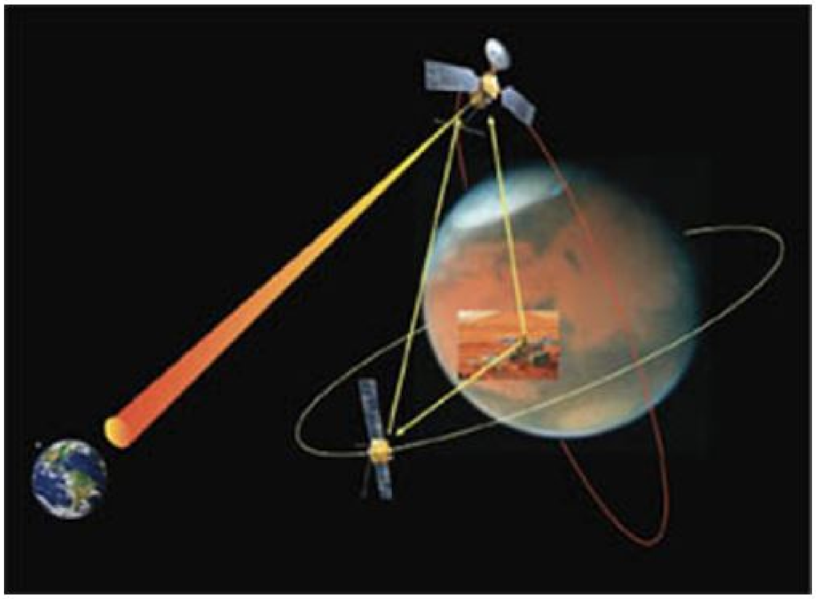
CMPT 727
Spring 2020

Programming Assignment 3

In this programming assignment, you will design algorithms for reliable communication in the presence of noise. We will learn state-of-the-art techniques that some NASA missions use to communicate from deep space, and in the process help The Martian return home.



The Mars Rover is trying to communicate with mission headquarters on Earth. The message the Mars Rover wants to transmit is a binary sequence $X \in \mathbf{B}^N$ of N bits, representing an image. Here $\mathbf{B} = \{0, 1\}$.

To deal with transmission noise, the Mars Rover appends N redundant bits to each message to allow for error correction (using an error-correcting code). The redundant bits are chosen using a clever scheme: the message is encoded as $Y = GX$ through a special matrix $G \in \mathbf{B}^{2N \times N}$, a generator matrix that you can treat as a "constant" for the purposes of this assignment. G is chosen such that the first N bits of Y are equal to X, while the remaining N bits are redundant "parity checks" (here GX is computed modulo 2, so that $Y \in \mathbf{B}^{2N}$). The encoded message $Y \in \mathbf{B}^{2N}$ obtained this way is special because it is a codeword: the Mars Rover and mission headquarters have agreed that all valid messages or codewords are such that $HY = 0$ where H is a pre-specified binary parity check matrix $H \in \mathbf{R}^{N \times 2N}$. The matrices G and H are paired, and chosen so that multiplying by G creates valid messages (codewords), and H can be used to check for errors in a received message (on Earth). Mathematically, $HG = 0$, so that $HY = HGX = 0$ for any input message X.

This codeword $Y \in \mathbf{B}^{2N}$ is then transmitted through deep space back to the mission control on Earth who then receives it as the noisy $\tilde{Y}$. The decoding process refers to the procedure of recovering the ground truth codeword Y (and thus also X, the first N bits of Y) from the noisy version $\tilde{Y}$. We will focus on error correcting codes based on highly sparse, low density parity check (LDPC) matrices H, and use the sum-product variant of the loopy belief propagation

(BP) algorithm to estimate partially corrupted message bits , and to bring our Martian safely back home.

To represent the problem using an undirected graphical model, there are two sets of factors you need to consider. The first are the unary factors associating Y_i with $\tilde{Y}_i$ (messages that are similar to the one received are more likely). You also need to include the parity checks, which are factors defined on Y (assigning zero probability to messages that are not valid codewords) which depend on H.

LDPC codes are specified by a binary parity check matrix $H \in \mathbf{R}^{N\times 2N}$, whose columns correspond to codeword bits, and rows to parity check constraints. We define $H_{ij} = 1$ if parity check P_i depends on codeword bit Y_j , and $H_{ij} = 0$ otherwise. Valid codewords are those for which the sum of the bits connected to each parity check, as indicated by H, equals zero in modulo-2 addition (i.e., the number of "active" bits must be even). Equivalently, the modulo-2 product of the parity check matrix with the 2N-bit codeword vector must equal a N-bit vector of zeros. As illustrated in Fig. 1, we can visualize these parity check constraints via a corresponding factor graph. The parity check matrix H can then be thought of as an adjacency matrix, where rows correspond to factor (parity) nodes P, columns to variable (message bit) nodes Y , and ones to edges linking factors to variables

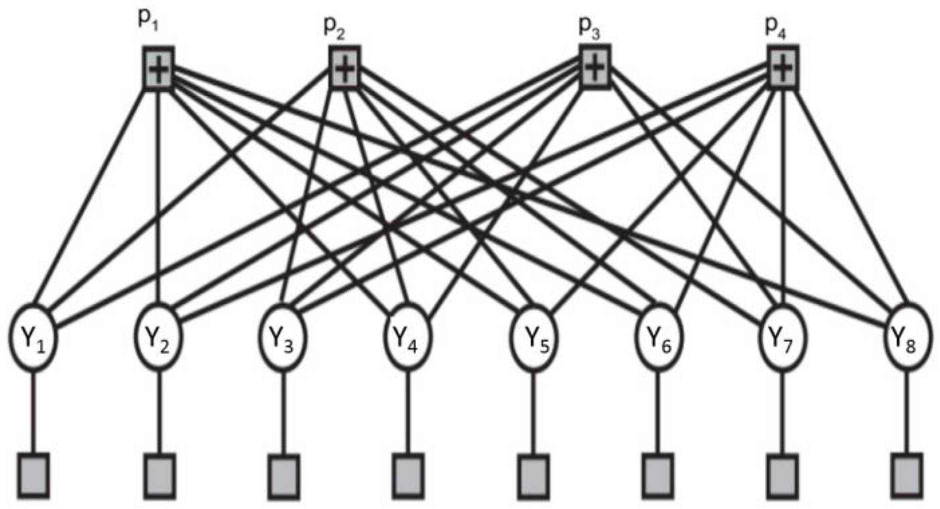


Figure 1: A factor graph representation of a LDPC code linking four factor (parity constraint) nodes to eight variable (message bit) nodes. The unary factors encode noisy observations of the message bits from the output of some communications channel.

We have provided some starter code for this assignment. You can download it here.

What to submit

Please submit the following three files to CourSYS:

- pa3.py
- factor_graph.py
- report.pdf – A pdf file answering all the questions in this assignment.

(4 points) Question 1

Implement constructFactorGraph() in pa3.py that, given an arbitrary parity check matrix H, constructs a corresponding factor graph. The parity check factors should evaluate to 1 if an even number of adjacent bits are active (equal 1), and 0 otherwise. For a given H matrix, define a small test case, and verify that your graphical model assigns zero probability to invalid codewords. The implementation of factors provided in factors.py.

Hint: For unary factors, values are $\begin{pmatrix} 1-\epsilon \\ \epsilon \end{pmatrix}$ if $\tilde{y} = 0$ and $\begin{pmatrix} \epsilon \\ 1-\epsilon \end{pmatrix}$ if $\tilde{y} = 1$.

For parity factors of m bits of 1's, the factor value has 2^m entries of the shape $\underbrace{2 \times ... \times 2}_{m}$, enumerating 2^m possible values of m binary-bits. Each entry is 1 if an even number of bits are active (equal 1), and 0 otherwise.

(6 points) Question 2

Implement loopy belief propagation (sum product) in factor_graph.py for the factor graphs.

(4 points) Question 3

Load the $N = 128bit$ LDPC code provided in ldpc36-128.mat. To evaluate decoding performance, we assume that the all-zeros codeword Y is sent, which always satisfies any set of parity checks. Using the `rand()` method, simulate the output $\tilde{Y}_i$ of a binary symmetric channel: each transmitted bit is flipped to its complement with error probability $\epsilon = 0.05$, and equal to the transmitted bit otherwise. Define unary factors for each variable node Y_i which equal $1 - \epsilon$ if that bit equals the "received" bit at the channel output, and ϵ otherwise.

Run loopy belief propagation for 30 iterations of a parallel message update schedule (update all messages in each iteration using BP equations, based on the messages from the previous iteration), initializing by setting all variable-to-factor messages to be constant. After the final iteration, plot the estimated posterior probability (conditioned on the received, noisy message) that each codeword bit equals one. If we decode by setting each bit to the maximum of its corresponding marginal, would we find the right codeword?

This question requires about 1 minutes of computing time.

(4 points) Question 4

Repeat the experiment from Question 3 for 8 random channel noise realizations with error probability $\epsilon = 0.06$. For each trial, run sum-product for 30 iterations. After each iteration, estimate the codeword by taking the maximum of each bit's marginal distribution, and evaluate the Hamming distance (number of differing bits) between the estimated and true (all-zeros) codeword. On a single plot, display 8 curves showing Hamming distance versus iteration for each trial. Is BP a reliable decoding algorithm?

This question requires about 10 minutes of computing time. Please plan ahead. And also you may choose not to use our provided classes or starter code.

Question 5

a) Give one short piece of feedback about the course so far. What have you found most interesting? Is there a topic that you had trouble understanding? Are there any changes that could improve the value of the course to you?

b) How many hours did you spend on this assignment?

Please provide your answers in your report.

Thanks to Stephano Ermon for giving us permission to adapt this assignment from Stanford CS 228.