LDPC Codes: Achieving the Capacity of the Binary Erasure Channel

Matthew C. Valenti

Lane Department of Computer Science and Electrical Engineering
West Virginia University
U.S.A.

Nov. 30, 2009

Outline

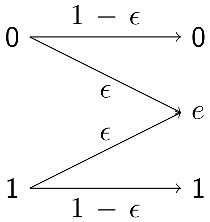
Outline

Error Control Codes

- Consider a data transmission system whereby binary data is segmented into *messages* $\mathbf{u}$ of length k bits.
- Each message is mapped to a unique *codeword* $\mathbf{c}$ of length n bits, where $n > k$.
- The ratio $R = k/n$ is called the *code rate*.
- Simple examples:
  - Repetition code: $k = 1$; Repeat bit n times; $R = 1/n$.
  - Single parity-check code: Codeword is the message and an additional "parity bit"; $n = k + 1$; $R = k/(k + 1)$.

The Binary Erasure Channel

- The BEC has two inputs (data 0 and data 1) and three outputs (data 0, data 1, and *erasure* e).
- A bit is erased with probability ϵ.
- A bit is correctly received with probability $1 - \epsilon$.



- Example applications:
  - Buffer overflows in network routers.
  - Fading in wireless channels.

Capacity of the BEC

- According to information theory, it is possible to reliably communicate over the BEC by using a rate $R = 1 - \epsilon$ code.
- Can be easily achieved if the *transmitter* knows the location of the erasures.
- Example: Transmit $\mathbf{u} = [1 \quad 0 \quad 1 \quad 1]$ with a rate $R = 4/6$ code:

$$\mathbf{c} \;=\; [\underbrace{1}_{c_1=u_1} \; \underbrace{e}_{c_2=X} \; \underbrace{0}_{c_3=u_2} \; \underbrace{e}_{c_4=X} \; \underbrace{1}_{c_5=u_3} \; \underbrace{1}_{c_6=u_4}]$$

  where X can be anything (does not matter, since erased).
- This scheme is not practical, since normally the transmitter won't know where the erasures are located, and therefore doesn't know where to place the message bits.
- Finding practical codes which require only the *receiver* to know the location of the erasures is a challenging problem.

Single Parity-Check Codes

- Consider the following rate $R = 5/6$ parity-check code:

$$\mathbf{c} = [\underbrace{1 \quad 0 \quad 1 \quad 0 \quad 1}_{\mathbf{u}} \quad \underbrace{\quad 1 \quad}_{\text{parity bit}} \quad]$$

- One erasure in *any* position may be corrected:

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

- Problem with using SPC's is that it can only correct a single erasure.

Product Codes: Encoding

- Place data into a k by k rectangular array.
  - Encode each row with a SPC.
  - Encode each column with a SPC.
  - Result is a rate $R = k^2/(k+1)^2$ code.
- Example $k = 2$.

$c_1 = u_1$	$c_2 = u_2$	$c_3 = c_1 \oplus c_2$
$c_4 = u_3$	$c_5 = u_4$	$c_6 = c_4 \oplus c_5$
$c_7 = c_1 \oplus c_4$	$c_8 = c_2 \oplus c_5$	$c_9 = c_3 \oplus c_6$

$=$

1	0	1
1	1	0
0	1	1

Product Codes: Decoding

- Decoding may be performed by iteratively decoding the SPC on each row and column.

Received word:

e	e	1
1	e	0
0	e	e

$\Rightarrow$ Row decoding:

e	e	1
1	1	0
0	e	e

$\Rightarrow$ Column decoding:

1	e	1
1	1	0
0	e	1

$\Rightarrow$ Row decoding:

1	0	1
1	1	0
0	1	1

- Does not achieve capacity. Try decoding:

e	e	1
e	e	0
0	1	1

Linear Codes

$c_1 = u_1$	$c_2 = u_2$	$c_3 = c_1 \oplus c_2$
$c_4 = u_3$	$c_5 = u_4$	$c_6 = c_4 \oplus c_5$
$c_7 = c_1 \oplus c_4$	$c_8 = c_2 \oplus c_5$	$c_9 = c_3 \oplus c_6$

- The example product code is characterized by the set of five linearly-independent equations:

$$\begin{aligned}
c_3 = c_1 \oplus c_2 \quad &\Rightarrow \quad c_1 \oplus c_2 \oplus c_3 = 0 \\
c_6 = c_4 \oplus c_5 \quad &\Rightarrow \quad c_4 \oplus c_5 \oplus c_6 = 0 \\
c_7 = c_1 \oplus c_4 \quad &\Rightarrow \quad c_1 \oplus c_4 \oplus c_7 = 0 \\
c_8 = c_2 \oplus c_5 \quad &\Rightarrow \quad c_2 \oplus c_4 \oplus c_8 = 0 \\
c_9 = c_3 \oplus c_6 \quad &\Rightarrow \quad c_3 \oplus c_6 \oplus c_9 = 0
\end{aligned}$$

- In general, it takes $(n - k)$ linearly-independent equations to specify a *linear* code.

Parity-check Matrices

- The system of equations may be expressed in matrix form as:

$$\mathbf{c}H^T = \mathbf{0}$$

where H is a *parity-check* matrix.

- Example:

$$\begin{array}{rcl} c_1 \oplus c_2 \oplus c_3 & = & 0 \\ c_4 \oplus c_5 \oplus c_6 & = & 0 \\ c_1 \oplus c_4 \oplus c_7 & = & 0 \\ c_2 \oplus c_4 \oplus c_8 & = & 0 \\ c_3 \oplus c_6 \oplus c_9 & = & 0 \end{array} \quad \Leftrightarrow \quad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

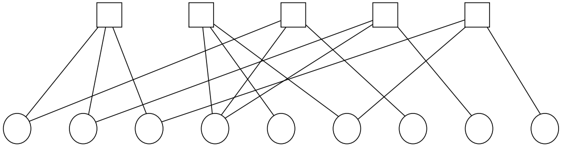
System of equations Parity-check matrix

Tanner Graphs

- The parity-check matrix may be represented by a *Tanner* graph.
- Bipartite graph:
  - Check nodes: Represent the $n-k$ parity-check equations.
  - Variable nodes: Represent the n code bits.
- If $H_{i,j} = 1$, then i^{th} check node is connected to j^{th} variable node.
- Example: For the parity-check matrix:

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

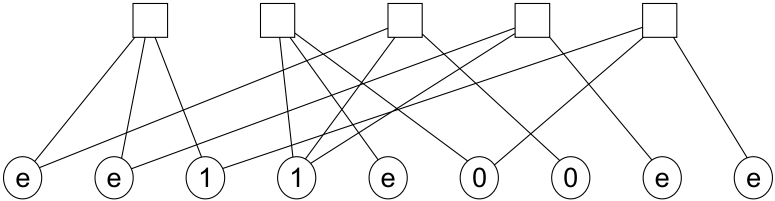
The Tanner Graph is:



Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

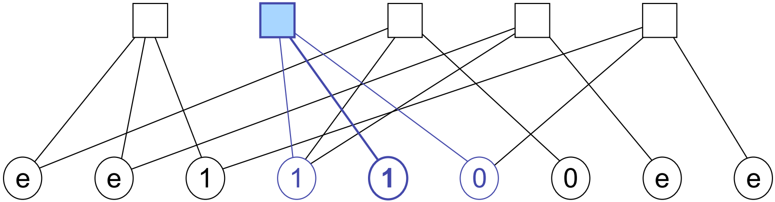
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

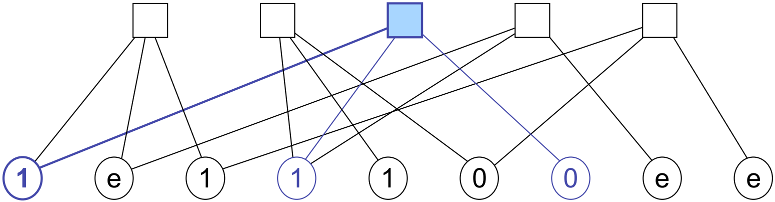
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

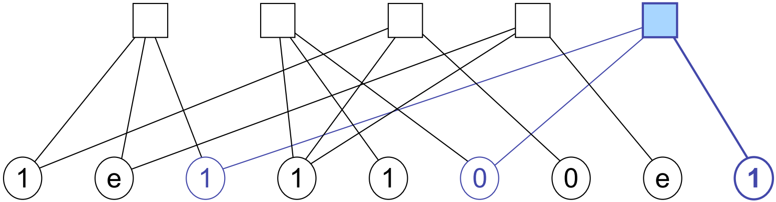
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

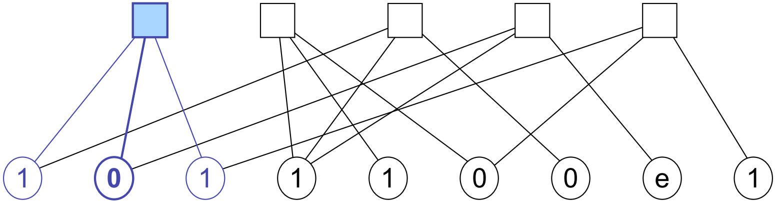
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

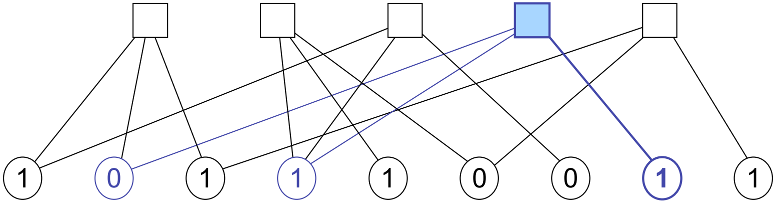
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Decoding on the Tanner Graph

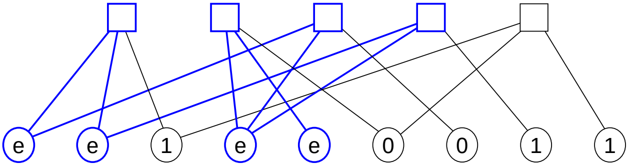
Decoding can be performed on the Tanner graph.

- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Stopping sets

- A *stopping set* $\mathcal{V}$ is a set of erased variable nodes that cannot be corrected, regardless of the state of the other variable nodes.



- Let $\mathcal{G}$ be the neighbors of $\mathcal{V}$.
- Every check node in $\mathcal{G}$ touches at least two variable nodes in $\mathcal{V}$.
- The *minimimum* stopping set $\mathcal{V}_{min}$ is the stopping set containing the fewest variable nodes.
- Let $d_{min} = |\mathcal{V}_{min}|$ be the size of the minimum stopping set.
  - There exists at least one pattern of d_{min} erasures that cannot be corrected.
  - The erasure correcting capability of the code is $d_{min} - 1$, which is the maximum number of erasures that can always be corrected.

Outline

1. Coding and the BEC
2. LDPC Codes
3. Density Evolution
4. Irregular LDPC Codes
5. Conclusion

LDPC Codes

- Observations:
  - The decoder's complexity depends on the degree of the check nodes.
  - The degree of a check node is equal to the Hamming weight of the corresponding row of the parity-check matrix.
  - To achieve capacity, a long code is needed.
  - It is desirable to have a code that is long, yet with small row weight.
- Low-density parity-check codes:
  - An LDPC code is characterized by a *sparse* parity-check matrix.
  - The row/column weights are independent of length.
  - Decoder complexity grows only linear with block length.
- Historical note:
  - LDPC codes were the subject of Robert Gallager's 1960 dissertation.
  - Were forgotten because the decoder could not be implemented.
  - Were "rediscovered" in the mid-1990's after turbo codes were developed.

Example LDPC Code

- A code from MacKay and Neal (1996):

$$
\mathbf{H} = \left[\begin{array}{cccc|cccc|cccc}
1 & & & & & 1 & & 1 & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
 & 1 & & & 1 & & 1 & & 1 & & & \\
 & & 1 & & & 1 & & & & & 1 & 1 \\
 & & 1 & & & & 1 & 1 & & & & 1 \\
 & 1 & & & 1 & & & & 1 & & 1 & \\
1 & & & 1 & & & 1 & & & 1 & & \\
 & 1 & & & & 1 & & 1 & & 1 & & \\
 & & 1 & 1 & & & & & 1 & & & 1
\end{array} \right]
$$

- The code is *regular* because:
  - The rows have constant weight (check-nodes constant degree).
  - The columns have constant weight (variable-nodes constant degree).
- This is called a $(3, 4)$ regular code because the variable nodes have degree 3 and the check nodes have degree 4.

Outline

Density Evolution

- For a (d_v, d_c) regular code, the probability that a variable-node remains erased after the ℓ^{th} iteration is

$$\epsilon_\ell = \epsilon_0 \left(1 - (1 - \epsilon_{\ell-1})^{d_c - 1}\right)^{d_v - 1} \quad (1)$$

  where d_v is the variable-node degree, d_c is the check-node degree, and the initial condition is $\epsilon_0 = \epsilon$.
- The above result assumes independent messages, which is achieved when the girth of the Tanner graph is sufficiently large.
- If $\epsilon_\ell \to 0$ as $\ell \to \infty$ for a particular channel erasure probability ϵ, then a code drawn from the ensemble of all such (d_v, d_c) regular LDPC codes will be able to correctly decode.
- The *threshold* ϵ^* is the maximum ϵ for which $\epsilon_\ell \to 0$ as $\ell \to \infty$.
- For the $(3, 6)$ regular code, the threshold is $\epsilon^* = 0.4294$

Proof of (1), Part I/II

- Decoding involves the exchange of messages between variable nodes and check nodes.
  - Let $p_\uparrow$ denote the probability of an erased message going *up* from the variable nodes to the check nodes.
  - Let $p_\downarrow$ denote the probability of an erased message going *down* from the check nodes to the variable nodes.
- Consider the degree d_c check node.
  - An outgoing message sent over a particular edge is a function of the incoming messages arriving over the other $d_c - 1$ edges.
  - For the outgoing message to be correct, all $d_c - 1$ incoming messages must be correct.
  - The outgoing message will be an erasure if any of the $d_c - 1$ incoming messages is an erasure.
  - The probability of the check node sending an erasure is:

$$p_\downarrow = 1 - (1 - p_\uparrow)^{d_c - 1} \quad (2)$$

Proof of (1), Part II/II

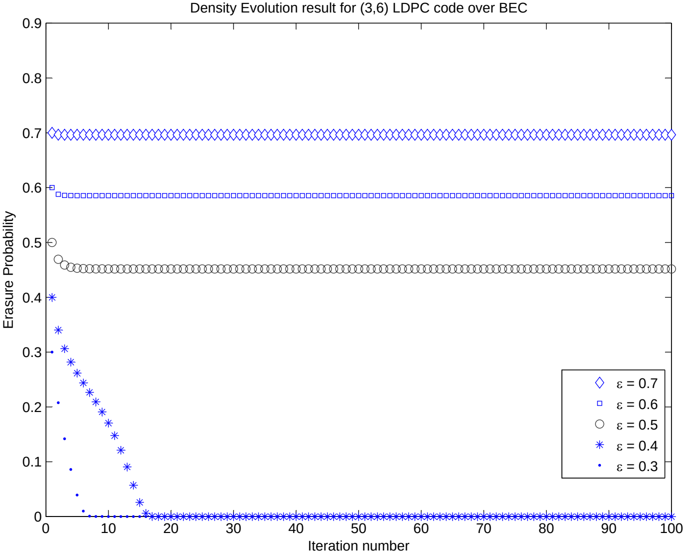
- Consider the degree d_v check node.
  - An outgoing message sent over a particular edge is a function of the incoming messages arriving over the other $d_v - 1$ edges.
  - An outgoing message will be an erasure if the variable node was initially erased *and* all of the arriving messages are erasures.
  - The probability of the variable node sending an erasure is:

$$p_{\uparrow} \quad = \quad \epsilon_0 p_{\downarrow}^{d_v - 1} \tag{3}$$

- Letting ϵ_ℓ equal the value of $p_{\uparrow}$ after the ℓ^{th} iteration, and substituting (2) into (3) yields the recursion given by (1):

$$\epsilon_\ell \quad = \quad \epsilon_0 \left(1 - (1 - \epsilon_{\ell-1})^{d_c - 1} \right)^{d_v - 1}$$

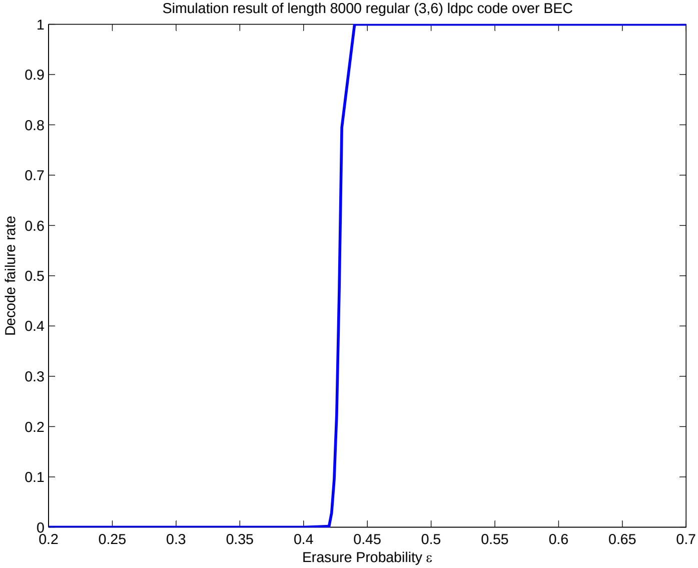
DE Example


Density Evolution result for (3,6) LDPC code over BEC
Erasure Probability
Iteration number
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80
90
100
ε = 0.7
ε = 0.6
ε = 0.5
ε = 0.4
ε = 0.3

Code Realization

- Density evolution only describes the asymptotic performance of the *ensemble* of LDPC codes.
- Implementation requires that an H matrix be generated by drawing from the ensemble of all (d_v, d_c) LDPC codes.
- Goals of good H design:
  - High girth.
  - Full rank.
  - Large minimum stopping set.
- If the girth is too low, the short cycles invalidate the iterative decoder.
- High girth achieved through girth conditioning algorithms such as progressive edge growth (PEG).
- If H is not full rank, then the rate will be reduced according to the number of dependent equations.
- Small stopping sets give rise to an *error floor*.
- A database of good regular LDPC codes can be found on MacKay's website.

Performance of an Actual Code



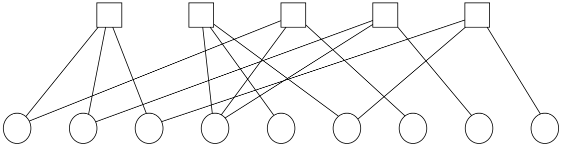
Outline

Irregular LDPC Codes

- Although regular LDPC codes perform well, they are not capable of achieving capacity.
- Properly designed irregular LDPC codes are capable of achieving capacity.
  - The degree distribution of the variable nodes is not constant.
  - The check-node degrees are often still constant (or close to it).
  - Here "designing" means picking the proper degree distribution.

Degree Distribution

- Edge-perspective degree distributions:
  - ρ_i is the fraction of *edges* touching degree i check nodes.
  - λ_i is the fraction of *edges* touching degree i variable nodes.
- For example, consider the Tanner graph:



  - 15 edges.
  - All are connected to degree-3 check nodes, so $\rho_3 = 15/15 = 1$.
  - Four are connected to degree-1 variable nodes, so $\lambda_1 = 4/15$.
  - Eight are connected to degree-2 variable nodes, so $\lambda_2 = 8/15$.
  - Three are connected to the degree-3 variable node, so $\lambda_3 = 3/15$.

DE for Irregular LDPC

- The degree distributions are described in polynomial form:
  - $\rho(x) = \sum_i \rho_i x^{i-1}$ for check nodes.
  - $\lambda(x) = \sum_i \lambda_i x^{i-1}$ for variable nodes.
- For an irregular code, the probability that a variable-node remains erased after the ℓ^{th} iteration is

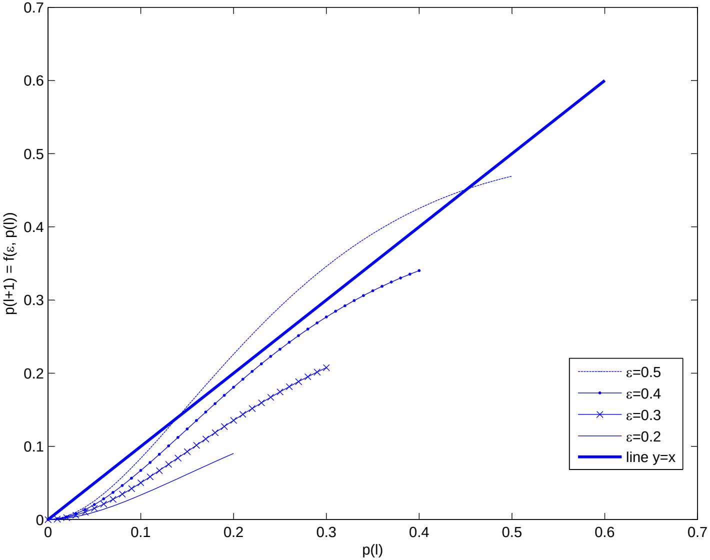
$$\epsilon_\ell = \epsilon_0 \lambda \left(1 - \rho \left(1 - \epsilon_{\ell-1}\right)\right)$$

  The proof follows from the Theorem on Total Probability.
- Convergence:
  - Error-free decoding requires that the erasure probability goes down from one iteration to the next.
  - Define the related function:

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

  - Error-free decoding is possible iff $f(\epsilon, x) \leq x$ for all $0 \leq x \leq \epsilon$.

Convergence



Optimization

- The threshold is

$$\epsilon^* = \sup\{\epsilon : f(\epsilon, x) < x, \forall x, 0 < x \leq \epsilon\}$$

- Solving $f(\epsilon, x) = x$ for ϵ

$$\begin{aligned} x &= f(\epsilon, x) \\ &= \epsilon\lambda\left(1 - \rho\left(1 - x\right)\right) \\ \epsilon &= \frac{x}{\lambda\left(1 - \rho\left(1 - x\right)\right)} \end{aligned}$$

  which is a function of x, and henceforth expressed as $\epsilon(x)$.
- This allows the threshold to be rewritten as:

$$\epsilon^* = \min\{\epsilon(x) : \epsilon(x) \geq x\}$$

Optimization with Linear Programming

- Our goal is to find the degree distribution which yields maximum threshold

$$\max_{\varepsilon^*} \{\varepsilon^* = \min(\varepsilon(x) : \varepsilon(x) \geq x)\};$$

- Several Constraints

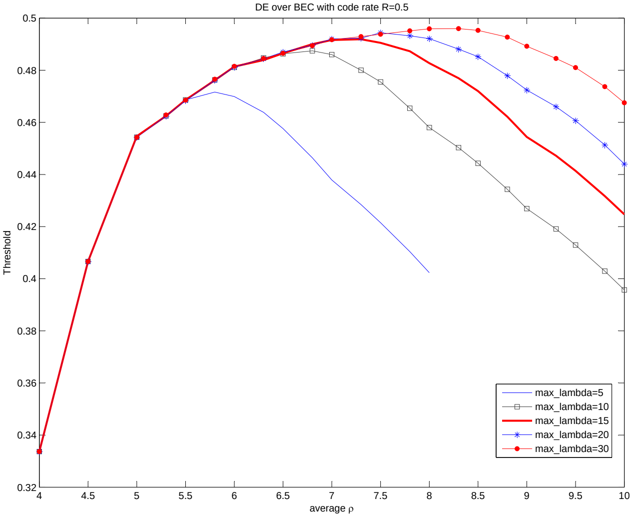
$$\frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} = 1 - R$$

$$\sum_{i \geq 2} \lambda_i = 1; \sum_{i \geq 2} \rho_i = 1;$$

$$x \in [0, 1]$$

- Which can be modeled as a optimization problem using linear programming
  - Can use Matlab's Optimization Toolbox.

Optimization Results ($\epsilon^* = 0.49596$)



Outline

Conclusion

- Conclusions:
  - Irregular LDPC codes can achieve the capacity of the BEC channel.
  - Density evolution predicts asymptotic performance.
  - Key to design is picking the degree distributions.
- Related Issues:
  - Predicting performance of finite-length codes (and designing them).
  - Dealing with unknown ϵ (rateless coding).
  - Dealing with other channels (AWGN, etc.).
- A plug:
  - EE 567: Coding Theory.
  - T/H 5:00-6:15 PM on Evansdale Campus.
  - Will cover linear codes in general and LDPC codes in particular.
  - All you need is graduate-level mathematical maturity and a sense of inquisitiveness.

Thank You.