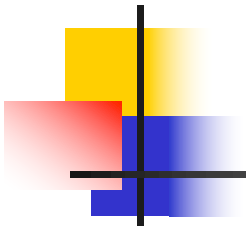
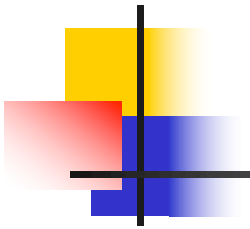
LDPC Codes: Decoding

Andrew Thangaraj
andrew@iitm.ac.in

Low-Density Parity-Check Codes



- Linear codes can be specified using either a generator matrix or a parity-check matrix
- LDPC codes are linear codes with a sparse parity-check matrix
  - Sparse: most of the entries in the matrix are 0s
- Proposed in 60s
- Since 1990's, intense research in the area
- Adopted in many standards
  - Wimax, Wifi, DVB etc
  - Now, 5G too



Decoding

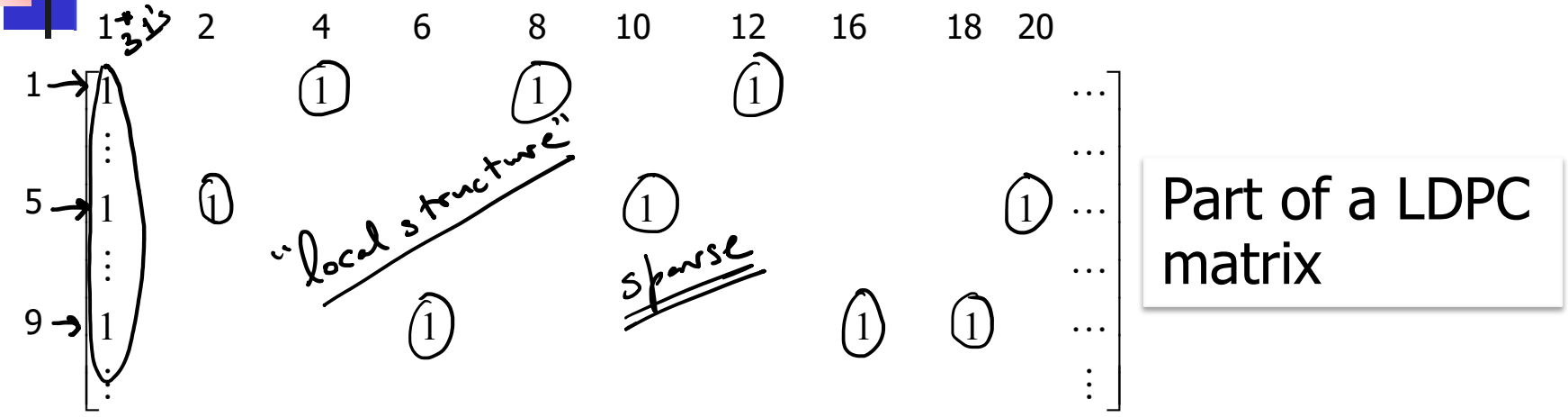
- Soft-input soft-output
- Iterative
- Message passing
  - Approximation: minsum
- Extensive analysis in literature

Soft-decision Decoding

- Process real values received from the channel
- Channel: BPSK over AWGN(σ)
  - $\mathbf{c} = [c_1\ c_2\ \ldots\ c_n]$; $\mathbf{r} = [r_1\ r_2\ \ldots\ r_n]$
  - $r_i = s_i + \text{noise}$
  - c_i : codeword bit {0,1}; $s_i = 1 - 2c_i$
  - LLR, $l_i = \log [\text{Prob}\{c_i=0|r_i\}/\text{Prob}\{c_i=1|r_i\}]$

    $= \log [p(r_i|c_i=0)/p(r_i|c_i=1)]$
  - For BPSK over AWGN, $l_i = 2r_i/\sigma^2$
- We seek
  - $L_i = \log [\text{Prob}\{c_i=0|\mathbf{r}\}/\text{Prob}\{c_i=1|\mathbf{r}\}]$

Decoding one bit



Part of a LDPC matrix

- Use the received value corresponding to Bit 1
  - 1st estimate, $l_{10} = \text{LLR} \mid r_1 = 2r_1 / \sigma^2$
- Use parity checks that involve the bit
  - Row 1: $c_1 + c_4 + c_8 + c_{12} = 0$; 2nd estimate, $l_{11} = \text{LLR} \mid r_4, r_8, r_{12}$
  - Row 5: $c_1 + c_2 + c_{10} + c_{20} = 0$; 3rd estimate, $l_{15} = \text{LLR} \mid r_2, r_{10}, r_{20}$
  - Row 9: $c_1 + c_6 + c_{16} + c_{18} = 0$; 4th estimate, $l_{19} = \text{LLR} \mid r_6, r_{16}, r_{18}$
- Same can be done for all bits

Efficient Calculation of Conditional LLR

- Suppose z = x + y; x,y,z: binary RVs (indep) ← mod 2 (or) XOR
  - $(1 - 2z) = (1 - 2x)(1 - 2y)$ ← same as XOR
  - $(1 - 2p_z(1)) = (1 - 2p_x(1))(1 - 2p_y(1))$
  - $\tanh(l_z/2) = \tanh(l_x/2)\tanh(l_y/2)$
    - $\text{sgn}(l_z) = \text{sgn}(l_x)\,\text{sgn}(l_y)$
    - $\text{abs}(\tanh(|l_z|/2)) = \text{abs}(\tanh(|l_x|/2))\,\text{abs}(\tanh(|l_y|/2))$

Take expectations

$l_x = \log \frac{p_x(0)}{p_x(1)}$

l_y, l_z

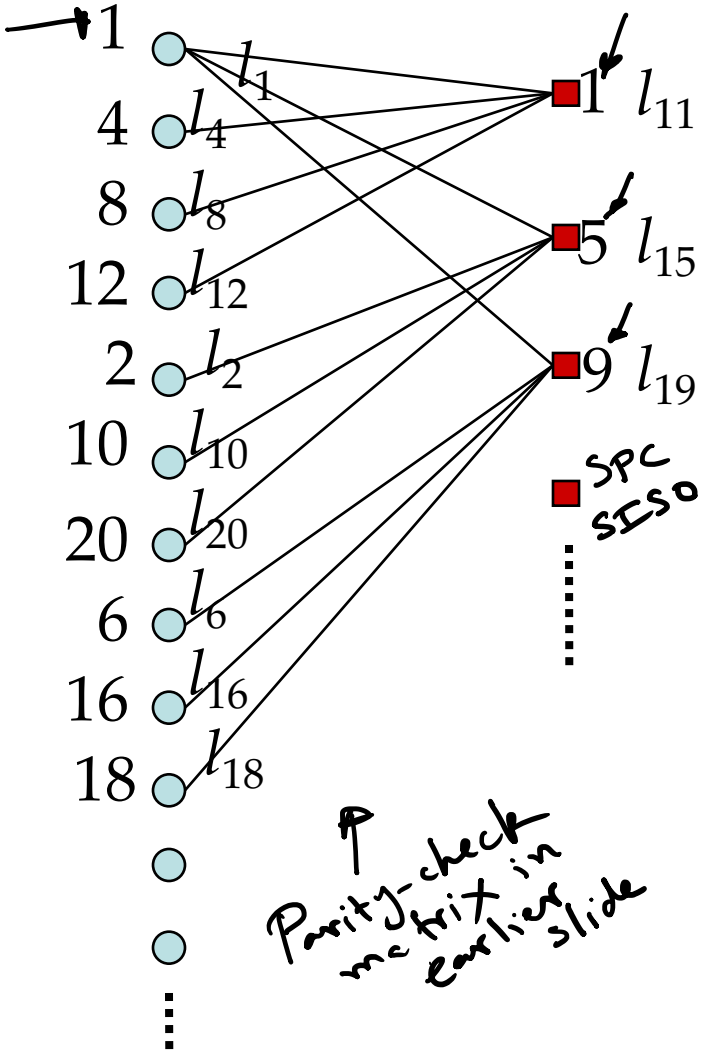
- Suppose $z = x_1 + x_2 + x_3 + \ldots + x_w$
  - $\text{sgn}(l_z) = \text{sgn}(l_1)\,\text{sgn}(l_2) \ldots \text{sgn}(l_w)$
  - $|l_z| = \text{abs}(f\,[\,f(l_1) + f(l_2) + \ldots + f(l_w)\,])$
    - $f(x) = \log \tanh(|x|/2)$

SPC SISO is easy to compute

Minimum approximation

- $|l_z| = \text{abs}(f\,[\, f(l_1) + f(l_2) + \ldots + f(l_w)\,])$
  - $f(x) = \log \tanh (|x|/2)$
- $|l_z| \approx \min\{\text{abs}(l_1), \text{abs}(l_2), \ldots, \text{abs}(l_w)\}$
- Saves computation of nonlinear function
- Used in a slightly modified format
- Almost no loss in the approximation in practice

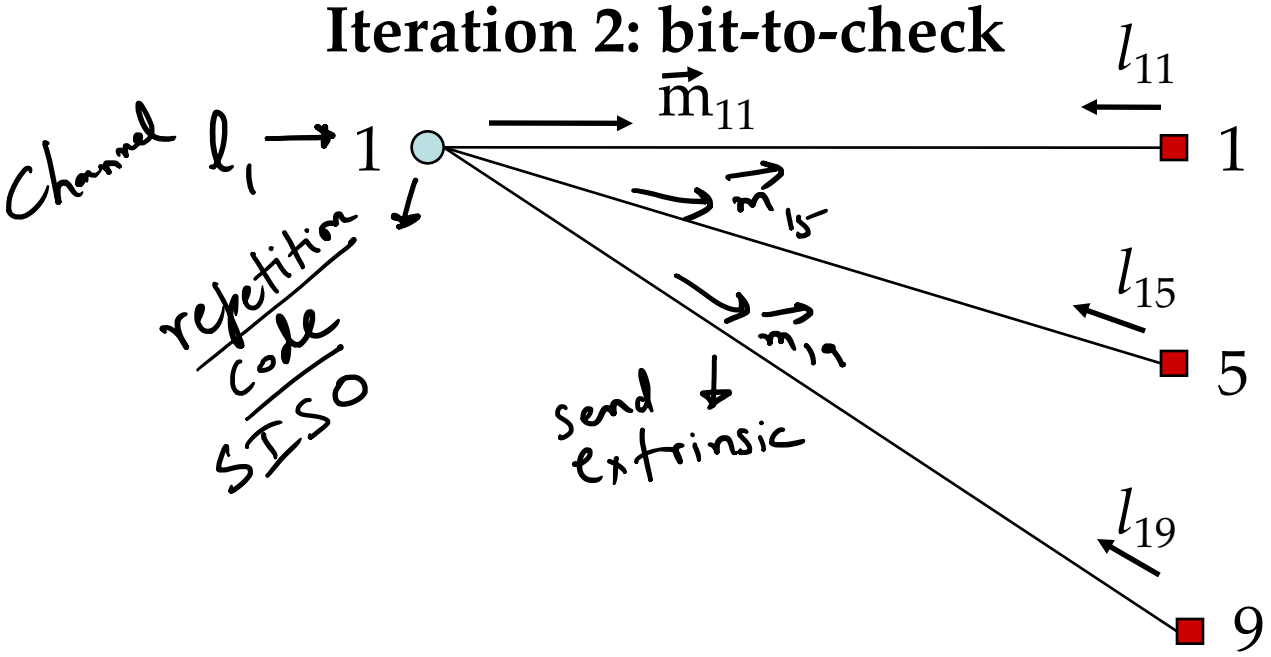
First Iteration in Tanner Graph



- The first estimate l_1 comes from the channel
- l_i : passed from bit node i to all neighbouring check nodes
- Estimates for Bit 1: l_{11}, l_{15}, l_{19} are calculated at check nodes 1, 5, 9
- Estimates passed from check nodes to neighbouring bit nodes
- This is done for all nodes in parallel

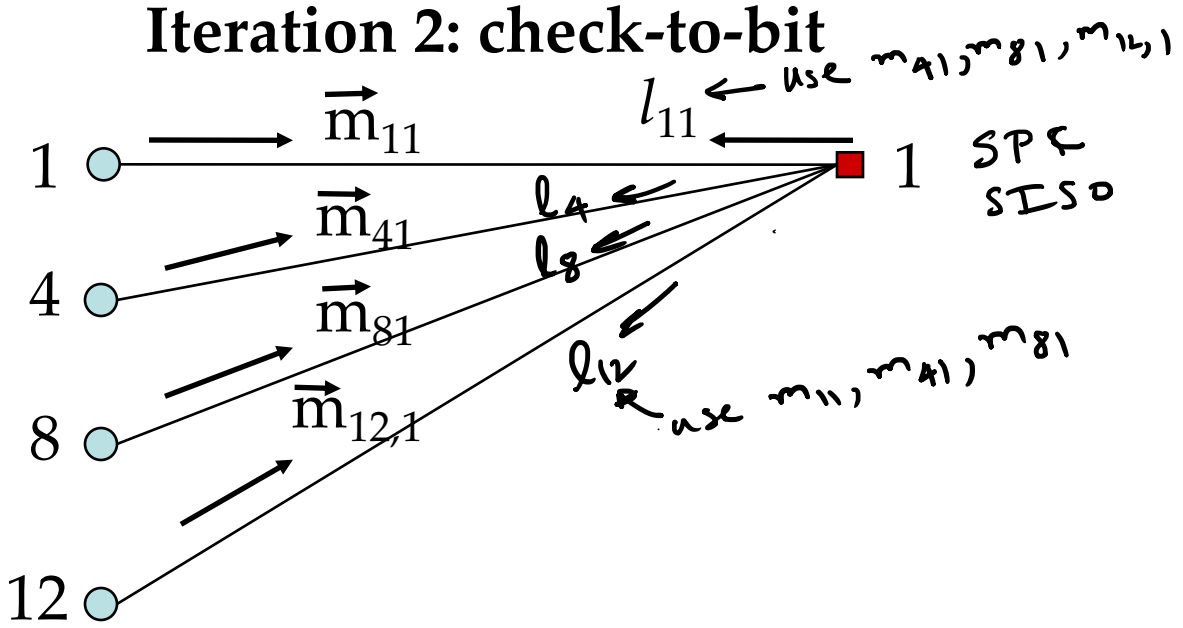
Next Iteration (i)

- Iterations are done to involve more bits and checks in the decoding



- $\vec{m}_{11} = l_1 + l_{15} + l_{19}$; $\vec{m}_{15} = l_1 + l_{11} + l_{19}$; $\vec{m}_{19} = l_1 + l_{11} + l_{15}$
- Similar rule for other bit nodes

Next Iteration (ii)



- Repeat computation of l_{11}, l_{15} and l_{19} using $\vec{m}_{41}$, $\vec{m}_{81}$ and $\vec{m}_{12,1}$
- Similar for all check nodes

Summary

- Row iteration
  - Check node computation SPC SISO
  - Computes conditional LLRs for one bit
- Column iteration
  - Bit node computation Rep SISO
  - Consolidate conditional LLRs
- Repeat for multiple iterations