CONSTRUCTION OF CAPACITY ACHIEVING PROTOGRAPH CODES

A Project Report

submitted by

JAYANTH RAMESH

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2015

THESIS CERTIFICATE

This is to certify that the thesis titled **CONSTRUCTION OF CAPACITY ACHIEVING PROTOGRAPH CODES**, submitted by **Jayanth Ramesh**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Andrew Thangaraj
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 21st May 2015

ACKNOWLEDGEMENTS

I am extremely grateful to my B.Tech Project advisor Prof. Andrew Thangaraj, Department of Electrical Engineering, IIT Madras for the help and guidance that he had extended to me, without which I could not have completed this project. I would like to thank Surajkumar Harikumar and Manikandan Srinivasan for being with me not just during the course of the project, but through the ups and downs in my life at the institute. I am thankful to all my laboratory partners and laboratory supervisors for their kind co-operation. I take this opportunity to thank all the department faculty members for their help and support. I wish to express my gratitude to one and all who have been involved directly or indirectly with this project. Finally, I would like to thank my parents, my sister and my family members for their unrelenting support throughout the course of the project.

ABSTRACT

KEYWORDS: Protograph ; Density Evolution; Heuristics for Protograph Construction; Bounds on Threshold;

The main focus of this project is to come up with simple ways of constructing protograph codes which have thresholds close to capacity. The concept of protographs as Generalized LDPC codes is explained. The basic properties of the evolution of erasure probabilities along different edges in a protograph in a BEC are presented. Closed form upper bounds on the threshold of protographs are proposed, which are used to construct optimal small sized protograph codes. Using ideas from these upper bounds and observed features of good protographs, a heuristic algorithm for the construction of good protograph codes is introduced.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AWGN	Additive White Gaussian Noise
BEC	Binary Erasure Channel
BI-AWGN	Binary Input-Additive White Gaussian Noise
BSC	Binary Symmetric Channel
DE	Density Evolution
GLDPC	Generalized Low-Denstiy Parity-Check
LDPC	Low-Denstiy Parity-Check

NOTATION

C	Capacity of a code
C	Set of check nodes of a protograph
c_j	j^{th} Check node in a protograph
d_{ji}	Number of edges between variable node v_i and check node c_j
E	Set of edges in a protograph
e_{ij}^k	k^{th} edge between variable node v_i and check node c_j
e_{ij}	Any of the edge between variable node v_i and check node c_j
ϵ	Probability of Erasure in a BEC channel
ϵ^*	Erasure Threshold for a Protograph Code
f_{ij}	Density evolution function for edge e_{ij}
H	The parity check matrix of a protograph code
V	Set of variable nodes of a protograph
v_i	i^{th} variable node in a protograph
R	Rate of a protograph code
x_{ij}	Probability of erasure for message sent from a bit node to check node
$\mathbf{x}$	Vector of erasure probabilities of all edges
y_{ij}	Probability of erasure for message sent from a check node to bit node

CHAPTER 1

INTRODUCTION

LDPC codes have emerged as one of the top contenders for near channel capacity error correction. They are described by their sparse parity check matrix, which are efficiently represented as bipartite graphs known as Tanner Graphs. Protographs refer to a new class of generalized LDPC codes, that are derived from a base template. This template is called the protograph, from which the LDPC Tanner graph is constructed using the copy and permute operation (Thorpe, 2003).

1.1 Graphical Representation and Matrix Representation

A protograph can be any Tanner graph, which typically consists of relatively small number of nodes. A protograph can be represented by $G = (V \cup C, E)$, where V represents the set of variable/bit nodes, C represents the set of check nodes and E represents the set of edges in the protograph. Each edge $e \in E$ connects a variable node $v \in V$ to a check node $c \in C$. In general, we consider the number of variable/bit nodes to be N and the number of check nodes to be M. The k^{th} edge connected between variable node v_i and check node c_j is denoted by $e_{ij}^{(k)}$.

It is convenient to represent a protograph by a $M \times N$ matrix H, where each entry $H(j, i)$ denotes the number of edges d_{ji} between variable node v_i and check node c_j. The matrix corresponding to the graphical representation is

$$H_1 = \begin{pmatrix} 4 & 2 & 3 \\ 2 & 1 & 2 \end{pmatrix} \tag{1.1}$$

The corresponding graphical representation of the sample protograph is given by

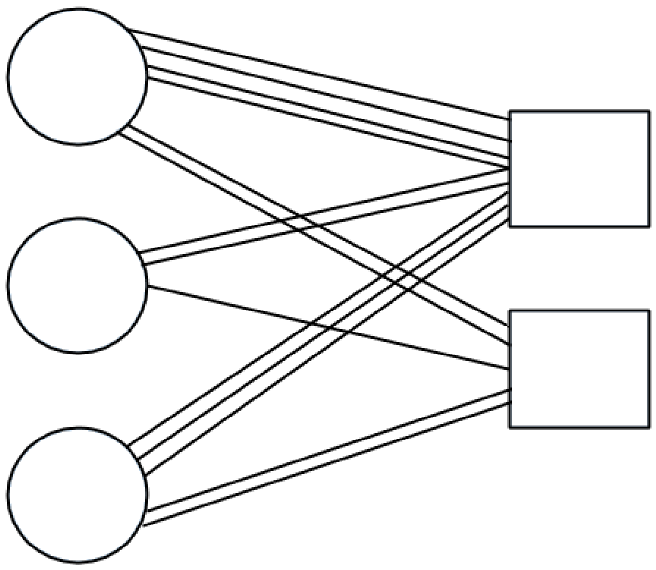
Figure 1.1: Protograph corresponding to the base matrix H_1.

1.2 Copy-Permute Operation to generate Protograph LDPC code

The copy-permute operation as the name suggests, consists of two major steps (Thorpe, 2003). The first step involves copying the base protograph a given number of times (say T). The second step involves permuting the edges of the same kind, that is permuting the endpoints of each edge among the T variable and T check nodes connected to the set of T edges, which are copied from the same edge in the protograph. The derived graph will have NT variable nodes and MT check nodes. This graph obtained after the copy-permute operation is called as **derived graph**, denoted by G'. The design rate of the protograph is given by $R = 1 - \frac{M}{N}$.

A property of the protograph codes is that any local neighbourhood of a node in the derived graph (G') is completely specified by the protograph (G).

Consider the following protograph

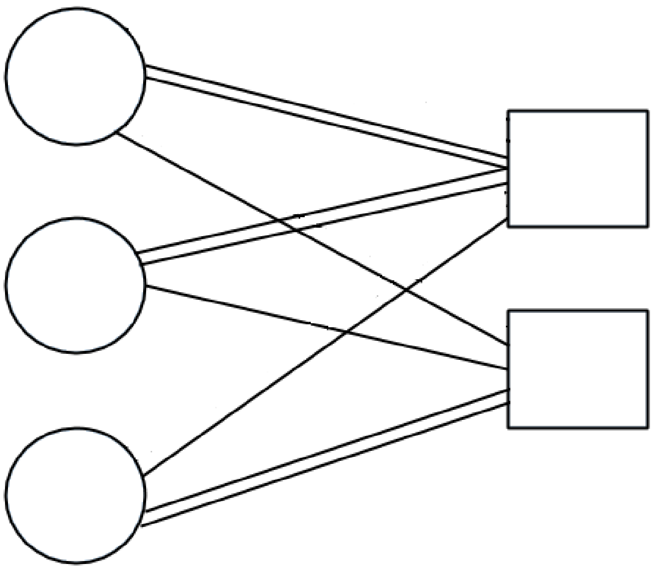

Figure 1.2: Graphical Representation of a sample Protograph

The derived graph obtained after copying the protograph two times and permuting the edges of the same type is shown below.

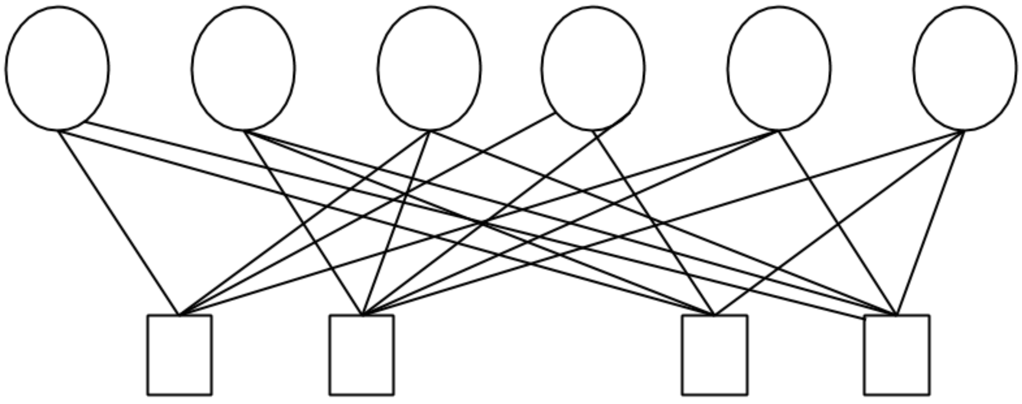

Figure 1.3: Derived Graph corresponding to the protograph in 1.2

CHAPTER 2

Density Evolution for Protograph Codes

We use the standard message passing decoder over a Binary Erasure Channel BEC(ϵ). In the case of the socket ensemble, density evolution tracks one probability of erasure. The protograph imposes a structure on the neighbourhood of each node. So, we need to track the erasure probability for each edge-type in the protograph separately.

Since every edge between v_i and c_j has the same neighbourhood, erasure probabilities of parallel edges connecting v_i to c_j evolve the same way. Let $x_{ij}^{(l)}$ be the probability that an erasure is sent from v_i to c_j along any of the parallel edges connecting them. Similarly, let $y_{ij}^{(l)}$ be the probability that an erasure is sent from c_j to v_i along any of the parallel edges connecting them. To keep the notation simple, we will denote by e_{ij} a representative edge between v_i and c_j, and refer to it as an edge type. Let $N(v_i)$ and $N(c_j)$ denote the neighbors of the nodes v_i and c_j, repectively.

The density evolution recursion (Richardson and Urbanke, 2001) over BEC(ϵ) is given by

$$\begin{aligned} x_{ij}^{(0)} &= \epsilon, \\ y_{ij}^{(l+1)} &= 1 - (1 - x_{ij}^{(l)})^{d_{ji}-1} \prod_{k \in N(c_j) \setminus v_i} (1 - x_{kj}^{(l)})^{d_{jk}}, \\ x_{ij}^{(l+1)} &= \epsilon \ (y_{ij}^{(l+1)})^{d_{ji}-1} \prod_{k \in N(v_i) \setminus c_j} (y_{ik}^{(l+1)})^{d_{ki}} \end{aligned} \tag{2.1}$$

for every pair (i, j) for which an edge e_{ij} exists. From the above equations, we can obtain a recursion of the form

$$\begin{aligned} x_{ij}^{(0)} &= \epsilon \\ x_{ij}^{(l+1)} &= \epsilon f_{ij} \left(x_{11}^{(l)}, \dots, x_{MN}^{(l)} \right), \end{aligned} \tag{2.2}$$

for every edge type e_{ij}. The functions f_{ij} are polynomials in a subset of the variables $\{x_{ij}\}$, $1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant M$, and their properties play a crucial role in the convergence of the iteration. The function f_{ij} can be written as

$$f_{ij}(\mathbf{x}) = \left(1 - (1 - x_{ij}^{(l)})^{d_{ji}-1} \prod_{k \in N(c_j) \setminus v_i} (1 - x_{kj}^{(l)})^{d_{jk}}\right)^{d_{ji}-1}$$
$$\prod_{k \in N(v_i) \setminus c_j} \left(1 - (1 - x_{ik}^{(l)})^{d_{ki}-1} \prod_{k' \in N(c_k) \setminus v_i} (1 - x_{k'k}^{(l)})^{d_{kk'}}\right)^{d_{ki}}, \qquad (2.3)$$

where expressions for $y_{ij}^{(l+1)}$ and $y_{ik}^{(l+1)}$ from (2.1) have been used and $\mathbf{x}$ denotes the vector of arguments for f_{ij}.

The threshold of density evolution, denoted by ϵ^*, is the supremum over ϵ for which erasure probability on each edge of the protograph tends to zero, as $l \to \infty$. In notation,

$$\epsilon^* = \sup\{\epsilon : x_{ij}^{(l)} \to 0 \text{ for all } e_{ij}\}. \qquad (2.4)$$

For $\epsilon > \epsilon^*$, there is some $x_{ij}^{(l)}$ that converges to a non-zero value as $l \to \infty$.

2.1 Properties of Density Evolution Functions

In this section, we describe some of the basic properties of the coupled DE recursion relations, which are useful for determining bounds on the threshold of the protograph.

The first lemma is about monotonicity of the DE functions as in the case of LDPC codes (Richardson and Urbanke, 2008).

Lemma 1. *DE functions $f_{ij}(\boldsymbol{x})$ are monotonically increasing with each variable $x_{mn}, 1 \leqslant m \leqslant M, 1 \leqslant n \leqslant N$.*

Proof. If $\partial f_{ij}/\partial x_{mn} = 0$, then the above result is trivially true as f_{ij} has no dependence on x_{mn}. Otherwise f_{ij} is of the form

$$f_{ij}(\mathbf{x}) = (1 - (1 - x_{mn})^{k_1} g_{mn}(\mathbf{x}))^{k_2} \times h_{mn}(\mathbf{x})$$

where $\partial g_{mn}/\partial x_{mn} = \partial h_{mn}/\partial x_{mn} = 0$. Consider x_{mn_1}, x_{mn_2} as the components of $\mathbf{x_1}$ and $\mathbf{x_2}$, such that $x_{mn_1} > x_{mn_2}$. Now, we consider

$$\begin{aligned}
(1 - x_{mn_1}) \quad &< \quad (1 - x_{mn_2}) \\
f_{ij}(\mathbf{x_1}) = (1 - (1 - x_{mn_1})^{k_1} g_{mn}(\mathbf{x}))^{k_2} \quad &\times \quad h_{mn}(\mathbf{x}) > \\
(1 - (1 - x_{mn_2})^{k_1} g_{mn}(\mathbf{x}))^{k_2} \times h_{mn}(\mathbf{x}) = f_{ij}(\mathbf{x_2}) &
\end{aligned}$$

Thus, without the loss of generality, $f_{ij}(\mathbf{x})$ is monotonically increasing in each variable.

□

Lemma 2. *For all $(i, j), x_{ij}^{(l+1)} \leqslant x_{ij}^{(l)}$, i.e the sequence $\left\{x_{ij}^{(l)}\right\}_{l=1}^{l=\infty}$ is decreasing.*

Proof. We prove this lemma by using induction. We have $f_{ij} \leqslant 1 \ \forall \ 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant M$. Given $\epsilon < 1$, we have $x_{ij}^{(1)} < x_{ij}^{(0)}$. Now let us assume that, $\forall (i, j) \quad x_{ij}^{(k+1)} < x_{ij}^{(k)}$. Now,

$$x_{ij}^{(k+2)} = \epsilon f_{ij}(x_{11}^{(k+1)}, \ldots, x_{MN}^{(k+1)})$$

By using the previous lemma, f_{ij} are monotonically increasing functions with each variable. Therefore

$$\begin{aligned}
\epsilon f_{ij}(x_{11}^{(k+1)}, \ldots, x_{MN}^{(k+1)}) \quad &\leqslant \quad \epsilon f_{ij}(x_{11}^{(k)}, \ldots, x_{MN}^{(k)}) \\
x_{ij}^{(k+2)} \quad &\leqslant \quad x_{ij}^{(k+1)}
\end{aligned}$$

Without any loss of generality, $\forall (i, j), \ x_{ij}^{(k+2)} \leqslant x_{ij}^{(k+1)}$. By the Principle of Mathematical Induction, $x_{ij}^{(l+1)} \leqslant x_{ij}^{(l)}, \ \forall \ l \geqslant 1$. Therefore the sequence $\left\{x_{ij}^{(l)}\right\}_{l=1}^{l=\infty}$ is decreasing. □

Before moving on to the next lemma, we consider an important lemma from mathematics literature (Yeh, 2006), which is relevant to DE equations.

Lemma 3. *If a sequence of real numbers is decreasing and bounded below, then its infimum is the limit.*

Proof. Consider a decreasing sequence $\{a_n\}$, which is bounded below.

Since $\{a_n\}$ is non-empty, and bounded below, we have by the Greatest Lower Bound Property of real numbers, $s = \inf\{a_n\}$ exists and is finite. Now for every $\epsilon > 0$, there exits N such that, $a_N < s + \epsilon$, otherwise $s + \epsilon$ is the lower bound, which contradicts the fact that s is the infimum. As $\{a_n\}$ is decreasing, if $n > N$, we have $|a_n - s| \leqslant |a_N - s| < \epsilon$, which implies that $\{a_n\}_{n\to\infty} = s = \inf\{a_n\}$. $\square$

Using the above lemma, the following result can be shown.

Lemma 4. *For any* $0 \leqslant \epsilon < 1$*, the sequence* $\left\{x_{ij}^{(l)}\right\}_{l=1}^{l=\infty}$ *converges.*

Proof. We know that the sequence $\left\{x_{ij}^{(l)}\right\}_{l=1}^{l=\infty}$ is monotonically decreasing. Also, for each $l \geqslant 1,\ 0 \leqslant x_{ij}^{(l)} \leqslant \epsilon$. By monotone convergence theorem stated in the previous lemma, a sequence that is monotonic and bounded converges. Therefore, the given sequence converges always. $\square$

To illustrate these properties graphically, the evolution of erasure probabilities of different edges with iteration of the protograph H_2 is shown below.

$$H_2 = \begin{pmatrix} 4 & 2 & 2 \\ 2 & 1 & 2 \end{pmatrix} \tag{2.5}$$

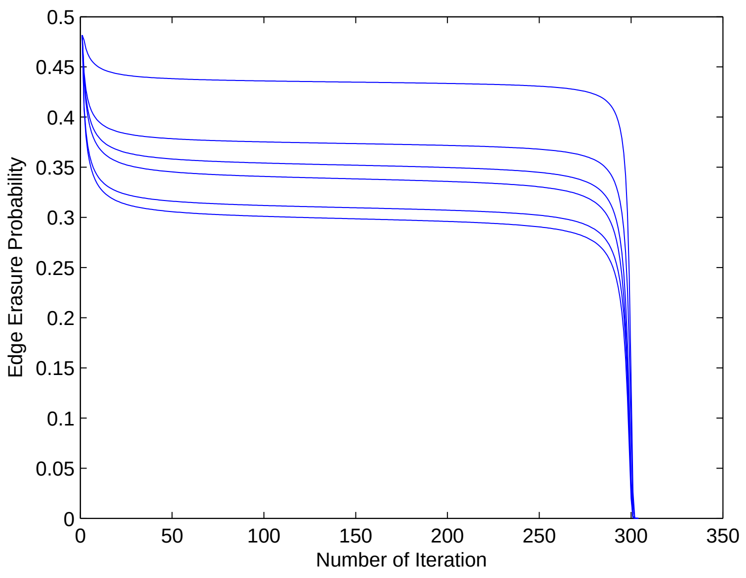

Figure 2.1: Evolution of Edge Erasure Probability with Iteration

CHAPTER 3

Closed Form Bounds on Threshold of Protographs

The threshold of a protograph code determines the ability of the coding scheme to decode errors when the message is transmitted in a BEC. The Density Evolution equation form a set of coupled equations and are difficult to analyse and hence be useful from the perspective of designing a code. Upper bounds on the threshold of protograph help us in gaining an understanding into the goodness of a protograph without much computation. In this chapter, we look at some closed form expression for bounds on threshold of protographs, that give insights about their design and guide the heuristic for improving the protograph threshold presented in the next chapter.

3.1 Preliminary Work

Before going into the closed form bounds, we shall first look at some simple bounds on the threshold of the protographs as given in (Srinivasan, 2015). We have the following useful results for bounds on threshold of protograph codes, which provide the framework for the closed form bounds presented in the next section.

Lemma 5. *If there exists a function $g(x)$ such that $g(x) \leqslant f_{ij}(x), \forall\ i \in V, j \in C, x \in (0,1]$, then an upper bound on threshold is*

$$\epsilon^* \leqslant \bar{\epsilon}_g^* = \min_{x \neq 0} \frac{x}{g(x)} \tag{3.1}$$

where $\bar{\epsilon}_g^*$ can be interpreted as the threshold for the single edge type density evolution with recursion function $g(\cdot)$ and $f_{ij}(x) = f_{ij}(x, \ldots, x)$, that is, all the MN variables are replaced by the single variable x.

The above lemma is useful in the sense that, it reduces the problem of getting an upper bound on the threshold of a protograph into a single variable optimization, thereby reducing computation. The following lemma (Srinivasan, 2015) shows the existence of a function $g(x)$ in terms of the DE function for some specific conditions on the check node degrees and bit node degrees on the protograph.

Lemma 6. *(Single Edge Bound) If there exists an edge $e_{\hat{i}\hat{j}}$ in the protograph that satisfies the conditions:*

- *Highest check node degree -* $d(c_{\hat{j}}) \geqslant d(c_j)$
- *Dominant Column -* $d_{k\hat{i}} \geqslant d_{ki}, \forall \;\; k \in C, \;\; i \in V$

then the threshold has an upper bound

$$\epsilon^* \leqslant \bar{\epsilon}^*_{\hat{i}\hat{j}} = \min_{x \neq 0} \frac{x}{f_{\hat{i}\hat{j}}(x)} \tag{3.2}$$

The **dominant column.** is defined as the column $\hat{i}$, which satisfies $d_{j\hat{i}} \geqslant d_{ji} \; \forall \; 1 \leqslant i \leqslant N$ for every j.

It can be inferred from the structure of DE equations that f_{ij} is of the form $\prod_{i=1}^{N}(1-(1-x)^{p_i})^{q_i}$. In the coming section, we show how to loosen the above form of equation to get a closed form expression for the upper bound on the threshold of protograph, explicitly in terms of the protograph parameters.

3.2 1-term closed form bound

Consider the a minimization of the form

$$W_1 = \min_{x \neq 0} \frac{x}{(1-(1-x)^p)^q} \tag{3.3}$$

with only 1 term in the denominator.

Lemma 7. *(1-TCF Bound) The one variable minimization is upper bounded by*

$$W_1 \leqslant \left(p^q \; e^{-(q-1)} \; \left[\frac{2(q-1)}{pq} \right]^{q-1} \right)^{-1} \tag{3.4}$$

when $q > 1$

Proof. The basic inequality we use to prove this is

$$\sqrt{x}\ \ln\frac{1}{x} \leqslant 1 - x \leqslant e^{-x}; \quad x \in (0,1] \tag{3.5}$$

We show that the function in W_1 can be upper bounded

$$\begin{aligned}
(1-x)^p &\leqslant e^{-px} \\
1-(1-x)^p &\geqslant 1-e^{-px} \geqslant \sqrt{e^{-px}}\ \ln\frac{1}{e^{-px}} \\
1-(1-x)^p &\geqslant (px)\ e^{-px/2} \\
(1-(1-x)^p)^q &\geqslant p^q x^q\ e^{-pqx/2} \\
\frac{x}{(1-(1-x)^p)^q} &\leqslant \frac{x}{p^q x^q\ e^{-pqx/2}}
\end{aligned} \tag{3.6}$$

So, we can upper bound the minimization as

$$W_1 = \min_{x\neq 0} \frac{x}{(1-(1-x)^p)^q} \leqslant \min_{x\neq 0} \frac{1}{(p^q x^{q-1}\ e^{-pqx/2})} \tag{3.7}$$

We can rewrite this as

$$W_1 \leqslant \frac{1}{\max_{x\neq 0}(p^q x^{q-1}\ e^{-pqx/2})} = \frac{1}{J} \tag{3.8}$$

We can perform the maximization for J by performing

$$\frac{dJ}{dx} = 0 \implies x = \left[\frac{2(q-1)}{pq}\right];\ q > 1 \tag{3.9}$$

Substituting this into 3.8, we obtain

$$W_1 \leqslant \left(p^q\ e^{-(q-1)}\ \left[\frac{2(q-1)}{pq}\right]^{q-1}\right)^{-1} \tag{3.10}$$

□

which proves the lemma.

3.3 N-term closed form bound

We can similarly obtain an upper bound for N product terms in the denominator.

$$W_N = \min_{x \neq 0} \frac{x}{\prod_{i=1}^{N} (1-(1-x)^{p_i})^{q_i}} \tag{3.11}$$

Lemma 8. *(N-TCF Bound) The one variable minimization is upper bounded by*

$$W_N \leqslant \left(\left(\prod p_i^{q_i} \right) \, e^{-q'} \, \left[\frac{2q'}{p'} \right]^{q'} \right)^{-1} \tag{3.12}$$

where $q' = \sum q_i - 1$*, and* $p' = \sum p_i \, q_i$

Proof. The proof proceeds in a manner very similar to the previous one. We can use the inequality on each of the product terms individually. Using (3.6)

$$\frac{x}{(1-(1-x)^{p_i})^{q_i}} \leqslant \frac{x}{p_i^{q_i} x^{q_i} \, e^{-p_i q_i x/2}}, \quad i = 1, \ldots, N \tag{3.13}$$

So we have

$$\frac{x}{\prod_{i=1}^{N} (1-(1-x)^{p_i})^{q_i}} \leqslant \left(\left(\prod p_i^{q_i} \right) x^{q'} \, e^{-p'x/2} \right)^{-1} \tag{3.14}$$

The minima will also follow this inequality, yielding

$$W_N \leqslant \frac{1}{\max_{x \neq 0} \left(\left(\prod p_i^{q_i} \right) x^{q'} \, e^{-p'x/2} \right)} = \frac{1}{J_2} \tag{3.15}$$

We can perform the maximization for J_2 by setting

$$\frac{dJ_2}{dx} = 0 \implies x = \left[\frac{2q'}{p'} \right]; \;\; q' > 1 \tag{3.16}$$

where the condition $q' > 1$ is obviously satisfied for $N > 1$. Substituting this into 3.15, we obtain

$$W_N \leqslant \left(\left(\prod p_i^{q_i} \right) \, e^{-q'} \, \left[\frac{2q'}{p'} \right]^{q'} \right)^{-1} \tag{3.17}$$

which proves the lemma.

$\square$

A special case of this which we use later is the 2-TCF bound, given by

$$\min_{x\neq 0} \frac{x}{\left(1-(1-x)^{p_1}\right)^{q_1}\left(1-(1-x)^{p_2}\right)^{q_2}} \leqslant \left(p_1^{q_1}\, p_2^{q_2}\, e^{-(q_1+q_2-1)} \left[\frac{2(q_1+q_2-1)}{p_1q_1+p_2q_2}\right]^{q_1+q_2-1}\right)^{-1} \tag{3.18}$$

We can use the all-edge and single edge bound with these expressions to obtain looser upper bounds on threshold. While these upper bound values are not very tight, they help us understand the properties of high threshold protographs, and guide our heuristic for improving protograph thersholds.

3.4 Design of Protograph based on Closed Form Bound

The objective of protograph design is to find protographs with maximum threshold. Since we don't have a closed form expression for the actual threshold, we try find the parameters of the protograph to maximize the upper bound. We now use the closed form bounds obtained to help design optimal protographs of small size.

Also, the single edge bound is a closer bound than the others, hence we consider only those protographs with a dominant column in our design. We provide an intuitive justification for why a dominant column yields high threshold protographs in Section 4.3. We restrict our attention only to these protographs.

3.4.1 Designing a 1 x 2 protograph

Consider the general 1×2 protograph matrix given by $H_{A1} = [a, b]$. In order for this to be a valid protogrpah (with non-trivial threshold), the protograph entries must satisfy $a \geqslant 2, b \geqslant 2$. Without loss of generality, we can assume $a \geqslant b$, which forms a dominant bit node. The Single-Edge Upper Bound, as given by Lemma 6 is

$$\epsilon^* \leqslant \overline{\epsilon}^*_{\hat{1}1} = \min_{x\neq 0} \frac{x}{\left(1-(1-x)^{a+b-1}\right)^{a-1}} \tag{3.19}$$

We now apply the 1-TCF bound in Lemma 7 to $\bar{\epsilon}^*_{\hat{1}1}$ giving

$$\epsilon^* \leqslant \bar{\epsilon}^*_{\hat{1}1} \leqslant \left(p^q e^{-(q-1)} \left[\frac{2(q-1)}{pq} \right]^{q-1} \right)^{-1} \tag{3.20}$$

where $p = a + b - 1, q = a - 1$. This is a valid bound only if $q > 1, a > 2$.

Any protograph of the form $[a, b]$ has an upper bound of this form. The design criterion is to find the values of a, b which maximize the 1-TCF bound. This amounts to finding

$$\begin{aligned}
(p, q) &= \arg\min_{p,q} \; p^q e^{-(q-1)} \left[\frac{2(q-1)}{pq} \right]^{q-1} \\
&= \arg\min_{p,q} \; p e^{-(q-1)} \left[\frac{2(q-1)}{q} \right]^{q-1} \\
&= \arg\min_{p,q} \; \log(p) - (q-1) + (q-1) \log \left(\frac{2(q-1)}{q} \right) \\
&= \arg\min_{p,q} \; J(p, q)
\end{aligned}$$

It is evident that for a given $q = a-1$, $J(p, q)$ is minimum when $b = p-q$ is minimum. Since $b > 1$, $b = 2$ is the best choice.

Now consider

$$\frac{\partial J(p, q)}{\partial q} = -1 + \log 2 + \log \left(1 - \frac{1}{q} \right) + \frac{1}{q} \tag{3.21}$$

which is negative for $q > 1$. So $J(p, q)$ is a decreasing function of q. Since $q > 1$, the minimum is attained when $q = 2, a = 3$. Note that $a = 2, b = 2$ is a valid protograph, but doesn't satisfy our bound conditions, so it is not considered.

Hence the best 1×2 protograph, as given by our design method is $H_{A1} = [3, 2]$. The threshold of this protograph is $\epsilon^* = 0.4448$.

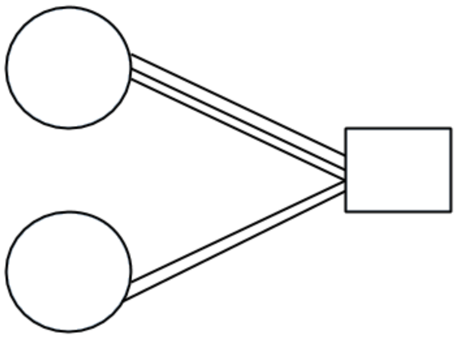

Figure 3.1: Best 1×2 protograph using Closed Form Bound

The following figure shows the evolution of erasure probability along the different edges.

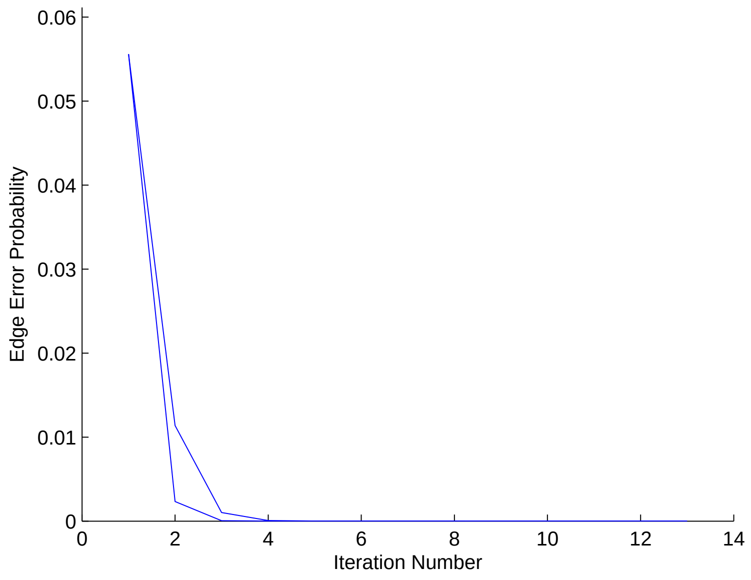

Figure 3.2: Edge Erasure Probability Evolution for best 1×2 code at $\epsilon < \epsilon^*$

3.4.2 Designing a 2 x 3 protograph

Consider a general 2×3 protograph matrix given by

$$H_{A2} = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \tag{3.22}$$

We consider the first column of the H_{A2} matrix as the dominant column. Without loss of generality, we can assume $a \geqslant b$. The single edge upper bound as given by Lemma

6 is

$$\epsilon^* \leqslant \bar{\epsilon}^*_{\hat{1}1} = \min_{x \neq 0} \frac{x}{\left(1-(1-x)^{a+b+c-1}\right)^{a-1}\left(1-(1-x)^{d+e+f-1}\right)^{d}} \tag{3.23}$$

We now apply the 2-TCF bound in Lemma 8 to the above equation to get

$$\epsilon^* \leqslant \bar{\epsilon}^*_{\hat{1}1} = \left(p_1^{q_1}\; p_2^{q_2}\; e^{-(q_1+q_2-1)}\; \left[\frac{2(q_1+q_2-1)}{p_1q_1+p_2q_2}\right]^{q_1+q_2-1}\right)^{-1} \tag{3.24}$$

where $p_1 = a+b+c-1, q_1 = a-1, p_2 = d+e+f-1$ and $q_2 = d.$

Our aim here is to find the values of p_1, q_1, p_2, q_2 which maximize the 2-TCF bound. Therefore, we have

$$\begin{aligned}
(p_1,q_1,p_2,q_2) &= \arg \min_{p_1,q_1,p_2,q_2} \left(p_1^{q_1}\; p_2^{q_2}\; e^{-(q_1+q_2-1)}\; \left[\frac{2(q_1+q_2-1)}{p_1q_1+p_2q_2}\right]^{q_1+q_2-1}\right) \\
&= \arg\min_{p,q}\; q_1\log(p_1) + q_2\log(p_2) - (q_1+q_2-1) + \\
&\qquad (q_1+q_2-1)\log\left(\frac{2(q_1+q_2-1)}{p_1q_1+p_2q_2}\right) \\
&= \arg\min_{p,q}\; J(p_1,q_1,p_2,q_2)
\end{aligned}$$

For minimizing $J(p_1, q_1, p_2, q_2)$, we enumerate the partial derivatives of J with respect to p_1, q_1, p_2 and q_2 to zero.

$$\frac{\partial J(p_1,q_1,p_2,q_2)}{\partial p_1} = \frac{q_1}{p_1} - \frac{(q_1+q_2-1)q_1}{p_1q_1+p_2q_2} \tag{3.25}$$

Setting the above partial derivative to zero results in

$$\frac{1}{p_1} - \left(\frac{q_1+q_2-1}{p_1q_1+p_2q_2}\right) = 0 \tag{3.26}$$

Similarly, $\frac{\partial J}{\partial p_2}$ results in

$$\frac{1}{p_2} - \left(\frac{q_1+q_2-1}{p_1q_1+p_2q_2}\right) = 0 \tag{3.27}$$

Solving the above two equations results in

$$p_1 = p_2 \tag{3.28}$$

This condition means that the check node degrees of both the check nodes in the protograph are equal.

Using $p_1 = p_2$, we have

$$\begin{aligned}\frac{\partial J(p_1, q_1, p_2, q_2)}{\partial q_1} &= \log\left(\frac{2(q_1+q_2-1)}{q_1+q_2}\right) - \frac{q_1+q_2-1}{q_1+q_2} \\ &= -1 + \log(2) + \log\left(1 - \frac{1}{q_1+q_2}\right) - \frac{1}{q_1+q_2} \qquad (3.29)\end{aligned}$$

which is negative for $q_1 + q_2 > 1$. So $J(p_1, q_1, p_2, q_2)$ is a decreasing function in $q_1 + q_2$. This implies that, $q_1 + q_2$ should be as less as possible and in this case it is equal to 2.

Using the above results, we get the following conditions on the optimal protograph.

- $a \geqslant b \geqslant c$ and $d \geqslant e \geqslant f$, for the first column to be dominant as assumed initailly
- $a + b + c = d + e + f$, which follows from $p_1 = p_2$
- $b + c + e + f \geqslant 5$, for a valid protograph without degree 2 cycles
- $a + b = 3$, for a higher closed form bound arising from $q_1 + q_2 > 1$

Since all the constraints cannot be satisfied at once, we choose to relax the constraints as little as possible without violating the dominant column condition and inducing degree two cycles. In this case, we relax the check node degree equality constraint.

The above set of conditions yield the following protograph.

$$H_{A2} = \begin{pmatrix} 2 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad (3.30)$$

Hence the best 2×3 protograph by our design is given by H_{A2} above with a threshold $\epsilon^* = 0.5947$.

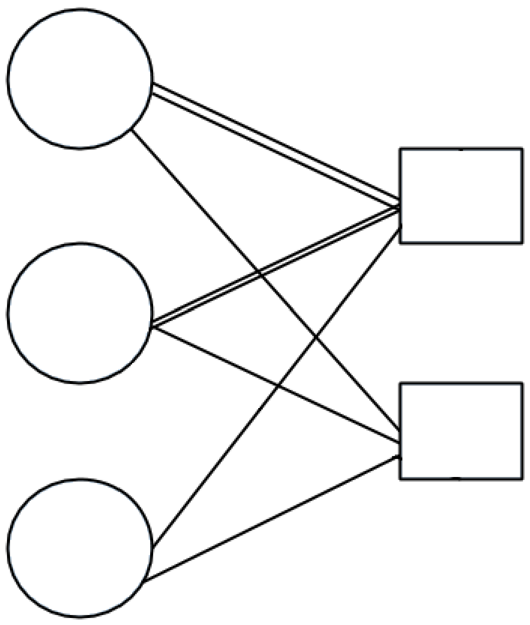

Figure 3.3: Best 2×3 protograph using Closed Form Bound

The following figure shows the evolution of erasure probability along the different edges.

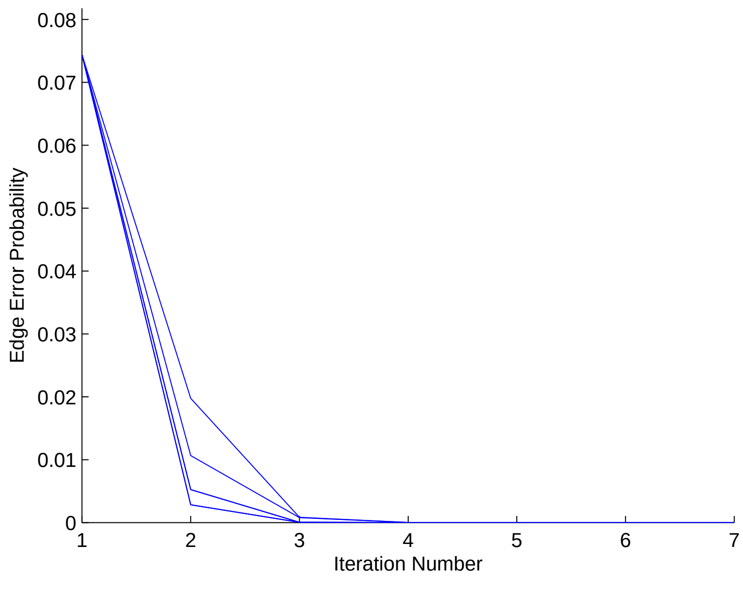

Figure 3.4: Edge Erasure Probability Evolution for best 2×3 code at $\epsilon < \epsilon^*$

CHAPTER 4

Heuristics for Construction of Capacity Achieving Protograph Codes

This chapter describes in detail the process of construction of capacity achieving protograph codes based on the observed characteristics of good protographs, and also justifies why those characteristics contribute to a higher erasure threshold for the protograph code in the BEC channel.

4.1 Commonly Observed Features of Good Protographs

Let us examine some of the protographs that have erasure threshold in a BEC channel close to capacity.

Consider the following 3×6 protgraph

$$H_3 = \begin{pmatrix} 2 & 2 & 1 & 1 & 1 & 0 \\ 4 & 0 & 1 & 0 & 1 & 1 \\ 3 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4803, \quad \mathbf{C} = 0.5 \tag{4.1}$$

An optimized 4×8 protograph given in (Pradhan *et al.*, 2013)

$$H_4 = \begin{pmatrix} 1 & 2 & 2 & 3 & 4 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 3 & 0 & 4 & 1 \\ 1 & 0 & 1 & 0 & 6 & 1 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.479, \quad \mathbf{C} = 0.5 \tag{4.2}$$

An optimized 8×16 protograph given in (Pradhan *et al.*, 2013)

$$H_5 = \begin{pmatrix} 1 & 2 & 0 & 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 2 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 3 & 1 & 2 & 1 & 0 & 0 & 0 & 4 & 0 & 0 & 3 & 2 & 2 & 0 & 3 \\ 0 & 5 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 3 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}; \ \epsilon^* = 0.4876, \ \mathbf{C} = 0.5 \tag{4.3}$$

The above protographs have threshold close to capacity. One striking feature common to these protographs is they have one column/bit node, the number of edges connected to which are significantly larger than the corresponding number of edges connected to the other columns/bit nodes. If we exclude this column from the protograph matrix, we see that the rest of the matrix is sparse with many entries being 0. These observation form the basis for the heuristic proposed for construction of capacity achieving protographs in the next section.

4.2 The Heuristic Algorithm

The heuristic consists of two major steps, which results in a improvement of the threshold of a given protograph code (H). The basic intuition follows from the major observations about the characteristics of good protograph codes made above. The first step involves creating a dominant bit node/column, as defined in the previous section, followed by reducing the entries of the protograph, which results in a sparser graph.

4.2.1 Creating a Dominant Bit Node and Making the Graph Sparse

The column with the highest weight in the H-matrix is identified. For each row in the matrix, entries from the other column are pushed into the dominant column. A dominant column creates a bundle of edges (corresponding to this bit node), whose erasure probability go to zero much faster than all the others. This can be observed in the DE plots. Since the erasure probability (for edges in the dominant column) are already very small (this happens because the dominant column edges are connected to multiple check nodes resulting in easier decoding), it allows the other edges to converge faster, hence speeding up the DE. Such protographs intuitively should give a higher value of threshold. A few points to be noted here are

- Pushing entries (edges) is done by reducing the H-entry in a given row, and adding the difference to the dominant column entry for the row. This preserves the degree of the check node.
- If an entry is the only one in it's row / column (only edge connecting the bit / check node), then it is left undisturbed.
- If the reduction of a matrix entry causes the graph to become disconnected (there no longer exists a single path connecting all nodes), then such a reduction is avoided.
- The step is terminated when any edge push causes the graph to become disconnected or induces degree two cycles or when the rest of the matrix becomes relatively sparse (that is each column degree other than dominant column is around 3-4).
- Repeating columns are avoided as much as possible. For instance, if the column to be reduced has $[3, 2]'$, and an existing column is $[2, 2]'$, preference is given to the combination $[3, 1]'$ over $[2, 2]'$.

4.2.2 Reducing the Dominant Column

This is the second step in the heuristic. The entries of the dominant column are reduced to make the graph sparse, while at the same time ensuring that the dominant column continues to remain so.

- All the check node degrees are computed first. Let the check node degrees be sorted in descending order and C_1 represent the check node of highest degree and C_M denote the check node with the least degree. If two check nodes have the same degree, then the check node with a higher entry in the dominant column is given precedence.
- The dominant column entry in C_1 are reduced to make the overall degree equal to (or just larger than) the degree of C_2.
- The dominant column entries in C_1 and C_2 together are reduced, to make their degree equal to (or just larger than) the degree of C_3.

- Once all the check node degrees are approximately equal, the entries in the dominant columns are reduced till the difference in the degree of the dominant and the second highest bit node degree column is about $3 \times (M-1)$.

4.3 Justification of the Heuristic

In this section, we try to reason out the various steps in our heuristic algorithm for improving the threshold of a given protograph.

4.3.1 Presence of Dominant Column

The first major step in the heuristic is to induce a dominant column in the protograph. The Density Evolution function corresponding to these edges have relatively smaller value compared to the other edges, owing to the higher value of the exponents. From 2.1, we have

$$\begin{aligned} x_{ij}^{(l+1)} &= \epsilon \; (y_{ij}^{(l+1)})^{d_{ji}-1} \prod_{k \in N(v_i) \backslash c_j} (y_{ik}^{(l+1)})^{d_{ki}} \\ y_{ij}^{(l+1)} &= 1 - (1 - x_{ij}^{(l)})^{d_{ji}-1} \prod_{k \in N(c_j) \backslash v_i} (1 - x_{kj}^{(l)})^{d_{jk}} \end{aligned} \tag{4.4}$$

Let j represent the dominant column. Then, the values of values of d_{ji} and d_{jk} are higher then the corresponding edges of other columns, resulting in smaller values for x_{ij} in subsequent iterations. This results in an earlier convergence of probability of erasure to zero for the edges present in dominant column. Intutively, the dominant column edges are connected to a large number of check nodes, which results in earlier convergence of erasure probabilities along these edges. This convergence induces the probability of erasure to go to zero in other edges, thereby resulting in a protograph code having a higher threshold for a BEC.

4.3.2 Preserving Connectivity

Another important point to be kept in mind while running the heuristic is to ensure that the connectivity of the graph isn't disturbed. This is because, if any step causes the graph to become disconnected, then it would result in two or more isolated graphs, thereby reducing the number of checks performed on the received bits, making error correction difficult, which leads to lower threshold.

4.3.3 Sparsity of Protograph

The concept of sparsity can also be explained using the ideas from the construction of the 2×3 protograph code. We had to minimize $J(p_1, q_1, p_2, q_2)$, which we observed was a decreasing function of $q_1 + q_2$, for given optimal values of p_1 and p_2. We can extend this again, using the N-TCF bound to show that $J(p_1, q_1, p_2, q_2, \ldots, p_n, q_n)$ is a decreasing function of $q_1 + q_2 + \ldots + q_n$ for given optimal values of $p_1, p_2, \ldots, p_n$. Therefore, for good thresholds, we need $\Sigma_{i=1}^{n} q_i$ to be minimum. This would force the dominant column entries in the protograph to be as low as possible, which would in-turn force the other entries of the protograph to be as little as possible. This throws light on the observation made about sparsity in good protographs in the last section.

4.3.4 Avoiding Repetition

It is also better to avoid repetitive columns as much as possible because, if there is repetition, it is possible to find non-repetitive columns which are more sparse than the repetitive case. This is because repetition adds more edges, without changing the connectivity state. For example, $[2, 1]'$ and $[2, 1]'$ can be reduced to $[2, 1]$ and $[1, 1]'$ without affecting the connectivity of the graph and at the same time making the graph more sparse.

4.3.5 Equality of Check Node Degrees

We mentioned about ensuring that the check node degrees of the different check nodes are equal or close to one another. The justification for this can be traced to the construction of 2×3 protograph using the closed form bounds discussed in the previous chapter. We found that, for a 2×3 protograph, the threshold maximizing code had check node degrees equal or nearly equal as in equation 3.28. It is also easy to see that in general, for a protograph with M checknodes, using the N-TCF bound from Lemma 8 discussed in chapter 3, we will reach the condition of check node degree equality using the partial derivative technique as described in section 3.4.2.

4.3.6 3 x (M-1) Difference Rule

It was also stated in the heuristic that the algorithm must be terminated when the difference in the degrees of the dominant column and the column with the second highest degree (number of edges connected to a bit node is defined as degree of the bit node) becomes around $3 \times (M-1)$. This is because, it is observed that, for good threshold protographs, the sparse part has at most one, two or three edges between a given variable node and check node, which results in the dominant column having about 3 edges connected to each check node on an average. Therefore, the number of edges connected to the dominant bit node is about $3 \times M$ on an average, where M is the number of check nodes in the protograph. The bit node with the next highest degree has about 3 edges connected to it. Therefore, the difference in the degree is about $3 \times (M-1)$. For instance, the optimized 8×16 protograph in 4.3 has this difference equal to 20 while $3 \times (M-1)$ is equal to 21.

4.4 Improving Protograph Threshold using Heuristic

Now we apply the heuristic to existing graphs and show the resulting protographs and their thresholds. Note that the final matrix is not unique, and different choices in each step, yield different results. However, all the resulting matrices have better thresholds after each step.

4.4.1 Example 1

Consider the following matrix .

$$H_6 = \begin{pmatrix} 5 & 2 & 1 \\ 4 & 1 & 2 \end{pmatrix}; \quad \epsilon^* = 0.4794 \tag{4.5}$$

H_6 is a rate-$1/3$ code. The BEC threshold of the graph is $\epsilon^* = 0.4794$. The highest achievable BEC threshold for all rate-$1/3$ codes is 0.6667 (Shannon Limit). The following matrices are constructed based on the heuristic.

$$H_6^P = \begin{pmatrix} 5 & 2 & 1 \\ 5 & 1 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4989$$

$$H_6^R = \begin{pmatrix} 2 & 2 & 1 \\ 3 & 1 & 1 \end{pmatrix}; \quad \epsilon^* = 0.5713 \tag{4.6}$$

H_6^P is obtained by the **push** operation, and column 1 is made the dominant column. The check node degrees are the same as the original graph, and any further pushes causes a loss of connectivity or degree-2 cycles. H_6^R is obtained by **reducing** the entries in the dominant column. The check node degrees are $7, 8$ respectively. We reduce $5 \rightarrow 4$ in accordance with the heuristic. The highest bit-node degree is now 9, and the next highest is 3, we can reduce the entries uniformly till the difference is around $3M - 3 = 3$. The BEC threshold of the graphs at each stage are shown next to the matrices. This demonstrates a 0.0919 increase in threshold through simple deterministic steps.

4.4.2 Example 2

Consider the 3×6 matrix H_7 reported in Liva *et al.* (2007). This has a BEC threshold .

$$H_7 = \begin{pmatrix} 1 & 2 & 2 & 1 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 & 1 \\ 2 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4395 \tag{4.7}$$

We now apply our heuristic to the above protograph to get.

$$H_7^P = H_7^R = \begin{pmatrix} 2 & 2 & 1 & 1 & 1 & 0 \\ 4 & 0 & 1 & 0 & 1 & 1 \\ 3 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4803 \tag{4.8}$$

H_8^P is obtained by the push operations, and a dominant column is created. Note that converting the column $[1, 1, 1]' \rightarrow [1, 1, 0]'$ would create a redundant column, and so is avoided. The dominant bit node degree is 9, and the next highest bit-node-degree is 3, with the difference being equal to $3M - 3 = 6$. So the reduction step yields the result $H_7^R = H_7^P$, with a BEC threshold as shown. Since the original graph already has degree-2 cycles, and since the cycles cannot be broken by our heuristic, this code will have slow block error threshold decay.

4.4.3 Example 3

In this example, we apply the heuristic to a protograph of a slightly larger 4×8 size. Consider

$$H_8 = \begin{pmatrix} 2 & 2 & 0 & 1 & 1 & 0 & 0 & 2 \\ 4 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 1 & 2 & 1 & 0 & 0 \\ 2 & 1 & 2 & 0 & 0 & 1 & 2 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4208 \tag{4.9}$$

Application of the heuristic results in

$$H_8^P = \begin{pmatrix} 4 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 4 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 1 & 2 & 1 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 & 2 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4513$$
$$H_8^R = \begin{pmatrix} 3 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 3 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 1 & 2 & 1 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 & 2 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4519 \tag{4.10}$$

H_8^P is obtained by pushing entries to the dominant column preserving the check node degrees, leading to the creation of a dominant first column. H_8^R is obtained by reducing the entries of the dominant column, till the $3 \times (M-1)$ rule is satisfied. We see that H_8^R has the degree difference of the dominant and second dominant as 10 , which is quite close to the $3M - 3 = 9$.

4.4.4 Example 4

Consider the 4×8 protograph matrix H_9. The Shannon Limit for a rate-1/2 code is 0.5, so this already quite close.

$$H_9 = \begin{pmatrix} 1 & 3 & 2 & 3 & 4 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 4 & 0 & 4 & 1 \\ 1 & 0 & 1 & 0 & 6 & 1 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4785 \tag{4.11}$$

We now apply our heuristic to the above protograph in the following steps.

$$\begin{aligned} H_9^{P_1} &= \begin{pmatrix} 0 & 3 & 2 & 3 & 6 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 4 & 0 & 4 & 1 \\ 1 & 0 & 1 & 0 & 6 & 1 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4789 \\ H_9^{P_2} &= \begin{pmatrix} 0 & 2 & 2 & 3 & 7 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 4 & 0 & 4 & 1 \\ 1 & 0 & 1 & 0 & 6 & 1 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4803 \\ H_9^{R} &= \begin{pmatrix} 0 & 2 & 2 & 3 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 5 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 4 & 0 & 4 & 1 \\ 1 & 0 & 1 & 0 & 6 & 1 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4807 \end{aligned} \tag{4.12}$$

$H_9^{P_1}$ and $H_9^{P_2}$ is obtained from the push operation of the heuristic and H_9^R is obtained using the reduce operation. Here, we couldn't further reduce the entries in dominant column to make the check nodes approximately equal, hence we don't move on to the next step of enforcing the $3 \times (M - 1)$ rule. We improved an already good threshold graph using the simple steps in the heuristic.

The heuristic gives a resulting graph whose threshold is usually higher than the base graph. Protograph matrices which are already very sparse, or exhibit the characteristics of a "good" matrix (which already exhibit the dominant column and sparse properties our heuristic seeks to induce), the improvement is milder. Even in this case, the threshold is atleast as high, and in most cases the heuristic gives a higher threshold protograph.

4.5 Construction of Good Protograph Codes using the Heuristic

In the previous section, we described a heuristic which takes an existing protograph matrix and returns a protograph with a higher BEC threshold. Protographs with high thresholds can possibly be made better by the heuristic. In this section, we tackle the problem of protograph construction.

We choose an arbitrary starting matrix and apply our heuristic to get a higher threshold protorgraph. Some of the smaller examples were done by hand, and we wrote a script which implements the heuristic.

4.5.1 Example 1

We consider a general starting point matrix as the one with all the entries as the same. Consider the 2×3 rate-1/3 protograph H_9 with all entries as 3.

$$H_{10} = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \end{pmatrix}; \quad \epsilon^* = 0.4035 \tag{4.13}$$

Applying the first step of heuristic results in

$$H_{10}^P = \begin{pmatrix} 7 & 1 & 1 \\ 6 & 2 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4570$$

Applying the second step of heuristic results in

$$H_{10}^R = \begin{pmatrix} 3 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix}; \quad \epsilon^* = 0.5713 \tag{4.14}$$

We see that $H_{10}^R = H_6^R$, which we obtained for H_6. The Shannon Limit on threshold for rate-1/3 codes is 0.6667

4.5.2 Example 2

Consider now the 2×4 rate-1/2 protograph H_{11} with all entries as 3.

$$H_{11} = \begin{pmatrix} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{pmatrix}; \quad \epsilon^* = 0.3075 \tag{4.15}$$

Applying the first step of heuristic results in

$$H_{11}^P = \begin{pmatrix} 8 & 2 & 1 & 1 \\ 9 & 1 & 2 & 1 \end{pmatrix}; \quad \epsilon^* = 0.3721$$

The application of second step of heuristic leads to

$$H_{11}^R = \begin{pmatrix} 2 & 2 & 1 & 1 \\ 3 & 1 & 1 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4505 \tag{4.16}$$

The Shannon Limit on threshold for rate-1/2 codes is 0.5.

4.5.3 Example 3

Consider now the 4×8 rate-1/2 protograph H_{12} with all entries as 3.

$$H_{12} = \begin{pmatrix} 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \end{pmatrix}; \quad \epsilon^* = 0.1953 \tag{4.17}$$

Applying our heuristic results in the following

$$H_{12}^{P_1} = \begin{pmatrix} 11 & 0 & 1 & 2 & 1 & 3 & 3 & 3 \\ 14 & 0 & 0 & 0 & 1 & 3 & 3 & 3 \\ 11 & 1 & 1 & 1 & 1 & 3 & 3 & 3 \\ 13 & 1 & 1 & 0 & 0 & 3 & 3 & 3 \end{pmatrix}; \quad \epsilon^* = 0.2345$$

$$H_{12}^{P_2} = \begin{pmatrix} 18 & 0 & 1 & 2 & 1 & 1 & 1 & 0 \\ 20 & 0 & 0 & 0 & 1 & 1 & 0 & 2 \\ 18 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 22 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.2675$$

$$H_{12}^{R} = \begin{pmatrix} 3 & 1 & 1 & 2 & 0 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 & 1 & 0 & 2 \\ 2 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 5 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.4783 \tag{4.18}$$

Here $H_{12}^{P_1}$ and $H_{12}^{P_2}$ denote the matrices obtained after the push operation, and H_{12}^{R} denotes the matrix obtained after the reduce operation.

Let us also consider an example, where we impose a structure on the sparse part of the protograph matrix. Starting with the same all 3-matrix as before, we have

$$H_{13}^{P_1} = \begin{pmatrix} 12 & 3 & 0 & 0 & 0 & 3 & 3 & 3 \\ 12 & 0 & 3 & 0 & 0 & 3 & 3 & 3 \\ 12 & 0 & 0 & 3 & 0 & 3 & 3 & 3 \\ 12 & 0 & 0 & 0 & 3 & 3 & 3 & 3 \end{pmatrix}; \quad \epsilon^* = 0.2325$$

$$H_{13}^{P_2} = \begin{pmatrix} 20 & 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 19 & 0 & 3 & 0 & 0 & 1 & 1 & 0 \\ 19 & 0 & 0 & 3 & 0 & 0 & 1 & 1 \\ 20 & 0 & 0 & 0 & 3 & 0 & 0 & 1 \end{pmatrix}; \quad \epsilon^* = 0.2670$$

$$H_{13}^{R} = \begin{pmatrix} 4 & 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 3 & 0 & 0 & 1 & 1 & 0 \\ 3 & 0 & 0 & 3 & 0 & 0 & 1 & 1 \\ 4 & 0 & 0 & 0 & 3 & 0 & 0 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4544 \tag{4.19}$$

We can see that restriction on the sparse part of the matrix reduces the number of possible graphs that can be reached from a given starting matrix and applying the heuristics. Therefore, it is possible that in such cases, we could miss out on some good protographs.

4.5.4 Example 4

We now show the construction of a 8×16 protograph. As in the previous sections, we start with

$$H_{14} = \begin{pmatrix} 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \end{pmatrix}; \quad \epsilon^* = 0.1174$$

The application of the heuristic results in

$$
\begin{aligned}
H_{14}^{P_1} &= \begin{pmatrix}
24 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
24 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
24 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
24 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
24 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
24 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
24 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3
\end{pmatrix}; \quad \epsilon^* = 0.1497 \\
H_{14}^{P_2} &= \begin{pmatrix}
43 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
43 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
43 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
43 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
43 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
44 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
43 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
43 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 0 & 1 & 0
\end{pmatrix}; \quad \epsilon^* = 0.1733 \\
H_{14}^{R} &= \begin{pmatrix}
3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
3 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
4 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 0 & 1 & 0
\end{pmatrix}; \quad \epsilon^* = 0.4826
\end{aligned}
\tag{4.20}
$$

The above protgraph has a good threshold for a code with capacity $C = 0.5$

Let us consider a example, where significant structure is imposed on the sparse portion of the matrix. We again have the same starting matrix. The heuristic yields

$$H_{15}^{P_1} = \begin{pmatrix} 24 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 24 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 24 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 24 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 24 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 24 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 24 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \end{pmatrix}; \quad \epsilon^* = 0.1497$$

$$H_{15}^{P_2} = \begin{pmatrix} 43 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 43 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 43 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 43 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 43 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 44 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 43 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 43 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}; \quad \epsilon^* = 0.1726$$

$$H_{15}^{R} = \begin{pmatrix} 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}; \quad \epsilon^* = 0.4806 \tag{4.21}$$

Here, we see that, despite enforcing conditions on the sparse matrix, we were still able to end up with a protograph with good threshold in a few simple steps.

In the next example, let us see the construction of a 10×20 protograph, where we have a different starting matrix.

4.5.5 Example 5

Consider the starting matrix

$$H_{16} = \begin{pmatrix} 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 \\ 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \\ 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 \\ 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \\ 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 \\ 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \\ 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 \\ 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \\ 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 & 4 \\ 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \end{pmatrix}; \quad \epsilon^* = 0.1171$$

Applying our heuristic on the above protograph results in

$$H_{16}^P = \begin{pmatrix} 0 & 45 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 43 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 46 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 43 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 46 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 43 & 0 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 46 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 46 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 43 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 44 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}; \quad \epsilon^* = 0.1716$$

$$H_{16}^{R_1} = \begin{pmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 4 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} ; \quad \epsilon^* = 0.4792$$

$$H_{16}^{R_2} = \begin{pmatrix} 0 & 4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 3 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} ; \quad \epsilon^* = 0.4853 \tag{4.22}$$

4.5.6 Example 6

In this section, we present a few protographs generated by the matlab code we wrote to implement our heuristic algorithm. For every column, a random column is generated such that the subtraction of the two would yield a bit-node, whose degree varies between two and four and the subtracted entries are pushed to the dominant column. The columns are generated such that there is no repetition and no degree 2 cycles. The reduction step is then carried out till the 3M-3 difference rule is met. Many of the protographs generated this way had reasonably high thresholds (0.465 for 0.5 capacity).

Thresholds of Protographs-Code Generated		
Protograph size	Matrix	Threshold
4×8	H_{17}	0.4787
4×8	H_{18}	0.4771
8×16	H_{19}	0.4713
8×16	H_{20}	0.4642
9×18	H_{21}	0.4664
9×18	H_{22}	0.4630

Table 4.1: Threshold of protographs generated by code

$$H_{17} = \begin{pmatrix} 3 & 0 & 1 & 2 & 1 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 & 1 & 0 & 2 \\ 2 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 5 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$H_{18} = \begin{pmatrix} 5 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 2 & 1 & 0 & 1 & 1 \\ 2 & 0 & 2 & 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix};$$

$$H_{19} = \begin{pmatrix} 4 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 6 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 3 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$H_{20} = \begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 2 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 4 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 4 & 0 & 1 & 1 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$H_{21} = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix};$$

$$H_{22} = \begin{pmatrix} 5 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 3 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}; \qquad (4.23)$$

It is important to note that the heuristic provides simple guidelines for construction of good protographs. We could end up in different resultant protographs from the same starting matrix, because there exist many possible ways of coming up with the dominant column and making the graph sparse. Also, there could be protographs, which do not obey all the properties stated in the heuristic and still have good threshold.

CHAPTER 5

Conclusion and Future Work

In the final chapter, we conclude by describing the progress made towards our goal of understanding the properties of capacity achieving protograph codes and their construction. We also suggest possible pathways for future research that can build up on our current work.

5.1 Conclusion

The aim of the thesis has been construction of capacity achieving protograph codes. In the first chapter, we introduced protograph codes, using which LDPC codes are generated by the copy-permute operation. In the subsequent chapters, we presented some basic properties of the Density Evolution curves, based on which we came up with some simple closed upper form bounds on protograph thresholds, which helped us in understanding and designing good protograph codes. We used these closed form bounds to construct optimal small sized protographs. Using ideas from these closed form bounds and observed properties of good protographs, we came up with a heuristic to improve the threshold of a given protograph code, which we extended for construction of good protograph codes from a generic protograph (like an all-three protograph matrix) as the starting point.

The heuristic algorithm was motivated by the common features observed in good threshold protographs. They act as a guideline for construction of good protographs, but there could be protographs with good threshold which don't follow all the properties stated in the heuristic (say, the dominant column property). Nevertheless, the protographs generated using the heuristic have generally very good thresholds (around 0.47-0.48 for C=0.5). We showed examples of protographs constructed using our heuristic for different protograph sizes, which were observed to have good thresholds for a BEC.

5.2 Suggestions for Future Work

The heuristic algorithm proposed in our work is based on the observed properties of good protograph codes. While the justification of the steps followed in heuristic have a mathematical basis, rigorous mathematical analysis could be done to further strengthen the arguments.

Another possible line of work could be in characterizing the sparse portion of the matrix of capacity achieving protograph codes. Analysis could be done on how a change in the degree of a variable node in the sparse region of a protograh affects the threshold of the protograph.

Future research could also be along the lines of observing the performance of the codes generated by the heuristic algorithm on other channels like the BI-AWGN channel, Gaussian channel or the BSC channel, which can give a better sense of the generalizability of the algorithm.

5.3 Summary

To sum up, we have come up with simple closed form upper bounds for protograph thresholds. We proposed a simple heuristic algorithm that allows us to improve the threshold of a given protograph and also construct good protographs with thresholds close to capacity using a few simple steps.

APPENDIX A

Matlab Codes

A.1 Code for Computing Threshold of Protograph

The following code computes the threshold of a protograph. It has three components. The first snippet given below performs the iterative DE on a protograph for a given threshold.

```
% Performs the iterative density evolution on a protograph for a
    given
% threshold

function [x,xmat,status,conv] = proto_de_iter(A,eps,Niter)

    [m,n]=size(A);
    xmat=zeros(Niter,m*n);
    x=zeros(1,m*n);
    y=zeros(1,m*n);
    x(:)=eps;
    xmat(1,:)=x;
    for k=2:Niter
        x_old=x;
        for i=1:m
            x1=x((i-1)*n+1:(i)*n);
            x2=(1-x1).^A(i,:);
            y((i-1)*n+1:(i)*n)=1-(prod(x2)./(1-x1));
        end

        for j=1:n
            y1=y(j:n:end);
            y2=y1.^(A(:,j)');
            y3=eps*(prod(y2))./y1;
            y3(y1==0)=0;
            x(j:n:end)=y3;
        end
        xmat(k,:)=x;

        if isempty(find(x>10^-15))
        %if sum(find(x<10^-12)) > 1
            conv=k;
            status=1;
            return;
        end
    end
    status=0;
    conv=Niter;
end
```

The second part computes the threshold of a given protograph

```

%Function to find the threshold of a given protograph by checking for
%convergence to zero of the erasure probabilities

function [eps]=proto_thresh_bec_brute(A)
    step=0.01;
    Niter=2000;
    count=0;

    for eps=0:step:1
        [x,xmat,status,complete]=proto_de_iter(A,eps,Niter);
        if status==0
            break
        end
    end
    while count<10
        step=step/2;
        if status ~= 0
            eps = eps+step;
        else
            eps = eps-step;
        end
        [x,xmat,status,complete]=proto_de_iter(A,eps,Niter);
        count=count+1;
    end
    if status == 1
        eps=eps+step;
    end
end
```

The following function plots the DE equations for the given protograph, when probability of erasure is less than threshold, to give an idea of the evolution of DE equations.

```

% Finding the threshold and plotting the DE curves

A=  [0 4 1 0 0 0 1 0 0 2 0 1 0 0 0 0;
     2 4 1 0 1 0 1 0 0 0 0 0 0 0 0 0;
     0 4 0 1 0 1 0 0 0 0 1 0 0 1 0 0;
     0 4 0 0 1 0 1 0 1 0 0 0 1 0 1 0;
     0 4 0 0 0 0 0 1 0 0 0 0 1 1 1 2;
     0 3 0 2 0 0 0 1 1 0 2 0 0 0 0 0;
     0 4 0 0 0 1 0 0 0 0 0 1 0 0 1 0;
     1 3 0 0 2 0 0 0 1 0 0 0 1 0 0 0] ;

[M,N]=size(A);
Niter=2000;

eps=proto_thresh_bec_brute(A)

epsp = eps/8;
[x,xmat,status ,complete]=proto_de_iter(A,epsp ,Niter); %Plot DE when
    erasure less than threshold

xmat=xmat(1:complete ,:)';
figure;        %Plotting the DE functions
hold on;
for i=1:size(xmat,1)
    j=i-1;

    a1=floor(j/N)+1;
    a2=mod(j,N)+1;
    if A(a1,a2) ~=0
        p3=semilogy(1:complete,xmat(i,:),'b');  %Plot DE only for
    valid edges
    end
end

ylim([0,epsp*1.1]);
xlabel('Iteration Number');
ylabel('Edge Error Probability');
```

A.2 Code for Protograph Construction based on the Heuristic

```
Hinit=[4,1,0,3,3,1,1,2;
    5,0,1,0,1,0,0,0;
    4,0,1,0,0,1,4,0;
    6,1,0,0,0,1,0,1];    %Starting Protograph Matrix

%proto_thresh_bec_brute(H)
[M,N]=size(Hinit);

allowed_sum=[2];
desired_sum=2:3;
nox=1;
H=Hinit;
thresh=proto_thresh_bec_brute(H)  %Compute protograph threshold
thresh_best = thresh;
Hbest=Hinit;
allindex=1;
allthresh=zeros(1,200);
while thresh < 0.485
    H=Hinit;
    j={zeros(1,M)};
    l=1;
    for i=2:N
        flag=0;
        k=0;
        if mod(k,100)==1
            k
        end
        z=H(:,i)';
        if ( sum(find(allowed_sum==sum(z))) || (length(find(z~=0))
==1) ) && (sum(z)==2)  %Check for Degree 2 column
            l=l+1;
            j{l}=z;
            v=2;
            linit=l;
            while v<=linit
                jnew=mod(j{v}+z,2);   %To ensure no degree two cycle
                if sum(jnew)==2
                    l=l+1;
                    j{l}=jnew;
                end
                v=v+1;
            end
            continue;
        end
        % The following performs
        % Generating the random column and arriving at a 2-4 degree
bitnode
        % Pushing the entries to the dominant column
        while ~flag
            x1=rand(1,length(z)).*z;
            x1=round(x1);
            sx1=sum(x1);
            p=z-x1;
            t=find(p~=0);

            if sum(find(desired_sum==sum(z)-sx1)) || ( (length(t)==1)
 && sum(find(p(t)==3:4))) %
                if ~(length(t)==1 && p(t)==2)
                    jmat=reshape(cell2mat(j),M,[])';
                    check=any( (jmat-repmat(p,1,1))' );
                    if ~sum(find(check==0))
                        if (sum(p)==2)
                            l=l+1;
                        j{l}=p;
                        v=2;
                        linit=l;
                        while v<=linit
                            jnew=mod(j{v}+p,2);
                            if sum(jnew)==2
                                l=l+1;
                                j{l}=jnew;
                            end
                            v=v+1;
                        end
                    end
                    H(:,i)=p';
                    H(:,1)=H(:,1)+x1';
                    flag=1;
                end
            end
        end
    end
end

% Reducing the entries in the dominant column in accordance with the
% 3M-3 rule
f=H(:,1);
[val0,index]=sort(sum(H'),'descend');
val0=val0';
val=H(index,1);
sorted_colsum=sort(sum(H),'descend');
finer=find(sorted_colsum(1)==sorted_colsum);
colsum=sorted_colsum(1)-sorted_colsum(max(finer)+1);   % Difference in max and second max degree bit node
l=1;
while colsum > N + randi([-3,3],1,1) && sum(find(~(val==0)))
    l=l+1;
    if mod(l,100)==1
        l
    end

    i0=1;
```

```
        i1=find(val0(i0)==val0);
        if length(i1) >= M  % All check node degree same
            break;
        end
        i2=max(i1)+1;   %Next lowrst check node degree
        t=(val0(i0)-val0(i2));
       %Reduction step
        if t >= min(val(i1))    %If check node difference greater than least entry in larger check node row
            diff=min(val(i1));
            colsum=colsum-length(i1)*diff;
            val0(i1)=val0(i1)-repmat(diff,length(i1),1);
            val(i1)=val(i1)-repmat(diff,length(i1),1);
            break;
        else
            diff=(val0(i0)-val0(i2));
            colsum=colsum-length(i1)*diff;
            val0(i1)=val0(i1)-repmat(diff,length(i1),1);
            val(i1)=val(i1)-repmat(diff,length(i1),1);
        end
    end
    new1=min(round((colsum-N)/M),min(val));  %Implement 3M-3 rule
    if new1 > 0
        val=val-repmat(new1,M,1);
    end
    g1=randi([0 3],1,1);
    val(1)=val(1)+g1;
    if g1
        val(2)=val(2)+randi([0 1],1,1);
    end
    H(index,1)=val;
    nox=nox+1
    thresh=proto_thresh_bec_brute(H)
    if thresh > thresh_best
        thresh_best=thresh;
        Hbest=H;
    end
end
```

REFERENCES

1. **Liva**, **Gianluigi**, and **M.Chiani**, Protograph ldpc codes design based on exit analysis. *In Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*. 2007.

2. **Pradhan, A. K.**, **A. Subramanian**, and **A. Thangaraj** (2013). Deterministic constructions for large girth protograph LDPC codes. *CoRR*, **abs/1301.6301**. URL `http://arxiv.org/abs/1301.6301`.

3. **Richardson, T.** and **R. Urbanke** (2001). The capacity of low-density parity-check codes under message-passing decoding. *Information Theory, IEEE Transactions on*, **47**(2), 599–618. ISSN 0018-9448.

4. **Richardson, T.** and **R. Urbanke**, *Modern Coding Theory*. Cambridge University Press, 2008.

5. **Srinivasan, M.** (2015). Design and analysis of high-performance protograph based ldpc codes.

6. **Thorpe, J.** (2003). Low-density parity-check (ldpc) codes constructed from protographs. *IPN progress report*, **42**(154), 42–154.

7. **Yeh, J. J.**, *Real Analysis: Theory Of Measure And Integration*. World Scientific Publishing Co. Pte. Ltd, 2006.