LDPC Codes: Decoding

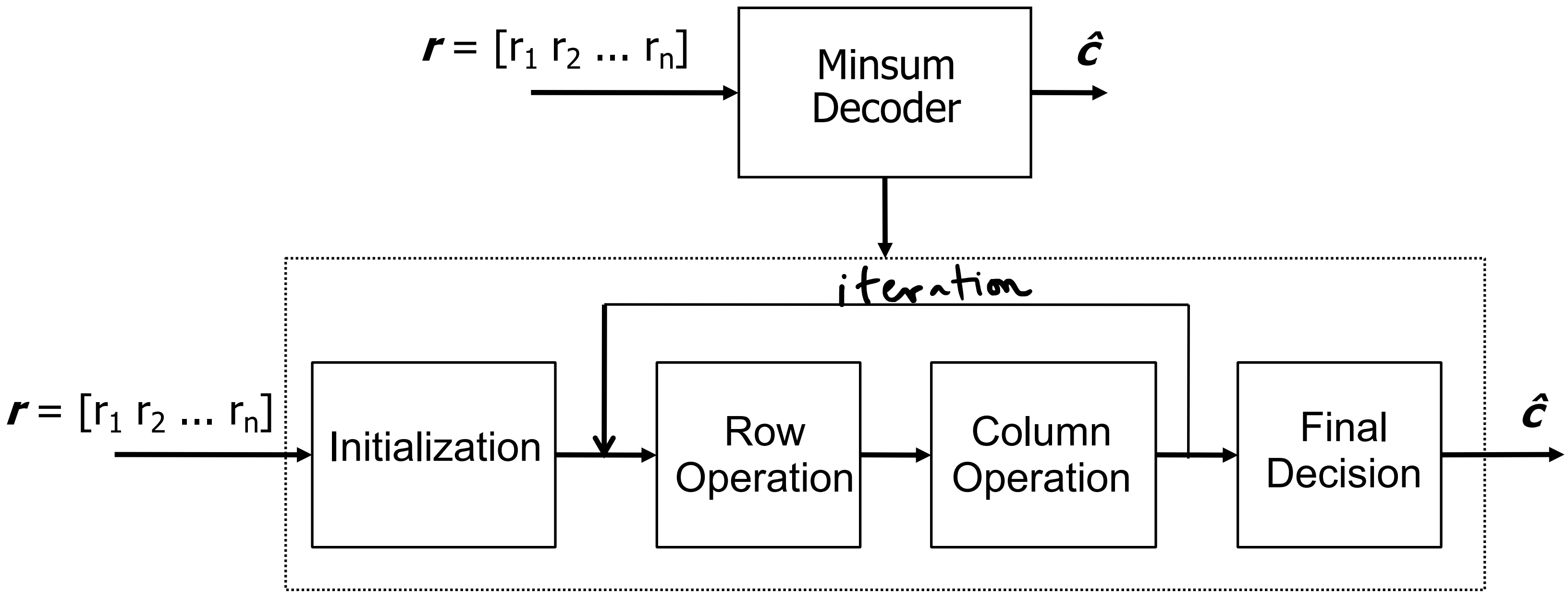
Andrew Thangaraj
andrew@iitm.ac.in

Toy Example for Illustration

$$H = \begin{array}{cc} & \begin{array}{ccccccc} & & & 7 & & & \end{array} \\ 4 & \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

- We will use above matrix for illuatration
- Typically, a much larger matrix is used!

Minsum Decoder



- Iterates row and column operations after initialization

Storage Matrix L

- L: Sparse matrix of same dimensions as parity-check matrix
  - NR, rate-1/2: L is a (22 x 46) * 48 sparse matrix
  - Toy example: L is a 4 x 7 matrix
- L[i,j]=0 if H[i,j]=0
  - L[i,j] can be nonzero only if H[i,j]=1

in-place computation using L

Toy Example for Illustration

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} x_1 & x_2 & x_3 & 0 & x_4 & 0 & 0 \\ 0 & x_5 & x_6 & x_7 & 0 & x_8 & 0 \\ x_9 & x_{10} & 0 & x_{11} & 0 & 0 & x_{12} \\ x_{13} & 0 & x_{14} & 0 & x_{15} & x_{16} & x_{17} \end{bmatrix}$$

17 non-zero values

Initialization Step

Channel $\longrightarrow$ $\boldsymbol{r} = [\, r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5 \quad r_6 \quad r_7 \,]$

1st col

$$L = \begin{bmatrix} r_1 & r_2 & r_3 & 0 & r_5 & 0 & 0 \\ 0 & r_2 & r_3 & r_4 & 0 & r_6 & 0 \\ r_1 & r_2 & 0 & r_4 & 0 & 0 & r_7 \\ r_1 & 0 & r_3 & 0 & r_5 & r_6 & r_7 \end{bmatrix}$$

- L[N[j,i], j]=r_j for j=1,2,..,n and for i=1,2,...,t[j]

Row Operation

In-place computation using L

- For each row, minsum SPC SISO
  - Magnitude
    - Min1 = minimum absolute value of all nonzero entries in row
    - Min2 = next higher absolute value
    - Set magnitude of all values (except minimum) = Min1
    - Set magnitude of minimum value = Min2
  - Sign
    - Parity = product of signs of entries in row
    - New sign of an entry = (Old Sign) (Parity)

Toy Example for Illustration

<u>Initialization</u>

r = [0.2 -0.3 1.2 -0.5 0.8 0.6 -1.1]

$$L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

Toy Example for Illustration

$$L = \begin{bmatrix} \underset{\text{min1}}{0.2} & \underset{\text{min2}}{-0.3} & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix} \begin{array}{l} \overset{\text{parity}}{\leftarrow -1} \\ \\ \\ \\ \end{array}$$

<u>Row Operation on Row 1</u>

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

Toy Example for Illustration

$$L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & \underset{\text{min1}}{-0.3} & 1.2 & \underset{\text{min2}}{-0.5} & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix} \begin{array}{l} \\ \text{parity} \leftarrow +1 \\ \\ \\ \end{array}$$

<u>Row Operation on Row 2</u>

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

Toy Example for Illustration

$$L=\begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

<u>After Row Operation</u>

$$L=\begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

Column Operation

In-place computation using L

- For each column, Repetition code SISO
  - New values
    - $Sum_j = r_j +$ sum of all entries in Column j ← total "belief"
    - New Entry = Sum – (Old entry) extrinsic

Toy Example for Illustration

<u>Column Operation on Column 1</u>

r = [0.2 -0.3 1.2 -0.5 0.8 0.6 -1.1]

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

Sum = -1

$-1-(-0.3)$

$-1-(-0.3)$

$-1-(-0.6)$

$$L = \begin{bmatrix} -0.7 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.7 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.4 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

Toy Example for Illustration

Column Operation on Column 2

$$
\boldsymbol{r} = [\ 0.2 \quad \boxed{-0.3} \quad 1.2 \quad -0.5 \quad 0.8 \quad 0.6 \quad -1.1\ ]
$$

$$
L = \begin{bmatrix} -0.7 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.7 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.4 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}
$$

Sum =[-1 -0.4]

$$
L = \begin{bmatrix} -0.7 & -0.6 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & 0.1 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.7 & -0.6 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.4 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}
$$

Toy Example for Illustration

After Column Operation

$r =$ [0.2 -0.3 1.2 -0.5 0.8 0.6 -1.1] ← from channel

$\hat{c} =$ [0 1 0 1 0 0 1

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

Sum =[-1 -0.4 1.1 -0.6 0.4 0.7 -0.7] ← updated belief after 1st iteration

$\hat{c} =$ [1 1 0 1 0 0 1]

$$L = \begin{bmatrix} -0.7 & -0.6 & 1.3 & 0 & 0.6 & 0 & 0 \\ 0 & 0.1 & 0.8 & -0.3 & 0 & 0.4 & 0 \\ -0.7 & -0.6 & 0 & -0.8 & 0 & 0 & -0.9 \\ -0.4 & 0 & 1.3 & 0 & 0.6 & 0.9 & -0.9 \end{bmatrix}$$

Decision

- If $Sum_j > 0$, Decision on Bit j = 0
- If $Sum_j < 0$, Decision on Bit j = 1
  - Assuming BPSK 0 -> +1 and 1 -> -1

After first iteration,

Sum =[	-1	-0.4	1.1	-0.6	0.4	0.7	-0.7]
Dec =[	1	1	0	1	0	0	1]

- For more iterations, continue with new L

Some Observations

- More iterations => Better performance
  - About 5 to 8 is typically enough
- Storage and efficient retrieval from the matrix L is vital for a good implementation