IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

LDPC Codes

Lab 1: LDPC Codes

- Format of Parity Check matrices
- Implementation of a Gallager-A decoder

Please send comment to (version 1.0 – October 23, 2020):

Elsa Dupraz
elsa.dupraz@imt-atlantique.fr

Preliminary remarks

The objective of this lab is to implement an LDPC Gallager-A decoder and to simulate it onto two parity check matrices. The first part of the lab consists of studying the two LDPC parity check matrices (size, degrees, sparsity, etc.). The second part is about the implementation of the Gallager-A decoder. The code of the lab can be done either with **Python**, or with **Matlab**. In case your choose Matlab, you may setup Octave (a free clone of Matlab) onto your computer.

The lab will be evaluated. Report and code should be sent by e-mail to elsa.dupraz@imt-atlantique.fr before the 4th of November, 9 PM. The quality of the report will be part of the evaluation. If you choose to do the lab with another student ("en binome"), please mention it explicitly at the beginning of your report.

1 Parity Check Matrices

Two LDPC parity check matrices H are provided in mat and csv formats. You can load them by using function load in Matlab or read_csv from the library pandas in Python.

Question 1

Load the two parity check matrices. Give their rates and degrees. Are they regular or irregular?

Question 2

Measure the matrices sparsity (ratio between number of non-zero components and total number of components of the matrices).

2 Gallager-A decoder

The objective of this part is to implement the Gallager-A decoder which was described in the course. For the Gallager A decoder, $b = d_v - 1$. We consider a Binary Symmetric Channel (BSC) with parameter p.

A theoretical result related to LDPC codes states that the decoder's performance does not depend on the transmitted codeword, but only on the channel realizations. Therefore, in the following, we will assume that the all-zero codeword $\mathbf{x} = \mathbf{0}$ is transmitted. This means that there is not need to compute the generator matrix nor to do the encoding when performing the simulations.

Question 3

Implement the BSC and verify that your implementation is correct.

Question 4

Implement the Gallager decoder by using the pseudo-code provided below.

Hints for the implementation of the Gallager-A decoder: You may store the messages exchanged in the decoder in two matrices (one for messages from Variable Nodes, one for messages from Check Nodes) of the same size as the parity check H. In this case, it is highly recommended to also store the positions of the non-zero components on each row and each column of H. In order to identify the positions of the non-zero components in H, you may use the function find in Matlab, or argwhere from the library Numpy in Python.

In Matlab (but can be adapted easily in Python), the code may be structured as follows:

```
% Compute initial messages
m0 = ...
% L iterations of the decoder
for k=1:L
        % Check Node messages computation
        for m=1:M
                for i=1:dc
                % Computes message from CN m to its neighbor VN i
                end
        end
        % Variable Node messages computation
        for n=1:N
                for j=1:dv
                % Computes message from VN n to its neighbor CN j
                end
        end
        % Computes decisions
        % Checks stopping condition
end
```

Question 5

Evaluate the Bit Error Rate (BER) and Frame Error Rate (FER) performance of the Gallager-A decoder by generating a large number of frames for various values of p, for the two provided codes. Comment.

Hints for BER and FER computation: The BER is the average proportion of incorrect bits in each frame. The FER is the proportion of frames which were not correctly decoded.

Question 6

How can you modify the algorithm in order to reduce its complexity?