DEPARTMENT OF EE - IIT MADRAS

Assignment 3
Protograph Thresholds and Code Design

Surajkumar Harikumar (EE11B075)

November 30, 2014

1 PROBLEM STATEMENT

Given the base parity check matrix of a protograph, find its BEC threshold. Design a protograph with the highest threshold, given the number of variable nodes, check nodes, and total number of edges.

2 THRESHOLD FINDING

A protograph is completely specified by its base matrix, which denote the number of connections from a variable node to a check node. We introduce the notion of an edge type, so that we can write out exact equations for message passing (rather than the averaged ones we had in LDPC codes). We can write a set of vector-density evolution equations for the protograph, based on message-passing decoding.

For simplicity all edges between the same variable node and check node are assumed to be of the same type (as they have the same neighbourhood on both end nodes). So we have one edge type corresponding to each entry in the H-matrix, and each edge type has a multiplicity specified by its H-matrix entry

The protograph is then copy-permuted, where we copy the base protograph, say L-times, and then permute the edges of the same type. The benefit of doing this is that the density evolution equations for the resulting LDPC-code is the same as that of the base protograph. So we have an exact way to track the evolution of messages in the LDPC code.

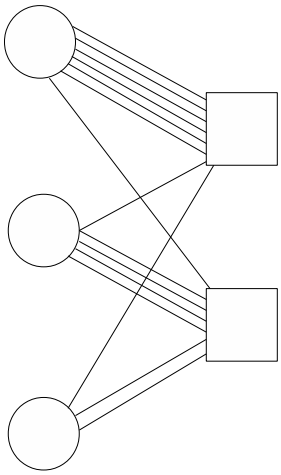

Figure 2.1: Rate 1/3 protograph with 3 variable nodes, 2 check nodes, and 15 edges.

2.1 Density evolution

We formulated density evolution equations for protographs as seen in [1]. The density evolution equation is completely specified by the base matrix, and for any $e_{i,j}$-edge type, we need only look at the row and column neighbourhood. The script **proto_de_iter.m** takes in the parity check matrix, ϵ and number of iterations as input, and returns the edge-erasure probabilities in each iteration. The **status** variable, tells us if DE converges to zero during the iterations.

2.2 Threshold Finding and Simulation

For finding the threshold, we simply iterate over values of ϵ till DE stops converging to zero. We start with low values of ϵ and keep increasing till DE stops converging to zero say at ϵ_1(status variable holds False). Say the immediate previous value was $\epsilon = \epsilon_0$. We now check at $\epsilon_2 = 0.5(\epsilon_0 + \epsilon_1)$. If this converges, the threshold is above ϵ_2, so repeat setting $\epsilon_0 = \epsilon_2$. Else, set $\epsilon_1 = \epsilon_2$. We keep dividing the interval into 2, and we iterate till we get a 4-digit accurate threshold. The script **proto_thresh_brute.m** implements this.

We used the protograph shown in Figure (2.1), and ran the threshold finder. The BEC threshold was found to be $\epsilon^* = 0.4594$. Figure (2.2) shows the density evolution run at ϵ just below the threshold. Each blue-line represents a different edge-type. We see that it converges to zero sharply after about 200 iterations. The script **sub_test.m** is used to implement this.

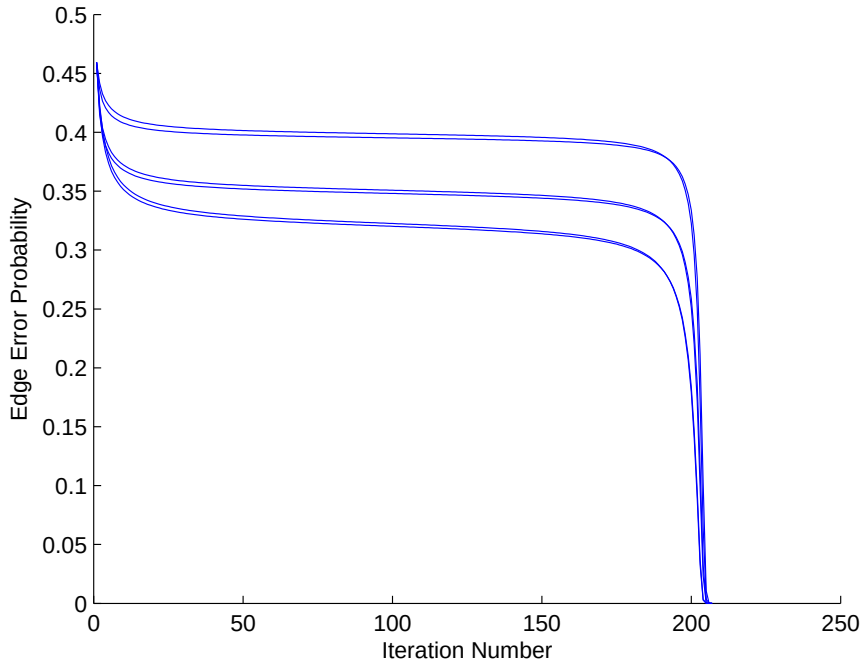

Figure 2.2: Density evolution run at $\epsilon = 0.4593$

3 Finding the Best protograph

We need to find the protograph with the highest threshold, given the number of variable nodes N, check nodes M, and total number of edges L. This is equivalent to finding all parity check matrices, such that the sum of all elements in the H-matrix equals the number of edge types. This is no easy task, as the search space can be large.

We implemented a brute force search, specific to $N = 3, M = 1$ in the script **proto_best.m**. We scan through all possible H-matrices, find their thresholds and choose the best. Even for $L = 10$, there were about 37 possible protographs.

We found $\epsilon^*_{best} = 0.2897$, corresponding to $H = [5,3,2]$. For the rate 2/3 code, we know that $\epsilon* \leq 1 - R = 0.3333$. Our optimization gave $\epsilon^*_{best} = 0.2897$, which is reasonably close to the upper bound.

For us to extend this to the general case, we can use recursions and loops. However the search space is just too vast for a brute force search to work well.

References

[1] Lentmaier, M. ; Tavares, M.B.S. ; Fettweis, G.P. *"Exact erasure channel density evolution for protograph-based generalized LDPC codes"*, ISIT 2009