CT111: Introduction to Communication Systems

End-of-Semester Project

In the end-of-semester project, you will develop Message Passing Channel Decoder for LDPC Codes.

Project Groups and Evaluations

- ▷ For the final project, you are required to work in the group that you have been assigned with. See Moodle page for the group assignments. There are roughly 10 students per group.
- ▷ Each TA will supervise and evaluate a total of three groups.
- ▷ There will be an interim evaluation in the first week of April. The TA supervising your group will evaluate your work during this interim evaluation. You are required to set up the time with the TA when the TA can do this evaluation.
- ▷ At the end of the semester, e.g., *either* before or *after* your final exams are over, you will make the final presentation of your project work.
- ▷ At the time of evaluation, each group should come prepared with a presentation (e.g., in Powerpoint) of your work, and should have a working code. You will be asked to demonstrate the operation of the program.
- ▷ You are strongly encouraged to use C++ or Python for this project. You will need to take permission from your Supervisor-TA if you want to work in Matlab.
- ▷ Although you will be working as a part of a group, the project grades will be individually awarded. It is in your best interests, both from the learning point of view as well as getting a good evaluation, that each one of you does not sit on the sidelines or be satisfied with some secondary tasks (preparing the presentation, understanding someone else's code/logic, etc.). Instead you should insist on engaging in the actual program development and working as if this is your own individual project. Do not hope or expect that someone else in your group will do a thorough study of the algorithm performance, instead be proactive and take up any such task yourself.

Honor Code

You will be required to submit the following pledge at the time of the interim and final evaluations of the project work. This should be the opening page of your project presentation slides.

- We declare that
  - → The work that we are presenting is our own work.
  - → We have not copied the work (the code, the results, etc.) that someone else has done.
  - → Concepts, understanding and insights we will be describing are our own.
  - → We make this pledge truthfully. We know that violation of this solemn pledge can carry grave[1] consequences.
- Signed by: all the members of the project group.

Technical Deliverables

Following are the items that you (i.e., each project group) is expected to *deliver* at the conclusion of the project:

1. A general encoder and decoder of the (N, K) Product Code, where K (the number of information bits) and N (the number of encoded bits) are perfect squares and $N = K + 1 + 2\sqrt{K}$.
   - ▷ Both the encoder and decoder should be able to accept arbitrary values of K.
   - ▷ They should make use of the generator matrix $\mathbf{G}$ and the parity check matrix $\mathbf{H}$ that are automatically generated given the input value of K.
2. Models of (i) the binary erasure channel (BEC(p)) with the erasure probability of p, and (ii) the binary symmetric channel (BSC(p)) with bit flipping probability of p.
3. Performance (expressed[2] as (i) convergence of the algorithm and (ii) the probability of decoding success) of the product coding scheme for different values of (N, K) on (i) BEC(p) and (ii) BSC(p) for different values of p.
4. Performance (expressed as (i) convergence of the algorithm and (ii) the probability of decoding success) of the LDPC scheme for several different parity-check $\mathbf{H}$ matrices on (i) BEC(p) and (ii) BSC(p) for different values of p.

Your work will be evaluated on the basis of how well you are able to deliver the above tasks. You will be judged on the basis of whether (i) you fall short of meeting the above deliverables, (ii) you have met these deliverables, or (iii) you have exceeded the expectations and done an outstanding job. To ensure that your evaluations are in category (iii), you should perform detailed studies required for Deliverables 3 and 4, and do a deep analysis and thinking on *why* does your code behave in the manner that you have observed.

Technical Instructions

Following are several technical notes about the project.

1. *Product Code*:

[1] E.g., the entire project group will be awarded 0 marks even if there is a slightest hint of plagiarism. It will be an extremely *foolish* mistake to engage in the plagiarism, or to accept someone in your group who is attempting to plagiarize.

[2] This is explained later.

- ▷ Begin by implementing the encoder and the decoder of a basic $(N = 9, K = 4)$ Product Code that we have studied in the class.
- ▷ You should first build an encoder that takes any (of 16 possible) binary string of length $K = 4$ bits and converts it into a product code of length $N = 9$ bits.
  - → Encoder can be implemented by generating $N - K = 5$ parity bits using the rectangular array formulation that we have studied in the class. Here, you take a randomly generated block of $K = 4$ bits, convert that into a 2×2 square array, and now generate 5 parity bits by deriving 5 SPC bits on each of the two rows and two columns of this 2×2 array. Refer to the lecture notes for a description of this.
- ▷ The above is, however, an interim optional step. It is mandatory that you convert this to the *generator* matrix $\mathbf{G}$ of this product code and generate the encoded word $\mathbf{c}$ as $\mathbf{c} = \mathbf{G}\,\mathbf{m}$ modulo-2.

2. *LDPC Code*: a total of 3 LDPC codes are given to you.
   - ▷ Two parity check $]mbfH$ matrices of two different LDPC codes are placed on Moodle folder. The third LDPC code is stated on Page 30 of `CT111 Topic 4 Channel Coding.pdf` in `Topic4ChannelCoding` folder of CT-111 Lecture Folder.
   - ▷ *Do not* implement an encoder (i.e., $\mathbf{G}$ matrix) for any of these three LDPC codes. Instead assume that an all-zero codeword has been transmitted.
3. Build a model for the binary erasure channel (BEC(p)) with the erasure probability of p.
4. Build the binary symmetric channel (BSC(p)) with the erasure probability of p.
5. Build a message passing decoder for the BEC and for the BSC. Note that the same decoder should work for both the product code and the LDPC code.
   - ▷ For the product code, you are asked to first generate the parity check matrix corresponding to (N, K) product code, for arbitrary value of K. You should develop a function to do this.
   - ▷ As mentioned earlier, the parity check matrix for the LDPC code is given to you.
   - ▷ Your code should automatically convert the pattern of ones and zeros in this parity check matrix $\mathbf{H}$ into a Tanner Graph model. The connections between the bit nodes and the check nodes of this bipartite graph are based on the locations of ones in $\mathbf{H}$. See Page 18 of of `CT111 Topic 4 Channel Coding.pdf`.
   - ▷ It is important that you make your code generic enough so that it does not fail to work with another matrix $\mathbf{H}$.
6. Your model of Tanner Graph should perform the message passing from bit nodes to variable nodes and vice versa. This is described on Pages 16 to 30 of `CT111 Topic 4 Channel Coding.pdf` in `Topic4ChannelCoding` folder of CT-111 Lecture Folder.
   - ▷ Recall that in the bipartite Tanner Graph, each bit node is connected to a total of d_v check nodes, and each check node is connected to a total of d_c bit nodes.
   - ▷ Each check node is essentially an SPC code. It enforces the even parity among all the bit nodes that are connected to it. Thus the message passed from the CN to the BN is based on the SPC decoding.

- ▷ Similarly, each bit node is essentially a repetition code. It requires that all the messages are the same (either 1 or 0). Thus, the message passed from the BN to the CN is based on the majority vote decoding of the repetition codes.
- ▷ Message passing for the BEC channel:
  - → From the Check Node to Bit Node:
    - ∗ a check node sends message to each of d_c bit nodes connected to it. When sending the message to k^{th} bit node, the check node "listens" to the messages received from $\{1, 2, \ldots, k-1, k+1, \ldots, d_c\}$ bit nodes.
    - ∗ *SPC decoding.* the message passed to k^{th} bit node is the correct value of that bit node provided *none* of $d_c - 1$ incoming message to this check node are erasures. Else the check node passes an erasure message to k^{th} bit node.
  - → From the Bit Node to Check Node:
    - ∗ a bit node sends message to each of d_v check nodes connected to it. When sending the message to k^{th} check node, the bit node "listens" to the messages received from $\{1, 2, \ldots, k-1, k+1, \ldots, d_v\}$ check nodes that it is connected to, and also the bit received over the BEC.
    - ∗ *majority vote decoding.* the message passed is the correct value of the bit node provided either the bit received over the BEC channel is not an erasure or at least one of the messages received from $d_v - 1$ check nodes connected to this bit node is not an erasure. Only when (i) the bit received over the BEC is an erasure *and* (ii) all $d_v - 1$ check nodes also send erasures to this bit node, this bit node sends erasure to the k^{th} check node.
- ▷ Message Passing for the BSC Channel:
  - → You are required to implement the decoder for the BSC channel.
  - → From the Check Node to Bit Node:
    - ∗ a check node sends message to each of d_c bit nodes connected to it. When sending the message to k^{th} bit node, the check node "listens" to the messages received from $\{1, 2, \ldots, k-1, k+1, \ldots, d_c\}$ bit nodes.
    - ∗ *SPC decoding.* the message passed by the check node to k^{th} bit node is sum modulo two of $d_c - 1$ incoming message to this check node. Thus, each check node gives k^{th} bit node connected to it the following message: "I see the values that your $d_c - 1$ neighbor bit nodes have. I, the parity check enforcer, am telling you that your value must be 0 since I am seeing an even number of 1's in your $d_c - 1$ neighbor bit nodes (or that your value must be 1 since I see an odd number of 1's in your $d_c - 1$ neighbors).
  - → From the Bit Node to Check Node:
    - ∗ a bit node sends message to each of d_v check nodes connected to it. When sending the message to k^{th} check node, the bit node "listens" to the messages received from $\{1, 2, \ldots, k-1, k+1, \ldots, d_v\}$ check nodes that it is connected to, and also the bit received over the BEC.
    - ∗ *Majority vote decoding.* the message passed by the bit node to k^{th} check node connected to it is the majority-vote based. The bit node sends to k^{th} check node a value of 1 if majority out of $d_v - 1$ check nodes and the bit received over the BSC itself is 1; else it sends 0 to k^{th} check node.

7. *Algorithm Convergence (one of the deliverables).* To show that your message passing decoder is working correctly, store the number of erasures for the BEC (and the number of bit errors for the BSC) as a function of the iteration index (one iteration of the message passing is said to have been completed when the messages complete one cycle, i.e., go from from bit nodes to check nodes, and then check nodes to bit nodes). Plot this number as a function of iteration index and see that this plot is converging to zero as the number of iterations increase.

8. *Probability of Successful Decoding (one of the deliverables).* You are asked to evaluate the performance of your algorithms in terms of the probability of decoding success. To evaluate this probability, you will need to implement the following logic:

   - ▷ For each value of p for the BEC and the BSC, and for each of the different codes (Product Codes for different values of K, and the three LDPC codes), perform a total of N_{sim} number of simulation experiments. Keep N_{sim} to be some large number such as 10000.
   - ▷ At n^{th} simulation experiment, where $n = 1, 2, \ldots, N_{sim}$, set a flag F_n to 1 if the decoding is successful. Your code should set $F_n = 1$ at n^{th} trial if the decoder is able to unerase all the erasures introduced by the BEC, or it is able to unflip all the bits that have been flipped by the BSC. If the erasures (for BEC) or the bit flips (for the BSC) remain (after a maximum number of iterations, say, 100), F_n is to be set to 0.
   - ▷ Probability of Successful Decoding is calculated after completing N_{sim} simulation trials as $\dfrac{1}{N_{sim}} \displaystyle\sum_{n=1}^{N_{sim}} F_n$.