Information and Coding Theory. Homework 3

Andrei Dzis

March 2019

1 Problem

1.1 Solution

1. First of all, let's find an interconnection between codewords and cycles in Tanner graph for $(J = 2, K)$ LDPC code.

2. Imagine we have a cycle m in Tanner Graph (denotations with respect to Gallager's paper).

3. Now let's take columns of parity-check matrix, which correspond to variables nodes in Tanner graph, which are used in given cycle. Using the fact, that cycle can't have inner subcycles (otherwise we will have cycle of smaller length m^*) and that Tanner graph is bipartite graph, rows of this submatrix can have only 2 ones in each row.

4. So vectors with non-zero bits in positions $i_1, \ldots, i_n$, where i_j denotes the $j-th$ column of parity check matrix from columns, which were taken in 3., or another words with corresponds to j-th variable node in Tanner graph, which belongs to the cycle m.

5. From statements above, we can assume for codeword weight:

$$w(C) = \frac{m}{2}$$

6. But as we know:

$$d_n = \min_{c \neq [0,\ldots,0]} w(c)$$

Thus, we obtain:

$$d_n \leq \frac{\min(m)}{2},$$

where $\min(m)$ denotes the minimal length cycle in Tanner graph. So we received the upper bound for code distance.

7. Now, looking on $(J = 2, K)$ LDPC code with length n, the goal is to bound the minimal length of cycle in Tanner Graph for given code.

8. We will do it, following Gallager's idea: goal is to build for given code the Tanner graph, with no cycles included.

9. We choose some variable node we choose two check node to go to, this variable node we'll remove from consideration after use, otherwise cycle can occur (after choosing every node in this algorithm we remove it from consideration).

10. From check nodes we can choose 2 variable nodes from available $K-1$ nodes. This is the end of the first step of algorithm.

11. The fascinating difference in the following steps of algorithm, that now in variable node step, while choosing check node, we can choose only one for variable node.

12. Iterations are stopped, when we've chosen all items from sets of nodes. From that step we have insufficient node to continue, without cycle occuring.

13. The length of cycle, occured at i-th step of algorithm to be denoted as:

$$m = \frac{i}{2}$$

14. For this number we can write down:

$$\sum_{i=1}^{m} 2(K-1)^i < n$$

From which we receive:

$$\frac{m}{2} \leq \frac{\log(\frac{Kn}{2}+1)}{\log(K-1)}$$

and finally:

$$d_n \leq 2\frac{\log(\frac{Kn}{2}+1)}{\log(K-1)}$$

15. Now let's write down the following limit to show where this code family asymptotically good or not:

$$\lim_{n\to\infty} \frac{d_n}{n} = \lim_{n\to\infty} \frac{2\log(\frac{Kn}{2}+1)}{n\log(K-1)} = C \lim_{n\to\infty} \frac{\log(Kn)}{n} = 0$$

16. From this, we see, that this family of codes isn't asymptotically good.

2 Problem

2.1 Solution

1. Let's consider function

$$g(z,\dots) = \prod_{i=1}^{K} g_i(z,\dots),$$

where $\dots$ denote other variables, i.e: $z_1, z_2, z_3, z_4, \dots$

2. For $g(z,\dots)$ we can write:

$$\max_{z_1,z_2,z_3,z_4,\dots} g(z,\dots) = \max_{z_1,z_1,z_2,z_3,z_4,\dots} \prod_{i=1}^{K} g_i(z,\dots) = \prod_{i=1}^{K} \max_{z_1,z_1,z_2,z_3,z_4,\dots} g_i(z,\dots)$$

3. Now let's represent $g_i(z,\dots)$ as:

$$g_i(z,\dots) = h(z, z_1, ..., z_J) \prod_{i=1}^{J} h_j(z_j, ...),$$

here we denote new function $h(z,\dots)$

4. Now for g_i we can write:

$$\max_{z_1,\dots} g_i(z, ...) = \max_{z_1,\dots} h(z, z_1, ..., z_J) \prod_{i=1}^{J} h_j(z_j, ...) = \max_{z_1,\dots} h(z, z_1, ..., z_J) \prod_{i=1}^{J} \max_{z_j,\dots} h_j(z_j, ...)$$

5. From previous points, we can assume, that marginalization rules for sum-product algorithm are similar with our case, keeping in mind that fact, that in sum-product algorithms we should substitute summarizing with maximizing.

6. From previous point, conclude that in updating rules for sum-product algorithm are similar, substituting summarizing with maximizing.

7. Thus, block-wise decoding rule for i-th bit:

$$x_i = arg\max_{x_i} \left[\max_{\mathbf{x}\in\mathcal{C}, x_i - fixed} \prod_{i=1}^{n} P(y_i|x_i) \right]$$

8. Generalising idea of substitution summarizing with maximizing, we assume that have same solution using factor graph, let's describe the messages produced by check and variable nodes.

9. Let's introduce the following denotations (mostly taken from correspondent lecture):

   - μ_i - message from i-th connected edge.
   - $r_i = \frac{\mu_i(1)}{\mu_i(-1)}$
   - K - degree of node

   For check nodes we obtain:

$$r = \frac{\mu(1)}{\mu(-1)} = \frac{\max_{x_1,\dots,x_J} \mathbf{1}_{\{\prod_{i=1}^{J} x_i = 1\}} \prod_{i=1}^{J} \mu_i(x_i)}{\max_{x_1,\dots,x_J} \mathbf{1}_{\{\prod_{i=1}^{J} x_i = -1\}} \prod_{i=1}^{J} \mu_i(x_i)}$$

   For variable nodes we obtain:

$$r = \frac{\mu(1)}{\mu(-1)} = \frac{\prod_{i=1}^{K} \mu_i(1)}{\prod_{i=1}^{K} \mu_i(-1)} = \prod_{i=1}^{K} r_i$$

3 Problem

3.1 Solution

1. Due to that fact that $P_{i,j}$ is a permutation matrix, it's generated from identity matrix of size $s \times s$, we can obviously wrire down its rank:

$$\mathrm{rg}(P_{i,j}) = s$$

2. For matrix $P_{i,j}$ column transformation, can be repserented as:

$$P_{i,j} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots & 0 \end{bmatrix}$$

3. So if we apply this transformation for first two column blocks of matrix H, or another words for matrices $P_{i,1}, P_{i,2}, i = \overline{1,m}$, we receive first $2s$ rows of matrix H as follows :

$$H = \begin{bmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots \\ 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \dots \\ \underbrace{1 \quad 0 \quad \dots \quad 0}_{\text{first s rows}} & \underbrace{1 \quad 0 \quad \dots \quad 0}_{\text{second s rows}} & \cdots \end{bmatrix}$$

4. From this representation it's obvious to see, that $2s$ first columns of parity check matrix are linear dependent (in origin non-trivial).

5. Due to that fact, that this code is linear, minimal distance d is upper-bounded by minimal number of dependent columns, or in another words (to make this inequality strict, I will rise the bound by adding 1 to result, otherwise $d \leq 2s$):

$$d < 2s + 1$$