

# LDPC Codes, Construction and Applications

Manu T S

Advisor : Prof. Sibi Raj B Pillai

IIT Bombay

October 31, 2013

# What is an LDPC Code

## A linear error correcting code

All codewords should satisfy

$$\mathbf{H}\mathbf{x} = \mathbf{0} \quad (1)$$

Any vector that satisfy above equation is a codeword

## A block code

If  $\mathbf{H}$  is  $M \times N$  and has rank  $R$  then the code has dimension

$$K = N - R$$

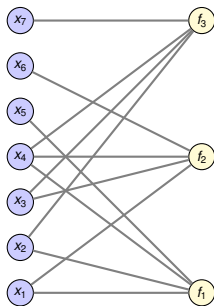
Block length of the code is  $N$ .

## Sparse $\mathbf{H}$

Here we are dealing with binary LDPC codes, so  $\mathbf{H}$  has few 1s rest all zeros

# Tanner Graph

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$



(a)

# Degree Distribution

Let  $\lambda$  and  $\rho$  be vectors such that their  $i^{th}$  component  $\lambda_i$  and  $\rho_i$  represent the fraction of edges connecting to a variable node of degree  $i$  and check node of degree  $i$  respectively. Thus

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \quad \text{and} \quad \rho(x) = \sum_i \rho_i x^{i-1}$$

are called variable and check degree distribution polynomials respectively.

- Regular LDPC Codes
- Irregular LDPC Codes
- Ensemble LDPC( $n, \lambda, \rho$ )

# Marginalization by Message Passing

For

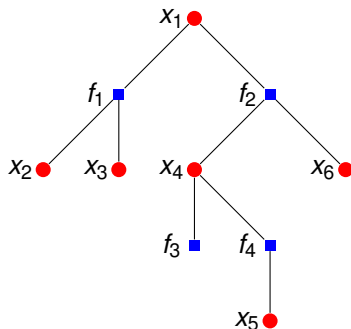
$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

$$\begin{aligned} f(x_1) &= \sum_{x_2, x_3, x_4, x_5, x_6} f(x_1, x_2, x_3, x_4, x_5, x_6) \\ &= \left[ \sum_{x_2, x_3} f_1(x_1, x_2, x_3) \right] \left[ \sum_{x_4} f_3(x_4) \left( \sum_{x_6} f_2(x_1, x_4, x_6) \right) \left( \sum_{x_5} f_4(x_4, x_5) \right) \right] \end{aligned}$$

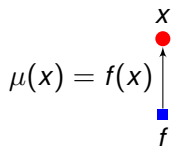
# Factor Graph

For

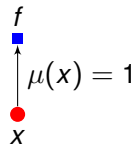
$$\left[ \sum_{x_2, x_3} f_1(x_1, x_2, x_3) \right] \left[ \sum_{x_4} f_3(x_4) \left( \sum_{x_6} f_2(x_1, x_4, x_6) \right) \left( \sum_{x_5} f_4(x_4, x_5) \right) \right]$$



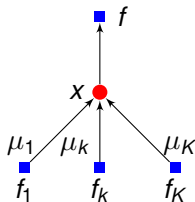
# Message Passing Rules



initialization at  
leaf nodes

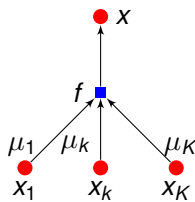


$$\mu(x) = \prod_{k=1}^K \mu_k(x)$$

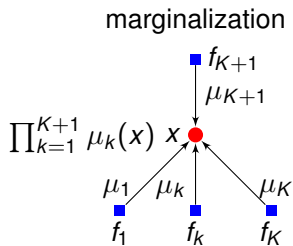


$$\mu(x) = \sum_{\sim x} f(x, x_1, \dots, x_K) \prod_{k=1}^K \mu_k(x)$$

variable/check  
node processing

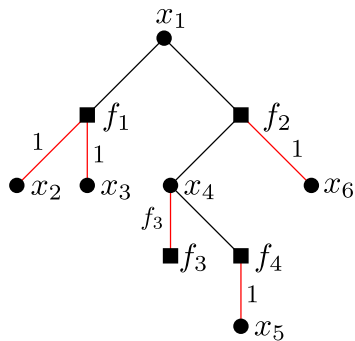
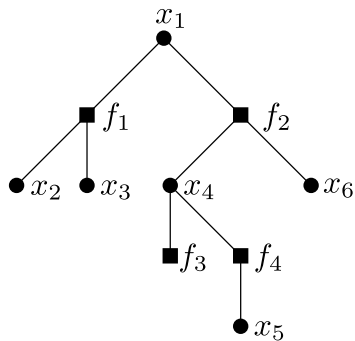


# Message Passing Rules

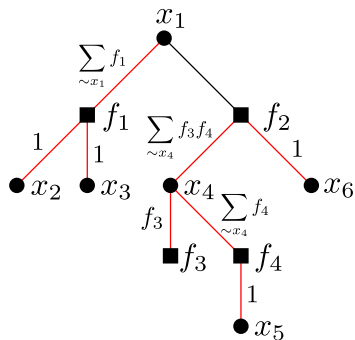
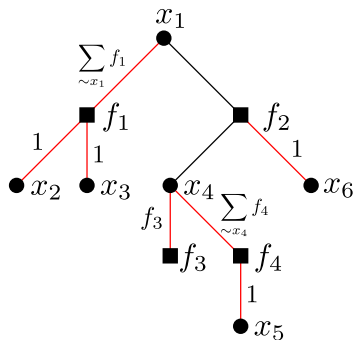




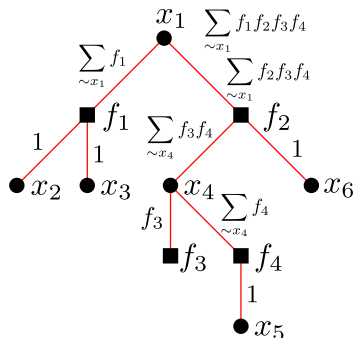
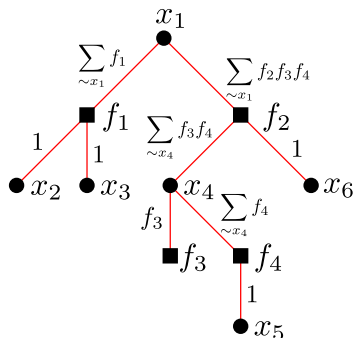
# Message Passing Example



# Message Passing Example



# Message Passing Example



# Belief Propagation Decoding

Bitwise maximum a posteriori (MAP) decoding for memoryless channel without feedback.

$$\hat{x}_i(y) = \arg \max_{x_i} p_{X_i|Y}(x_i|y) \quad (3)$$

$$= \arg \max_{x_i} \sum_{\sim x_i} p_{X|Y}(x|y) \quad (4)$$

$$= \arg \max_{x_i} \sum_{\sim x_i} p_{Y|X}(y|x) p_X(x) \quad (5)$$

$$= \arg \max_{x_i} \sum_{\sim x_i} \prod_j p_{Y_j|X_j}(y_j|x_j) p_X(x) \quad (6)$$

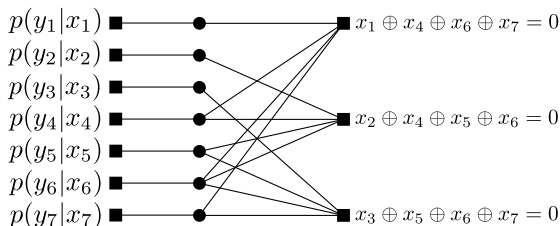
$$= \arg \max_{x_i} \sum_{\sim x_i} \prod_j p_{Y_j|X_j}(y_j|x_j) \mathbf{1}_{\{x \in C\}} \quad (7)$$

# An Example

Given the parity check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad (8)$$

$$\hat{x}_i(y) = \arg \max_{x_i} \sum_{\sim x_i} \prod_j p_{y_j|x_j}(y_j|x_j) \mathbf{1}_{\{x_1+x_4+x_6+x_7=0\}} \mathbf{1}_{\{x_2+x_4+x_5+x_6=0\}} \mathbf{1}_{\{x_3+x_5+x_6+x_7=0\}} \quad (9)$$



# Message Passing Rules - Simplifications

- In the binary case the message is  $(\mu(1), \mu(-1))$ .
- $(\mu(1), \mu(-1)) \leftarrow (p_{y_i|x_i}(y_i|1), p_{y_i|x_i}(y_i|-1))$
- At a variable node of degree  $K + 1$ ,

$$\mu(1) = \prod_{k=1}^K \mu_k(1), \quad \mu(-1) = \prod_{k=1}^K \mu_k(-1)$$

- $r_k \leftarrow \mu_k(1)/\mu_k(-1)$

$$r = \frac{\mu(1)}{\mu(-1)} = \frac{\prod_{k=1}^K \mu_k(1)}{\prod_{k=1}^K \mu_k(-1)} = \prod_k r_k$$

- If  $l_k = \log(r_k)$ , processing rule becomes  $l = \sum_k l_k$

# Message Passing Rules - Simplifications

- At a check node of degree  $J + 1$  the computation is

$$\mu(x) = \sum_{x_1, x_2, \dots, x_J} \mathbf{1}_{\{\prod_{j=1}^J x_j = x\}} \prod_{j=1}^J \mu_j(x_j) \quad (10)$$

$$r = \frac{\mu(1)}{\mu(-1)} = \frac{\sum_{\{x_1, x_2, \dots, x_J: \prod_{j=1}^J x_j = 1\}} \prod_{j=1}^J \mu_j(x_j)}{\sum_{\{x_1, x_2, \dots, x_J: \prod_{j=1}^J x_j = -1\}} \prod_{j=1}^J \mu_j(x_j)} \quad (11)$$

$$= \frac{\sum_{\{x_1, x_2, \dots, x_J: \prod_{j=1}^J x_j = 1\}} \prod_{j=1}^J \frac{\mu_j(x_j)}{\mu_j(-1)}}{\sum_{\{x_1, x_2, \dots, x_J: \prod_{j=1}^J x_j = -1\}} \prod_{j=1}^J \frac{\mu_j(x_j)}{\mu_j(-1)}} \quad (12)$$

# Message Passing Rules - Simplifications

$$r = \frac{\sum_{\{x_1, x_2, \dots, x_J: \prod_{j=1}^J x_j = 1\}} \prod_{j=1}^J r_j^{(1+x_j)/2}}{\sum_{\{x_1, x_2, \dots, x_J: \prod_{j=1}^J x_j = -1\}} \prod_{j=1}^J r_j^{(1+x_j)/2}} \quad (13)$$

$$= \frac{\prod_{j=1}^J (r_j + 1) + \prod_{j=1}^J (r_j - 1)}{\prod_{j=1}^J (r_j + 1) - \prod_{j=1}^J (r_j - 1)} \quad (14)$$

$$= \frac{1 + \frac{\prod_{j=1}^J (r_j - 1)}{\prod_{j=1}^J (r_j + 1)}}{1 - \frac{\prod_{j=1}^J (r_j - 1)}{\prod_{j=1}^J (r_j + 1)}} \quad (15)$$

$$\Rightarrow \frac{r - 1}{r + 1} = \prod_{j=1}^J \frac{r_j - 1}{r_j + 1} \quad (16)$$



# Message Passing Rules - Simplifications

$$r = \exp(l) \quad (17)$$

$$\Rightarrow \tanh(l/2) = \frac{r - 1}{r + 1} \quad (18)$$

$$= \prod_{j=1}^J \frac{r_j - 1}{r_j + 1} \quad (19)$$

$$= \prod_{j=1}^J \tanh(l_j/2) \quad (20)$$

$$\Rightarrow l = 2 \tanh^{-1} \left( \prod_{j=1}^J \tanh(l_j/2) \right) \quad (21)$$

# Construction of Parity Check Matrices

- Ideally we need tree.
- In practice we try to achieve a minimum girth in the tanner graph.
- Construction of Parity Check matrix from Reed Solomon codes.
  - Minimum girth of 4 is guaranteed.
  - Constructs a regular LDPC code.
- Progressive Edge Growth construction
  - Starts with a graph with no edges.
  - Progressively introduce edges such that it has minimum effect on the girth
  - Outputs both regular and irregular LDPC codes.

# Encoding

If the parity check matrix is given as  $[\mathbf{I} \ \mathbf{P}]$ , where  $\mathbf{I}$  is  $N - K \times N - K$  identity matrix,  $\mathbf{P}$  is some  $N - K \times K$  matrix, a codeword  $\mathbf{x} = [\mathbf{x}_p^T \ \mathbf{x}_d^T]^T$  can be formed by assigning

$$\mathbf{x}_p = \mathbf{P} \cdot \mathbf{x}_d$$

We obtain  $\mathbf{G} = [\mathbf{I} \ \mathbf{P}]$  from  $\mathbf{H}$  by

- Column permutation
- Row additions
- Row permutations

Suppose  $f$  represents the permutation in obtaining  $\mathbf{G}$  from  $\mathbf{H}$ , and  $\mathbf{G} \cdot \mathbf{x} = \mathbf{0}$ , then  $\mathbf{H} \cdot f(\mathbf{x}) = \mathbf{0}$

# Belief Propagation Decoding for 2 User Gaussian MAC

$$\hat{x}_i^{[1]} \triangleq \arg \max_{x_i} p_{X_i^{[1]}|Y}(x_i^{[1]}|y) \quad (22)$$

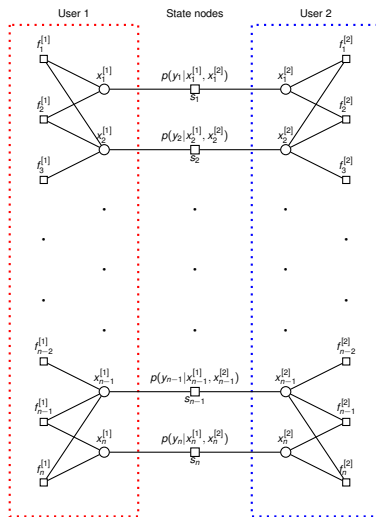
$$= \arg \max_{x_i} \sum_{\sim x_i^{[1]}} p_{X^{[1]}, X^{[2]}|Y}(x^{[1]}, x^{[2]}|y) \quad (23)$$

$$= \arg \max_{x_i} \sum_{\sim x_i^{[1]}} p_{Y|X^{[1]}, X^{[2]}}(y|x^{[1]}, x^{[2]}) p_{X^{[1]}, X^{[2]}}(x^{[1]}, x^{[2]}) \quad (24)$$

$$= \arg \max_{x_i} \sum_{\sim x_i^{[1]}} p_{Y|X^{[1]}, X^{[2]}}(y|x^{[1]}, x^{[2]}) p_{X^{[1]}}(x^{[1]}) p_{X^{[2]}}(x^{[2]}) \quad (25)$$

$$= \arg \max_{x_i} \sum_{\sim x_i^{[1]}} \prod_j p_{Y_j|X_j^{[1]}, X_j^{[2]}}(y_j|x_j^{[1]}, x_j^{[2]}) \mathbf{1}_{\{x^{[1]} \in \mathcal{C}^{[1]}\}} \mathbf{1}_{\{x^{[2]} \in \mathcal{C}^{[2]}\}} \quad (26)$$

# Factor Graph for Joint Decoding



# Message Passing Rules for Joint Decoding

- At variable nodes and at Check nodes the rules are the same.
- Rule for the function node

$$\mu_{s_i \rightarrow x_i^{[2]}} = \log \left( \frac{\exp(\mu_{x_i^{[1]} \rightarrow s_i}) p(y|x_i^{[1]} = -1, x_i^{[2]} = -1) + p(y|x_i^{[1]} = 1, x_i^{[2]} = -1)}{\exp(\mu_{x_i^{[1]} \rightarrow s_i}) p(y|x_i^{[1]} = -1, x_i^{[2]} = 1) + p(y|x_i^{[1]} = 1, x_i^{[2]} = 1)} \right) \quad (27)$$

$$\mu_{s_i \rightarrow x_i^{[1]}} = \log \left( \frac{\exp(\mu_{x_i^{[2]} \rightarrow s_i}) p(y|x_i^{[1]} = -1, x_i^{[2]} = -1) + p(y|x_i^{[1]} = -1, x_i^{[2]} = 1)}{\exp(\mu_{x_i^{[2]} \rightarrow s_i}) p(y|x_i^{[1]} = 1, x_i^{[2]} = -1) + p(y|x_i^{[1]} = 1, x_i^{[2]} = 1)} \right) \quad (28)$$

## Results - Bit Error Rate Performance

(3, 6) Regular Code of block length  $N = 96$ , dimension  $K = 50$ ,  
Number of parity checks  $M = 48$ ,  $10^7$  Bytes Transferred

	No Code		With LDPC	
sigma	Errors	BER	Errors	BER
0.3	4338	0.0004338	0	0
0.4	61814	0.0061814	0	0
0.5	227228	0.0227228	31	3.1e-06
0.6	477744	0.0477744	65855	0.0065855
0.7	765326	0.0765326	572393	0.0572393
0.8	1055848	0.1055848	968856	0.0968856
0.9	1332039	0.1332039	1291208	0.1291208
1.0	1585653	0.1585653	1565803	0.1565803

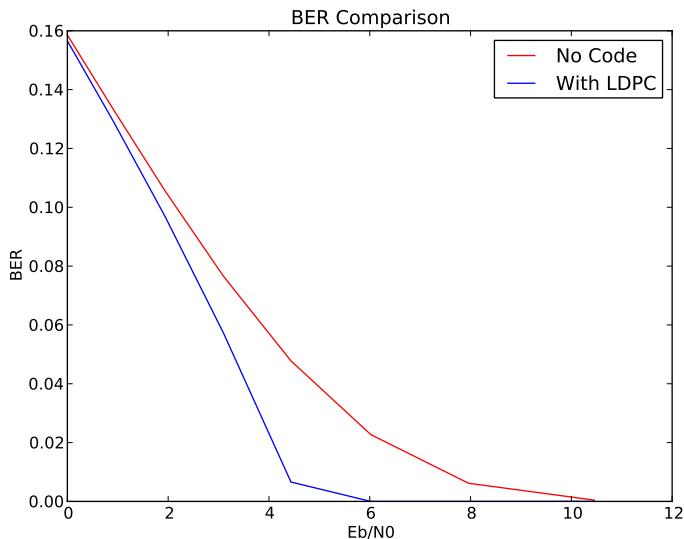
# Results - Block Error Rate Performance

LDPC generated using PEG Algorithm,  
Block length  $N = 1008$ , Dimension  $K = 504$   
Number of Parity Checks  $M = 504$

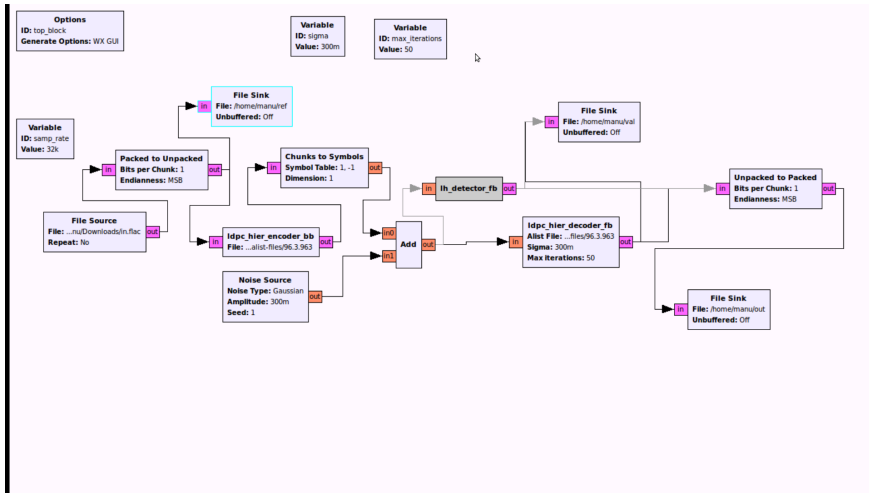
	No Code		With LDPC	
sigma	Errors	BER	Errors	BER
0.3	3898	0.196461	0	0
0.4	18938	0.954488	0	0
0.5	19840	0.999994	0	0
0.6	19841	1.0	17	0.0008568
0.7	19841	1.0	19841	1.0



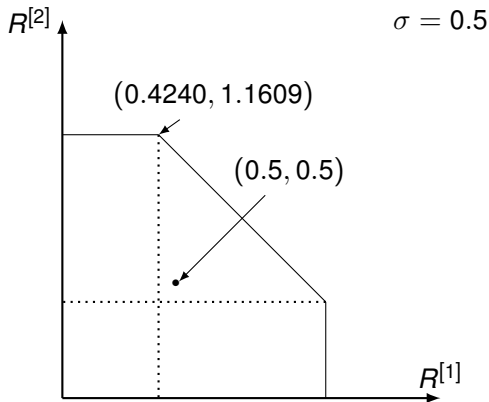
# Results - Plots



# gr-ldpc module in GNU Radio



## Rate - Capacity comparison



For single user at  $\sigma = 0.8$  (capacity = 0.678) all blocks were decoded correctly.

# Future Work

- Study of Density Evolution and EXIT Charts.
- 2 users with sample offset of 0.5 symbol duration.
- Encoder using sparse LU decomposition
- Using VOLK