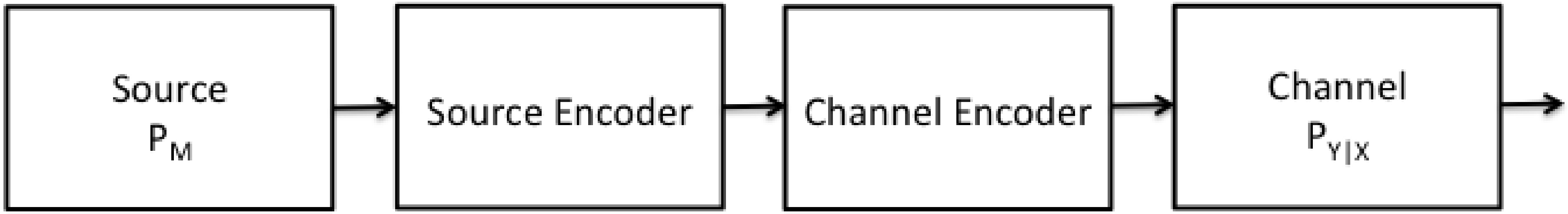
Channel Coding with LDPC codes

Fernando Pérez-Cruz

University Carlos III in Madrid

April 2012

Main Results in Information Theory



- Achievable Source Encoding:

$$R \geq H(M)$$

- Reliable Transmission Rate:

$$R \leq I(X;Y)$$

- Separation Principle: No loss of optimality if perform separately.

Channel Encoding

- The source (encoder) provides symbols $\mathbf{m} \in \{1, 2, \ldots, M\}$
- The channel encoder assigns them a bitstream of n symbols:

$$\mathbf{x} = [x_1 \; x_2 \; \cdots \; x_n]$$

- If the messages are also binary symbols:

$$\mathbf{m} = [m_1 \; m_2 \; \cdots \; m_k]$$

- The rate is given by:

$$R = \frac{\log_2 M}{n} = \frac{k}{n}$$

Linear Channel Encoding

- Linear transformation from source bits to encoded bits:

$$\mathbf{x} = \mathbf{m}\mathbf{G} = \mathbf{m}[\mathbf{I} \quad \mathbf{P}]$$

$\mathbf{G}$ is a $k \times n$ matrix that adds $n-k$ redundancy bits to $\mathbf{m}$.

- Independent channel realizations:

$$y_i = x_i + z_i \qquad \forall i = 1, \ldots, n,$$

z_i is iid noise.

- At the channel decoder we want to recover $\mathbf{x}$ from $\mathbf{y}$.

$$\hat{\mathbf{x}} = \operatorname*{argmax}_{\mathbf{x} \ a \ CW} p(\mathbf{y}|\mathbf{x}) = \operatorname*{argmax}_{\mathbf{x} \ a \ CW} \prod_{\ell=1}^{n} p(\mathbf{y}_\ell|\mathbf{x}_\ell)$$

Syndrome and Parity Check Matrix

- The dual space of the linear space $\mathbf{G}$:

$$\mathbf{H} = [-\mathbf{P}^\top \quad \mathbf{I}]$$

- Syndrome:

$$\mathbf{s} = \mathbf{y}\mathbf{H}^\top = (\mathbf{x}+\mathbf{z})\mathbf{H}^\top = \mathbf{z}\mathbf{H}^\top$$

- Why is $\mathbf{x}\mathbf{H}^\top = \mathbf{0}$?
- $\mathbf{s}$ uniquely identifies the error pattern.
- $\mathbf{s}$ has 2^{n-k} entries.

Solutions to Channel Coding

- Algebraic Code (40's-70's):
  - Linear encoding and decoding
  - Limited to minimum distance.
- Convolutional Codes (60's-80's):
  - Linear encoding and decoding.
  - Decoding exponential in the memory.
- LDPC and Turbo Codes (63 & 90's-10's):
  - Almost linear encoding and decoding.
  - Almost achieve capacity.

Tanner Graph (Factor Graph)

- Given a parity Check Matrix

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

  what do we know?

- What restriction do we have over x_1, x_5, x_6 and x_7?

Tanner Graph (Factor Graph)

- Given a parity Check Matrix

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

  what do we know?

- What restriction do we have over x_1, x_5, x_6 and x_7?

- and over x_2, x_4, x_6 and x_7? or x_3, x_4, x_5 and x_7?

Tanner Graph (Factor Graph)

- Given a parity Check Matrix

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

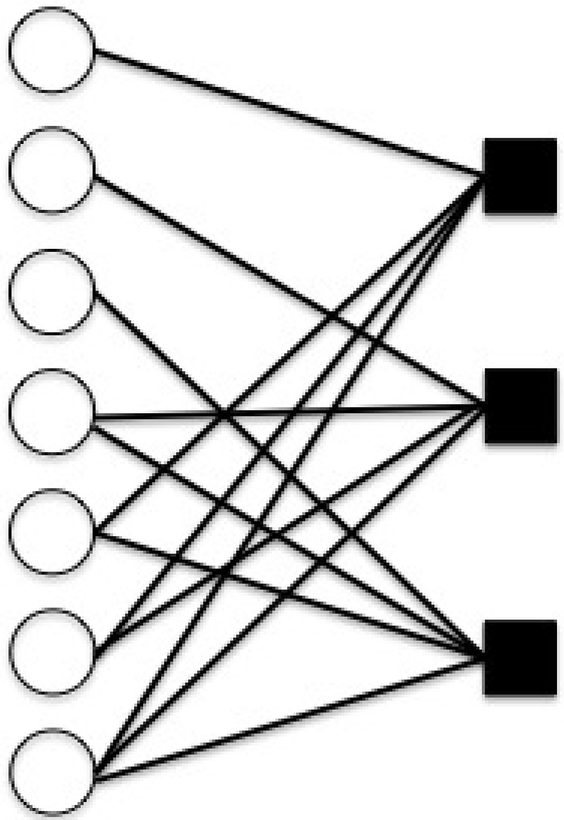
  what do we know?

- What restriction do we have over x_1, x_5, x_6 and x_7?
- and over x_2, x_4, x_6 and x_7? or x_3, x_4, x_5 and x_7?
- Remember that:

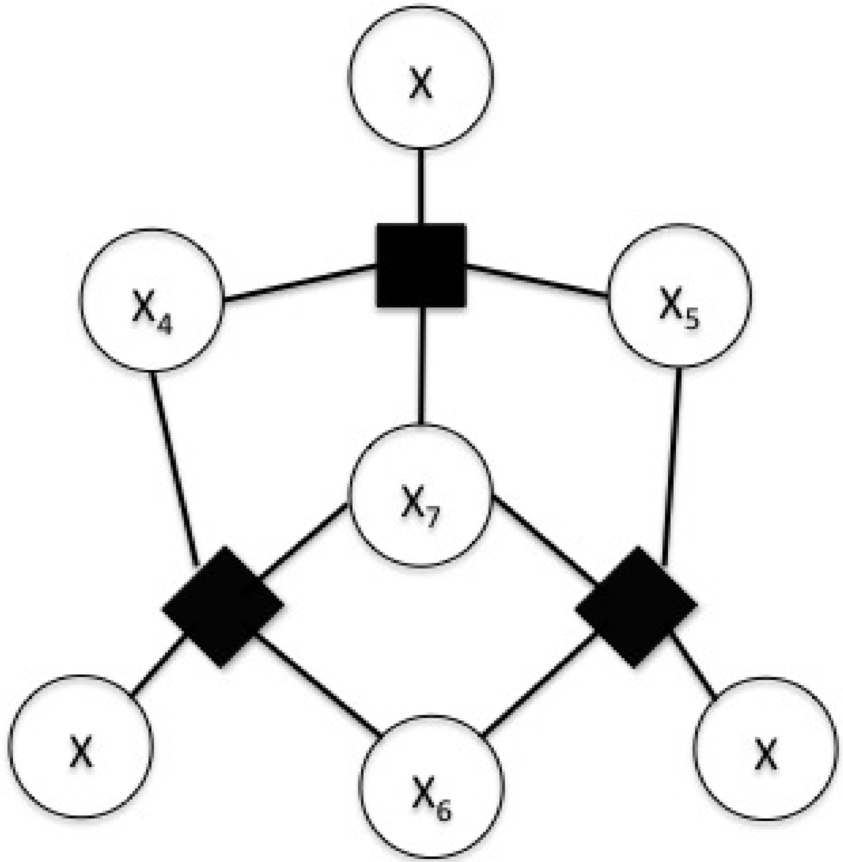
$$\mathbf{x}\mathbf{H}^\top = \mathbf{0}$$

- Hence?

Bipartite Graph



Bipartite Graph



Maximum Likelihood Decoder

- ML solution:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\ a\ CW}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmax}} \prod_{\ell=1}^{n} p(\mathbf{y}_\ell|\mathbf{x}_\ell) \prod_{j=1}^{n-k} \delta(\mathbf{x}\mathbf{h}_j^\top = 0)$$

- ML solution is exponential in n.
- Bitwise MAP solution:

$$\widehat{\mathbf{x}_i} = \underset{\mathbf{x}_i \in \{0,1\}}{\operatorname{argmax}}\, p(\mathbf{x}_i|\mathbf{y}) = \underset{v \in \{0,1\}}{\operatorname{argmax}} \sum_{\substack{\mathbf{x} \\ x_i = v}} \prod_{\ell=1}^{n} p(\mathbf{y}_\ell|\mathbf{x}_\ell) \prod_{j=1}^{n-k} \delta(\mathbf{x}\mathbf{h}_j^\top = 0)$$

- Still exponential in n, but ...

Computing $p(x_i)$

- If we ignore the graph structure:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} = \frac{1}{p(\mathbf{y})}\frac{1}{2^k}\prod_{\ell=1}^{n} p(\mathbf{y}_\ell|\mathbf{x}_\ell)\prod_{j=1}^{n-k}\delta(\mathbf{x}\mathbf{h}_j^\top = 0)$$

- For binary variables we need to compute 2^n elements and perform $2^n - 1$ sums for computing:

$$p(\mathbf{x}_i = v|\mathbf{y}) = \sum_{\substack{\mathbf{x} \\ x_i = v}} p(\mathbf{x}|\mathbf{y})$$

- Can we use the graph structure to reduced this computational complexity?

Computing $p(x_i)$

- If we ignore the graph structure:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} = \frac{1}{p(\mathbf{y})}\frac{1}{2^k}\prod_{\ell=1}^{n} p(\mathbf{y}_\ell|\mathbf{x}_\ell)\prod_{j=1}^{n-k}\delta(\mathbf{x}\mathbf{h}_j^\top = 0)$$

- For binary variables we need to compute 2^n elements and perform $2^n - 1$ sums for computing:

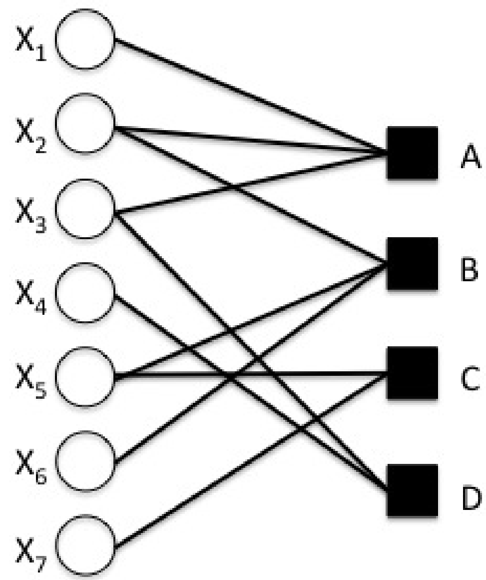
$$p(\mathbf{x}_i = v|\mathbf{y}) = \sum_{\substack{\mathbf{x} \\ x_i = v}} p(\mathbf{x}|\mathbf{y})$$

- Can we use the graph structure to reduced this computational complexity?

- Yes, we can!

An Easy Example

$$p(\mathbf{x}|\mathbf{y}) \propto \prod_{\ell=1}^{7} p(\mathbf{y}_\ell|\mathbf{x}_\ell) \prod_{j=1}^{4} \delta(\mathbf{x}\mathbf{h}_j^\top = 0)$$



- We can compute $p(x_1)$ from:

$$p(x_1) \propto \sum_{\substack{x_2,x_3,x_4 \\ x_5,x_6,x_7}} f_A(x_1,x_2,x_3) f_B(x_2,x_5,x_6) f_C(x_5,x_7) f_D(x_3,x_4)$$

An Easy Example

- We can express the sum as follows:

$$p(x_1) \propto \sum_{x_2,x_3} f_A(x_1,x_2,x_3) \sum_{x_4} f_D(x_3,x_4) \sum_{x_5,x_6} f_B(x_2,x_5,x_6) \sum_{x_7} f_C(x_5,x_7)$$

- We define:

$$r_{C\to x_5}(x_5) = \sum_{x_7} f_C(x_5,x_7)$$
$$r_{C\to x_5}(x_5=0) = f_C(x_5=0,x_7=0) + f_C(x_5=0,x_7=1)$$
$$r_{C\to x_5}(x_5=1) = f_C(x_5=1,x_7=0) + f_C(x_5=1,x_7=1)$$

- We need to compute 4 components and perform 2 sums.

An Easy Example

- We can express the sum as follows:

$$p(x_1) \propto \sum_{x_2,x_3} f_A(x_1,x_2,x_3) \sum_{x_4} f_D(x_3,x_4) \sum_{x_5,x_6} f_B(x_2,x_5,x_6) r_{C\to x_5}(x_5)$$

- Now we define:

$$r_{B\to x_2}(x_2) = \sum_{x_5,x_6} f_B(x_2,x_5,x_6) r_{C\to x_5}(x_5)$$
$$r_{D\to x_3}(x_3) = \sum_{x_4} f_D(x_3,x_4)$$

- We need to compute 8+4 components and perform 6+2 sums.

An Easy Example

- We can express the sum as follows:

$$p(x_1) \propto \sum_{x_2,x_3} f_A(x_1,x_2,x_3) r_{B\to x_2}(x_2) r_{D\to x_3}(x_3)$$

- Now we define:

$$r_{A\to x_1}(x_1) = \sum_{x_2,x_3} f_A(x_1,x_2,x_3) r_{B\to x_2}(x_2) r_{D\to x_3}(x_3)$$

- We need to compute 8 components and perform 6 sums.
- Leaving $p(x_1) \propto r_{A\to x_1}(x_1)$ and:

$$p(x_1 = 1) = \frac{r_{A\to x_1}(x_1 = 1)}{r_{A\to x_1}(x_1 = 1) + r_{A\to x_1}(x_1 = 0)}$$

An Easy Example

- We have computed $p(x_1)$ computing 26 terms and performing 17 sums.
- The direct enumeration would lead to computing 128 terms and performing 127 sums.
- Moreover the proposed approach gives us the partition function:

$$Z = r_{A \to x_1}(x_1 = 0) + r_{A \to x_1}(x_1 = 1)$$

- For the other marginals we need to do a bit more work.
- Drawback: We need to sort the variables.

An Easy Example

- We have computed $p(x_1)$ computing 26 terms and performing 17 sums.
- The direct enumeration would lead to computing 128 terms and performing 127 sums.
- Moreover the proposed approach gives us the partition function:

$$Z = r_{A\to x_1}(x_1 = 0) + r_{A\to x_1}(x_1 = 1)$$

- For the other marginals we need to do a bit more work.
- Drawback: We need to sort the variables. Do we really?

Message Passing Algorithms

- We do not need to sort the variables in order to compute Z or the marginals.
- We can use only local computations in the graph to obtain these quantities.
- Simple algorithm:
  - The variables nodes tell each factors about themselves.
  - The factors tell the variables what their value should be.
  - Iterate until convergence.
- Convergence is achieved in finite number of iterations.

Message Passing Algorithms

- Variable to factor:
  - Send the unknown information about the factor.
  - Send information to local factors.

- Factor to Variable:
  - Send the unknown information about the variable.
  - Send information to local variable nodes.

Message Passing Algorithms

- Variable to factor:
  - Send the unknown information about the factor.
  - Send information to local factors. x_2 to A, B and E_2.

- Factor to Variable:
  - Send the unknown information about the variable.
  - Send information to local variable nodes.

Message Passing Algorithms

- Variable to factor:
  - Send the unknown information about the factor.
  - Send information to local factors. x_2 to A, B and E_2.

- Factor to Variable:
  - Send the unknown information about the variable.
  - Send information to local variable nodes. A to x_1, x_2 and x_3.

Message Passing Algorithms

- Variable to factor:
  - Send the unknown information about the factor.
  - Send information to local factors. x_2 to A, B and E_2.

$$q_{x2\to A}(x_2) = r_{B\to x_2}(x_2) r_{E_2\to x_2}(x_2)$$

- Factor to Variable:
  - Send the unknown information about the variable.
  - Send information to local variable nodes. A to x_1, x_2 and x_3.

Message Passing Algorithms

- Variable to factor:
  - Send the unknown information about the factor.
  - Send information to local factors. x_2 to A, B and E_2.

$$q_{x2\to A}(x_2) = r_{B\to x_2}(x_2) r_{E_2\to x_2}(x_2)$$

- Factor to Variable:
  - Send the unknown information about the variable.
  - Send information to local variable nodes. A to x_1, x_2 and x_3.

$$r_{A\to x_2}(x_2) = \sum_{x_1,x_3} f_A(x_1,x_2,x_3) q_{x_1\to A}(x_1) q_{x_3\to A}(x_3)$$

Message Passing Algorithms

- Variable n to factor J:

$$q_{x_n \to J}(x_n) = \prod_{J' \in \mathcal{M}(n) \backslash J} r_{J' \to x_n}(x_n)$$

- Factor J to variable n:

$$r_{J \to x_n}(x_n) = \sum_{\mathbf{x}_J \backslash n} f_J(\mathbf{x}_J) \prod_{n' \in \mathcal{N}(J) \backslash n} q_{x_{n'} \to J}(x_{n'})$$

  - $\mathcal{M}(n)$ are the factors in which x_n is included.
  - $\mathcal{N}(J)$ are the variable nodes for factor J.
  - $\mathbf{x}_J$ are the variables for factor J.

Message Passing Algorithms

- We do not need to sort the sums.
- We do not need to know the structure of the whole graph.
- We only need local information:
  - which variable is connected to which factor.
  - which factor is connected to which variable.
- For tree-like graphs the solution is exact and it finishes in a finite number of iterations.
- For general graphs this algorithm is not applicable.
- ... but it typically works well.

Low Density Parity Check Codes

- If we have few ones in the parity check matrix,

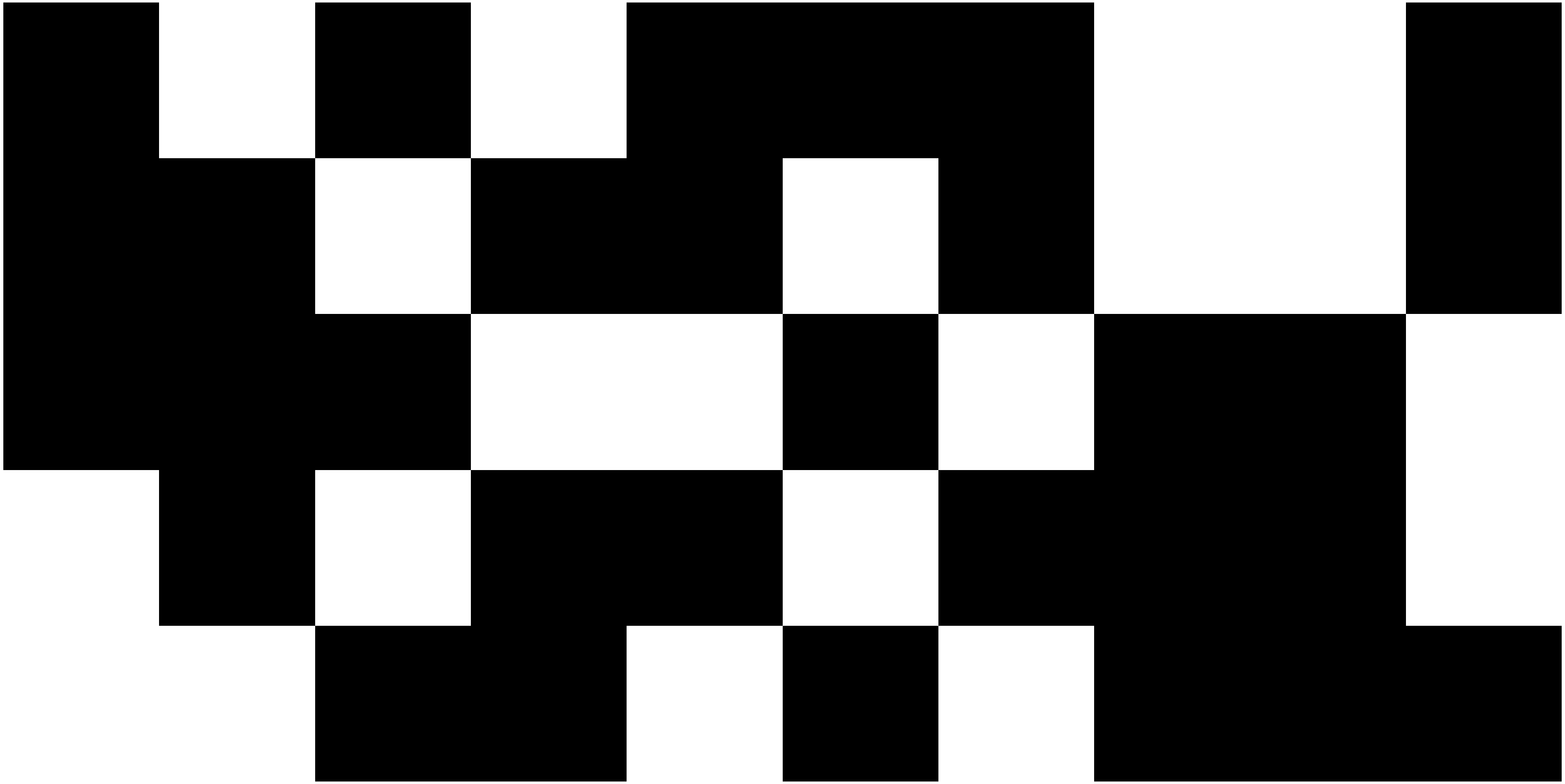
Low Density Parity Check Codes

- If we have few ones in the parity check matrix,
- ... we should expect few loops.
- Locally it will look like a tree

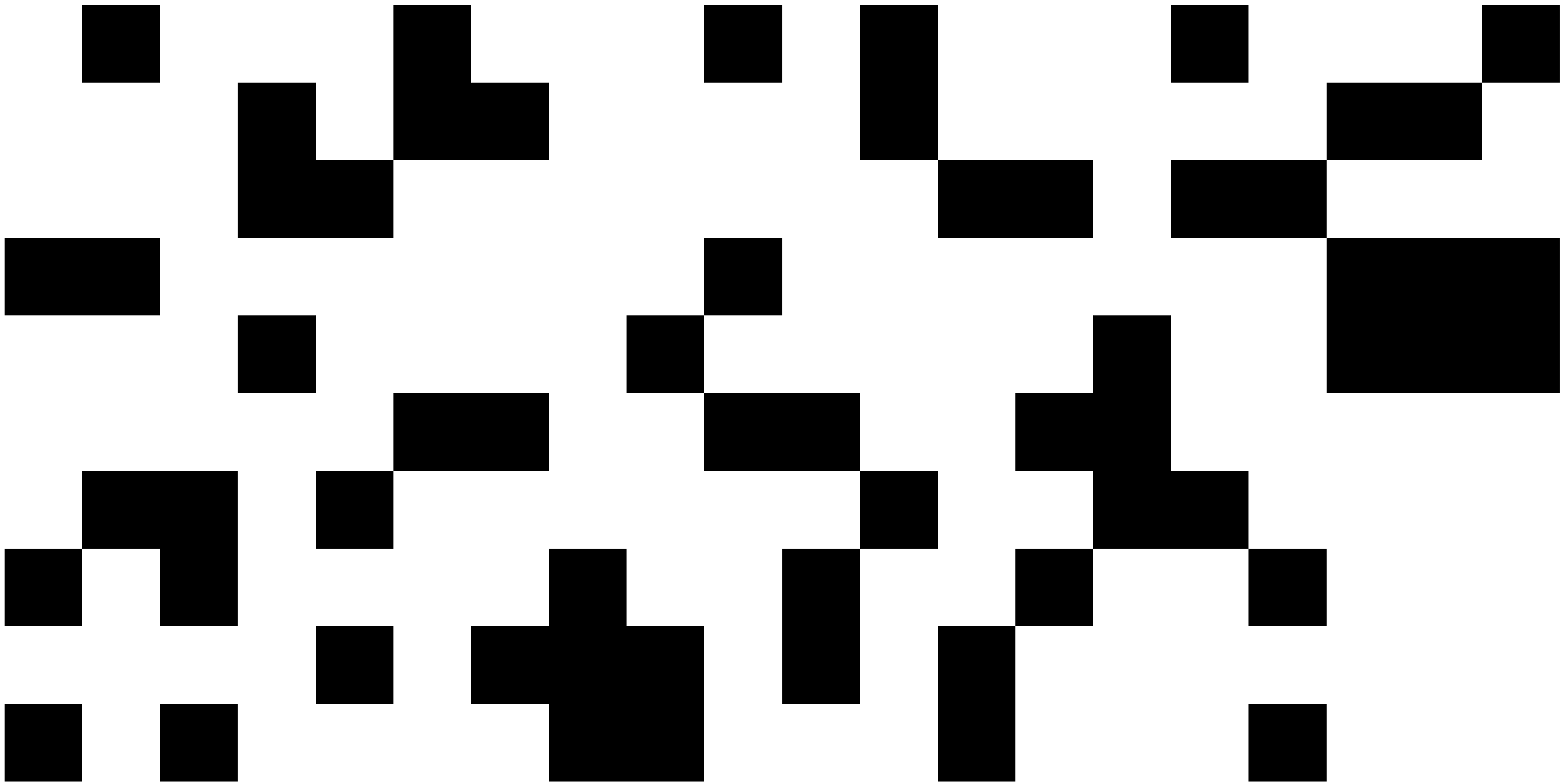
Low Density Parity Check Codes

- If we have few ones in the parity check matrix,
- ... we should expect few loops.
- Locally it will look like a tree
- ... and if we run the message passing algorithm for a finite iterations we will not get harmful feedback.
- And if the density is large enough it leads to 'good codes'.
- 3 ones per column seems to work ... good enough.

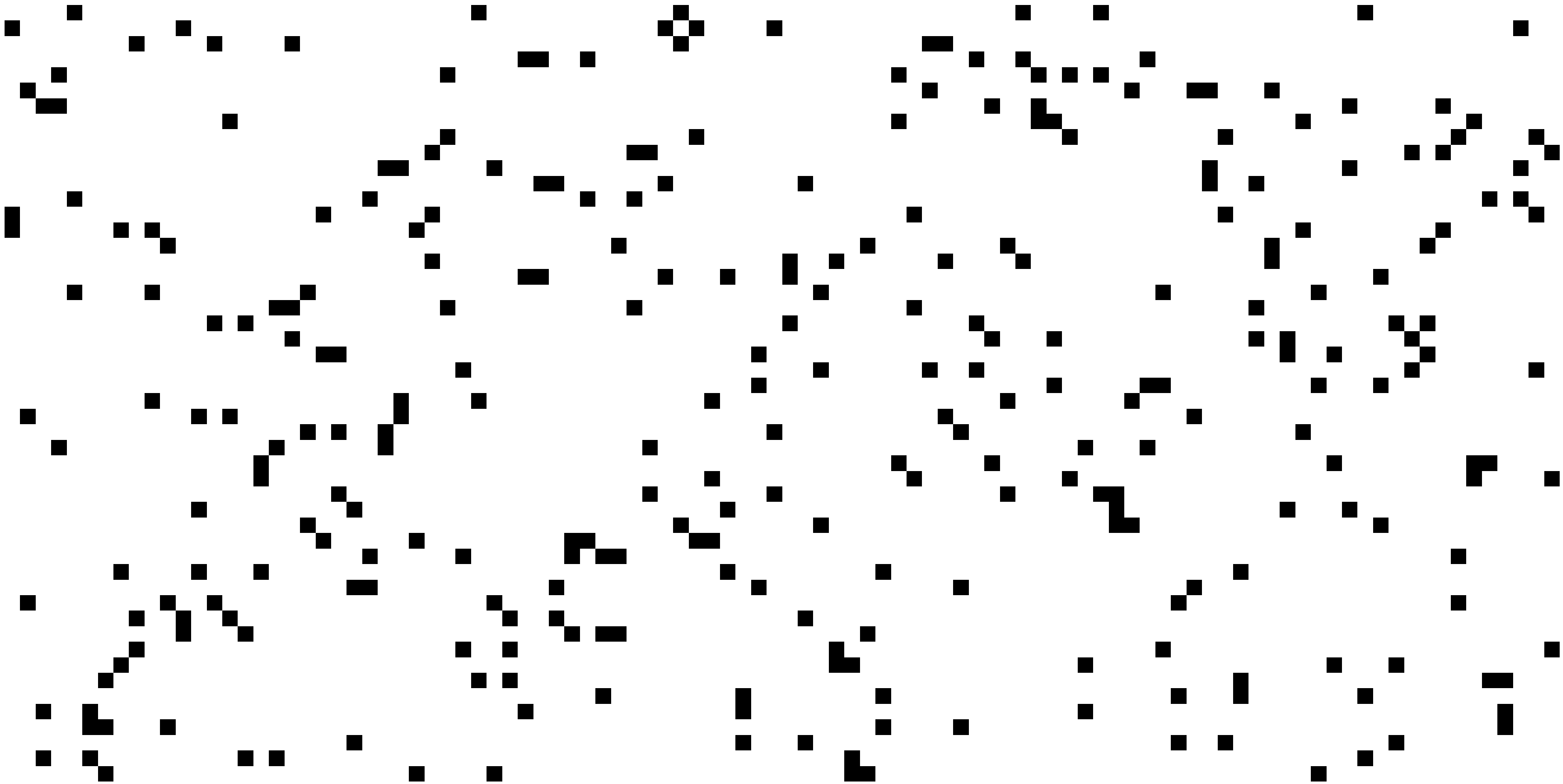
Low density?



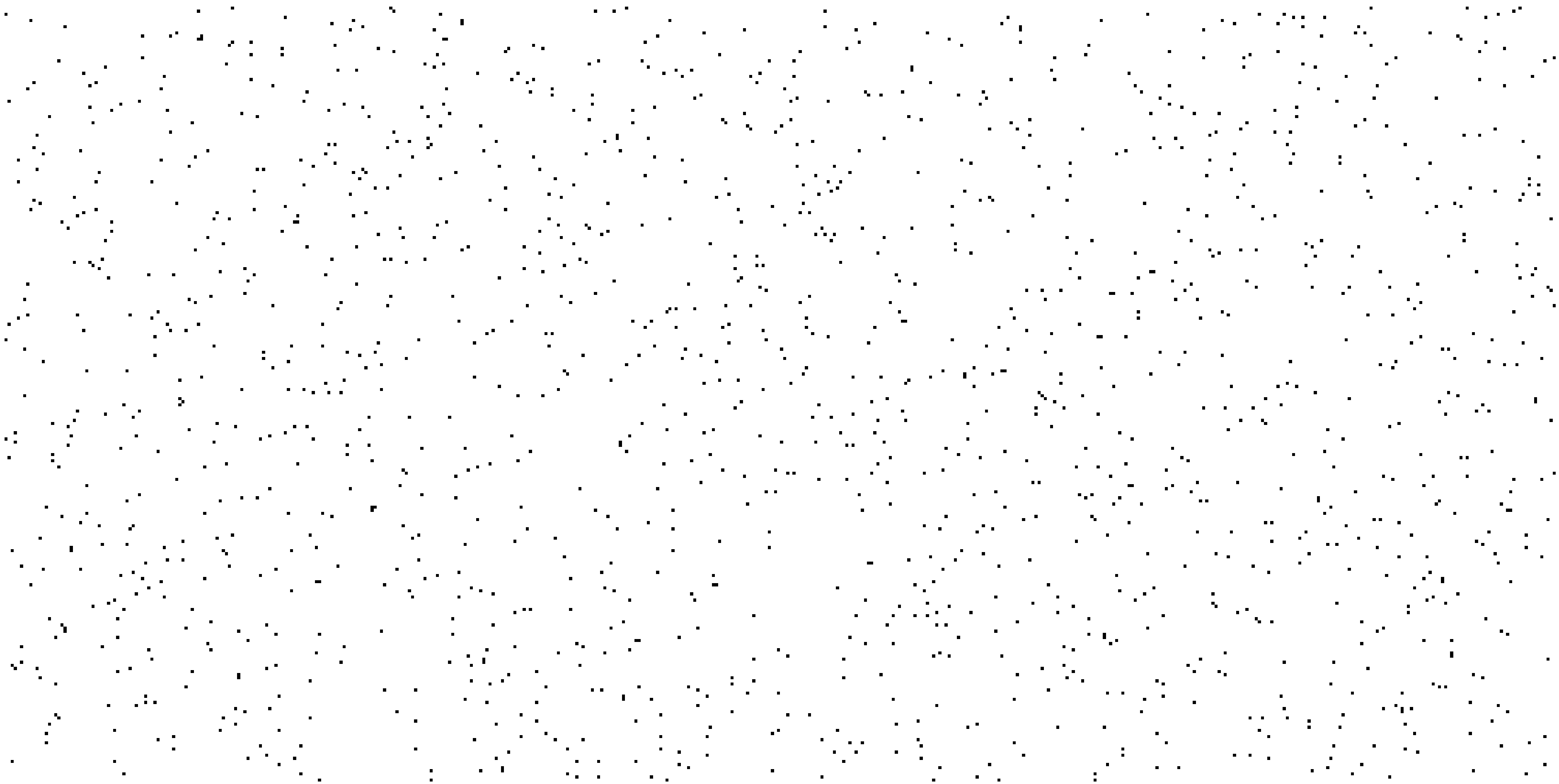
Low density?



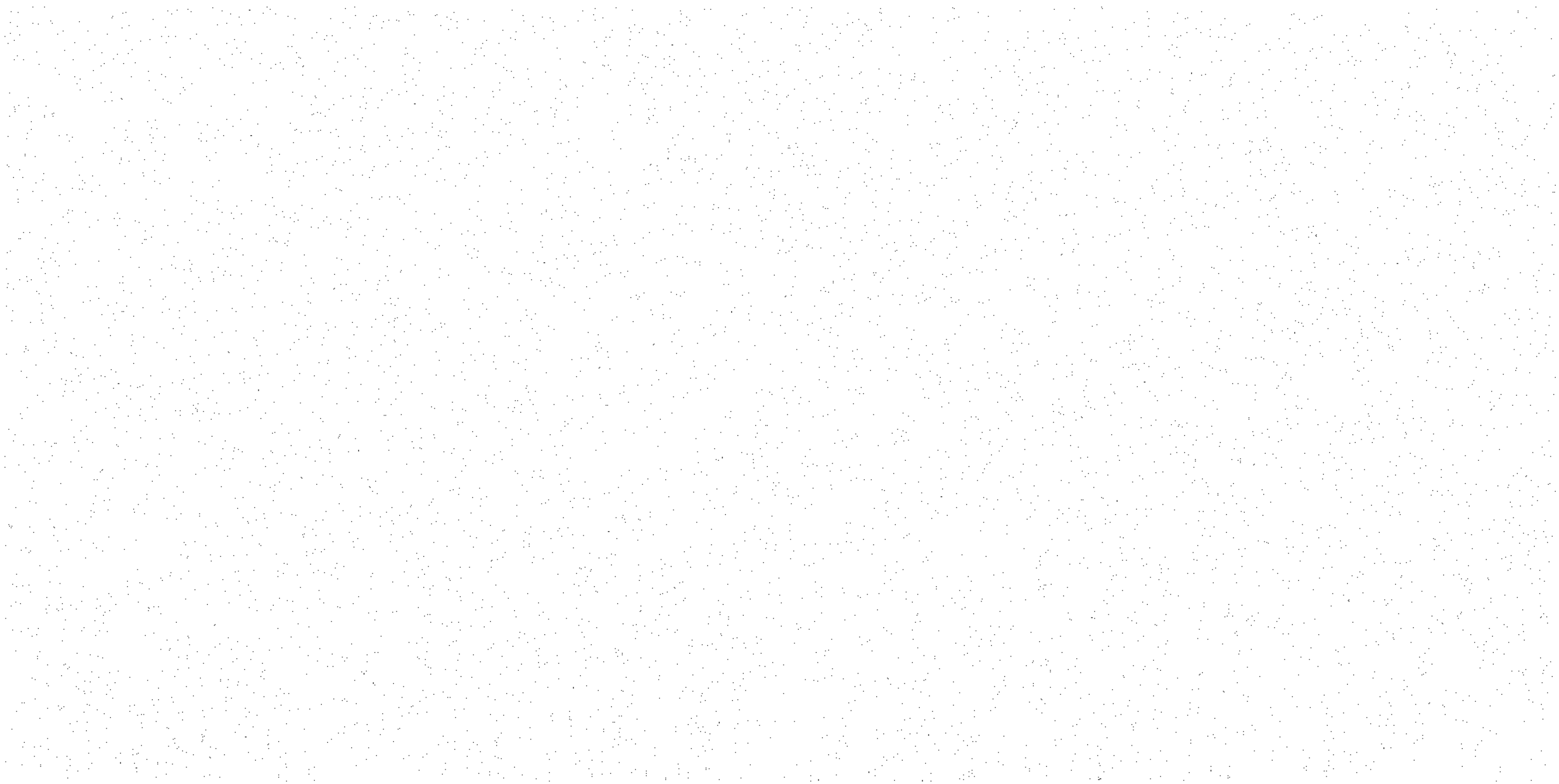
Low density



Low density



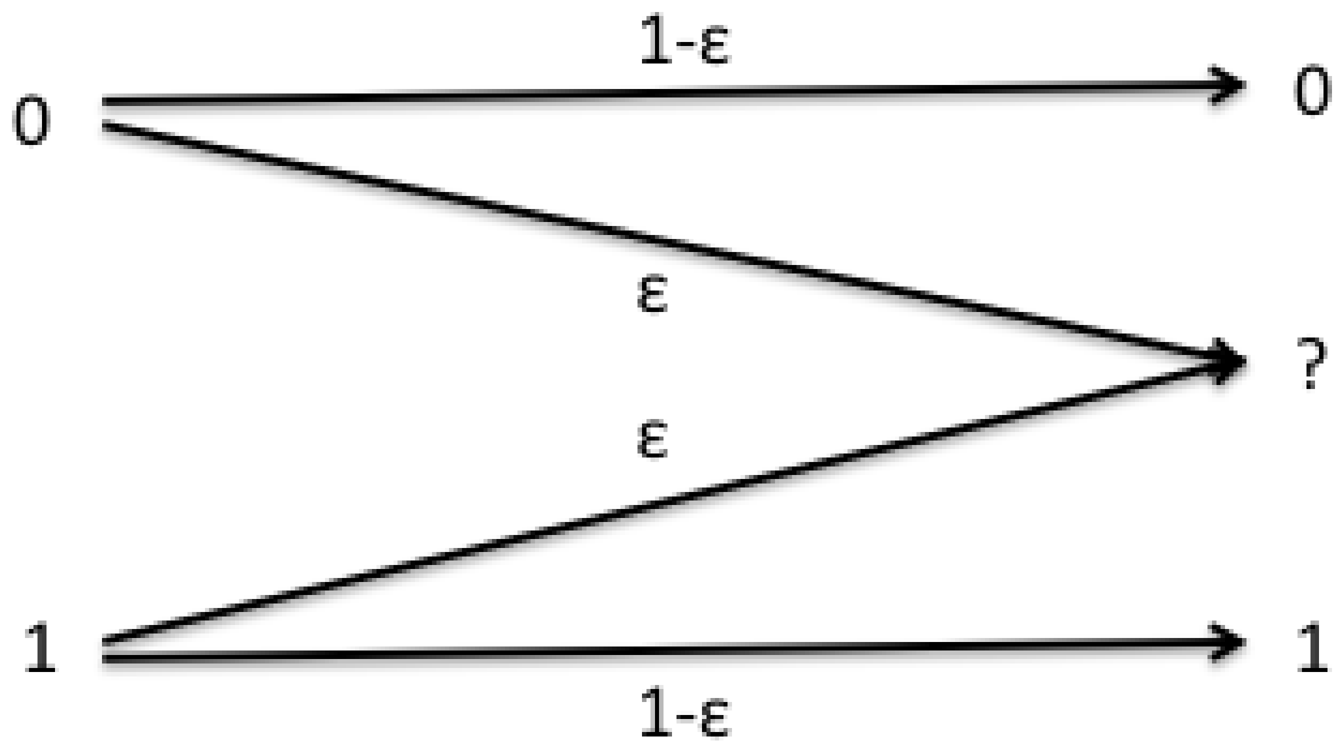
Low density



How do we know LDPC codes are ‘good’?

- We select a simple channel model:
  - Binary Erasure Channel (BEC).
- We simplify the message passing algorithm:
  - Peeling Decoder.
- We analyze its behavior:
  - Density Evolution.
- We show it achieves channel capacity.
  - Optimized LDPC codes.

Binary Erasure Channe



Binary Erasure Channe

- BEC is a simple channel, because ...

Binary Erasure Channe

- BEC is a simple channel, because ...
  - we either have total knowledge of the transmitted bit.
  - or we are completely cluless.
- if $y_\ell = 0$ or $y_\ell = 1$:

$$p(\mathbf{x}_\ell = 0 | y_\ell = 0) = 1 \qquad p(\mathbf{x}_\ell = 1 | y_\ell = 0) = 0$$
$$p(\mathbf{x}_\ell = 0 | y_\ell = 1) = 0 \qquad p(\mathbf{x}_\ell = 1 | y_\ell = 1) = 1$$

- if $y_\ell =?$:

$$p(\mathbf{x}_\ell = 0 | y_\ell =?) = 0.5 \qquad p(\mathbf{x}_\ell = 1 | y_\ell =?) = 0.5$$

Message Passing over the BEC

- Initial Message:

$$r_{\mathbf{x}_2 \to A}(x_2) \propto p(\mathbf{x}_2|y_2)$$

- Message from factors to Variables:

$$r_{A \to x_2}(x_2) = \sum_{x_1, x_3} f_A(x_1, x_2, x_3) q_{x_1 \to A}(x_1) q_{x_3 \to A}(x_3)$$

- What happens if either x_1 or x_2 are erased?
- What happens if neither are erased?
- After the first iteration:

$$p(x_2|y_2, y_1, y_2, y_5, y_6) \propto p(\mathbf{x}_2|y_2) r_{A \to x_2}(x_2) r_{B \to x_2}(x_2)$$

Message Passing over the BEC

- Initial Message:

$$r_{\mathbf{x}_2 \to A}(x_2) \propto p(\mathbf{x}_2|y_2)$$

- Message from factors to Variables:

$$r_{A \to x_2}(x_2) = \sum_{x_1, x_3} f_A(x_1, x_2, x_3) q_{x_1 \to A}(x_1) q_{x_3 \to A}(x_3)$$

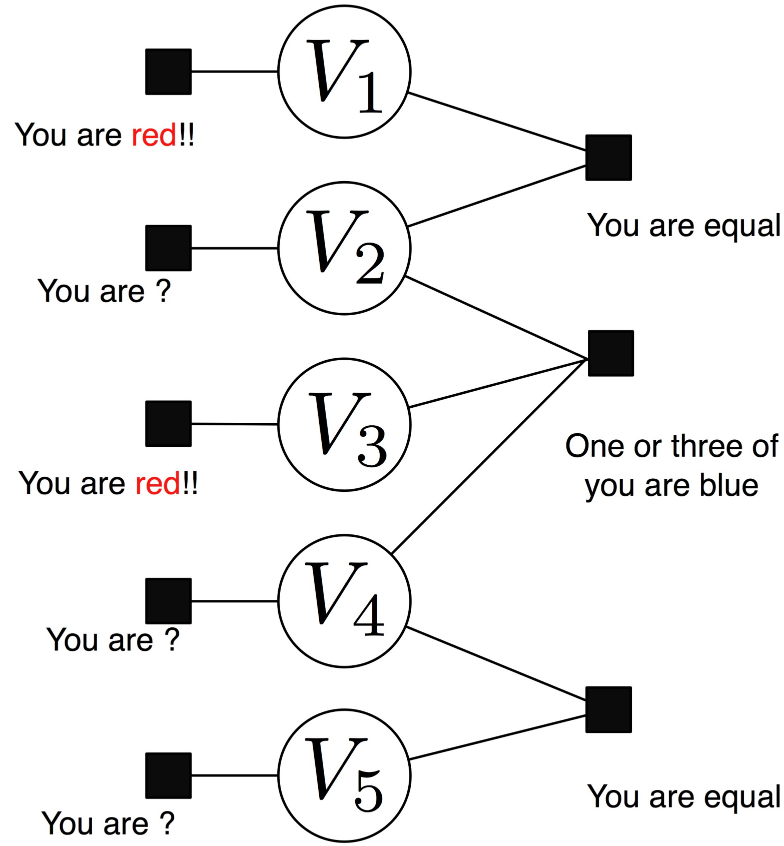
- What happens if either x_1 or x_2 are erased?
- What happens if neither are erased?
- After the first iteration:

$$p(x_2|y_2, y_1, y_2, y_5, y_6) \propto p(\mathbf{x}_2|y_2) r_{A \to x_2}(x_2) r_{B \to x_2}(x_2)$$

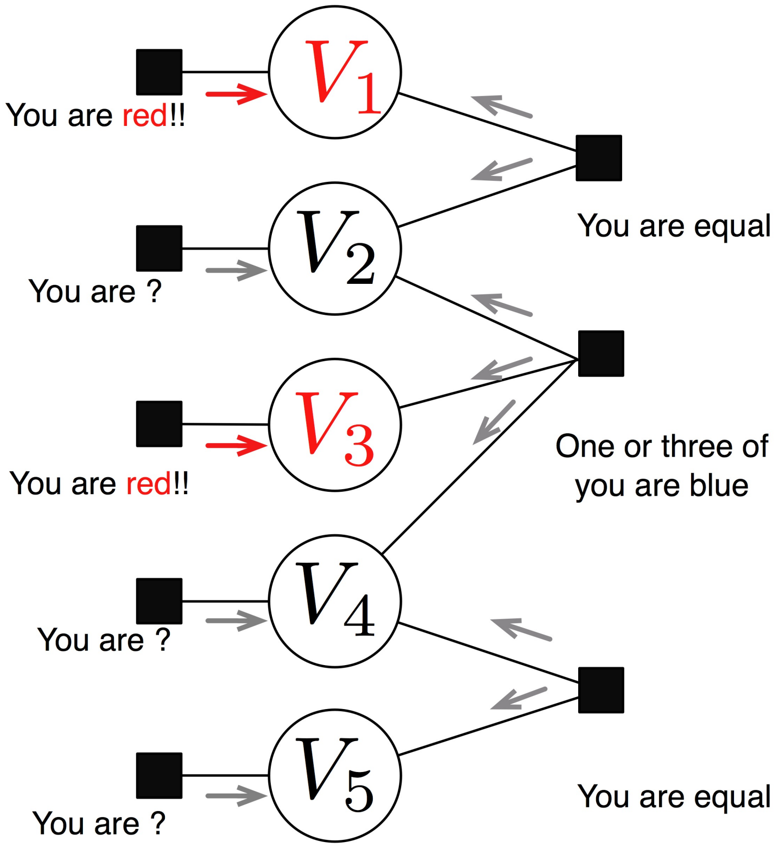
- What happens if one of them is not erased?

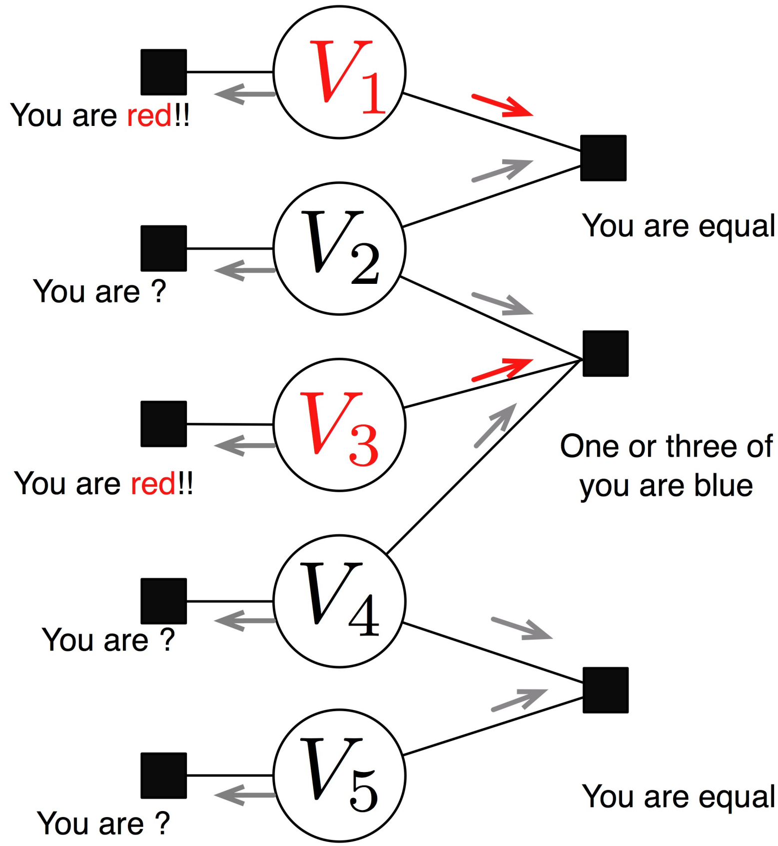
Example

- blue is zero.
- red is one.

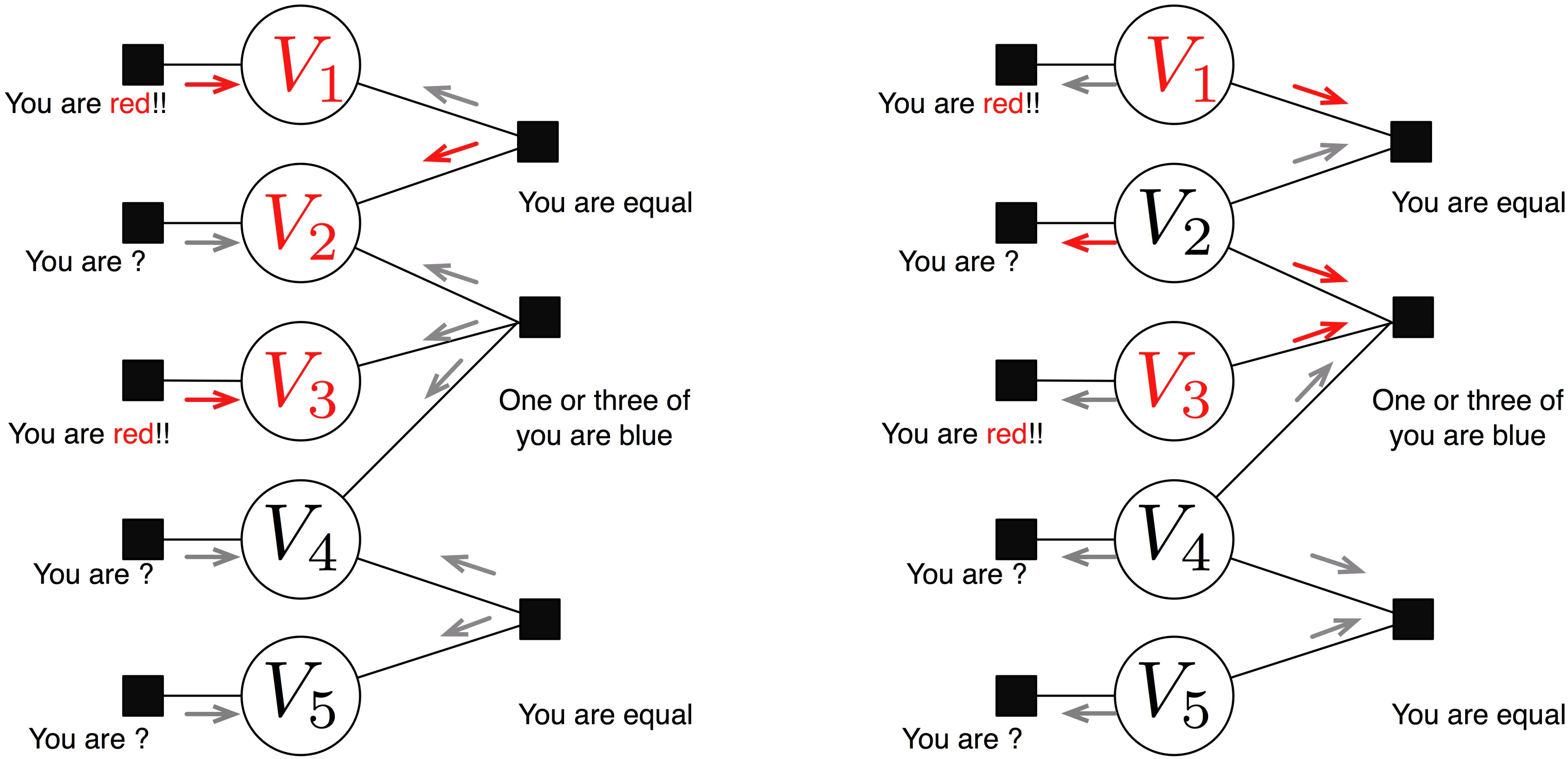


Example

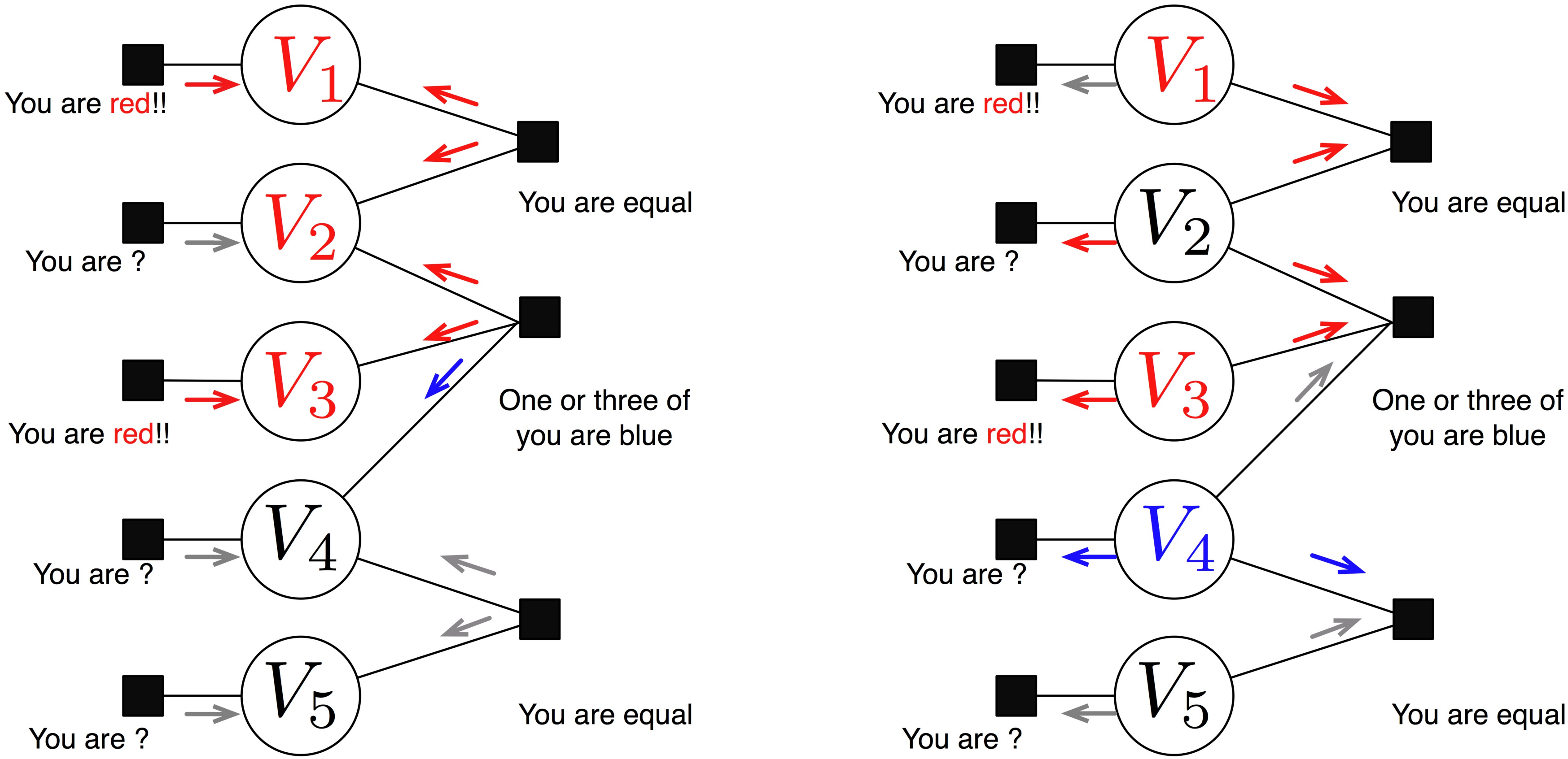

V_1
V_2
V_3
V_4
V_5
You are red!!
You are ?
You are red!!
You are ?
You are ?
You are equal
One or three of you are blue
You are equal


V_1
V_2
V_3
V_4
V_5
You are red!!
You are ?
You are red!!
You are ?
You are ?
You are equal
One or three of you are blue
You are equal

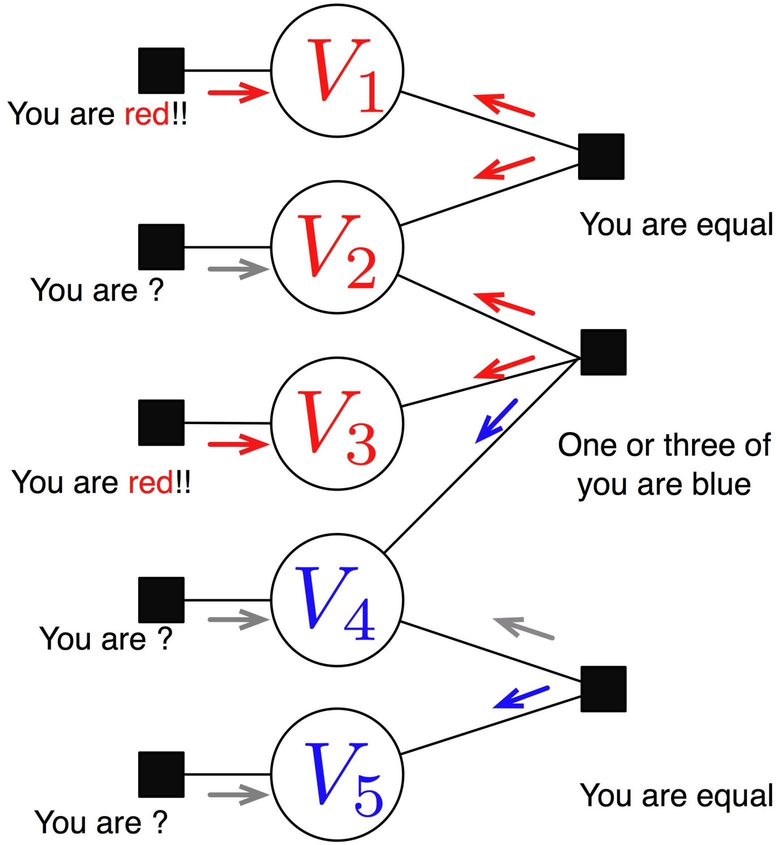
Example


V_1
You are red!!
V_2
You are ?
V_3
You are red!!
V_4
You are ?
V_5
You are ?
You are equal
One or three of you are blue
You are equal
V_1
You are red!!
V_2
You are ?
V_3
You are red!!
V_4
You are ?
V_5
You are ?
You are equal
One or three of you are blue
You are equal

Example


V_1
V_2
V_3
V_4
V_5
You are red!!
You are ?
You are red!!
You are ?
You are ?
You are equal
One or three of you are blue
You are equal
V_1
V_2
V_3
V_4
V_5
You are red!!
You are ?
You are red!!
You are ?
You are ?
You are equal
One or three of you are blue
You are equal

Example

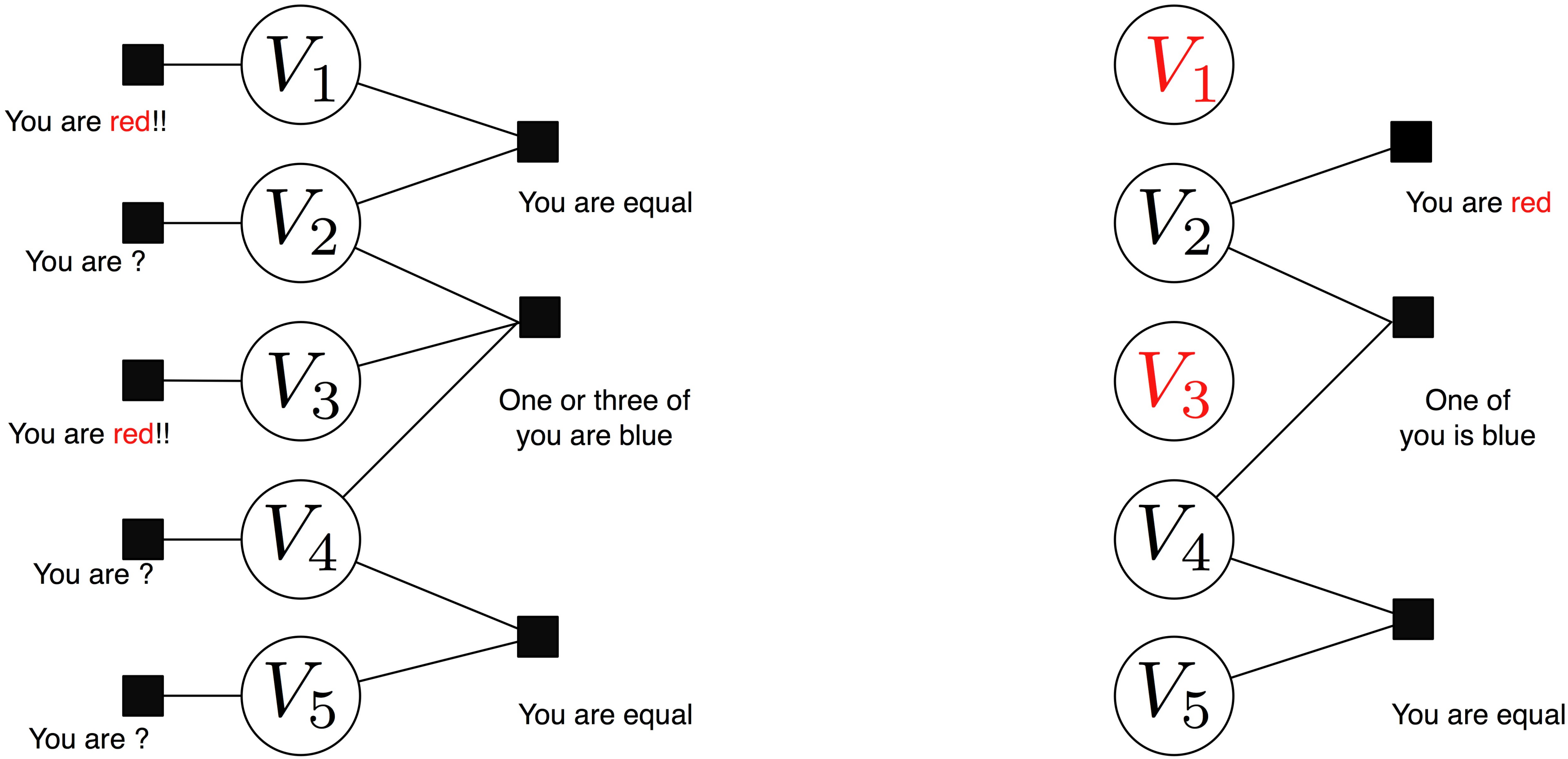


V_1
V_2
V_3
V_4
V_5
You are red!!
You are ?
You are red!!
You are ?
You are ?
You are equal
One or three of you are blue
You are equal

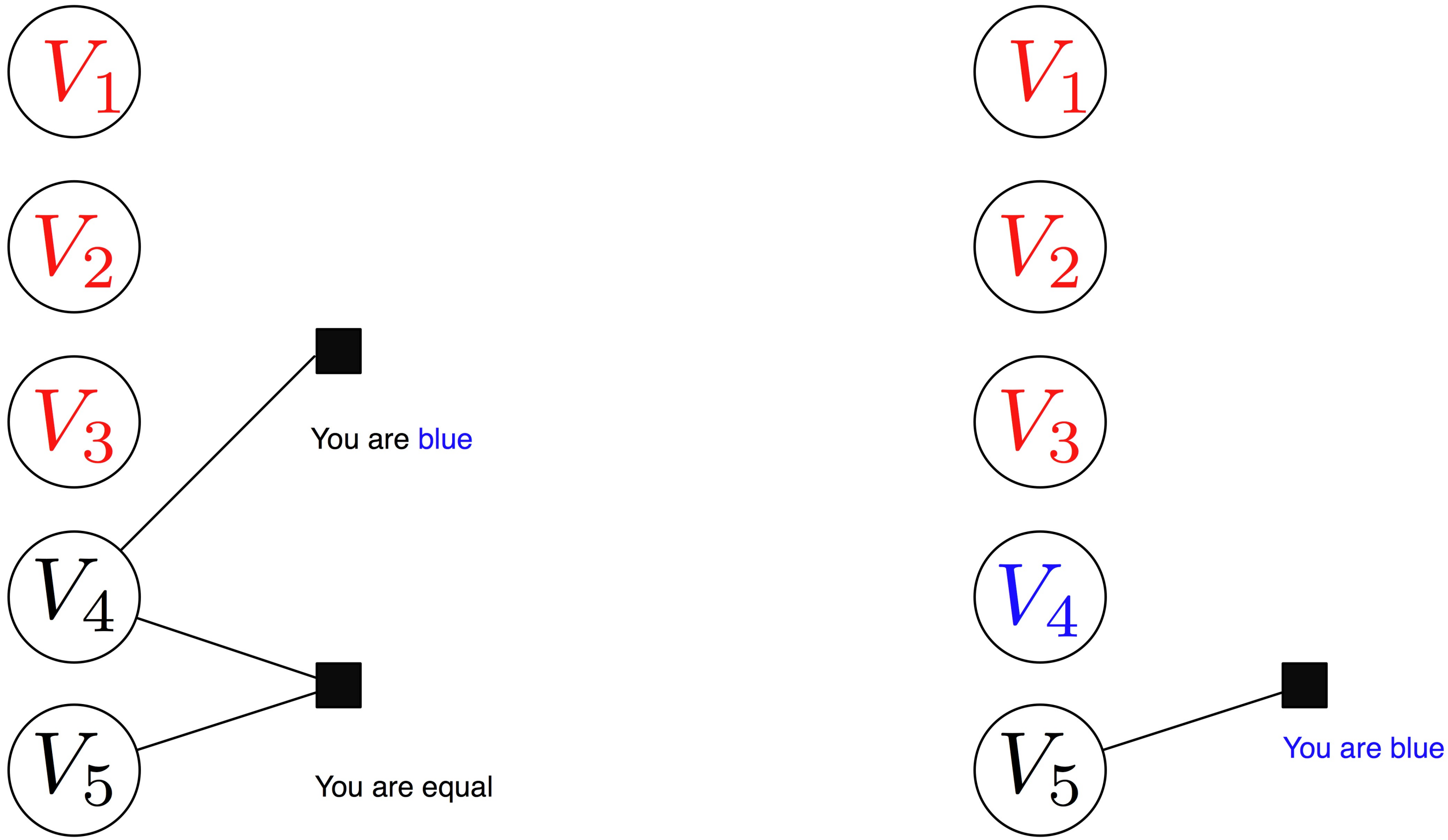
Message Passing over the BEC

- From this example we reduce the message passing algorithm to two simple rules:
  - Variable is de-erase if it gets a single de-erasure message.
  - Factor sends a de-erasure message if all the other variables are known.
- The Peeling decoder:
  1. Remove from the bipartite graph all known variables.
  2. Search for factors with a single variable.
  3. Remove those variables from the graph. Go to 2.

Example

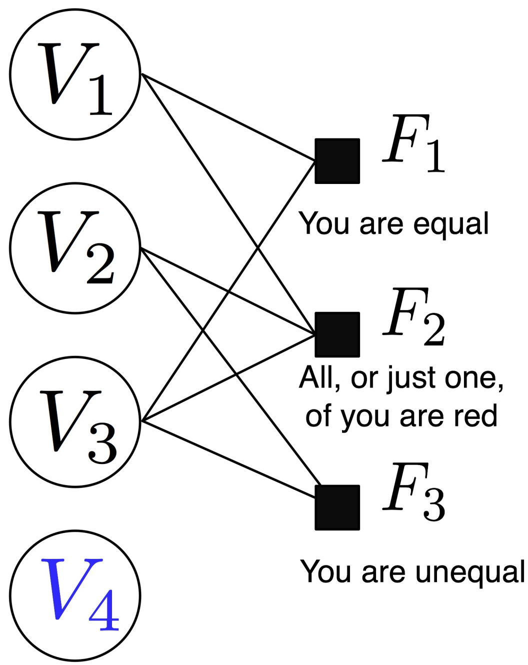

V_1
V_2
V_3
V_4
V_5
You are red!!
You are ?
You are red!!
You are ?
You are ?
You are equal
One or three of you are blue
You are equal
V_1
V_2
V_3
V_4
V_5
You are red
One of you is blue
You are equal

Example


V_1
V_2
V_3
V_4
V_5
You are blue
You are equal
V_1
V_2
V_3
V_4
V_5
You are blue

Can the Peeling Decoder fail?

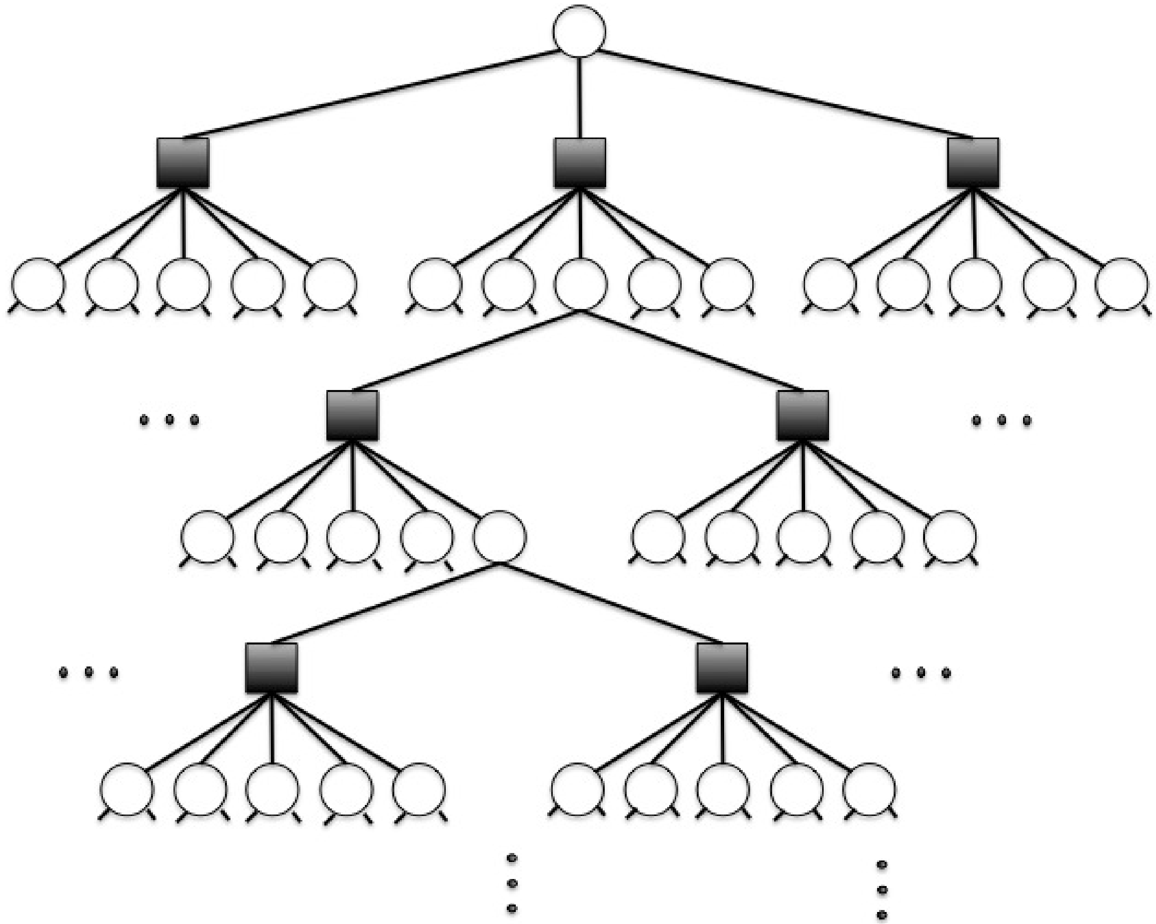
- When all the factors have two or more outputs.
- Can the solution still be unique?
- It dependens.
- It never makes a mistake, though



Analysis

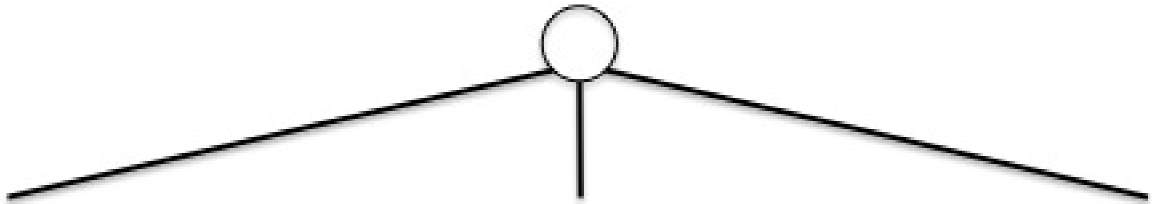
- How good is our LDPC code?
- What error rate in the channel can be decoded?
- Is it equal to the channel capacity?
- Density Evolution answers these questions for LDPC code.
- Reminder: For the BEC $C = 1 - \epsilon$.
- We first analyze a regular LDPC code with 3 ones per column and rate 1/2.

Detangle



First Step

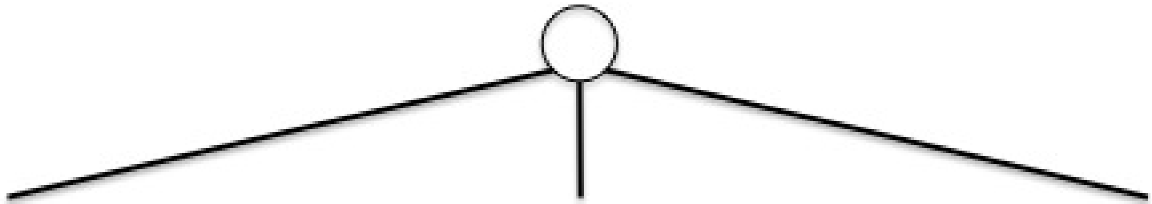
- If the erasure probability in the channel is ϵ, what is the probability that this variable is erased?



First Step

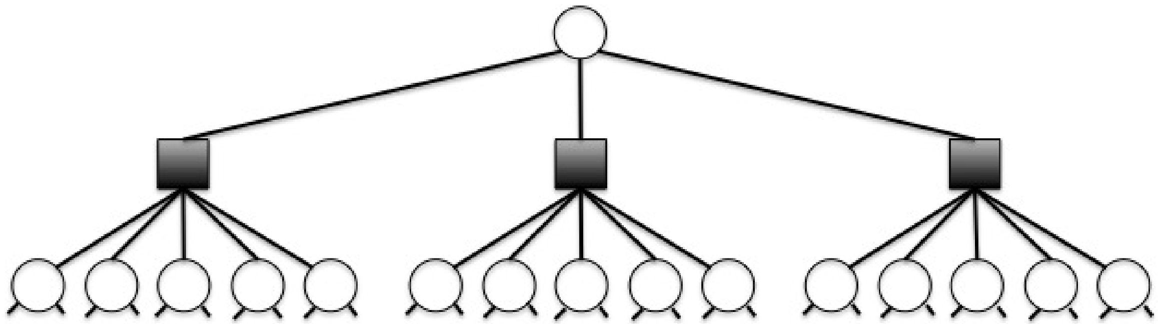
- If the erasure probability in the channel is ϵ, what is the probability that this variable is erased?

$$R_0 = \epsilon$$



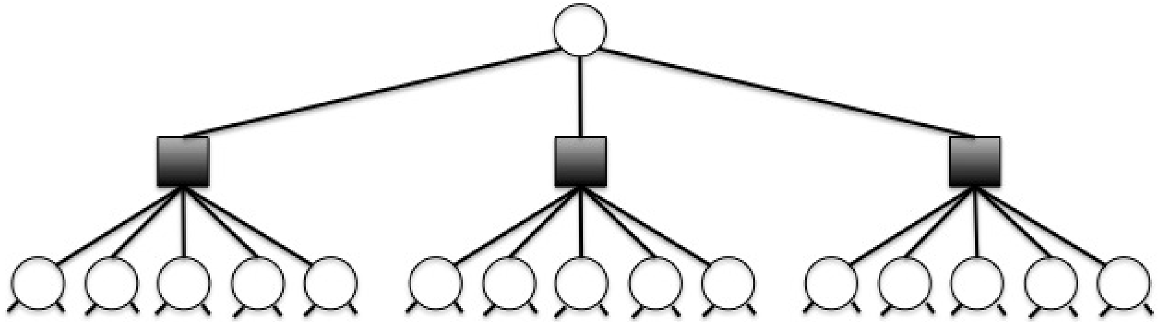
Second Step (a)

- If the erasure probability in the channel is ϵ, what is the probability that each factor sends an erased message?



Second Step (a)

- If the erasure probability in the channel is ϵ, what is the probability that each factor sends an erased message?

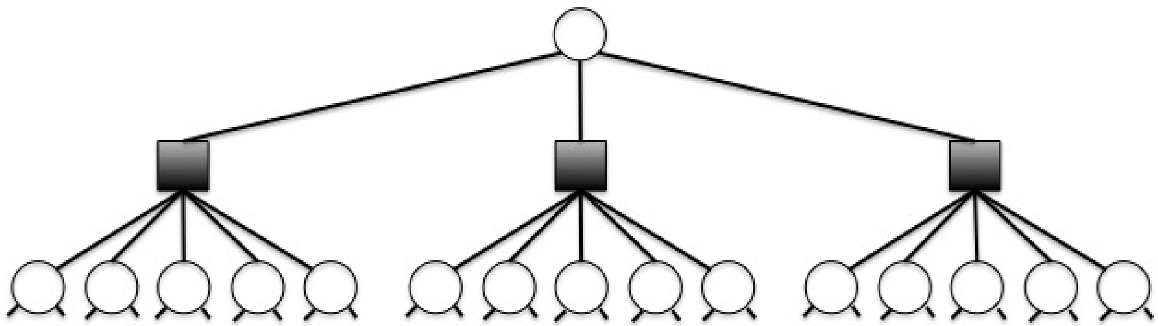


- It sends an erased message if any of the variables is erased:

$$L1 = 1 - (1 - \epsilon)^5$$

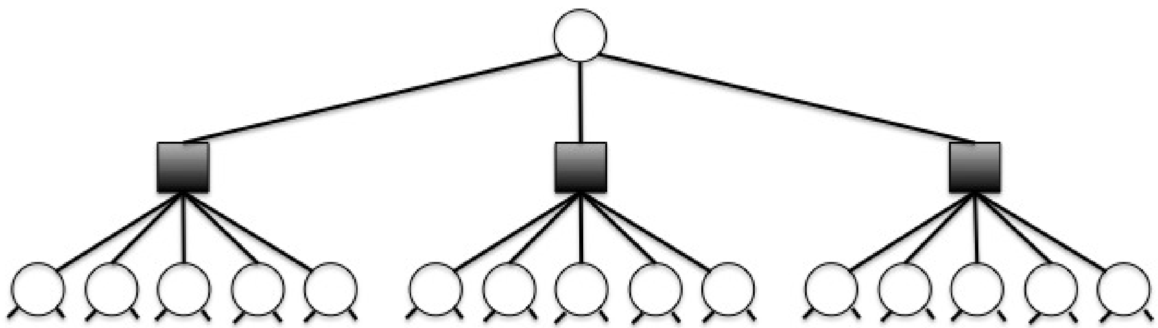
Second Step (b)

- If the erasure probability in the channel is ϵ, what is the probability that the top variable is erased once it has received the messages from the factors?



Second Step (b)

- If the erasure probability in the channel is ϵ, what is the probability that the top variable is erased once it has received the messages from the factors?



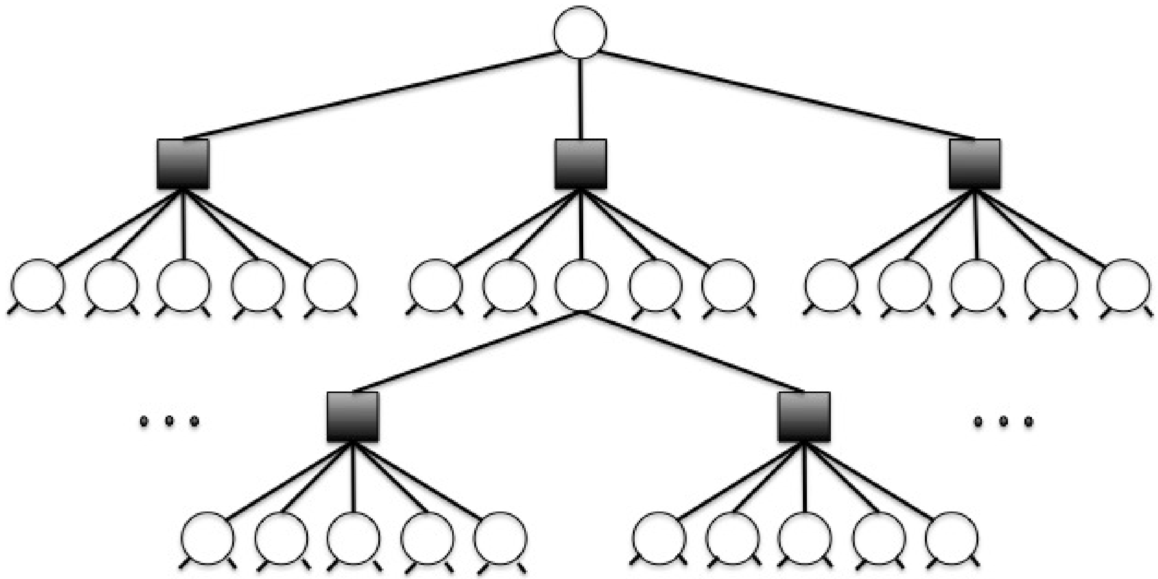
- It is erased is any of the messages are erased:

$$R1 = \epsilon(L1)^3 = \epsilon(1-(1-\epsilon)^5)^3$$

General Step

- The variable in next layer variable is erased with probability:

$$R_{t+1} = \epsilon(1 - (1 - R_t)^5)^2$$



- Why did I change the 3 for the 2?

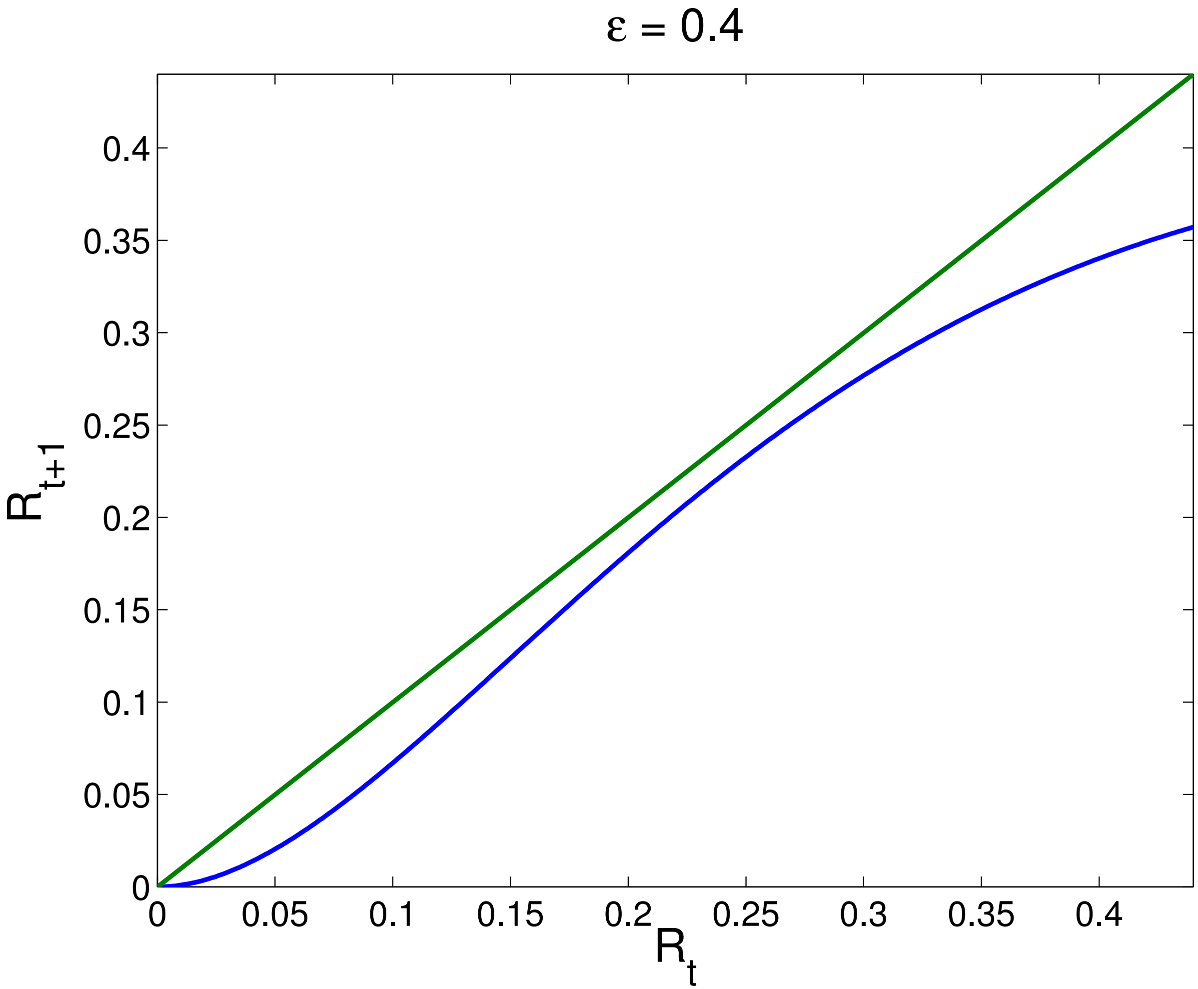
General Step

- The variable in next layer variable is erased with probability:

$$R_{t+1}(\epsilon, R_t) = \epsilon(1 - (1 - R_t)^5)^2$$

- In this recursion, we can expect two things to happen:
  - Either R_{t+1} is reduced in each iteration to zero.
  - Or for some t: $R_{t+1} = R_t$. The algorithm stops decoding.
- What it the maximum ϵ for which the algorithm recovers the transmitted word?

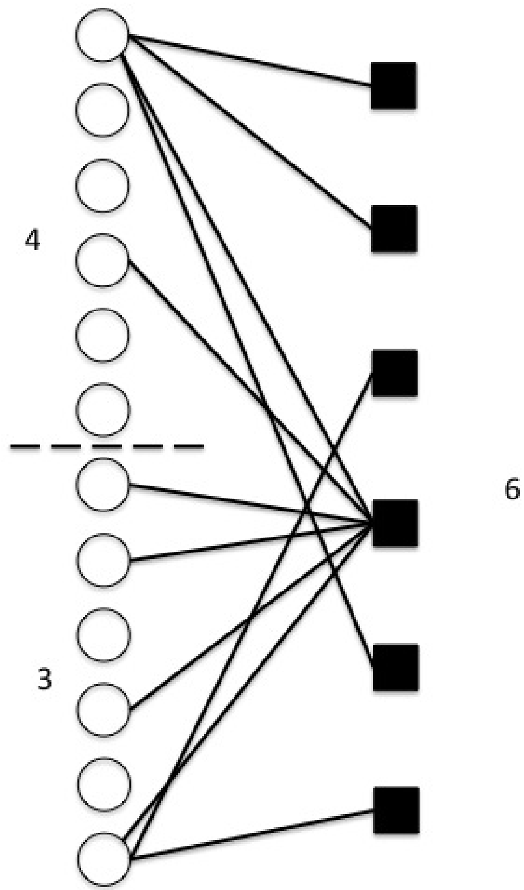
Matlab Demo


ε = 0.4
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
R t+1
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
R t

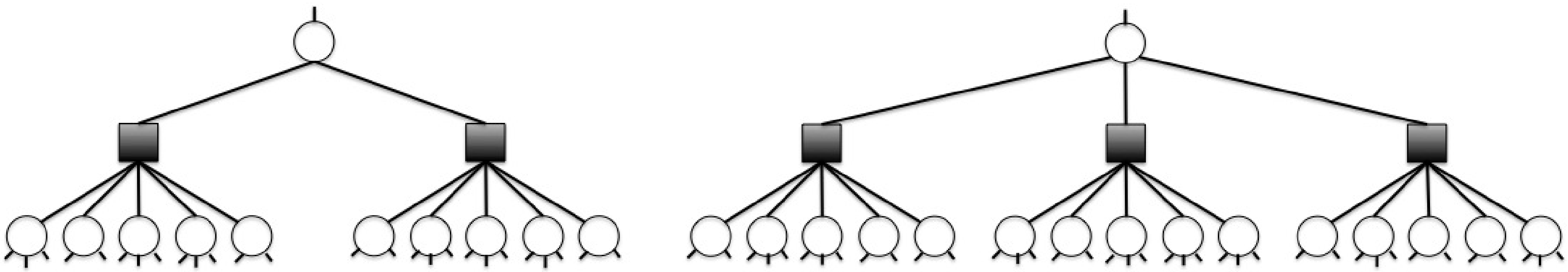
Irregular LDPC codes

- For the Regular LDPC codes maximum error 0.4294 with DP decoding.
- For the Regular LDPC codes maximum error 0.48815 with MAP decoding.
- Channel capacity 0.5.
- Can we get closer to capacity?
  - Using Irregular LDPC codes.

Irregular LDPC codes



Irregular LDPC codes

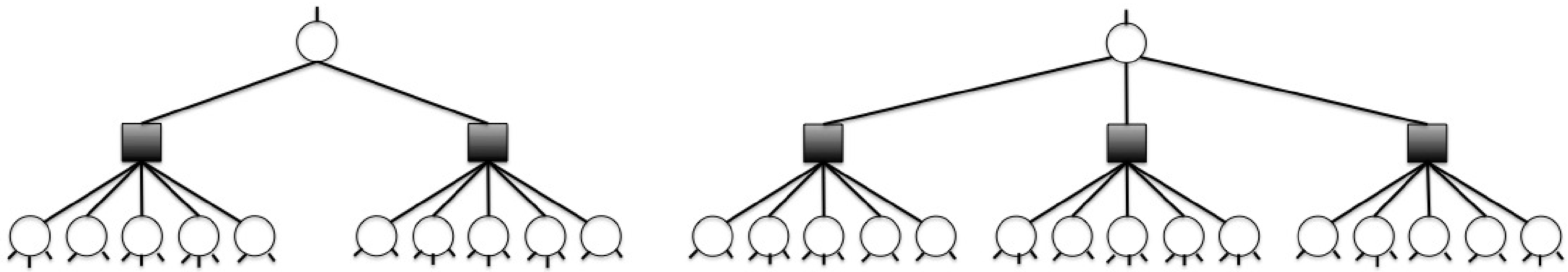


- 50% of the times we will have each variable.
- For the first case: $R_{t+1} = \epsilon(1-(1-R_t)^5)^2$
- For the second case: $R_{t+1} = \epsilon(1-(1-R_t)^5)^3$
- For the general case:

$$R_{t+1}(\epsilon, R_t) = 0.5\epsilon(1-(1-R_t)^5)^2 + 0.5\epsilon(1-(1-R_t)^5)^3$$

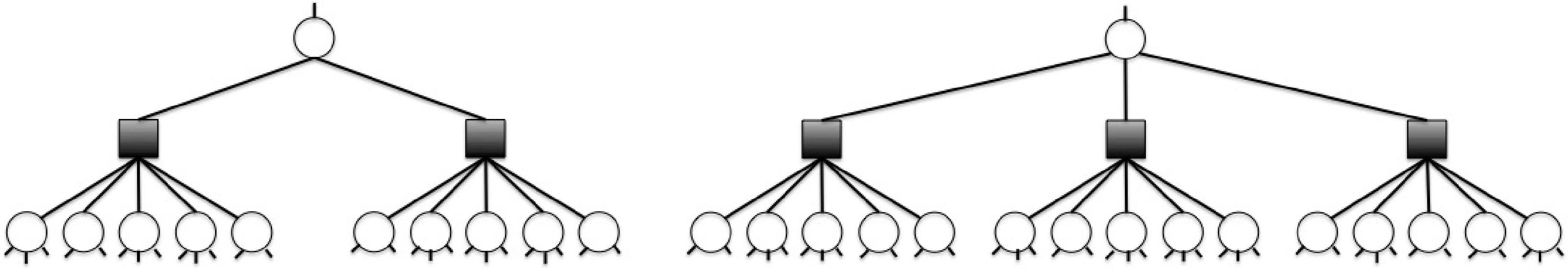
- What is wrong with this interpretation?

Irregular LDPC codes



- This is not the case, because we have more links to variables with 4 connections.
- From the point of view of the variables:
  - 6 variables of degree 3 and 6 variables of degree 4.
- From the point of view of the links:
  - 18 links to var. of degree 3 and 24 links to var. of degree 4.

Irregular LDPC codes



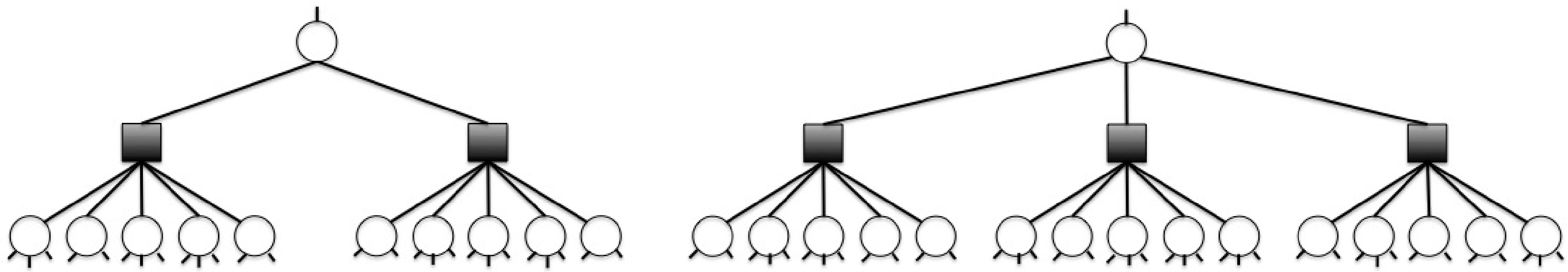
- For point of view of the variables:

$$\Lambda(x) = \sum_i \Lambda_i x^i$$

- For point of view of the links to the variables:

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

Irregular LDPC codes



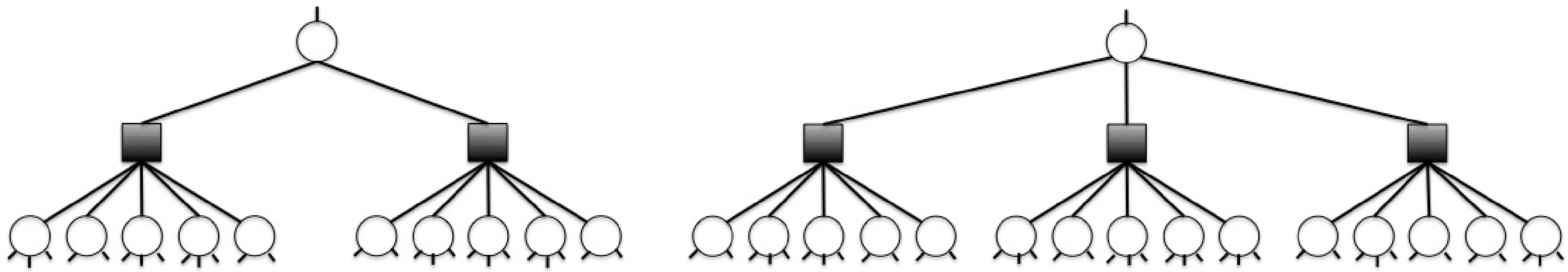
- For point of view of the variables:

$$\Lambda(x) = 6x^3 + 6x^4$$

- For point of view of the links to the variables:

$$\lambda(x) = 18x^2 + 24x^3$$

Irregular LDPC codes



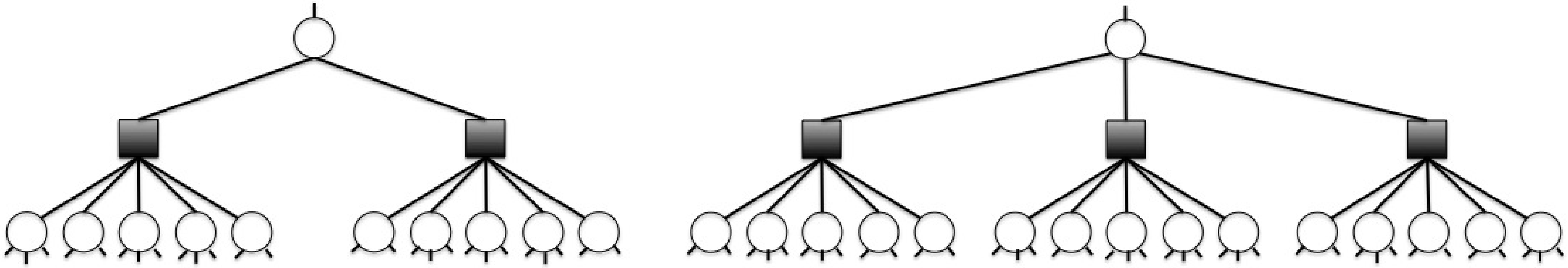
- For point of view of the variables:

$$L(x) = 0.5x^3 + 0.5x^4$$

- For point of view of the links to the variables:

$$\lambda(x) = 0.4286x^2 + 0.5714x^3$$

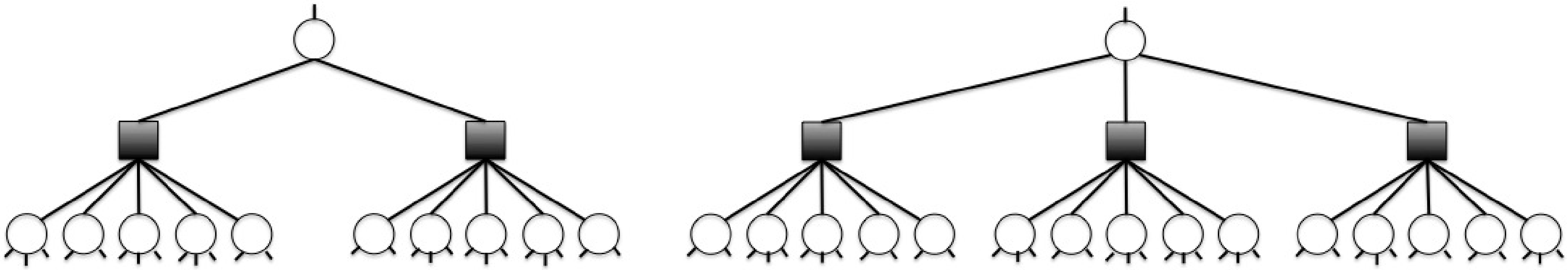
Irregular LDPC codes



- 42.86% 57.14%
- For the first case: $R_{t+1} = \epsilon(1-(1-R_t)^5)^2$
- For the second case: $R_{t+1} = \epsilon(1-(1-R_t)^5)^3$
- For the general case:

$$R_{t+1}(\epsilon, R_t) = 0.4286\epsilon(1-(1-R_t)^5)^2 + 0.5714\epsilon(1-(1-R_t)^5)^3$$

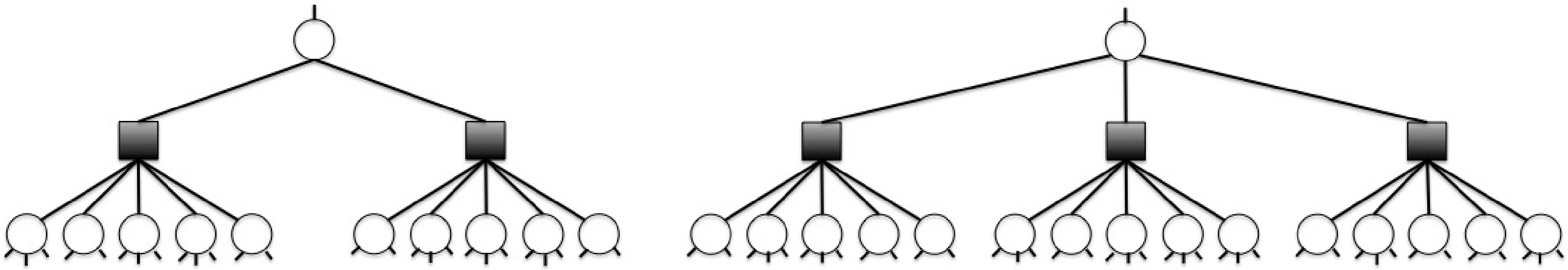
Irregular LDPC codes



- 42.86% 57.14%
- For the first case: $R_{t+1} = \epsilon(1 - (1 - R_t)^5)^2$
- For the second case: $R_{t+1} = \epsilon(1 - (1 - R_t)^5)^3$
- For the general case:

$$R_{t+1}(\epsilon, R_t) = \lambda\left(\epsilon(1 - (1 - R_t)^5)\right)$$

Irregular LDPC codes



- 42.86% 57.14%
- For the first case: $R_{t+1} = \epsilon(1-(1-R_t)^5)^2$
- For the second case: $R_{t+1} = \epsilon(1-(1-R_t)^5)^3$
- For the general case:

$$R_{t+1}(\epsilon, R_t) = \lambda\left(\epsilon(1-\rho(1-R_t))\right)$$

Definitions

- Number of variables with i links:

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

- Number of checks with i links:

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

- Number of variables: $\Lambda(1) = n$.
- Number of checks: $P(1) = n(1 - r)$.
- Rate: $r = 1 - \frac{P(1)}{\Lambda(1)}$.

Definitions

- Fraction of variables with i links:

$$L(x) = \frac{\Lambda(x)}{\Lambda(1)} = \frac{1}{\Lambda(1)} \sum_{i=1}^{\ell_{\max}} \Lambda_i x^i$$

- Fraction of checks with i links:

$$R(x) = \frac{P(x)}{P(1)} = \frac{1}{P(1)} \sum_{i=1}^{r_{\max}} P_i x^i$$

- $L(1) = 1$.
- $R(1) = 1$.
- Rate: $r = 1 - \frac{L'(1)}{R'(1)}$.

Definitions

- Fraction of links connected to variables with i links:

$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)} = \frac{L'(x)}{L'(1)} = \sum_{i=1}^{\ell_{\max}} \lambda_i x^{i-1}$$

- Fraction of links connected to checks with i links:

$$\rho(x) = \frac{P'(x)}{P'(1)} = \frac{R'(x)}{R'(1)} = \sum_{i=1}^{r_{\max}} \rho_i x^{i-1}$$

- $\ell_{avg} = \frac{1}{\int_0^1 \lambda(x)dx}$.
- $r_{avg} = \frac{1}{\int_0^1 \rho(x)dx}$.
- Rate: $r = 1 - \frac{\ell_{avg}}{r_{avg}} = 1 - \frac{\int_0^1 \rho(x)}{\int_0^1 \lambda(x)}$.

Sequence of Capacity Achieving Codes

- Multiplicative Gap to Capacity:

$$r(\lambda, \rho) = (1 - \delta)(1 - \epsilon_{BP})$$

If delta were zero the BP decoder achieves capacity.

- It can be proven that

$$\delta(\lambda, \rho) \geq \frac{r^{r_{avg}-1}(1-r)}{1 + r^{r_{avg}-1}(1-r)}$$

- Meaning that the avarage right degree distribution needs to go to infinity for the codes to get to capacity.

- We can only expect to design a sequence of capacity achieving codes.

Sequence of Capacity Achieving Codes

- We say that the sequence $\{\lambda^{(N)}, \rho^{(N)}\}_{N\geq 1}$ achieve capacity on the $BEC(\epsilon)$ if:

$$\lim_{N\to\infty} r(\lambda^{(N)}, \rho^{(N)}) = 1 - \epsilon$$
$$\lim_{N\to\infty} \delta(\lambda^{(N)}, \rho^{(N)}) = 0$$

- Example for $\alpha^{-1} \in \mathbb{N}$ and N:

$$\lambda_\alpha^{(N)}(x) = \frac{\widehat{\lambda}_\alpha^{(N)}(x)}{\widehat{\lambda}_\alpha^{(N)}(1)} \qquad \widehat{\lambda}_\alpha^{(N)}(x) = \sum_{i=1}^{N} \binom{\alpha}{i} (-1)^{i-1} x^i$$
$$\rho_\alpha(x) = \mathrm{x}^{1/\alpha}$$

Example of a Sequence of Capacity Achieving Codes

- We can obtain that:

$$r(\lambda,\rho) = \frac{\frac{N}{\alpha}\binom{\alpha}{N}(-1)^{N-1}\left(1-\frac{1}{N}\right)}{1-\frac{1}{N}\frac{N}{\alpha}\binom{\alpha}{N}(-1)^{N-1}}$$

$$\delta(\lambda,\rho) \leq \frac{1-\frac{N}{\alpha}\binom{\alpha}{N}(-1)^{N-1}}{N-\frac{N}{\alpha}\binom{\alpha}{N}(-1)^{N-1}}$$

- if you set $\frac{N}{\alpha}\binom{\alpha}{N}(-1)^{N-1} = 1-\epsilon$, we can reach capacity as $1/\alpha$ and N goes to infinity.

Optimization of Irregular LDPC codes

- Sufficient Condition for obtaining the ML codeword:

$$\lambda\left(\epsilon(1-\rho(1-x))\right)-x\leq 0 \qquad x\in[0,1]$$

- If we fixed $\rho(x)$, the previous equation is linear in λ_i.
- For a fixed $\rho(x)$, the rate is an increasing function of $\sum_i \lambda_i/i$.
- Optimization Procedure:

$$\max_{\lambda_i\geq 0}\left\{\sum_i \frac{\lambda_i}{i}\middle|\sum_{i=2}^{\ell_{\max}}\lambda_i=1;\lambda\left(\epsilon(1-\rho(1-x))\right)-x\leq 0, x\in[0,1]\right\}$$

- We can now fix $\lambda(x)$ and optimize $\rho(x)$ [Not necessary].

Optimization of Irregular LDPC codes

- Optimization procedure:
  - Fix r_{avg}:

$$\rho(x) = \frac{r(r+1-r_{avg})}{r_{avg}} x^{r-1} + \frac{r_{avg} - r(r+1-r_{avg})}{r_{avg}} x^{r}$$

  - Select the objective rate r and ℓ_{max}.
  - Run the optimization problem
- Example: $r_{avg} = 6$, $\ell_{\mathsf{max}} = 8$ and $r = 0.5$:

$$\lambda(x) = 0.409x + 0.202x^2 + 0.0768x^3 + 0.1971x^6 + 0.1151x^7$$

$$\rho(x) = x^5 \qquad\qquad r = 0.5004$$

Why Machine Learning can help?

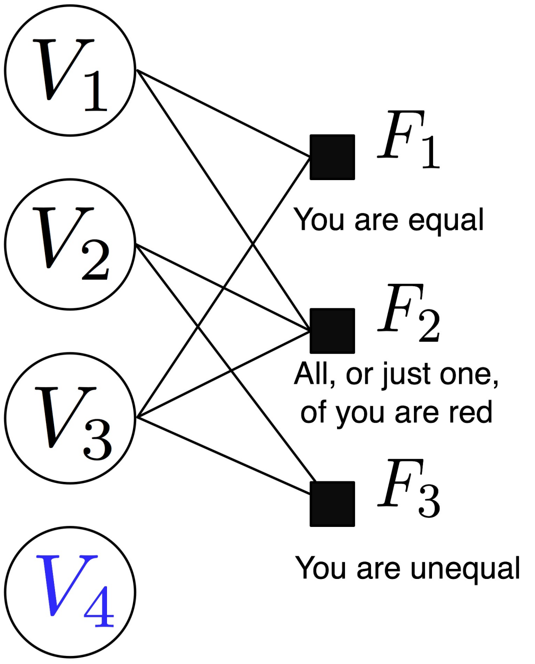
- The decoder of LDPC codes is based on BP.
- We have stronger Approximate Inference Algorithm.
- Expectation Propagation:

$$\widehat{q}(\mathbf{x}) = \arg\min D_{KL}(p(\mathbf{x}|\mathbf{y})||q(\mathbf{x}))$$

- For $q(\mathbf{x}) = \prod_{\ell=1}^{n} q(x_i)$, we recover BP.
- For $q(\mathbf{x}) = \prod_{\ell=1}^{n} q(x_i|\pi_{x_i})$
  - We impose a chain over the the variables.
  - We get a more accurate approximation.

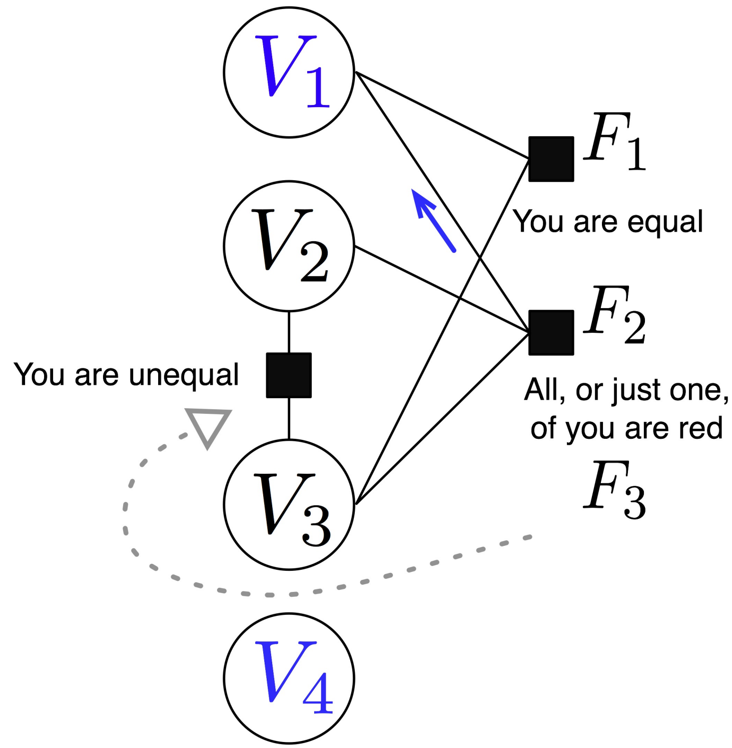
EP with a Tree-Structure

- For this set of variables the PD fails.
- Now we enforce $q(V_3|V_2)$.



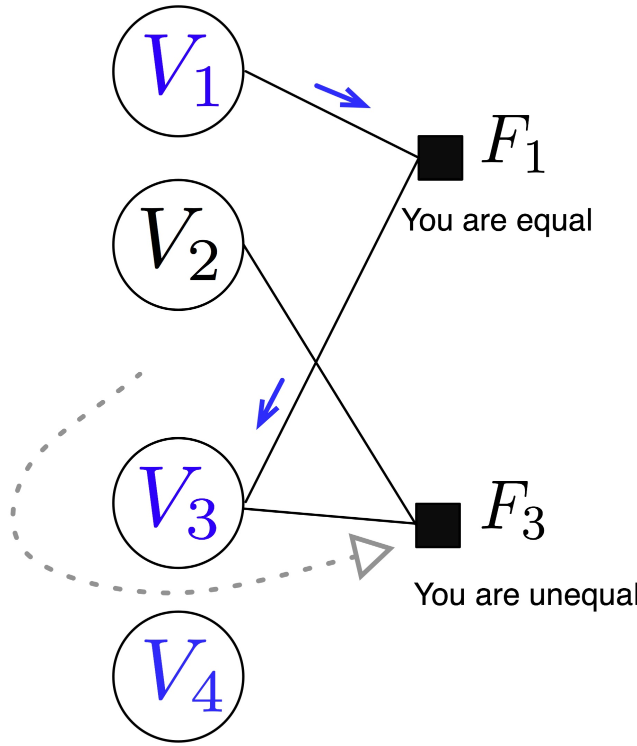
EP with a Tree-Structure

- For this set of variables the PD fails.
- Now we enforce $q(V_3|V_2)$.
- And F_2 tells V_1 that is blue.
- Why?



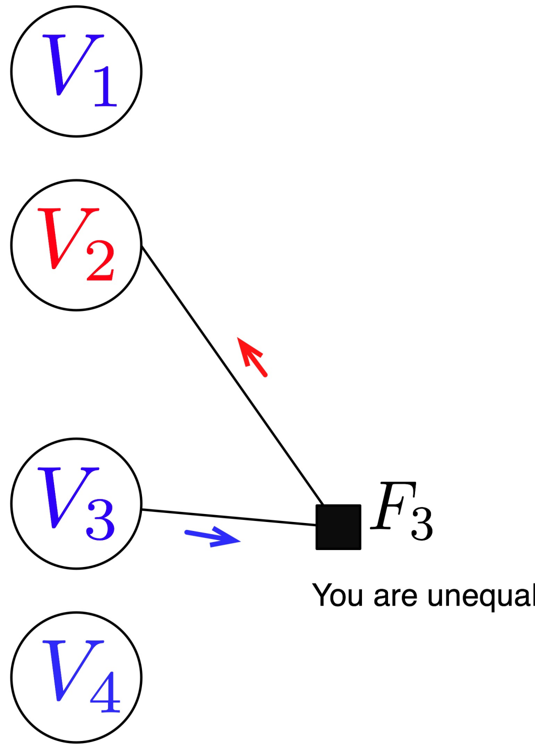
EP with a Tree-Structure

- For this set of variables the PD fails.
- Now we enforce $q(V_3|V_2)$.
- And F_2 tells V_1 that is blue.
- Now F_2 tells V_3 that is blue.



EP with a Tree-Structure

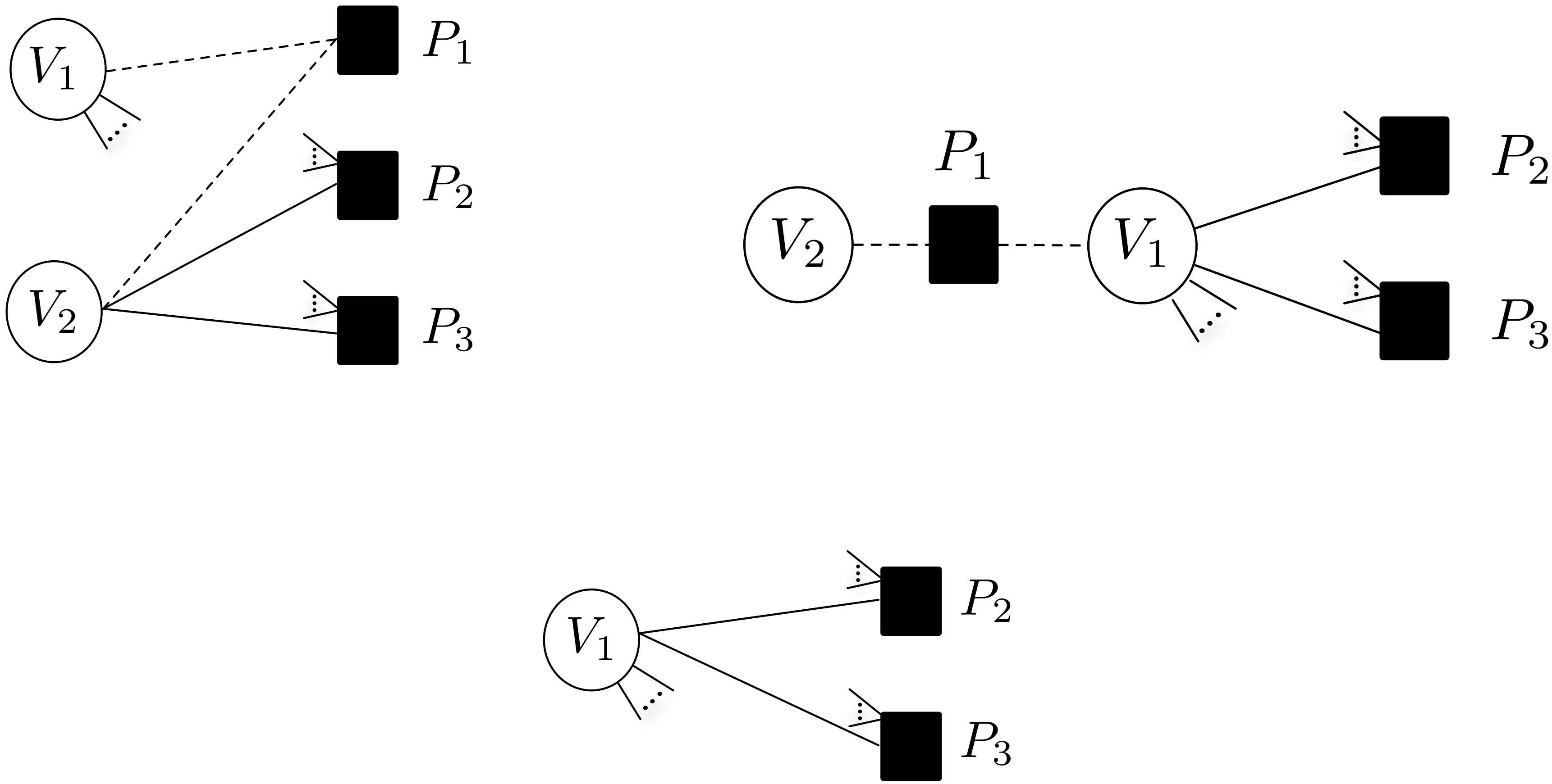
- For this set of variables the PD fails.
- Now we enforce $q(V_3|V_2)$.
- And F_2 tells V_1 that is blue.
- Now F_1 tells V_3 that is blue.
- Now F_3 tells V_2 that is red.



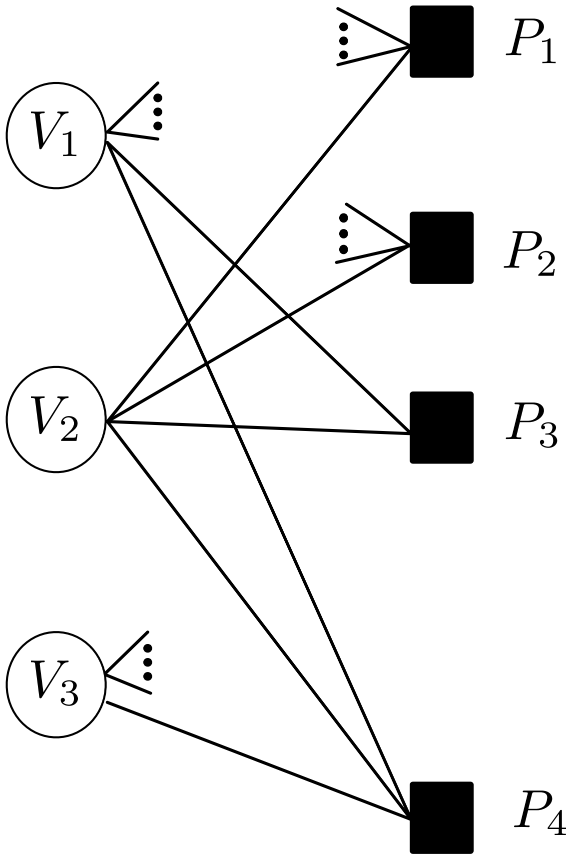
TEP Algorithm

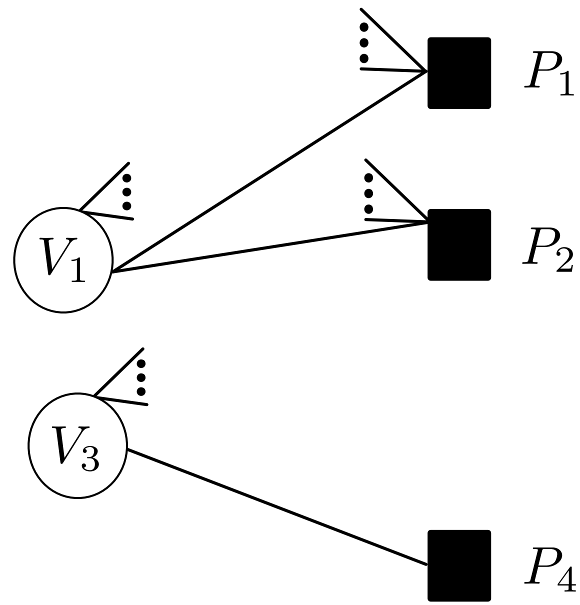
- Initialization: Remove all variables that have not been erased.
  - Change the parity of those that are equal to one.
- Iteration:
  - Look for a check node of degree 1: De-erase the associated variable.
  - Look for a Check node of degree 2: Substitute one variable by the other.
  - Repeat until decoding.

TEP Algorithm



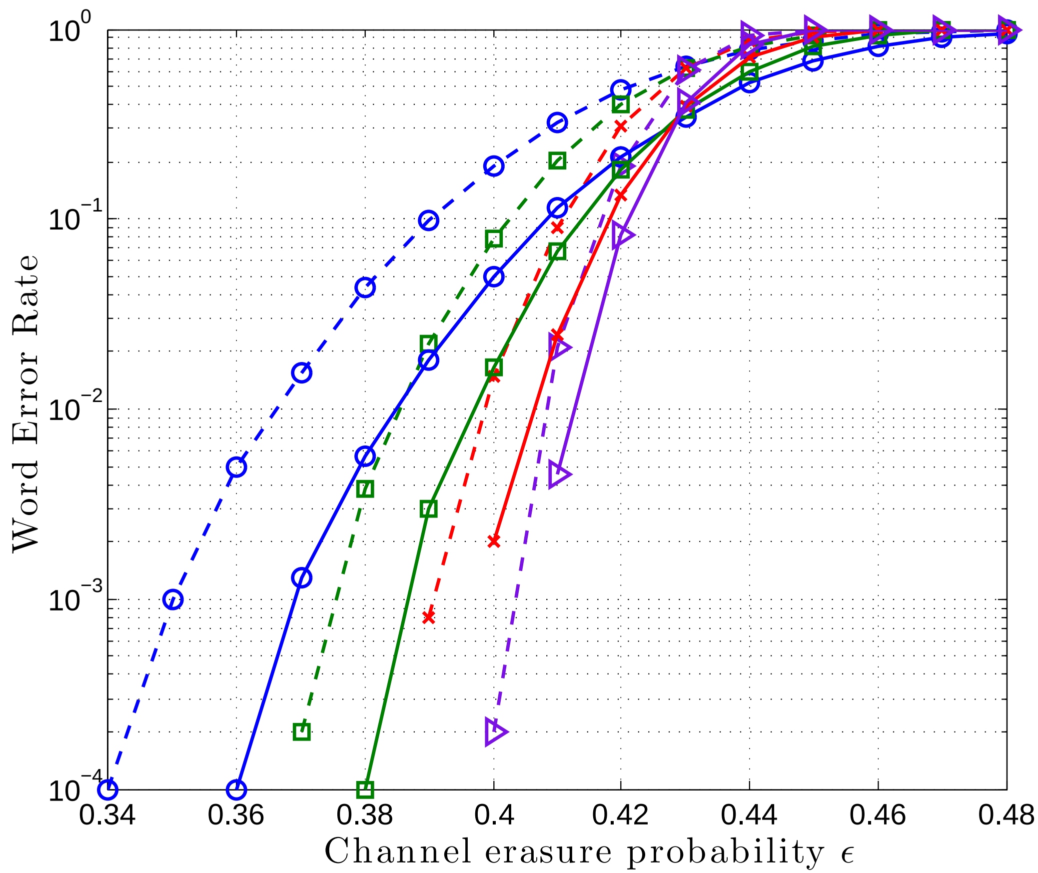
Useful TEP Algorithm





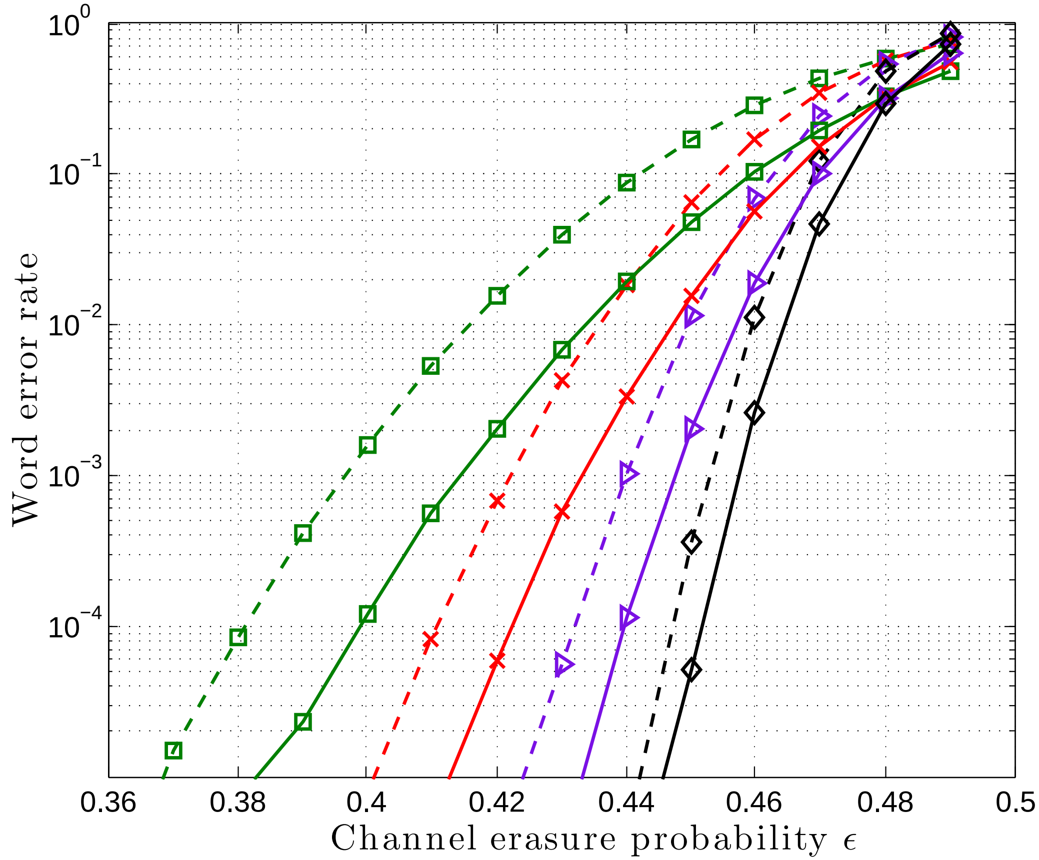
Results

- $\lambda = x^2$ and $\rho = x^5$.
- $n = 2^8$ ($\circ$),
  $n = 2^9$ ($\square$),
  $n = 2^{10}$ ($\times$) and,
  $n = 2^{11}$ ($\triangleright$).



Results

- $\lambda(x) = 1/6x + 5/6x^3$ and $\rho(x) = x^5$.
- $n = 2^9$ ($\square$), $n = 2^{10}$ ($\times$), $n = 2^{11}$ ($\triangleright$) and, $n = 2^{12}$ ($\diamond$).



Extensions

- These results can be extended to more realistic channels:
  - BSC.
  - AWGN.
- Using ExIT charts.
- Results only approximate.
- Polar Codes.

Take Home Message

- There are problems in Information Theory that can be solved using Information Theory
- My view:
  - Non-asymptotic Information Theory.
  - Rate-Distortion.
  - Network Coding.

Thanks!