Coding Theory
Lecture 14. LDPC codes and graphs.
Low-denoisty parity-check (LDPC) codes are linear codes whose parity-check matrices contain only a small number of non-zero entries. This class of linear codes invented by R. Gallager in the early 1960s, was almost completely forgotten until its rediscovery by D. Mackay in 1995. At that time, it became clear that LDPC codes are strong competitors of turbo codes. The sparsity of the LDPC code parity-check matrix allows low-complexity iterative decoding of these codes, which achieves performance close to the Shannon limit. Moreover, the LDPC code decoding procedure can be efficiently parallelized. These facts attract attention of many researches to this class of codes and make LDPC codes a good choice for modern communication standards.
Definition
A binary linear \([n, k]\)-code determined by a parity-check matrix \(H\) is called \((J, K)\)-regular LDPC code if each column of \(H\) contains \(J\) ones and each row contains \(K\) ones.

It is assumed that \(J\) and \(K\) are small compared to \(n\) and \(k\).

For sequences of codes of rate \(R = \frac{k}{n}\) when \(n\) and \(k\) grow, \(J\) and \(K\) remain constant.

Code rate can also be expressed via parameters \(J\) and \(K\) as

\[
R \geq 1 - \frac{J}{K}.
\]

which follows from the fact that the total number of nonzero elements in \(H\) is \(nJ = (n - k)K\).

LDPC codes whose parity-check matrices have a different number of ones in their rows and columns are called irregular LDPC codes.
For a wide variety of regular LDPC codes, their parity-check matrix can be represented in the form

\[
H = \begin{pmatrix}
H_1 \\
H_2 \\
\vdots \\
H_J
\end{pmatrix}
\]

where \(H_j\) are permutations of \(H_1\) consisting of columns of weight 1 and rows of weight \(K\).
LDPC codes. Parity-check matrix of the $(J = 3, K = 4)$-regular LDPC code

Example

$$H = \begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\end{pmatrix} \quad (1)$$

determines the $R > 1 - 6/8$ $(3,4)$-LDPC code. Indeed, 
$R = 1 - 4/8 = 1/2$, since the 4-th row is the sum of the 1st, 2d, and 3d rows (1st, 2d, 3d, and 4th rows are linearly dependent) and the 6-th row is the sum of the 3d, 4th, and 5th rows (3d, 4th, 5th, and 6th rows are linearly dependent).
LDPC codes. Parity-check matrix of the \((J = 3, K = 4)\)-regular LDPC code

The matrix (1) can be reduced to the following form

\[
H = \begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{pmatrix}
\]
Definition
A (simple) graph $G$ is determined by a set of vertices $\mathcal{V} = \{v_i\}$ and a set of edges $\mathcal{E} = \{e_i\}$, where each edge is a set with exactly two vertices (the edge connects those two vertices).

Definition
The degree of a vertex denotes the number of edges that are connected to it. If all vertices have the same degree $l$, the degree of the graph is $l$, or, in other words, the graph is $l$-regular.

Definition
If the set of vertices $\mathcal{V}$ of a graph is partitioned into two disjoint subsets $\mathcal{V}_k$, $k = 0, 1$, where $\mathcal{V} = \bigcup_{k=0}^{1} \mathcal{V}_k$ then a graph is called bipartite if no edge connects two vertices from the same set $\mathcal{V}_k$, $k = 0, 1$. 
Definition

A cycle of length \( g \) in a graph is an alternating sequence of \( g + 1 \) vertices \( v_i, \ i = 1, 2, ... g + 1 \), and \( g \) edges \( e_i, \ i = 1, 2, ..., g \) with \( e_i \neq e_{i+1} \) whose first and last vertices coincide, that is, \( v_1 = v_{g+1} \). A cycle is called simple if all its vertices and edges are distinct, except for the first and last vertex, which coincide. The length of the shortest simple cycle is called the girth of the graph.
Definition
The incidence matrix $\mathcal{A}$ of an undirected graph is a binary matrix whose rows correspond to the graph vertices and columns correspond to the graph edges. The matrix entry $\mathcal{A}_{ij}$ is equal to one iff the $i$-th vertex is connected with the $j$-th edge, otherwise $\mathcal{A}_{ij} = 0$.
Parity-check matrices of regular LDPC codes with $J = 2$ can be interpreted as incidence matrices of graphs.
Regular \((J = 2, K)\) LDPC codes have a close relation to the regular graphs. An incidence matrix of the regular graph can be considered as a parity-check matrix of a regular \((J = 2, K)\) LDPC code. The utility graph with incidence matrix:

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 2 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 3 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 4 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 5
\end{pmatrix}
\]
Its incidence matrix is a parity-check matrix of the \([9,3]\) tailbiting LDPC convolutional code determined by

\[
H = \begin{pmatrix}
1 & 1 & 1 \\
1 & D & D^2
\end{pmatrix}
\]
LDPC codes and their graph representations. Heawood graph.
LDPC codes and their graph representations. Heawood graph.

The **Heawood graph** has incidence matrix:

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 3 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 8 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 11 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 12 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 13
\end{pmatrix}
\]

It can be considered as a parity-check matrix of the [21,7] tailbiting convolutional code determined by

\[
\begin{pmatrix}
1 & 1 & 1 \\
1 & D & D^3
\end{pmatrix}
\]
Famous graphs. Balaban’s graph

(105,36,10)- TB code with parent convolutional of rate $R=7/21$
Famous graphs. Tutte’s graph

(189,64,12)-TB code
R=9/27 parent convolutional code
Definition

The \((s + p) \times (s + p)\) adjacency matrix \(B\) of a bipartite graph with \(s\) and \(p\) vertices belonging to the disjoint sets has the form

\[
B = \begin{pmatrix} 0 & C \\ C^T & 0 \end{pmatrix},
\]

where \(C\) is a matrix of size \(s \times p\) and all-zero matrices have sizes \(s \times s\) and \(p \times p\), respectively. The matrix \(C\) is called biadjacency matrix.
Definition

Tanner graph of a linear code determined by the parity-check matrix $H = \{h_{ij}\}$, $i = 1, \ldots, r$, $j = 1, \ldots, n$ is a bipartite graph whose one set of nodes corresponds to the checks of $H$ (check nodes) and the other set of nodes corresponds to the set of code symbols (variable nodes). A check node $c_i$ is connected with a variable node $v_j$ if $h_{ij} \neq 0$.

The code parity-check matrix $H$ can be interpreted as a biadjacency matrix of the code Tanner graph.
Example

Consider a parity-check matrix of the $(J = 2, K = 3)$-regular LDPC code of rate $R = (n - r)/n > 1/3$

$$H = \begin{pmatrix}
V_0 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 & c_0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & c_1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & c_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & c_3 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & c_4 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & c_5 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & \end{pmatrix}$$

The corresponding Tanner graph is shown in Fig. 2.
Example

Figure: Tanner graph of the (2, 3)-LDPC code

The variable node degree is equal to two, the check node degree is equal to three. The girth of this Tanner graph is equal to eight. Notice that in terms of graph theory, the parity-check matrix $H$ can be interpreted as the biadjacency matrix of the Tanner graph.
ML and MAP decoding. Reminder

Maximum a posteriori probability (MAP) decoding rule returns a codeword \( \hat{c} \) maximizing a posteriori probability of \( c \) given channel output \( y \):

\[
\hat{c} = \arg \max_{c \in C} p(c|y)
\]  

(2)

A posteriori probability can be expressed by using Bayes formula as

\[
p(c|y) = \frac{p(y|c)p(c)}{p(y)}.
\]

Denominator does not influence the decision. Therefore, the equivalent rule is

\[
\hat{c} = \arg \max_{c \in C} p(y|c)p(c).
\]  

(3)
ML and MAP decoding. Reminder

Maximum likelihood (ML) decoding rule returns a codeword \( \hat{c} \) maximizing probability of \( y \) over \( c \in C \):

\[
\hat{c} = \arg \max_{c \in C} p(y|c)
\] (4)

Form (3) and (4) we conclude that if a priori probabilities are unknown or uniformly distributed then two rules, ML and MAP are equivalent. In the BSC, the ML decoding rule reduces to

\[
\hat{c} = \arg \min_{c \in C} d(c, y),
\]

where \( d(c, y) \) denotes the Hamming distance between vectors \( c \) and \( y \).
Symbol MAP decoding

Let $C = \{c_m, m = 1, \ldots, M\} \subseteq \{0, 1\}^n$, be a binary block code and $y = (y_1, \ldots, y_n) \in Y^n$ is a channel output sequence.

A posteriori probability of $c \in \{0, 1\}$ at position $t$ is

$$p(c_t = c | y) = \frac{p(c_t = c, y)}{p(y)},$$

where

$$p(c_t = c, y) = \sum_{c \in C_t(c)} p(c, y),$$

and $C_t(c)$ is a set of codewords which have $c$ at position $t$, $p(c, y) = p(y | c)p(c)$. 
Unlike ML-decoding which gives a decision about the transmitted codeword, MAP-decoding gives optimal soft decision about each transmitted symbol.

Let a generator matrix have the form

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

Consider BSC with $p_0 = 0.01$. Let the channel output be $y = (0, 0, 1, 0, 1)$. 
### Symbol MAP decoding

| Information symbols $u_1, u_2$ | $c = (c_1, \ldots, c_5)$ | $p(y|c)$ | $p(c|y) = \frac{p(y|c)p(c)}{p(y)}$ |
|-------------------------------|--------------------------|----------|---------------------------------|
| $y = (0, 0, 1, 0, 1)$          |                          |          |                                 |
| 00                            | 00000                    | $p_0^2(1 - p_0)^3 = 0.00729$ | 0.45     |
| 01                            | 01011                    | $p_0^2(1 - p_0)^2 = 0.00081$ | 0.05     |
| 10                            | 11100                    | $p_0^2(1 - p_0)^2 = 0.00081$ | 0.05     |
| 11                            | 10111                    | $p_0^2(1 - p_0)^3 = 0.00729$ | 0.45     |

$p(y) = \sum_{m=0}^{3} p(y|c_m)p(c_m) = 0.00405$

| Symbols          | Output LLRs | $\ln \frac{p(1|y)}{p(0|y)}$ |
|------------------|-------------|-------------------------------|
| $u_1$            |             | 0                             |
| $u_2$            |             | 0                             |
| $c_1, c_3, c_4, c_5$ |             | 0                             |
| $c_2$            |             | $-2.1972$                     |
We consider symbol MAP-decoding for the single-parity-check code which can be interpreted as a constituent (row) code in the LDPC code construction. We start with proofs of auxiliary lemmas.
Lemma:
Consider a sequence of $n$ independent binary symbols. Let $p$ be a probability of one in each position. Then the probability that this sequence has an even weight is equal to

$$P_{\text{even}} = \frac{1 + (q - p)^n}{2}, \quad q = 1 - p$$

(5)

Proof.
Since probability of the sequence weight $i$ ones is $\binom{n}{i} p^i q^{n-i}$

$$P_{\text{even}} + P_{\text{odd}} = (q + p)^n = 1$$

$$P_{\text{even}} - P_{\text{odd}} = (q - p)^n$$

Half of the sum is equal to (5).
Lemma

Let in the binary sequence of length \( n \), \( p_i \) be the probability of one in position \( i \). Then the probability of even number of ones is

\[
P_{\text{even}} = \frac{1 + \prod_{i=1}^{n} (q_i - p_i)}{2}, \quad q_i = 1 - p_i.
\]

Proof.

Consider the product \( \prod_{i=1}^{n} (q_i - p_i) \) and represent it as the sum of \( 2^n \) terms. Positive terms correspond to probabilities of even weight sequences and negative terms correspond to odd weight sequences. Then we obtain

\[
P_{\text{even}} - P_{\text{odd}} = \prod_{i=1}^{n} (q_i - p_i).
\]  \hfill (6)

The assertion of the lemma follows similarly to the proof of lemma (5).
Let a binary \([n, n - 1]\) SPC code given by its parity-check matrix
\[
H = (1 \ 1 \ \ldots \ 1)
\]  \hspace{1cm} (7)

is used for transmitting information over a DMC, let \(y\) be the received sequence.

MAP-decoder computes \(p_i = P(x_i = c|y), \ c \in \{0, 1\}\). Since (7) has to be satisfied for any codeword, then depending on the value \(c\) of the \(i\)-th symbol, the total weight of the remaining \((n - 1)\) symbols has to be either even (for \(c = 0\)) or odd (for \(c = 1\)). Then MAP-decoding rule can be reformulated as follows

\[
P(x_i = c|y) = P(x_i = c|y, S) = \begin{cases} 
1 + \prod_{j=1, j \neq i}^{n} (1 - 2p_j), & c = 0 \\
1 - \prod_{j=1, j \neq i}^{n} (1 - 2p_j), & c = 1
\end{cases}
\]  \hspace{1cm} (8)

where \(S\) denotes the event that the check is satisfied.
Theorem

If checks containing a given symbol are independent, $S$ is the event that they all are satisfied then

$$\frac{P(x_i = 0|y, S)}{P(x_i = 1|y, S)} = \frac{(1 - p_i)}{p_i} \prod_{j=1}^{J} \frac{1}{1 - \prod_{h=1}^{K-1} (1 - 2p_{jh})} \left( 1 + \prod_{h=1}^{K-1} (1 - 2p_{jh}) \right).$$

Proof. A posteriori probability of symbol can be computed as

$$P(x_i = 0|y, S) = \frac{P(S|y, x_i = 0)P(x_i = 0|y)}{P(S|y)}. $$
Since checks are independent we have

\[ P(S|y, x_i = 0) = \prod_{j=1}^{J} P(S_j|y, x_i = 0) \]

If \( x_i = 0 \) then a check containing \( x_i \) is satisfied if the sum of other symbols in this check is zero.
Tanner graph representation of the LDPC code gives another interpretation of BP decoding as a kind of so-called *message passing algorithms*. At the horizontal step of BP decoding, one computes messages from check nodes to variable nodes belonging to them and at the vertical step, messages from variable nodes to check nodes are computed. From this point of view, it is easy to explain why check independence is so important for successful decoding.

The existence of short cycles in the Tanner graph reduces the efficiency of BP decoding due to violating the hypothesis on independence of parity checks in the Theorem. Another interpretation of this phenomenon is that if the error combination covers a cycle, then since check nodes involved in a cycle contain an even number of the corrupted variable nodes, they get zero syndrome values and do not help in error correction.
Decoding of LDPC codes

Initialization: \( q_{ji}^{(0)}(0) = 1 - p_j, \quad q_{ji}^{(0)}(1) = p_j. \)

Horizontal step: Then MAP decoding for the \( i \)-th row is performed according to the formula

\[
\begin{align*}
  r_{ij}^{(t)}(0) &= (1 + \prod_{h \neq j} (1 - 2q_{hi}^{(t-1)}(1))) / 2 \\
  r_{ij}^{(t)}(1) &= 1 - r_{ij}^{(t)}(0)
\end{align*}
\]

In the program we compute

\[
Z_{ij}^{(t)} = \frac{r_{ij}^{(t)}(0)}{r_{ij}^{(t)}(1)}
\]
Decoding of LDPC codes

Vertical step:

\[ q_{ji}(0) = K_{ij}(1 - p_j) \prod_{k \neq i} r_{kj}(0) \]

\[ q_{ji}(1) = K_{ij}p_j \prod_{k \neq i} r_{kj}(1), \]

where \( K_{ij} \) are chosen to ensure \( q_{ji}(0) + q_{ji}(1) = 1 \).

In the program we compute

\[ \text{softout}_{ji} = \frac{q_{ji}(0)}{q_{ji}(1)} \]
Decoding of LDPC codes

Compute a posteriori probabilities

\[ Q_j^{(t)}(0) = K_j(1-p_j) \prod_{k=1}^{J} r_{kj}(0), \quad Q_j^{(t)}(1) = K_j p_j \prod_{k=1}^{J} r_{kj}(1), \]

where \( K_n \) is chosen to guarantee

\[ Q_j^{(t)}(0) + Q_j^{(t)}(1) = 1. \]

Hard decoding:

\[ \hat{c}_j^{(t)} = \begin{cases} 1 & \text{if } Q_j^{(t)}(1) > 0.5 \\ 0 & \text{otherwise} \end{cases} \]
function [hard, steps] = BP(soft,NumIter,V,C)

Compute $n; \ r; \ rw; \ cw; \ Z=\text{zeros}(r,n)$

$\text{hard} = \text{soft} < 1; \ \text{softout} = \text{soft};$

if $\text{checksyndrome(hard,r,rw,V)}==0,$ steps$=0;$ return; end

for $i=1:r,$ $Z(i,V(i,1:rw(i))) = \text{soft}(V(i,1:rw(i)));$ end

for steps$=1:$NumIter

for $i=1:r$ \% loop over checks

\% positions to process
\% MAP decoding for the $i$-th check

$p=V(i,1:rw(i));\ Z(i,p) = \text{map}(Z(i,p));$

end \% end loop over checks

for $i=1:n$ \% loop over symbol nodes

\% nonzero positions in the $i$-th column
\% MAP decoding for the $i$-th check

$p=C(i,1:cw(i)); \ Z(p,i) = \text{softout}(i)/Z(p,i);$\end;

end;***************

$\text{hard} = \text{softout} < 1; \ \text{if checksyndrome(hard,r,rw,V)}==0,$ return; end

end \% end iterations