC codes (repetition)
- Irregular LDPC codes
- Density Evolution
- EXIT charts
- Encoding of LDPC codes
- Miniproject update
Repeat Codes

• The repeat code of length $N$ has only two codewords:

$$\mathcal{C} = \{[0 \ldots 0], [1 \ldots 1]\}$$

• A-posteriori probabilities

$$p_{p,n}^0 = \alpha \cdot \prod_{i=0}^{N-1} p_{ch,i}^0,$$
$$p_{p,n}^1 = \alpha \cdot \prod_{i=0}^{N-1} p_{ch,i}^1$$

• Extrinsic probabilities

$$p_{e,n}^0 = \alpha \cdot \prod_{i=0}^{N-1} p_{ch,i}^0,$$
$$p_{p,n}^1 = \alpha \cdot \prod_{i=0}^{N-1} p_{ch,i}^1$$
Definition
A regular \((d_v, d_c)\) LDPC code of length \(N\) is defined by a parity-check matrix \(H \in \mathbb{F}_2^{M \times N}\), with \(d_v\) ones in each column and \(d_c\) ones in each row. The dimension of the code (info word length) is \(K = N - \text{rank } H\).

Example 1

\[
H = \begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

Parameters: \(d_v = 3, d_c = 4, N = 8, M = 6, K = 4\) (!), \(R = 1/2\) (!)
**Regular LDPC Codes**

**Design Rate**

The true rate $R$ and the design rate $R_d$ are defined as

\[ R := \frac{K}{N}, \quad R_d := 1 - \frac{d_v}{d_c}, \]

and they are related by

\[ R \geq R_d \]

**Proof**

The number of ones in the check matrix is $Md_c = Nd_v$. Some check equations may be redundant, i.e., $M \geq N - K$, and thus

\[ \frac{K}{N} = 1 - \frac{N - K}{N} \geq 1 - \frac{M}{N} = 1 - \frac{d_v}{d_c} \]
**Factor Graphs**

**Definition**
A factor graph of a code is a graphical representation of the code constraints defined by a parity-check matrix of this code:

$$xH^T = 0$$

The factor graph is a bipartite graph with
- a **variable node for each code symbol**,
- a **check node for each check equation**,
- an edge between a variable node and a check node if the code symbol participates in the check equation.

Notice that each edge corresponds to one 1 in the check matrix.
Example (cont.)

\[ xH^T = 0 \]

\[ x_0 + x_3 + x_4 + x_5 = 0 \text{ (chk}_0) \]
\[ x_0 + x_2 + x_4 + x_5 = 0 \text{ (chk}_1) \]
\[ x_0 + x_2 + x_3 + x_5 = 0 \text{ (chk}_2) \]
\[ x_1 + x_3 + x_6 + x_7 = 0 \text{ (chk}_3) \]
\[ x_1 + x_4 + x_6 + x_7 = 0 \text{ (chk}_4) \]
\[ x_1 + x_2 + x_6 + x_7 = 0 \text{ (chk}_5) \]
Irregular LDPC Codes

- Generalization of regular LDPC codes
- Lower error rates, i.e., better performance
- Irregular number of ones per column and per row
- Variable nodes of different degrees
- Check nodes of different degrees
Irregular LDPC Codes

Example

\[ H = \begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}^T \]

Slides originally from I. Land – p.8
Irregular LDPC Codes

- Irregular number of ones per column and per row:
  \( l_i \): proportion of left (variable) nodes of degree \( i \)
  \( r_i \): proportion of right (check) nodes of degree \( i \)

- In example:
  \( l_3 = 5/8, l_4 = 1/8, l_5 = 2/8 \),
  \( r_4 = 3/6, r_5 = 1/6, r_6 = 2/6 \)

- Proportions of edges:
  \( \lambda_i \): proportion of edges incident to left nodes of degree \( i \)
  \( \rho_i \): proportion of edges incident to right nodes of degree \( i \)

- In example:
  \( \lambda_3 = 15/29, \lambda_4 = 4/29, \lambda_5 = 10/29 \),
  \( \rho_4 = 12/29, \rho_5 = 5/29, \rho_6 = 12/29 \)
Irregular LDPC Codes

Relation:

\[ \lambda_i = \frac{i \ l_i}{\sum_j j \ l_j} \]
\[ \rho_i = \frac{i \ r_i}{\sum_j j \ r_j} \]

Definition of irregular LDPC codes by the degree polynomials defining the degree distributions:

\[ \lambda(z) := \sum_i \lambda_i \ z^{i-1} \]
\[ \rho(z) := \sum_i \rho_i \ z^{i-1} \]

Remark: \( z \) is only a dummy variable
Irregular LDPC Codes

Design Rate

The design rate of irregular LDPC codes can be computed by

\[ R_d = 1 - \frac{\int_0^1 \rho(z) dz}{\int_0^1 \lambda(z) dz} \]

The proof follows the one for regular LDPC codes.

As in the regular case, we have the relation

\[ R \geq R_d \]
The factor graph represents a factorization of the **global code constraint**

$$xH^T = 0$$

Variable nodes and check nodes represent **local code constraints**

This is made explicit by the edge interleaver
Message Passing Algorithm

- LDPC codes can be iteratively decoded on the factor graph:
  - Nodes perform local decoding operations
  - Nodes exchange extrinsic soft-values called messages
  - Iteration is terminated when code symbol estimates form a valid codeword, i.e., if $\hat{x}H^T = 0$

Advantages

- The overall decoding complexity is only linear (!) in the code length
- The decoder performs close to the ML decoder

But nevertheless: This decoding algorithm is sub-optimal due to cycles in the graph
Messages and Node Operations

- Initial message to variable node $X_n$:

$$\mu_{ch}(n) = \text{fct}(y_n)$$

- Message from variable node $X_n$ to check node $\text{chk}_m$:

$$\mu_{vc}(n, m) = \text{fct}(\mu_{ch}(n), \mu_{cv}(m', n) : m' \in \mathcal{M}(n) \setminus m)$$

- Message from check node $\text{chk}_m$ to variable node $X_n$:

$$\mu_{cv}(m, n) = \text{fct}(\mu_{vc}(n', m) : n' \in \mathcal{N}(m) \setminus n)$$

- Variable-to-check messages and check-to-variable messages are **extrinsic** messages.
Messages and Node Operations (cont.)

- **Final message generated by variable node** $X_n$:

  $$\mu_{\nu}(n) = \text{fct}(\mu_{\text{ch}}(n), \mu_{\text{cv}}(m', n) : m' \in \mathcal{M}(n))$$

- **Estimate for code symbol** $X_n$

  $$\hat{x}_n = \begin{cases} 
  0 & \text{if } \Pr(X_n = 0|\mu_{\nu}(n)) > \Pr(X_n = 1|\mu_{\nu}(n)) \\
  1 & \text{otherwise}
  \end{cases}$$
**Optimality, Cycles, and Girth**

- Nodes assume that incoming messages are independent.

- **Independency assumption** holds only for first few iterations due to cycles in the graph.

- Length of smallest cycle is called the **girth**.

- For $N \to \infty$, the girth tends to $\infty$ and the MPA becomes optimal.
MPA based on L-values I

The symbol $\mu$ (for message) is commonly replaced by $l$ (for L-value) when the messages are L-values.

- **Initial messages:** $l_{ch}(n) = L(X_n|y_n)$
- **Variable-node operation**
  
  $$l_{vc}(n, m) = L(X_n \mid l_{ch}(n), l_{cv}(m', n) : m' \in M(n)\setminus m)$$
  
  $$= l_{ch}(n) + \sum_{m' \in M(n)\setminus m} l_{cv}(m', n)$$

- **Check-node operation**
  
  $$l_{cv}(m, n) = L(X_n \mid l_{vc}(n', m) : n' \in N(m)\setminus n)$$
  
  $$= \sum_{n' \in N(m)\setminus n} l_{vc}(n', m)$$
MPA based on L-values II

- Final variable-node operation
  \[ l_v(n) = L(X_n|l_{ch}(n), l_{cv}(m', n) : m' \in \mathcal{M}(n)) \]
  \[ = l_{ch}(n) + \sum_{m' \in \mathcal{M}(n)} l_{cv}(m', n) \]

- Hard decision
  \[ \hat{x}_n = \begin{cases} 
    0 & \text{if } l_v(n) > 0 \\
    1 & \text{if } l_v(n) < 0 
  \end{cases} \]

- Termination criterion
  \[ \hat{x}H^T = 0 \]
Some Remarks

- The MPA gives the a-posteriori probabilities (APPs) of the code symbols if the graph is cycle free.

- Otherwise, it approximates decoding for the maximum-likelihood (ML) codeword.

- For the BEC, the MPA may get stuck: A set of code symbols which is not resolvable is called a stopping set.

- The performance of a code of infinite length can be determined by tracking the evolution of the probability density function of the messages with respect to the iterations. This method is called density evolution.
Decoding Threshold

Consider a regular \((d_v, d_c)\) LDPC code with code length \(N \to \infty\).

Consider communication over a BEC, which is completely described by its erasure-probability \(\delta\).

The decoding threshold for this code on the BEC is the largest erasure-probability \(\delta\) (corresponding to the worst channel) for which decoding with the MPA is error-free.
Concentration Theorems

- We consider the average code performance over all graphs with degrees $d_v$ and $d_c$. (Interpretation: average over all edge interleavers.)

- **Concentration**: the probability that the performance of a specific code diverges by $\epsilon$ from the expected performance over all graphs converges to 0 exponentially in the code length $N$.

Thus: **expected performance suffices for** $N \to \infty$

- **Cycle-free behavior**: for any iteration number $i$, one can choose a code length $N$ so that the expected performance at iteration $i$ is as near as desired to the decoder performance under the independency assumption (tree assumption).

Thus: **independency assumption suffices for** $N \to \infty$
**Reminder: Decoding for the BEC**

**Variable-node decoder**

\[ \mu_{cv} = \Delta \text{ if and only if all incoming messages are } \Delta \]

**Check-node decoder**

\[ \mu_{cv} = \Delta \text{ if and only if at least one incoming message is } \Delta \]
Density Evolution

Variable-node decoder

- Probability of erasure
  \[ p_i := \Pr(\mu_{vc} = \Delta \text{ at } i) \]

- Evolution
  \[ p_0 = \delta \]
  \[ p_{i+1} = p_0 \cdot q_i^{d_{v}-1} \]

- Combination
  \[ p_{i+1} = \delta \left( 1 - (1 - p_i)^{d_c-1} \right)^{d_v-1} \]

Check-node decoder

- Probability of erasure
  \[ q_i := \Pr(\mu_{cv} = \Delta \text{ at } i) \]

- Evolution
  \[ q_i = 1 - (1 - p_i)^{d_c-1} \]
Iterative Decoding will be successful if

\[ p > \delta \left( 1 - (1 - p)^{d_c-1} \right)^{d_v-1} \text{ for all } 0 < p \leq \delta \]

A fixed point is where

\[ p^* = \delta \left( 1 - (1 - p^*)^{d_c-1} \right)^{d_v-1} \]

Iterative decoding will get stuck if there exists a fixed point \( p^* \) with \( 0 < p^* \leq \delta \).
The largest erasure probability $\delta$ (worst channel) for given degrees $(d_v, d_c)$ and rate $R$ such that there is no fixed point $0 < p^* \leq \delta$ is called the decoding threshold $\delta_{th}$ of the code.

For a given code rate $R$, the degrees should be determined such that the decoding threshold is minimal. (Of course.)

Code optimization for large code lengths via DE
For an irregular LDPC code with degree polynomials $\lambda(z)$ and $\rho(z)$, the result is:

Iterative Decoding will be successful if

$$p > \delta \lambda(1 - \rho(1 - p)) \quad \text{for all} \quad 0 < p \leq \delta$$

Using DE, the degree polynomials can be optimized for a given code rate such that the resulting codes are (almost) capacity-achieving.
Generalizations of Density Evolution

- The technique of DE can be generalized to messages with continuous range, like L-values.

- Thus, codes can also be optimized for other channels. For the AWGN channel, the decoding threshold is given in terms of the minimal SNR.

- Density evolution can also be applied if sub-optimal node operations are employed. A typical case is where the approximation for boxplus is used in the check-node decoder (to simplify the operation).