Лекция 3: Введение в современную теорию кодирования: LDPC коды.

Алексей Фролов

al.frolov@skoltech.ru

Сколковский институт науки и технологий (Сколтех)

Современные методы теории информации, оптимизации и управления

Сочи, Россия

2–23 августа, 2020

Outline

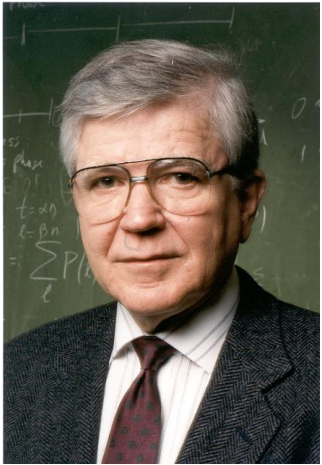
Skoltech
Skolkovo Institute of Science and Technology

Outline

Introduction

- LDPC codes were invented by Robert G. Gallager in the 1960s and forgotten for three decades.
- After Turbo codes were invented 1993, LDPC codes found new attention

Introduction



Robert G. Gallager

LDPC codes

- LDPC codes are defined with use of *sparse* parity-check matrix, i.e. the percentage of 1's in the parity check matrix for a LDPC code is low.
- A regular LDPC code has the property that:
  - every code digit is contained in the same number of equations,
  - each equation contains the same number of code symbols.
- An irregular LDPC code relaxes these conditions.

Definition by parity-check matrix: [Gallager, '62]

$$H = \begin{array}{|cccccccccccccccccccc|}
\hline
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1\\
0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0\\
\hline
0&0&1&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&1&0&0\\
1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1\\
\hline
0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
1&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
\end{array}$$

15×20

Code: $\{\mathbf{v} \mid \mathbf{v}\mathbf{H}^{\mathrm{T}} = \mathbf{0}\}$

(null space of a **sparse** parity-check matrix $\mathbf{H}$)

Definition by parity-check matrix: [Gallager, '62]

$$H = \begin{array}{|cccccccccccccccccccc|}
\hline
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&1\\
0&0&0&0&1&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0&0\\
\hline
0&0&1&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&1&0&0\\
1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1\\
\hline
0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
1&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
\end{array}$$

15×20

Code: $\{\mathbf{v} \mid \mathbf{v}\mathbf{H}^{\mathrm{T}} = \mathbf{0}\}$

(null space of a **sparse** parity-check matrix $\mathbf{H}$)

Regular LDPC code:

Column weight: $J = 3$

Row weight: $K = 4$

$$\boxed{R \geq 1 - \frac{J}{K}}$$

Definition by parity-check matrix: [Gallager, '62]

$$H = \begin{array}{c}
\begin{array}{cccccccccccccccccccc}
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&1\\
0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0\\
0&0&1&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&1&0&0\\
1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&1\\
0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
1&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1
\end{array}\\
15 \times 20
\end{array}$$

Code: $\{\mathbf{v} \mid \mathbf{v}\mathbf{H}^{\mathrm{T}} = \mathbf{0}\}$

(null space of a **sparse** parity-check matrix $\mathbf{H}$)

Regular LDPC code:

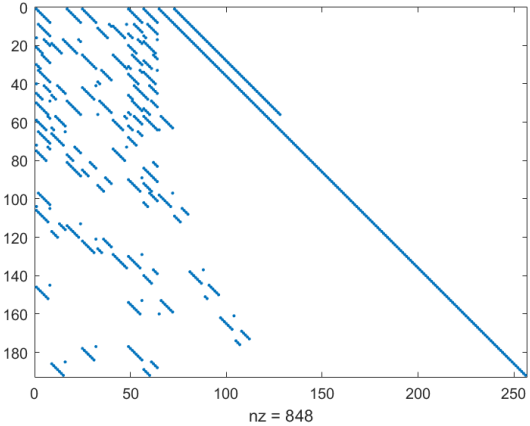
Column weight: $J = 3$

Row weight: $K = 4$

$$\boxed{R \geq 1 - \frac{J}{K}}$$

- If the row and column weights J and K are not constant, then the LDPC code is **irregular** (more later)

Irregular LDPC example, PCM with $R = 1/4$



Representation by bi-partite graph: [Tanner, '81]

$H =$

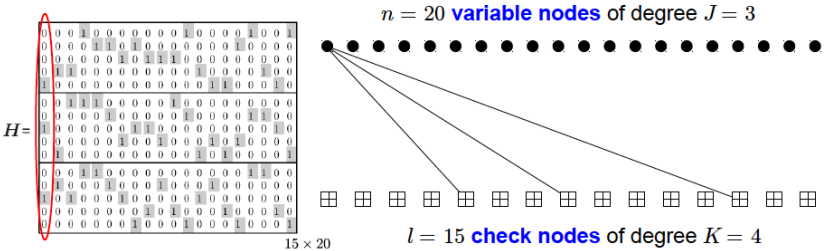
0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1

15×20

$n = 20$ **variable nodes** of degree $J = 3$

$l = 15$ **check nodes** of degree $K = 4$

Representation by bi-partite graph: [Tanner, '81]



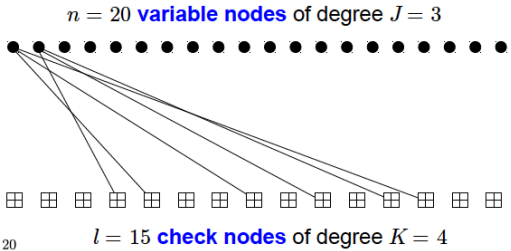
Tanner graph

Representation by bi-partite graph: [Tanner, '81]

$H =$

0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1

15×20



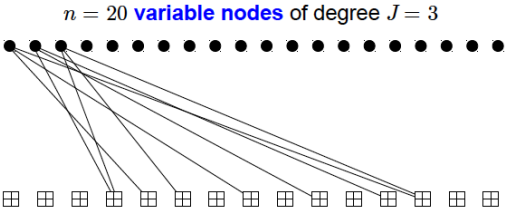
Tanner graph

Representation by bi-partite graph: [Tanner, '81]

$H =$

0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1
0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0
0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

15×20



$n = 20$ **variable nodes** of degree $J = 3$

$l = 15$ **check nodes** of degree $K = 4$

Tanner graph

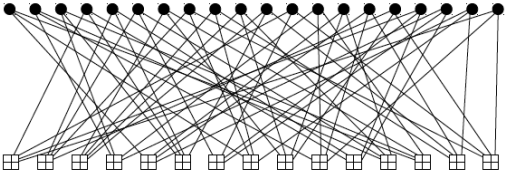
Representation by bi-partite graph: [Tanner, '81]

$H =$

0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

15×20

$n = 20$ **variable nodes** of degree $J = 3$



$l = 15$ **check nodes** of degree $K = 4$

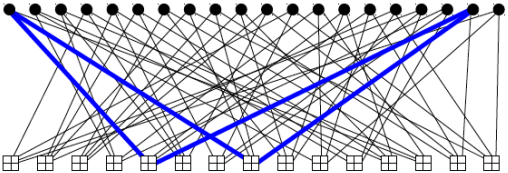
Representation by bi-partite graph: [Tanner, '81]

$H =$

0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

15×20

$n = 20$ **variable nodes** of degree $J = 3$



$l = 15$ **check nodes** of degree $K = 4$

- Tanner graphs typically contain **cycles** (length 4 cycle highlighted above)
- The **girth** of a Tanner graph is the length of the shortest cycle (4 in this example)

Gallager's ensemble of LDPC codes

Ensemble $\mathcal{E}(N, K, J)$

$$\mathbf{H} = \begin{pmatrix} \pi_1(\mathbf{H}_b) \\ \pi_2(\mathbf{H}_b) \\ \vdots \\ \pi_J(\mathbf{H}_b) \end{pmatrix}_{\ell b \times b n_0}$$

where

$$\mathbf{H}_b = \begin{pmatrix} 11\ldots1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & 11\ldots1 & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & 11\ldots1 \end{pmatrix}_{b \times bK}$$

Lower bound on the minimum distance

Theorem (Gallager'62)

For any $J > 2$ there exists $\delta^*(K, J) > 0$ such that:

- *there are codes in the ensemble $\mathcal{E}(N, K, J)$ for which the following inequality holds*

$$d(\mathcal{C}) \geq (\delta^* - \varepsilon)N \tag{1}$$

- *the number of such codes $(G(N, K, J))$ satisfy the following relation*

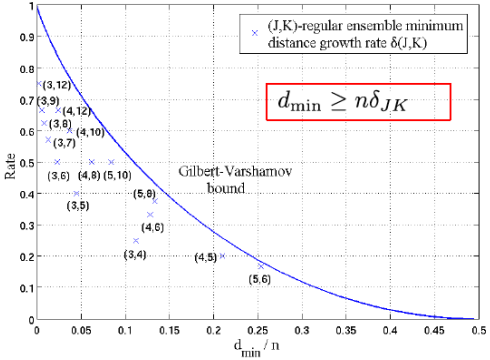
$$\lim_{N\to\infty} \frac{G(N, K, J)}{|\mathcal{E}(N, K, J)|} = 1.$$

The value δ^* is the smallest positive root of equation

$$(J-1)h(\delta) + J \max_{s>0} \left(\delta \log \delta - \frac{1}{K} g_0(s, K) \right) = 0.$$

Lower bound on the minimum distance

- δ_{JK} is called the **typical minimum distance ratio**, or **minimum distance growth rate**, of a code ensemble



Outline

Bit-flipping decoding algorithm

- If there exists a variable node, such that the number of unsatisfied check nodes is bigger then the number of satisfied check nodes $\Rightarrow$ flip the bit.
- Continue until such variable nodes exist.

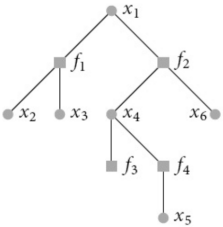
Distributive law

$$ab + ac = a(b + c)$$

$$\sum_{i,j} a_i b_j \qquad\qquad (\sum_i a_i)(\sum_j b_j)$$

Example

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



$$f(x_1) = \sum_{x_2, x_3, x_4, x_5, x_6} f(x_1, x_2, x_3, x_4, x_5, x_6) = \sum_{\sim x_1} f(x_1, x_2, x_3, x_4, x_5, x_6)$$

Note: $f(x_1)$ is a function; therefore, it takes on a distinct value for each value of x_1

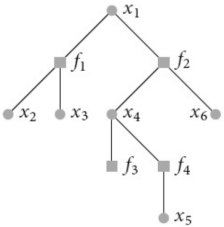
$|\mathcal{X}|$ alphabet

$\Theta\left(|\mathcal{X}|^6\right)$ brute force complexity

Example

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

$\Theta\left(|\mathcal{X}|^3\right)$ complexity

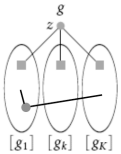


Does there exist a systematic way to find this low complexity scheme using the structure of the graph?

Marginalization via Message Passing for Trees

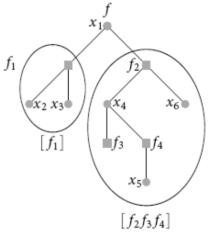
$$g(z) = \sum_{\sim z} g(z, \dots)$$

$$g(z, \dots) = \prod_{k=1}^{K} [g_k(z, \dots)]$$



$$f(x_1) = \sum_{\sim x_1} f(x_1, x_2, x_3, x_4, x_5, x_6)$$

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



Note: the individual functions $g_k(z, \dots)$ only share the variable z; all other variables are "private"

otherwise ... graph is not a tree

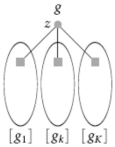
Marginalization via Message Passing for Trees

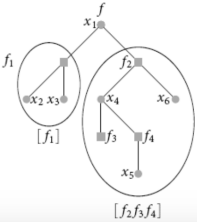
recall that g(z) is a function, taking a distinct value for each value of z

instead of computing g(z) directly by brute force we can first compute each of the functions $g_k(z)$; we then get g(z) by multiplying these functions $g_k(z)$

$$\sum_{\sim z} g(z,\dots) = \underbrace{\sum_{\sim z}\prod_{k=1}^{K}[g_k(z,\dots)]}_{\text{marginal of product}} = \underbrace{\prod_{k=1}^{K}\Big[\sum_{\sim z} g_k(z,\dots)\Big]}_{\text{product of marginals}}$$

$$f(x_1) = \Big[\sum_{\sim x_1} f_1(x_1,x_2,x_3)\Big]\Big[\sum_{\sim x_1} f_2(x_1,x_4,x_6)f_3(x_4)f_4(x_4,x_5)\Big]$$





marginal $\sum_{\sim z} g(z,\dots)$ is the product of the individual marginals

Marginalization via Message Passing for Trees

$$\sum_{\sim z} g(z,\dots) = \underbrace{\sum_{\sim z}\prod_{k=1}^{K}[g_k(z,\dots)]}_{\text{marginal of product}} = \underbrace{\prod_{k=1}^{K}\Big[\sum_{\sim z} g_k(z,\dots)\Big]}_{\text{product of marginals}}$$

$$g_k(z,\dots)$$

$$g_k(z,\dots) = \underbrace{h(z,z_1,\dots,z_J)}_{\text{kernel}}\prod_{j=1}^{J}\underbrace{\big[h_j(z_j,\dots)\big]}_{\text{factors}}$$

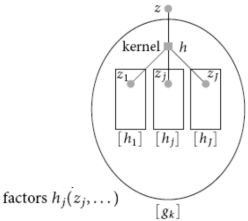
"kernel" $h(z,z_1,\dots,z_J)$

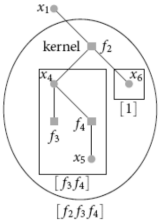
Marginalization via Message Passing for Trees

$$g_k(z,\dots) = \underbrace{h(z,z_1,\dots,z_J)}_{\text{kernel}} \prod_{j=1}^{J} \underbrace{[h_j(z_j,\dots)]}_{\text{factors}}$$

"kernel" $h(z,z_1,\dots,z_J)$

$g_k(z,\dots)$





$$f_2(x_1,x_4,x_6)f_3(x_4)f_4(x_4,x_5) = \underbrace{f_2(x_1,x_4,x_6)}_{\text{kernel}}\underbrace{[f_3(x_4)f_4(x_4,x_5)]}_{x_4}\underbrace{[1]}_{x_6}.$$

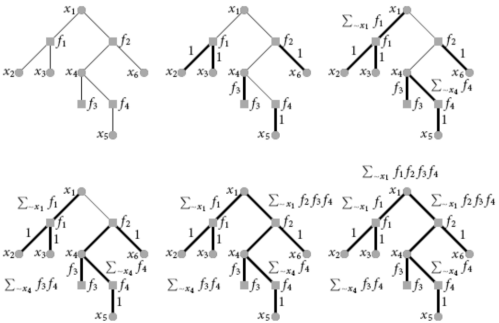
Marginalization via Message Passing for Trees

$$\sum_{\sim z} g_k(z,\dots) = \sum_{\sim z} h(z,z_1,\dots,z_J) \prod_{j=1}^{J} [h_j(z_j,\dots)]$$

$$= \sum_{\sim z} h(z,z_1,\dots,z_J) \underbrace{\prod_{j=1}^{J} \Big[\sum_{\sim z_j} h_j(z_j,\dots)\Big]}_{\text{product of marginals}}$$

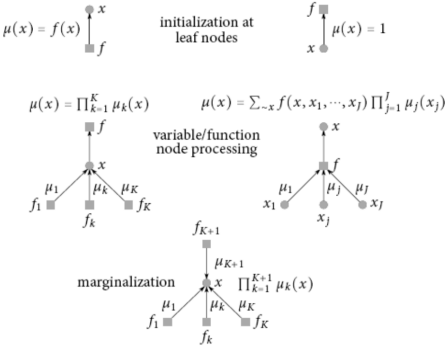
$$f(x_1) = \Big[\sum_{\sim x_1} f_1(x_1,x_2,x_3)\Big]\Big[\sum_{\sim x_1} f_2(x_1,x_4,x_6) f_3(x_4) f_4(x_4,x_5)\Big]$$

Marginalization via Message Passing for Trees



complexity proportional to highest degree

Message passing rules

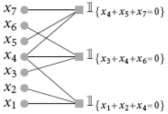


Example

$$H = \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

$$f(x_1, \ldots, x_7) = \mathbb{1}_{\{x \in C(H)\}} = \begin{cases} 1, & \text{if } Hx^T = 0^T, \\ 0, & \text{otherwise.} \end{cases}$$

$$f(x_1, \ldots, x_7) = \mathbb{1}_{\{x_1+x_2+x_4=0\}} \mathbb{1}_{\{x_3+x_4+x_6=0\}} \mathbb{1}_{\{x_4+x_5+x_7=0\}}$$



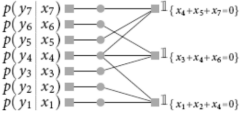
again a tree

Bitwise MAP decoding

$$
\begin{aligned}
\hat{x}_i^{\mathrm{MAP}}(y) &= \operatorname{argmax}_{x_i \in \{\pm 1\}} p_{X_i \mid Y}(x_i \mid y) \\
\text{(law of total probability)} \quad &= \operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} p_{X \mid Y}(x \mid y) \\
\text{(Bayes')} \quad &= \operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} p_{Y \mid X}(y \mid x) p_X(x) \\
&= \operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \Big(\prod_j p_{Y_j \mid X_j}(y_j \mid x_j) \Big) \mathbb{1}_{\{x \in C\}}
\end{aligned}
$$

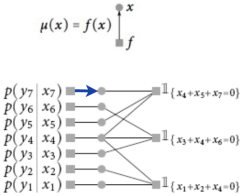
$$
\operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \Big(\prod_{i=1}^{7} p_{Y_i \mid X_j}(y_j \mid x_j) \Big) \mathbb{1}_{\{x_1+x_2+x_4=0\}} \mathbb{1}_{\{x_3+x_4+x_6=0\}} \mathbb{1}_{\{x_4+x_5+x_7=0\}}
$$

$$
H = \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}
$$



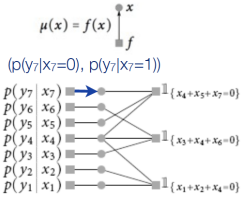
Decoding for Trees via Message Passing

Initial messages from leaf check nodes on the left:



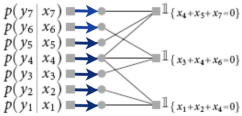
Decoding for Trees via Message Passing

Initial messages from leaf check nodes on the left:

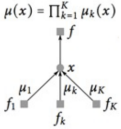


Decoding for Trees via Message Passing

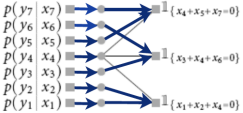
same for all other leafs



Decoding for Trees via Message Passing

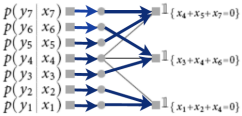


at variables we multiply messages

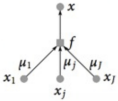


Decoding for Trees via Message Passing

at check nodes we multiply messages,
multiply with kernel and marginalize

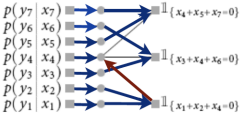


$$\mu(x) = \sum_{\sim x} f(x, x_1, \cdots, x_J) \prod_{j=1}^{J} \mu_j(x_j)$$



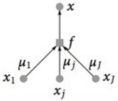
Decoding for Trees via Message Passing

at check nodes we multiply messages, multiply with kernel and marginalize



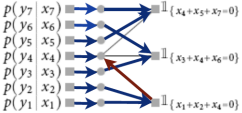
$\mu(x_4)=\sum_{\sim x_4} \mathbb{1}_{\{x_1+x_2+x_4=0\}}\, p(y_1|x_1)\, p(y_2|x_2)$

$$\mu(x) = \sum_{\sim x} f(x, x_1, \cdots, x_J) \prod_{j=1}^{J} \mu_j(x_j)$$



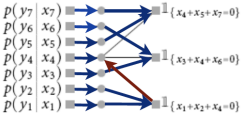
Decoding for Trees via Message Passing

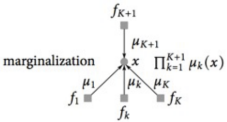
continue in this fashion until all messages along all edges have been determined



Decoding for Trees via Message Passing

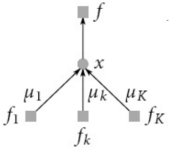
the final decision for each variable is given by the product of all incoming messages





Simplification of Message-Passing Rules for Decoding

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

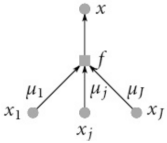


$$(\mu(1), \mu(-1))$$

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

$$l = \sum_{k=1}^{K} l_k$$

$$\mu(x) = \sum_{\sim x} f(x, x_1, \cdots, x_J) \prod_{j=1}^{J} \mu_j(x_j)$$



$$(p_{Y_i|X_i}(y_i|1), p_{Y_i|X_i}(y_i|-1))$$

$$r = \frac{1 + \prod_j \frac{r_j - 1}{r_j + 1}}{1 - \prod_j \frac{r_j - 1}{r_j + 1}}$$

$$l = 2\tanh^{-1}\Big(\prod_{j=1}^{J} \tanh(l_j/2)\Big)$$

Simplification of Message-Passing Rules for Decoding

$$\prod_{j=1}^{3}(r_j+1) = 1 + r_1 + r_2 + r_3 + r_1r_2 + r_1r_3 + r_2r_3 + r_1r_2r_3$$

$$\prod_{j=1}^{3}(r_j-1) = -1 + r_1 + r_2 + r_3 - r_1r_2 - r_1r_3 - r_2r_3 + r_1r_2r_3$$

$$\prod_{j=1}^{J}(r_j+1) + \prod_{j=1}^{J}(r_j-1) = 2 \sum_{x_1,\ldots,x_J:\prod_{j=1}^{J} x_j=1} \prod_{j=1}^{J} r_j^{(1+x_j)/2}.$$

Simplification of Message-Passing Rules for Decoding

$$r = \frac{\mu(1)}{\mu(-1)} = \frac{\sum_{\sim x} f(1, x_1, \ldots, x_J) \prod_{j=1}^{J} \mu_j(x_j)}{\sum_{\sim x} f(-1, x_1, \ldots, x_J) \prod_{j=1}^{J} \mu_j(x_j)} \qquad f(x, x_1, \ldots, x_J) = \mathbb{1}_{\{\prod_{j=1}^{J} x_j = x\}}$$

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

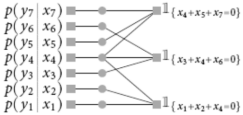
$$= \frac{\sum_{x_1,\ldots,x_J : \prod_{j=1}^{J} x_j = 1} \prod_{j=1}^{J} r_j^{(1+x_j)/2}}{\sum_{x_1,\ldots,x_J : \prod_{j=1}^{J} x_j = -1} \prod_{j=1}^{J} r_j^{(1+x_j)/2}} = \frac{\prod_{j=1}^{J}(r_j + 1) + \prod_{j=1}^{J}(r_j - 1)}{\prod_{j=1}^{J}(r_j + 1) - \prod_{j=1}^{J}(r_j - 1)}.$$

$$r = \frac{1 + \prod_j \frac{r_j - 1}{r_j + 1}}{1 - \prod_j \frac{r_j - 1}{r_j + 1}} \qquad \frac{r-1}{r+1} = \prod_j \frac{r_j - 1}{r_j + 1} \qquad r = \mathrm{e}^{l} \qquad \frac{r-1}{r+1} = \tanh(l/2).$$

$$\tanh(l/2) = \frac{r-1}{r+1} = \prod_{j=1}^{J} \frac{r_j - 1}{r_j + 1} = \prod_{j=1}^{J} \tanh(l_j/2). \qquad l = 2\tanh^{-1}\Big(\prod_{j=1}^{J} \tanh(l_j/2)\Big).$$

Summary and Limitations

$$l = \sum_{k=1}^{K} l_k$$



$$l = 2\tanh^{-1}\Big(\prod_{j=1}^{J} \tanh(l_j/2)\Big)$$

Lemma 2.24 (Bad News about Cycle-Free Codes). Let C be a binary linear code of rate r that admits a binary Tanner graph that is a forest. Then C contains at least $\frac{2r-1}{2}n$ codewords of weight 2.

Outline

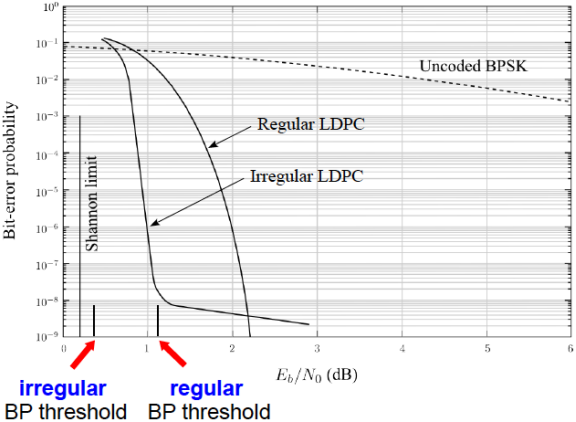
Problems

In what follows the decoding algorithm is fixed.

The decoding algorithm is suboptimal (there are cycles in the Tanner graph).

How to optimize LDPC parity-check matrices for this decoder?

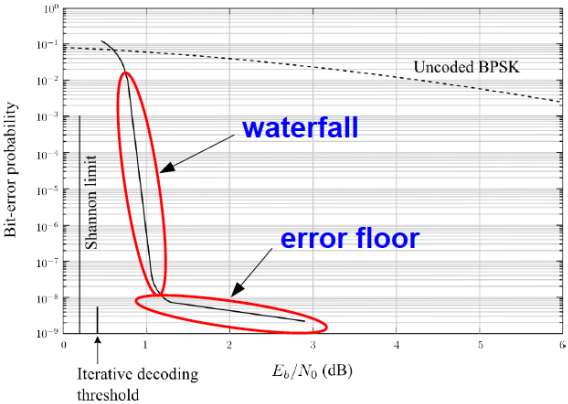
Regular vs Irregular LDPC codes



- **Irregular** LDPC code ensembles can have optimized thresholds **close to capacity**
- **Regular** LDPC code ensembles are asymptotically good and have good graph properties, resulting in a **low error floor**

Waterfall vs error floor

- The **Shannon limit** defines capacity and is a property of the physical channel.



- The **iterative decoding threshold** depends on the code structure and iterative decoding algorithm in use.
- Capacity-approaching LDPC codes typically display a **waterfall** (related to their threshold) and an **error-floor** (related to their graph/distance properties).

How to define the ensemble of irregular LDPC codes?

Degree (weight if we consider PCM) distribution polynomials

$$\Lambda(x) = \sum_{i=1}^{l_{\max}} \Lambda_i x^i \text{ (variable nodes)}$$

and

$$P(x) = \sum_{i=1}^{r_{\max}} P_i x^i \text{ (check nodes)},$$

where Λ_i and P_i are numbers of variable/check nodes of degree i.

Properties:

$$\Lambda(1) = n, P(1) = (1-R)n, R = 1 - \frac{P(1)}{\Lambda(1)}$$

How to define the ensemble of irregular LDPC codes?

Degree (weight if we consider PCM) distribution polynomials

$$L(x) = \frac{\Lambda(x)}{\Lambda(1)}$$

and

$$Q(x) = \frac{P(x)}{P(1)},$$

where L_i and Q_i are *fractions* of variable/check nodes of degree i.

Edge perspective

For the asymptotic analysis it is more convenient to take on an edge perspective. Define:

$$\lambda(x) = \sum_i \lambda_i x^{i-1}$$

and

$$\rho(x) = \sum_i \rho_i x^{i-1},$$

where λ_i and ρ_i are *fractions* of *edges* that connect to variable(check) nodes of degree i.

Properties:

$$\lambda(x) = \frac{L'(x)}{L'(1)}, \rho(x) = \frac{Q'(x)}{Q'(1)}.$$

Example

Consider $[7, 4]$ Hamming code

$$\mathbf{H}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \Lambda(x) &= 3x + 3x^2 + x^3 \\ L(x) &= \frac{3}{7}x + \frac{3}{7}x^2 + \frac{1}{7}x^3 \\ \lambda(x) &= \frac{1}{4} + \frac{1}{2}x + \frac{1}{4}x^2 \end{aligned}$$

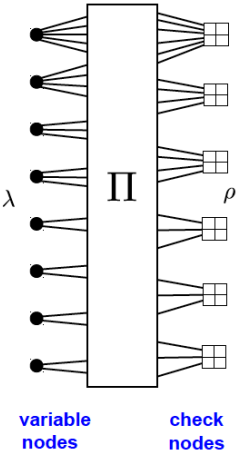
How to define the ensemble of irregular LDPC codes?

- **Node degrees**: random variables [Luby, et al., '97]

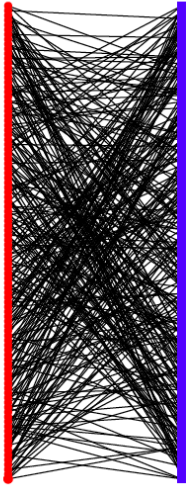
$$\lambda(x) = \sum_k \lambda_k x^{k-1} \leftarrow \text{variable node distribution}$$

$$\rho(x) = \sum_k \lambda_k x^{k-1} \leftarrow \text{check node distribution}$$

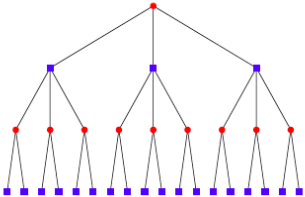
LDPC Ensemble:



Computational graph



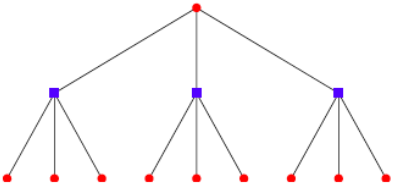
$$\lim_{n\to\infty} \mathbb{E}[P_b(\mathrm{G}, n, \ell)]$$



probability that computation graph
of fixed depth becomes tree
tends to 1 as n tends to infinity

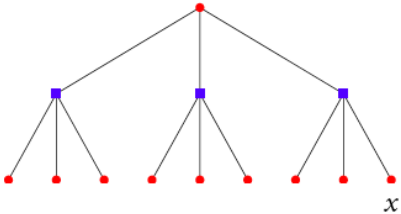
Density evolution, BEC

Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



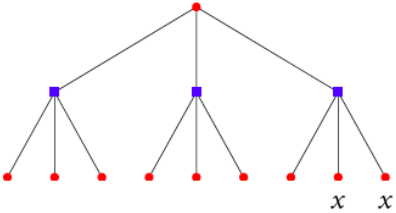
Density evolution, BEC

Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



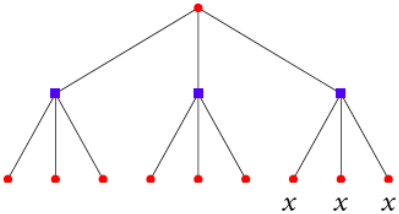
Density evolution, BEC

Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



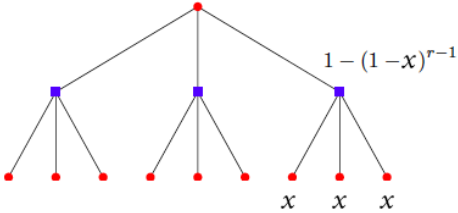
Density evolution, BEC

Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



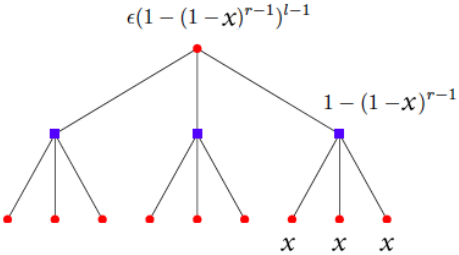
Density evolution, BEC

Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



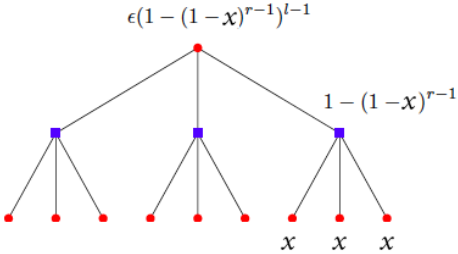
Density evolution, BEC

Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



Density evolution, BEC

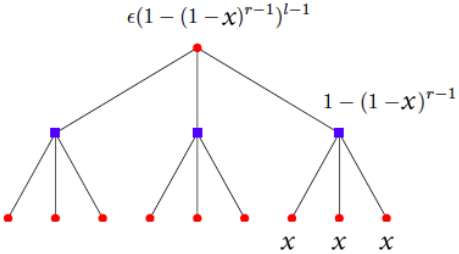
Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



$$x_t = \epsilon\left(1-(1-x_{t-1})^{r-1}\right)^{l-1}$$

Density evolution, BEC

Luby, Mitzenmacher, Shokrollahi, Spielman, and Steman '97



$$x_\ell = \epsilon\lambda\left(1-\rho\left(1-x_{\ell-1}\right)\right)$$

Density evolution, BEC

EXAMPLE 3.52 (DENSITY EVOLUTION FOR $(\lambda(x) = x^2, \rho(x) = x^5)$). For the degree distribution pair $(\lambda(x) = x^2, \rho(x) = x^5)$ we have $x_0 = \epsilon$ and for $\ell \geq 1$, $x_\ell = \epsilon(1 - (1 - x_{\ell-1})^5)^2$. For example, for $\epsilon = 0.4$ the sequence of values of x_ℓ is 0.4, 0.34, 0.306, 0.2818, 0.2617, 0.2438, and so forth. $\diamond$

Density evolution, BEC

EXAMPLE 3.52 (DENSITY EVOLUTION FOR $(\lambda(x) = x^2, \rho(x) = x^5)$). For the degree distribution pair $(\lambda(x) = x^2, \rho(x) = x^5)$ we have $x_0 = \epsilon$ and for $\ell \geq 1$, $x_\ell = \epsilon(1 - (1 - x_{\ell-1})^5)^2$. For example, for $\epsilon = 0.4$ the sequence of values of x_ℓ is 0.4, 0.34, 0.306, 0.2818, 0.2617, 0.2438, and so forth. $\diamond$

Does this sequence converge to 0 ?

Density evolution, BEC

LEMMA 3.53 (MONOTONICITY OF $f(\cdot,\cdot)$). For a given degree distribution pair (λ, ρ) define $f(\epsilon, x) = \epsilon\lambda(1 - \rho(1 - x))$. Then $f(\epsilon, x)$ is increasing in both its arguments for $x, \epsilon \in [0, 1]$.

LEMMA 3.54 (MONOTONICITY WITH RESPECT TO CHANNEL). Let (λ, ρ) be a degree distribution pair and $\epsilon \in [0, 1]$. If $P^{BP}_{\mathcal{T}_\ell}(\epsilon) \xrightarrow{\ell\to\infty} 0$ then $P^{BP}_{\mathcal{T}_\ell}(\epsilon') \xrightarrow{\ell\to\infty} 0$ for all $0 \le \epsilon' \le \epsilon$.

phase transition: ϵ^{BP} so that

$$x_t \to 0 \text{ for } \epsilon < \epsilon^{BP}$$

$$x_t \to x_\infty > 0 \text{ for } \epsilon > \epsilon^{BP}$$

Threshold

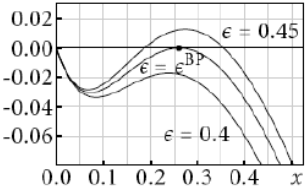
DEFINITION 3.56 (THRESHOLD OF DEGREE DISTRIBUTION PAIR). The *threshold* associated with the degree distribution pair (λ, ρ), call it $\epsilon^{\mathrm{BP}}(\lambda, \rho)$, is defined as

$$\epsilon^{\mathrm{BP}}(\lambda, \rho) = \sup\{\epsilon \in [0, 1] : \mathrm{P}^{\mathrm{BP}}_{\mathcal{T}_\ell(\lambda,\rho)}(\epsilon) \xrightarrow{\ell \to \infty} 0\}. \qquad \nabla$$

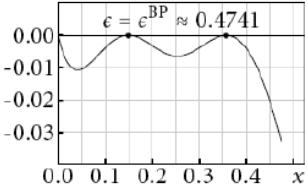
EXAMPLE 3.57 (THRESHOLD OF $(\lambda(x) = x^2, \rho = x^5)$). Numerical experiments show that $\epsilon^{\mathrm{BP}}(3, 6) \approx 0.42944$. $\diamond$

Fixed point characterization, BEC

$$f(\epsilon, x) = \epsilon\lambda(1 - \rho(1 - x))$$



$$f(\epsilon, x) - x = \epsilon(1 - (1 - x)^5)^2 - x$$

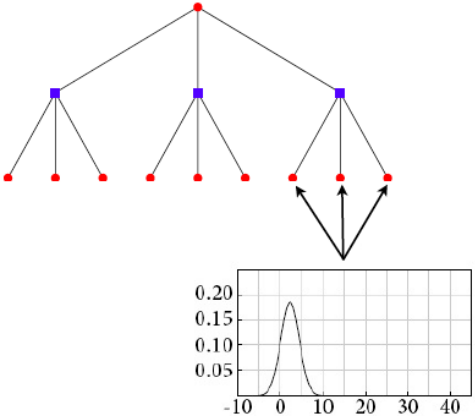


$$(\lambda, \rho) = (x^2, x^5)$$

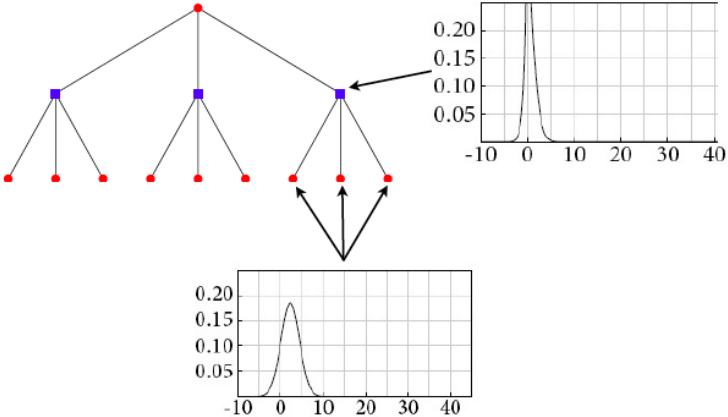
$$\lambda(x) = 0.106257x + 0.486659x^2 + 0.010390x^{10} + 0.396694x^{19}$$

$$\rho(x) = 0.5x^7 + 0.5x^8$$

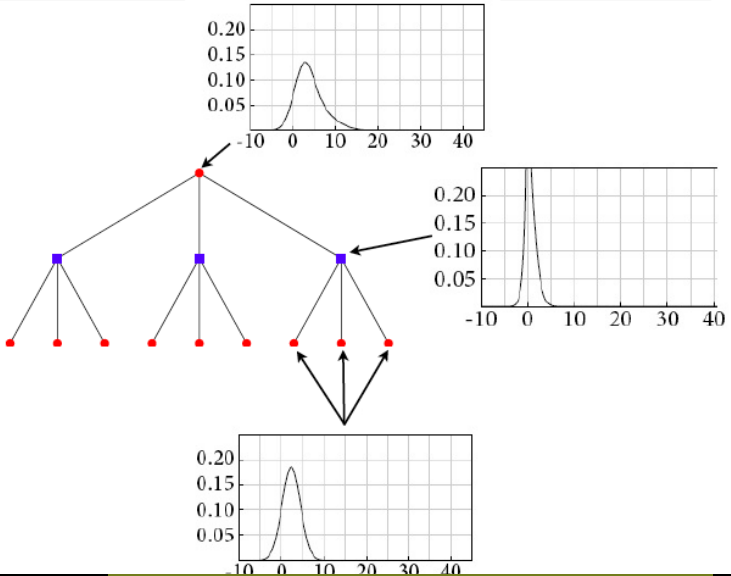
Density evolution, AWGNC



Density evolution, AWGNC



Density evolution, AWGNC



Error floor

- Error events in the **waterfall** typically result from **large** decoding failures (a large number of symbols decoded incorrectly)
- Error events in the **error floor** typically result from **small** decoding failures (only a few symbols decoded incorrectly)

- The minimum distance is a **code property**; under ML decoding, a large minimum distance results in a low error floor
- Under sub-optimal **iterative BP decoding**, the error floor is also affected by small failures arising due to **weaknesses in the Tanner graph**

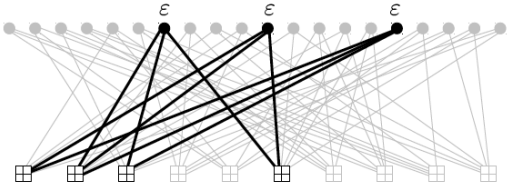
- These graphical weaknesses have been studied extensively for a variety of channels and are known collectively as **pseudocodewords** [Frey et al '98], **stopping sets** [Di et al '02], **near-codewords** [MacKay & Postol '03], **trapping sets** [Richardson '03], **elementary trapping sets** [Laendner & Milenkovic '05], and **absorbing sets** [Dolecek et al '07].

Skoltech
Skolkovo Institute of Science and Technology

Error floor, BEC

- On the BEC, the cause of failures is **stopping sets** [Di, et al. '02].

Definition: A stopping set is a subset S of the variable nodes such that all neighboring check nodes are connected to S at least twice



Example stopping set in a $(3,6)$-regular Tanner graph

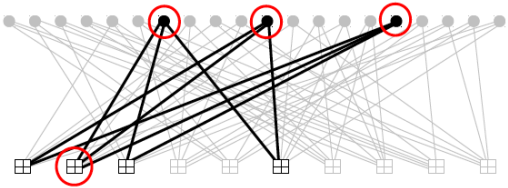
- If the highlighted nodes are all erasures then the BP decoder will fail to correct them

➡ Message-passing decoding is suboptimal! The MAP decoder fails if and only if the set of erasures contains the set of all non-zero positions in the codeword.

Error floor, AWGNC

- On the AWGNC, failures are attributed to **trapping sets** [Richardson '03].

Definition: An (a,b) general trapping set $\tau_{a,b}$ of a bipartite graph is a set of a variable nodes which induce a subgraph with exactly b odd-degree check nodes.



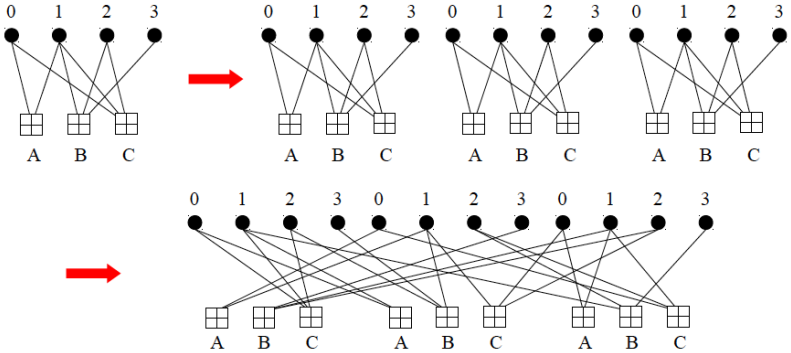
A $(3,1)$ trapping set in a $(3,6)$-regular Tanner graph

- Low connectivity outside the set causes the iterative decoder to become trapped and fail to correct the symbols in the set
- Certain types of trapping sets with small a and b, such as **elementary trapping sets** and **absorbing sets**, are known to be particularly harmful

Outline

Protograph-based LDPC codes

- Codes can be constructed from a **protograph** using a **copy-and-permute** operation



[Tho05] J. Thorpe, "Low-Density Parity-Check (LDPC) codes constructed from protographs", *Jet Propulsion Laboratory INP Progress Report*, Vol. 42-154 Aug. 2003.

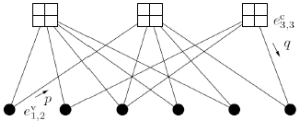
Protograph-based LDPC codes

- **Compact representation** of a permutation matrix based ensemble by a base matrix:

$$\mathbf{H} = \underbrace{\begin{bmatrix} \mathbf{\Pi}_{1,1} & \mathbf{\Pi}_{1,2} & \mathbf{\Pi}_{1,3} & \mathbf{\Pi}_{1,4} & \mathbf{\Pi}_{1,5} & \mathbf{0} \\ \mathbf{\Pi}_{2,1} & \mathbf{0} & \mathbf{\Pi}_{2,3} & \mathbf{\Pi}_{2,4} & \mathbf{\Pi}_{2,5} & \mathbf{\Pi}_{2,6} \\ \mathbf{0} & \mathbf{\Pi}_{3,2} & \mathbf{\Pi}_{3,3} & \mathbf{0} & \mathbf{0} & \mathbf{\Pi}_{3,6} \end{bmatrix}}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

base matrix



protograph

[Tho05] J. Thorpe, “Low-Density Parity-Check (LDPC) codes constructed from protographs”, *Jet Propulsion Laboratory INP Progress Report*, Vol. 42-154 Aug. 2003.

Quasi-cyclic LDPC codes

Replace permutation matrices with circulant matrices (usually of weight 1).

Why this code is quasi-cyclic?

Спасибо за внимание!