An Introduction to Low-Density Parity-Check Codes

Paul H. Siegel

Electrical and Computer Engineering

University of California, San Diego

5/ 31/ 07

Center for Wireless Communications

CMRR

Center for Magnetic Recording Research

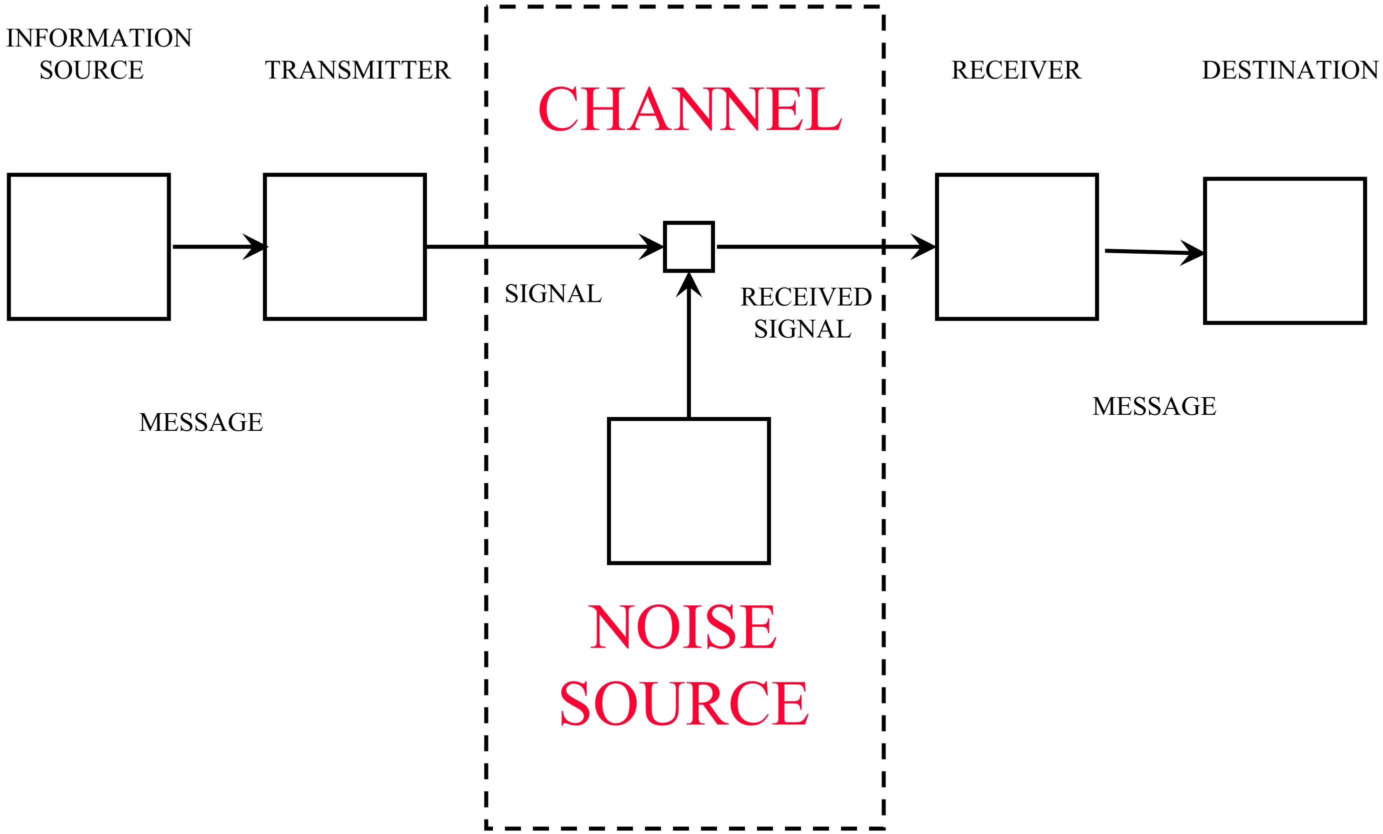
Outline

- Shannon's Channel Coding Theorem
- Error-Correcting Codes – State-of-the-Art
- LDPC Code Basics
  - Encoding
  - Decoding
- LDPC Code Design
  - Asymptotic performance analysis
  - Design optimization

Outline

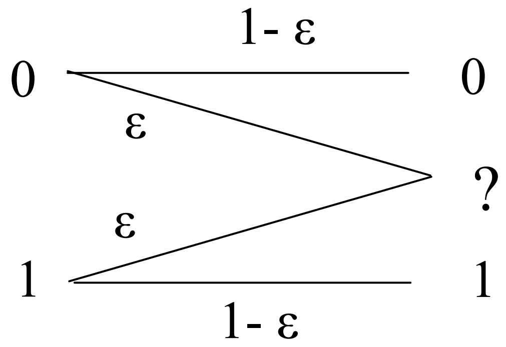
- EXIT Chart Analysis
- Applications
  - Binary Erasure Channel
  - Binary Symmetric Channel
  - AWGN Channel
  - Rayleigh Fading Channel
  - Partial-Response Channel
- Basic References

A Noisy Communication System

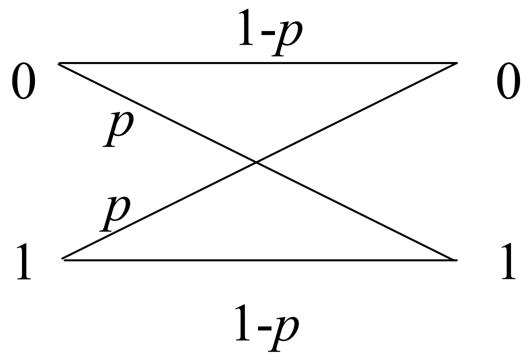


Channels

- Binary erasure channel BEC(ε)

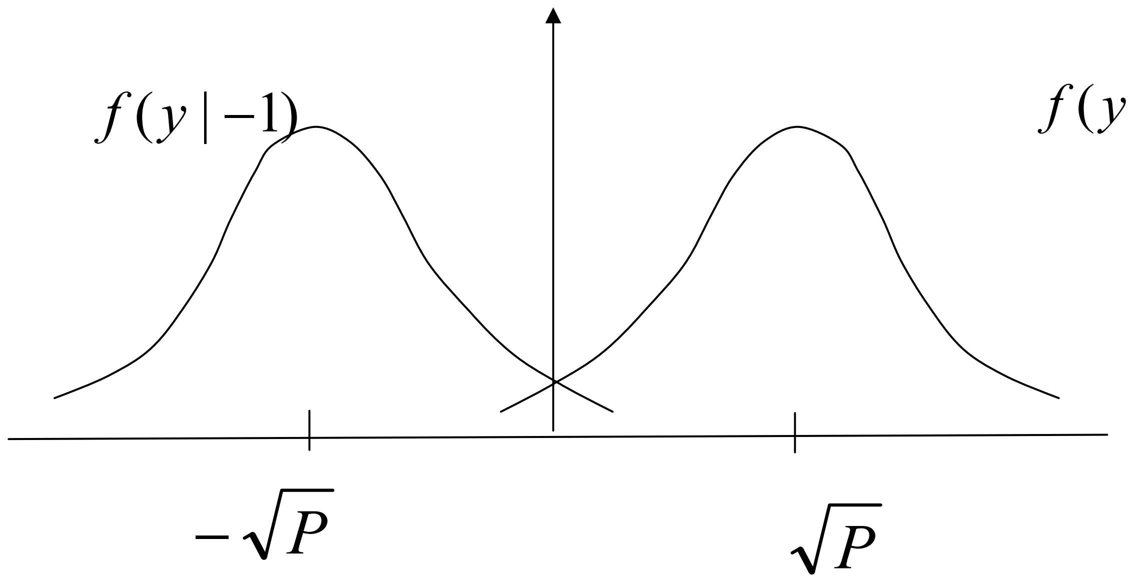


- Binary symmetric channel BSC(p)

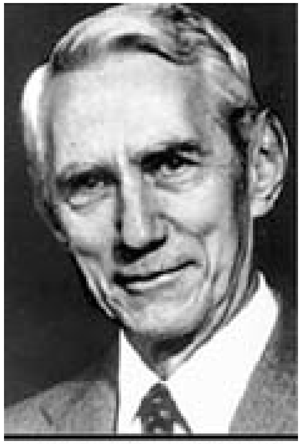


More Channels

- Additive white Gaussian noise channel AWGN



Shannon Capacity



Claude Elwood Shannon
1916 - 2001

Every communication channel is characterized by a single number C, called the **channel capacity.**

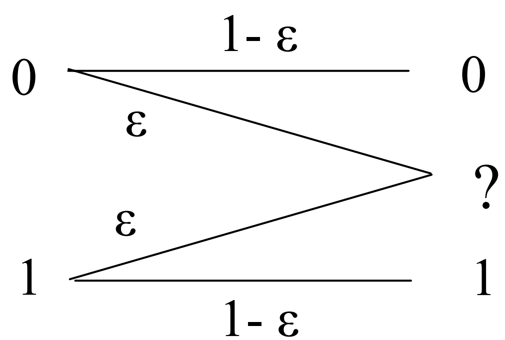
It is possible to transmit information over this channel reliably (with probability of error $\rightarrow 0$) if and only if:

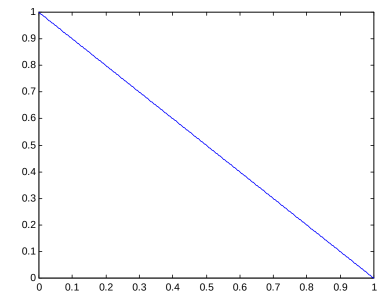
$$R \overset{def}{=} \frac{\#\,\text{information bits}}{\text{channel use}} < C$$

Channels and Capacities

- Binary erasure channel BEC(ε)

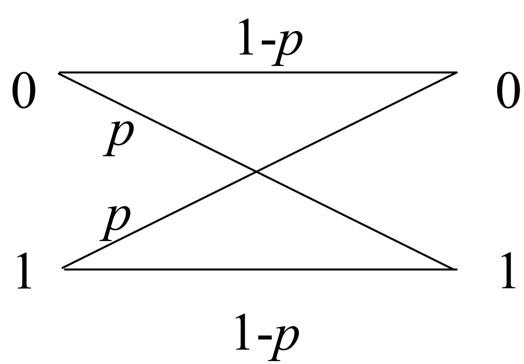
$$C = 1 - \varepsilon$$

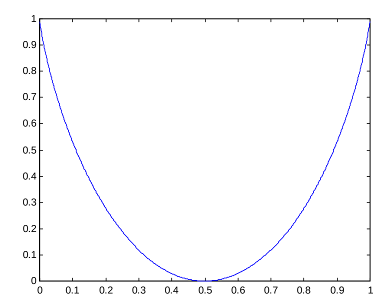




- Binary symmetric channel BSC(p)

$$C = 1 - H_2(p)$$

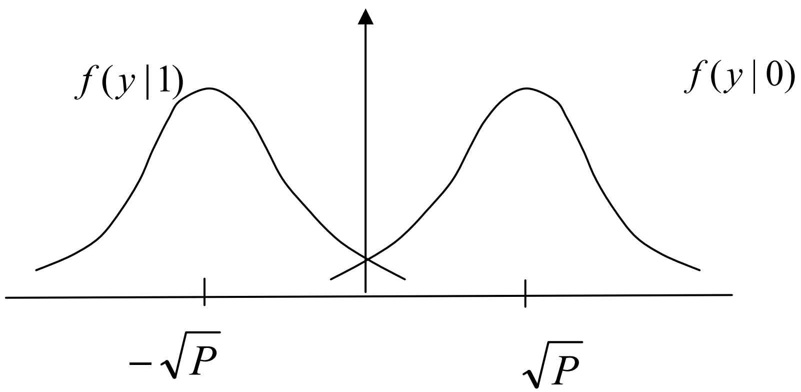




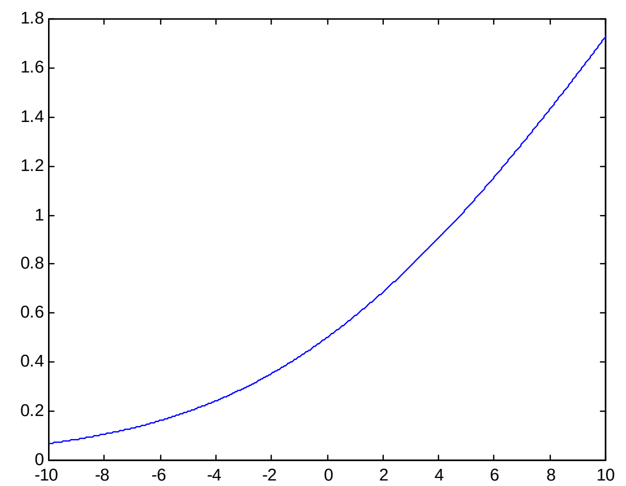
$$H_2(p) = -p\log_2 p - (1-p)\log_2(1-p)$$

More Channels and Capacities

- Additive white Gaussian noise channel AWGN

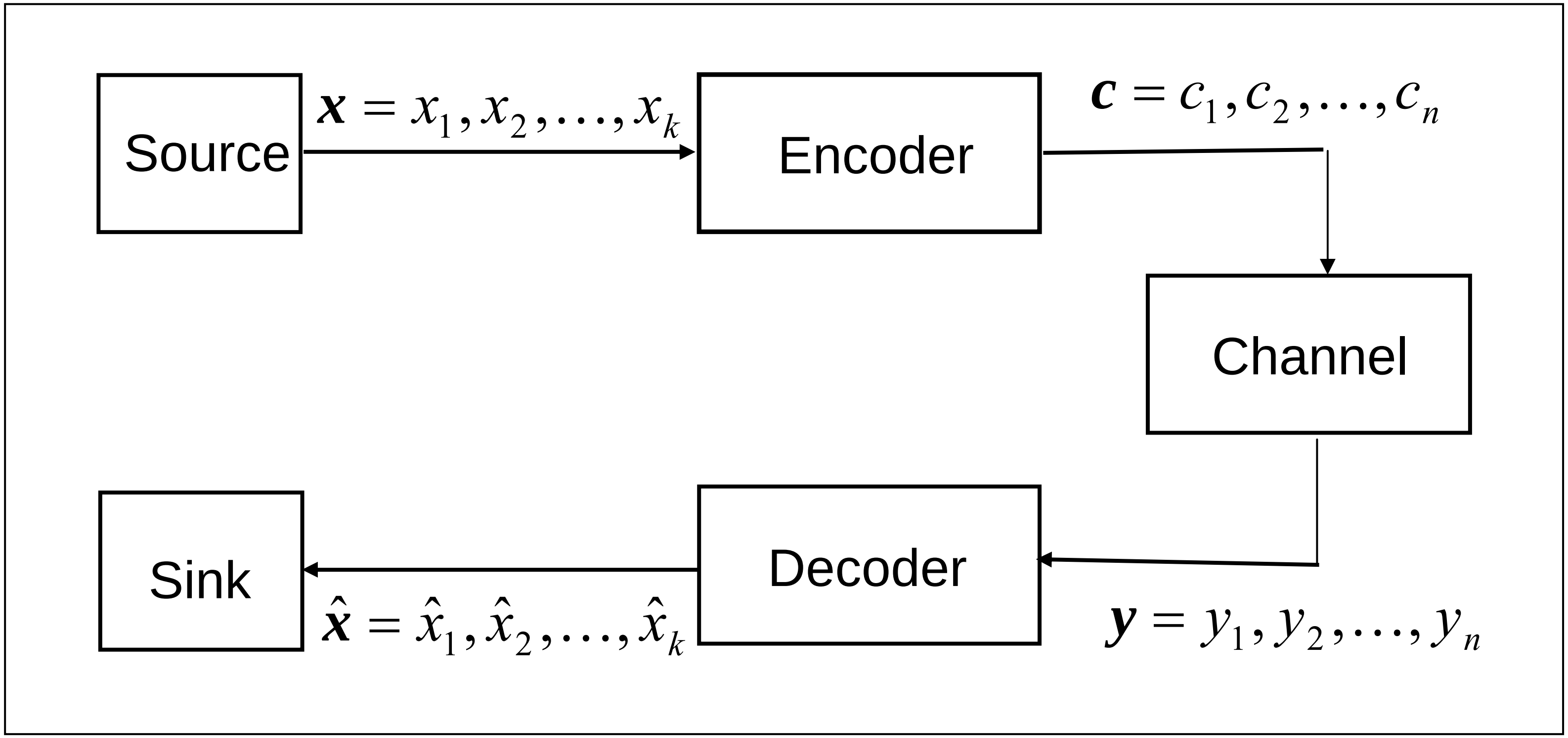


$$C = \tfrac{1}{2}\log_2\left(1 + \frac{P}{\sigma^2}\right)$$



Coding

We use a **code** to communicate over the noisy channel.



Code rate: $R = \dfrac{k}{n}$

Shannon's Coding Theorems

$\uparrow$

$C = \text{Max}\,(H(x) - H_y(x))$

If **C** is a code with rate *R*>*C*, then the probability of error in decoding this code is bounded away from 0. (In other words, at any rate *R*>*C*, reliable communication is **not** possible.)

For any information rate $R < C$ and any $\delta > 0$, there exists a code **C** of length n_δ and rate R, such that the probability of error in maximum likelihood decoding of this code is at most δ.

Proof: **Non-constructive!**

Review of Shannon's Paper

- A pioneering paper:

  Shannon, C. E. "A mathematical theory of communication. *Bell System Tech. J.* **27,** (1948). 379–423, 623–656

- A regrettable review:

  Doob, J.L., Mathematical Reviews, **MR0026286 (10,133e)**

  > "The discussion is suggestive throughout, rather than mathematical, and it is not always clear that the author's mathematical intentions are honorable."

  Cover, T. "Shannon's Contributions to Shannon Theory," AMS Notices, vol. 49, no. 1, p. 11, January 2002

  > "Doob has recanted this remark many times, saying that it and his naming of super martingales (processes that go down instead of up) are his two big regrets."

Finding Good Codes

- Ingredients of Shannon's proof:
  - Random code
  - Large block length
  - Optimal decoding

- Problem

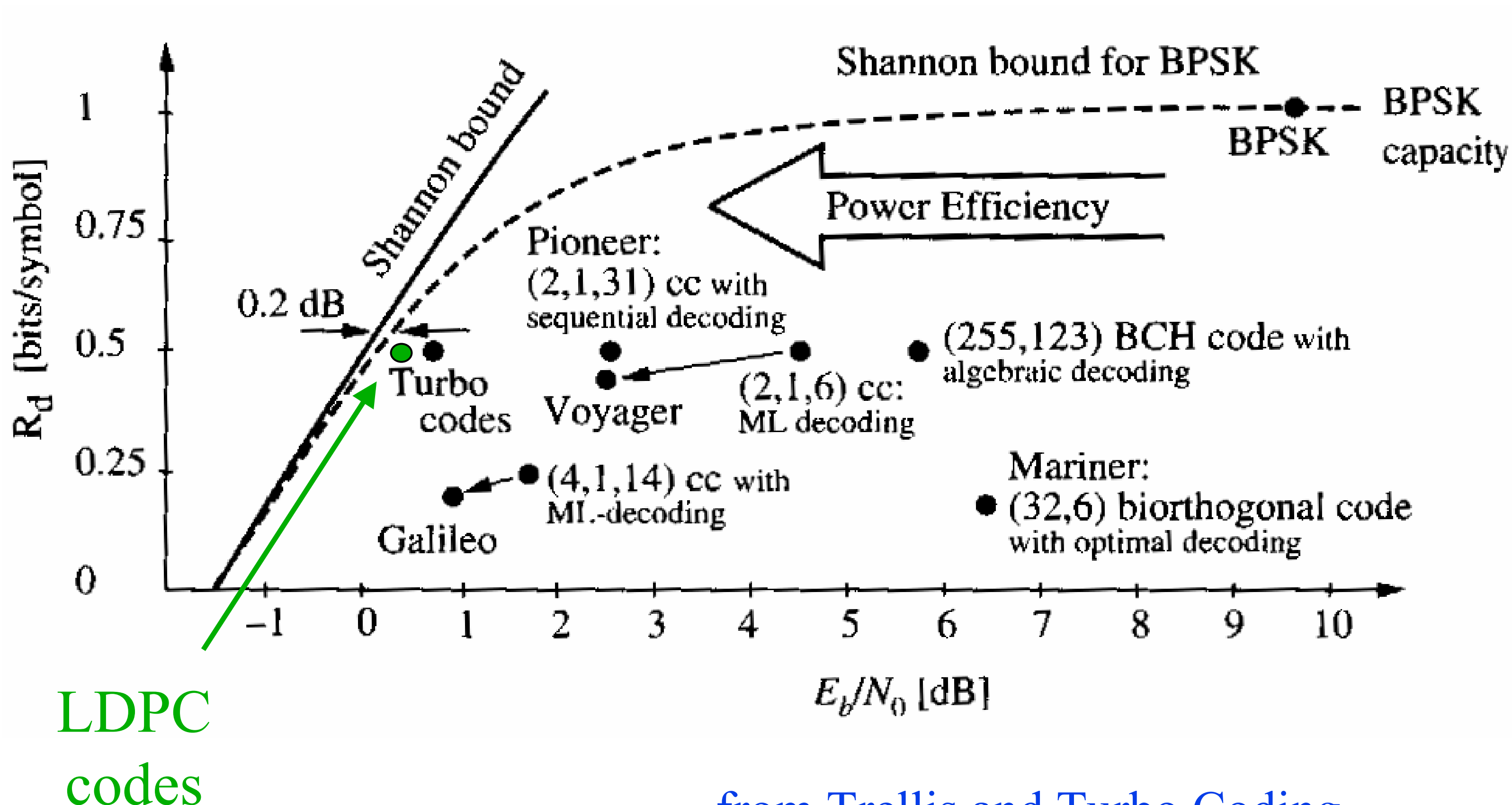
Randomness + large block length + optimal decoding =

COMPLEXITY!

State-of-the-Art

- Solution
  - Long, structured, “pseudorandom” codes
  - Practical, near-optimal decoding algorithms
- Examples
  - Turbo codes (1993)
  - Low-density parity-check (LDPC) codes (1960, 1999)
- State-of-the-art
  - Turbo codes and LDPC codes have brought Shannon limits to within reach on a wide range of channels.

Evolution of Coding Technology



from Trellis and Turbo Coding,
Schlegel and Perez, IEEE Press, 2004

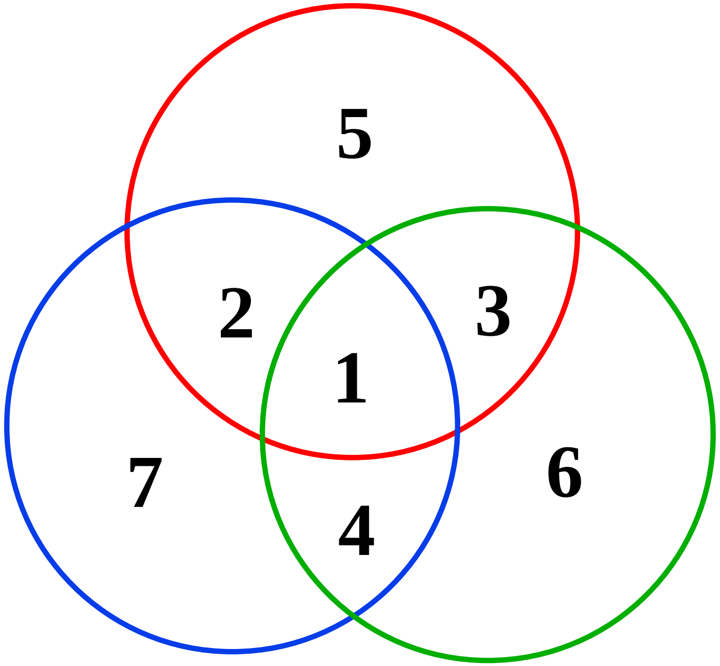
Linear Block Codes - Basics

- Parameters of binary linear block code C
  - k = number of information bits
  - n = number of code bits
  - R = k/n
  - $d_{\min}$ = minimum distance
- There are many ways to describe C
  - Codebook (list)
  - Parity-check matrix / generator matrix
  - Graphical representation ("Tanner graph")

Example: (7,4) Hamming Code

- $(n,k) = (7,4)$, $R = 4/7$
- $d_{\min} = 3$
  - single error correcting
  - double erasure correcting
- Encoding rule:

1. Insert data bits in 1, 2, 3, 4.
2. Insert "parity" bits in 5, 6, 7 to ensure an even number of 1's in each circle

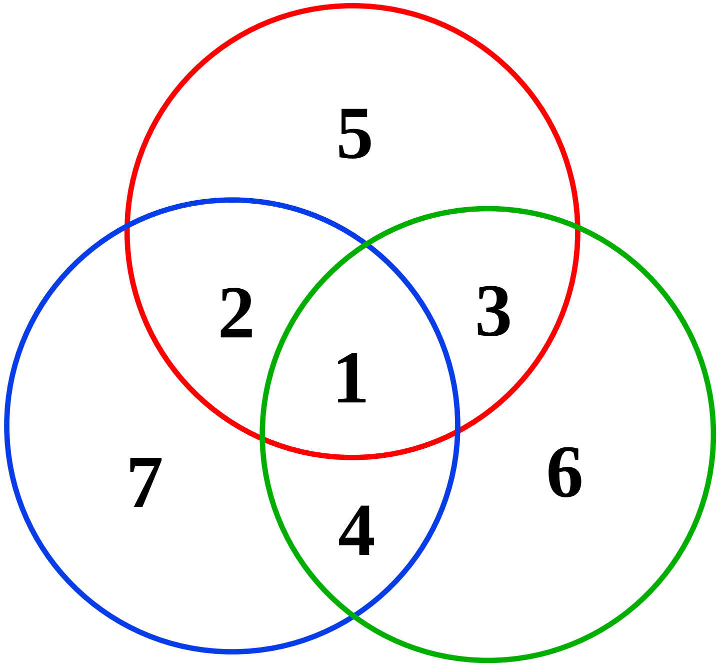


Example: (7,4) Hamming Code

- 2^k=16 codewords
- Systematic encoder places input bits in positions 1, 2, 3, 4
- Parity bits are in positions 5, 6, 7

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

Hamming Code – Parity Checks



$$
\begin{array}{ccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}
$$

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

Hamming Code: Matrix Perspective

- Parity check matrix H

$$\underline{c} = \left[c_1, c_2, c_3, c_4, c_5, c_6, c_7\right]$$

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

$$H\underline{c}^{\mathrm{T}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- Generator matrix G

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

$$\underline{u} = \left[u_1, u_2, u_3 . u_4\right]$$

$$\underline{c} = \left[c_1, c_2, c_3, c_4, c_5, c_6, c_7\right]$$

$$\underline{u} \cdot G = \underline{c}$$

Parity-Check Equations

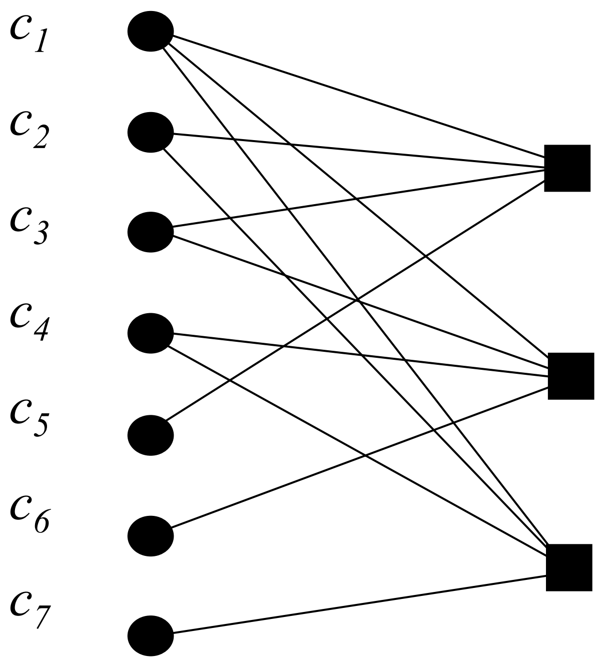
- Parity-check matrix implies system of linear equations.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{array}{ccccccccc} c_1 & + & c_2 & + & c_3 & + & c_5 & = & 0 \\ c_1 & + & c_3 & + & c_4 & + & c_6 & = & 0 \\ c_1 & + & c_2 & + & c_4 & + & c_7 & = & 0 \end{array}$$

- Parity-check matrix is not unique.
- Any set of vectors that span the rowspace generated by H can serve as the rows of a parity check matrix (including sets with more than 3 vectors).

Hamming Code: Tanner Graph

- Bi-partite graph representing parity-check equations

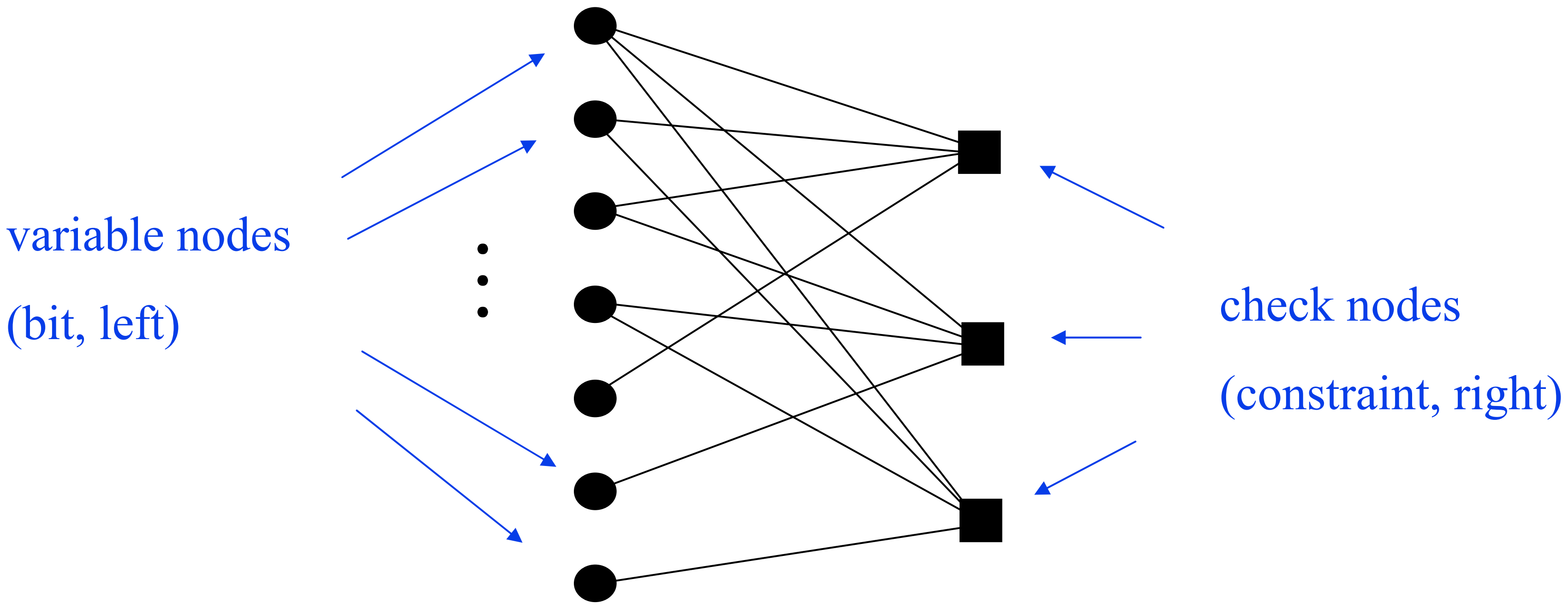


$$c_1 + c_2 + c_3 + c_5 = 0$$

$$c_1 + c_3 + c_4 + c_6 = 0$$

$$c_1 + c_2 + c_4 + c_7 = 0$$

Tanner Graph Terminology



The degree of a node is the number of edges connected to it.

Low-Density Parity-Check Codes

- Proposed by Gallager (1960)
- "Sparseness" of matrix and graph descriptions
  - Number of 1's in H grows linearly with block length
  - Number of edges in Tanner graph grows linearly with block length
- "Randomness" of construction in:
  - Placement of 1's in H
  - Connectivity of variable and check nodes
- Iterative, message-passing decoder
  - Simple "local" decoding at nodes
  - Iterative exchange of information (message-passing)

Review of Gallager's Paper

- Another pioneering work:

  Gallager, R. G., *Low-Density Parity-Check Codes*, M.I.T. Press, Cambridge, Mass: 1963.

- A more enlightened review:

  Horstein, M., IEEE Trans. Inform. Thoery, vol. 10, no. 2, p. 172, April 1964,

  "This book is an extremely lucid and circumspect exposition of an important piece of research. A comparison with other coding and decoding procedures designed for high-reliability transmission ... is difficult...Furthermore, many hours of computer simulation are needed to evaluate a probabilistic decoding scheme... It appears, however, that LDPC codes have a sufficient number of desirable features to make them highly competitive with ... other schemes ...."

Gallager's LDPC Codes

- Now called "regular" LDPC codes
- Parameters (n,j,k)
  - $n =$ codeword length
  - $j =$ # of parity-check equations involving each code bit
    $=$ degree of each variable node
  - $k =$ # code bits involved in each parity-check equation
    $=$ degree of each check node
- Locations of 1's can be chosen randomly, subject to (j,k) constraints.

Gallager's Construction

$(n,j,k) = (20,3,4)$

- First ***n/k*** =5 rows have ***k***=*4* 1's each, descending.
- Next ***j*-1**=2 submatrices of size ***n/k* x *n*** =5 x 20 obtained by applying randomly chosen column permutation to first submatrix.
- Result: ***jn/k*** x ***n*** = 15 x 20 parity check matrix for a **(*n,j,k*)** =(20,3,4) LDPC code.

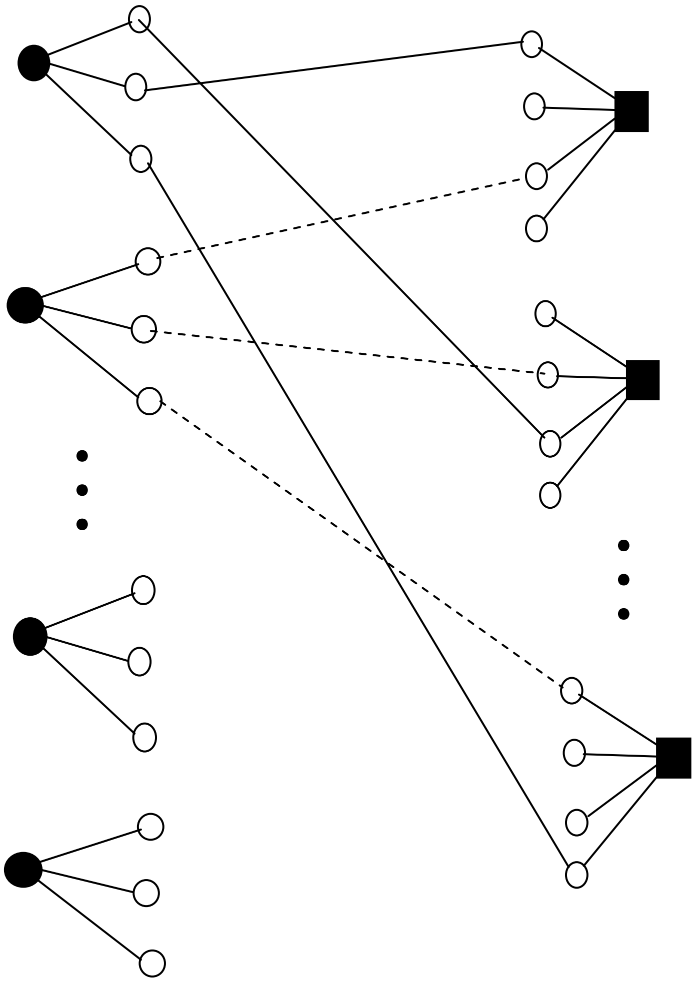
$$
\begin{array}{c|cccccccccccccccccccc|}
\hline
 & 1&1&1&1& & & & & & & & & & & & & & & & \\
 & & & & &1&1&1&1& & & & & & & & & & & & \\
 & & & & & & & & &1&1&1&1& & & & & & & & \\
 & & & & & & & & & & & & &1&1&1&1& & & & \\
 & & & & & & & & & & & & & & & & &1&1&1&1\\
\hdashline
\pi_1 &1& & & &1& & & &1& & & &1& & & & & & & \\
 & &1& & & &1& & & &1& & & & & & &1& & & \\
 & & &1& & & &1& & & & & & &1& & & &1& & \\
 & & & &1& & & & & & &1& & & &1& & & &1& \\
 & & & & & & & &1& & & &1& & & &1& & & &1\\
\hdashline
\pi_2 &1& & & & &1& & & & & &1& & & & & &1& & \\
 & &1& & & & &1& & & &1& & & & &1& & & & \\
 & & &1& & & & &1& & & & &1& & & & & &1& \\
 & & & &1& & & & &1& & & & &1& & &1& & & \\
 & & & & &1& & & & &1& & & & &1& & & &1\\
\hline
\end{array}
$$

Regular LDPC Code – Tanner Graph

n = 20 variable nodes

left degree ***j*** = 3

nj = 60 edges



nj/k = 15 check

right degree **k** = 4

nj = 60 edges

Properties of Regular LDPC Codes

- Design rate: $R(j,k) = 1 - j/k$
  - Linear dependencies can increase rate
  - Design rate achieved with high probability as n increases
  - Example: $(n,j,k)=(20,3,4)$ with $R = 1 - 3/4 = 1/4$.
- For $j \geq 3$, the "typical" minimum distance of codes in the (j,k) ensemble grows **linearly** in the codeword length n.
- Their performance under maximum-likelihood decoding on BSC(p) is "at least as good...as the optimum code of a somewhat higher rate." [Gallager, 1960]

Performance of Regular LDPC Codes

Gallager, 1963

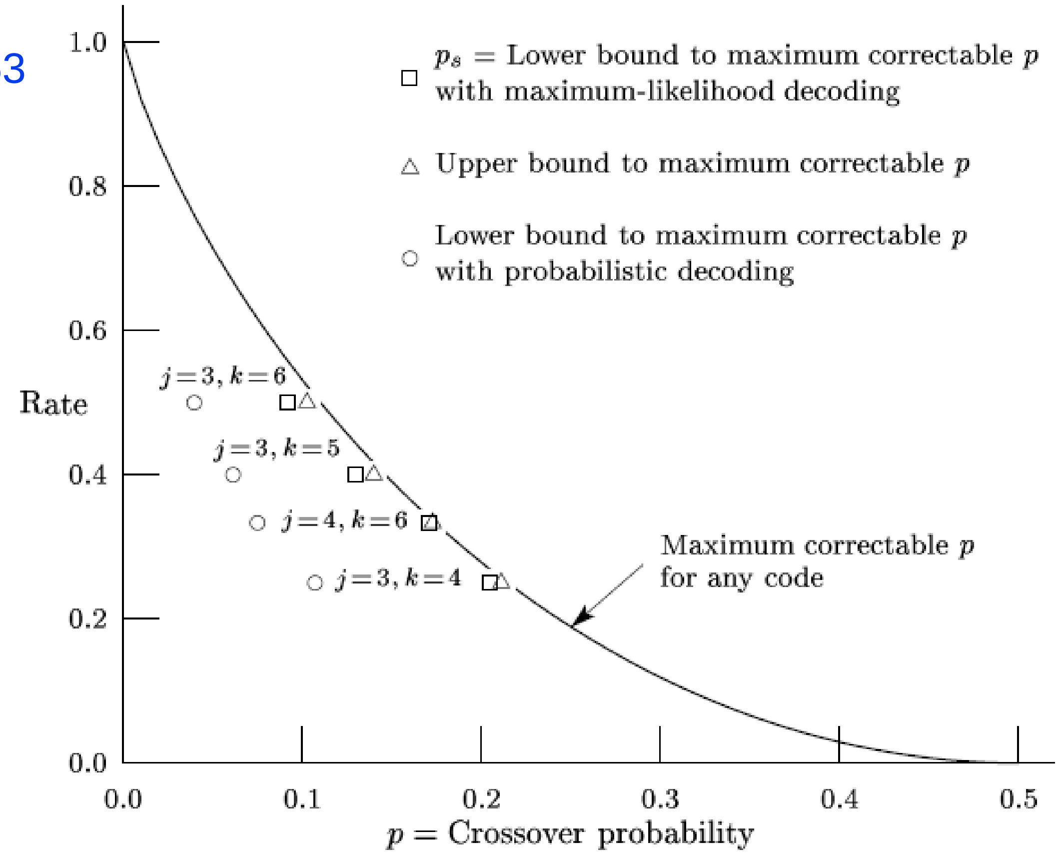


Figure 3.5: Error-correcting properties of (n, j, k) codes on BSC as function of rate for large n.

Performance of Regular LDPC Codes

Gallager, 1963

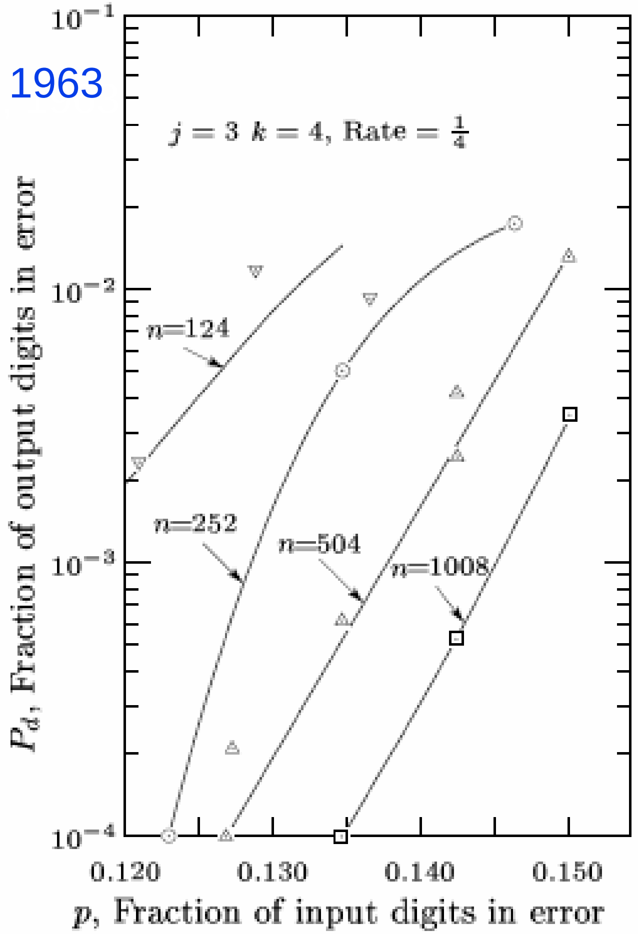


Figure 6.1: Experimental results on BSC.

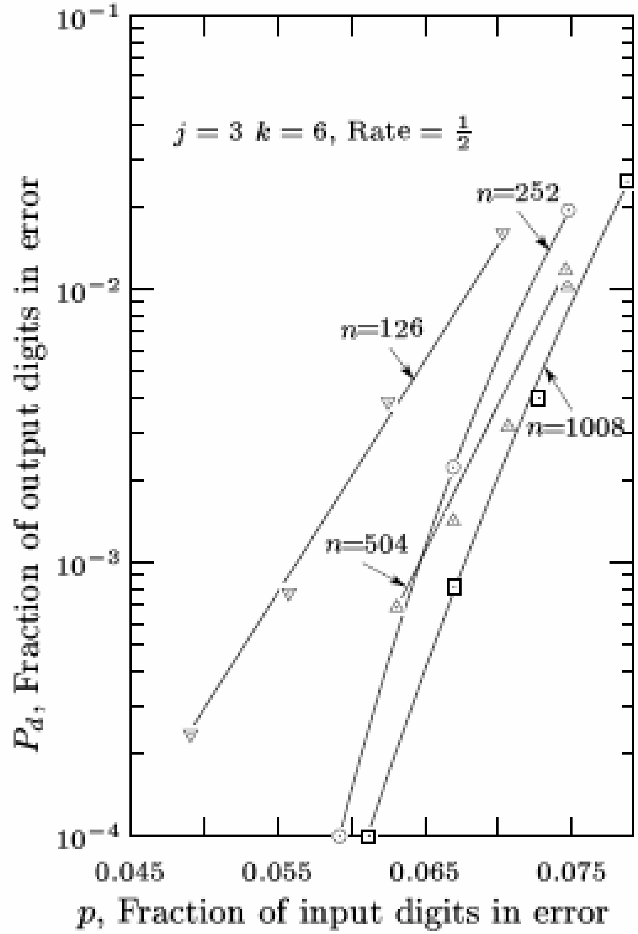


Figure 6.2: Experimental results on BSC.

Performance of Regular LDPC Codes

Gallager, 1963

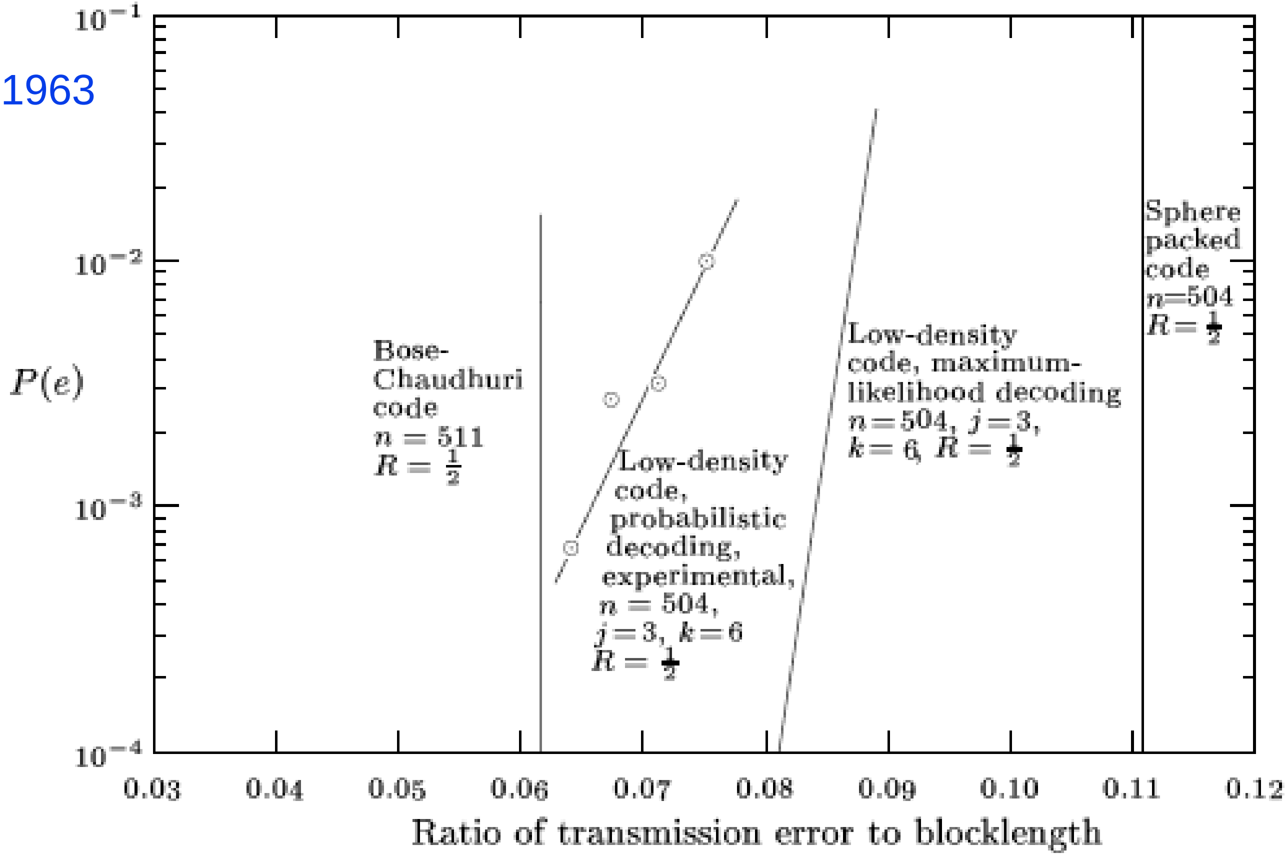
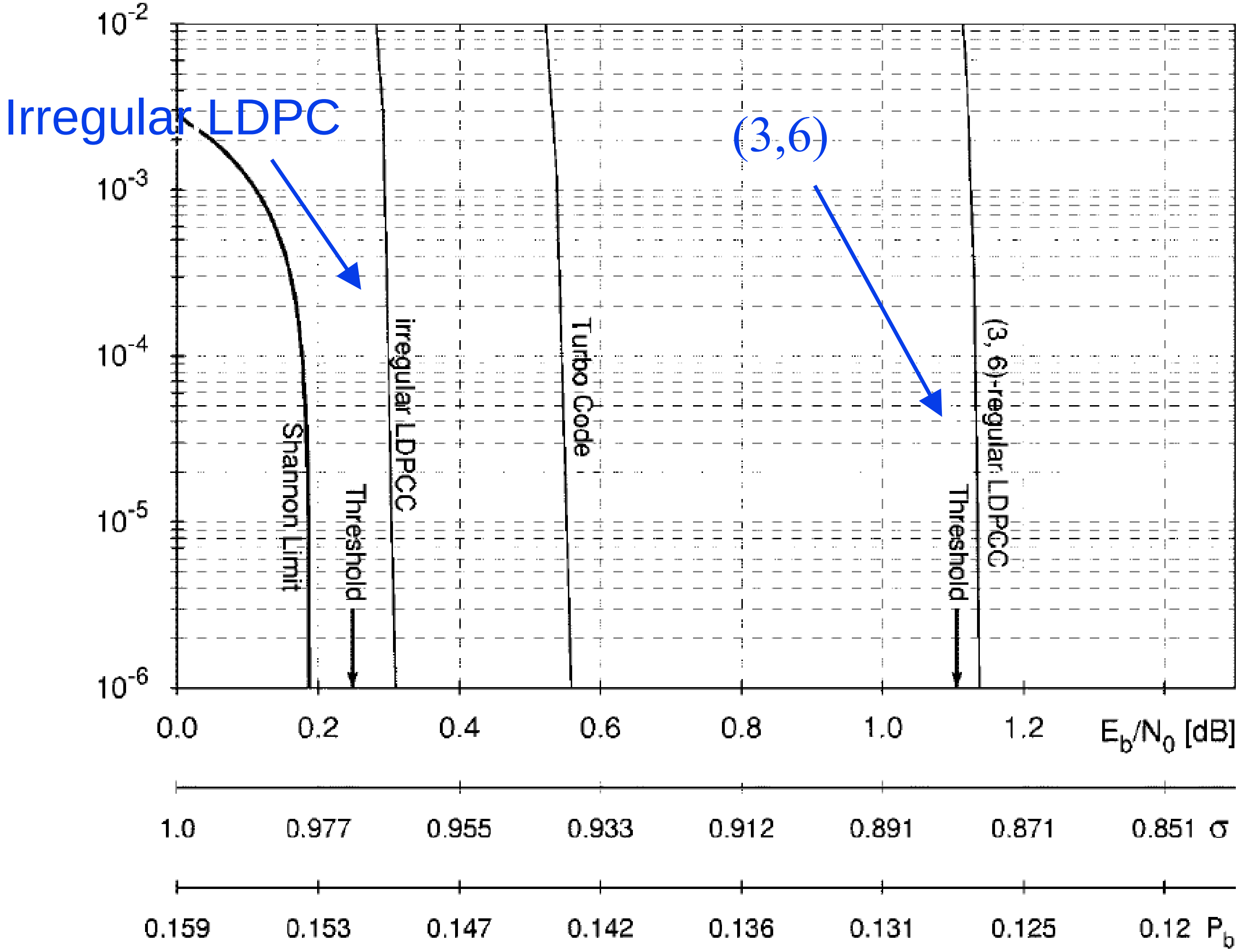


Figure 6.5: Comparison of experimental results using probabilistic decoding to theoretical results with maximum-likelihood decoding.

Performance of Regular LDPC Codes



Richardson, Shokrollahi, and Urbanke, 2001

$n=10^6$

$R=1/2$

Irregular LDPC Codes

- Irregular LDPC codes are a natural generalization of Gallager's LDPC codes.
- The degrees of variable and check nodes need not be constant.
- Ensemble defined by "node degree distribution" functions.

$$\Lambda(x) = \sum_{i=1}^{d_v} \Lambda_i x^i \qquad \mathrm{P}(x) = \sum_{i=2}^{d_c} \mathrm{P}_i x^i$$

Λ_i = number of variable nodes of degree i

P_i = number of check nodes of degree i

- Normalize for fraction of nodes of specified degree

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

Irregular LDPC Codes

- Often, we use the degree distribution from the edge perspective

$$\lambda(x) = \sum_{i=1}^{d_v} \lambda_i x^{i-1} \qquad \rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1}$$

λ_i = fraction of edges connected to variable nodes of degree i

ρ_i = fraction of edges connected to check nodes of degree i

- Conversion rule

$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)} = \frac{L'(x)}{L'(1)} \qquad \rho(x) = \frac{P'(x)}{P'(1)} = \frac{R'(x)}{R'(1)}$$

Irregular LDPC Codes

- Design rate

$$R(\lambda,\rho) = 1 - \frac{\sum_i \frac{\rho_i}{i}}{\sum_i \frac{\lambda_i}{i}} = 1 - \frac{\int_0^1 \rho(x)\,dx}{\int_0^1 \lambda(x)\,dx}$$

- Under certain conditions related to codewords of weight $\approx n/2$, the design rate is achieved with high probability as n increases.

Examples of Degree Distribution Pairs

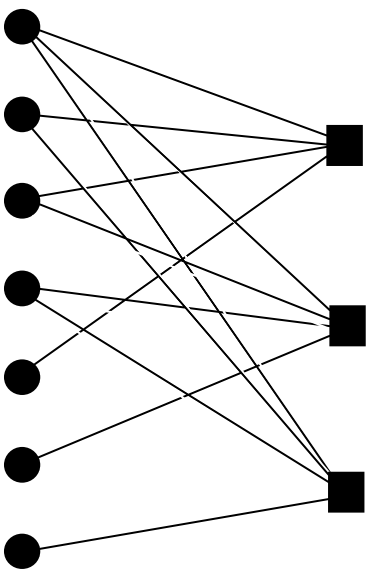
- Hamming (7,4) code

$$\Lambda(x) = 3x + 3x^2 + x^3$$
$$\mathrm{P}(x) = 3x^4$$

edges = 12

$$\lambda(x) = \tfrac{1}{4} + \tfrac{1}{2}x + \tfrac{1}{4}x^2$$
$$\rho(x) = x^3$$

$$R(\lambda, \rho) = 1 - \frac{3}{7} = \frac{4}{7}$$



- (j,k) – regular LDPC code, length-n

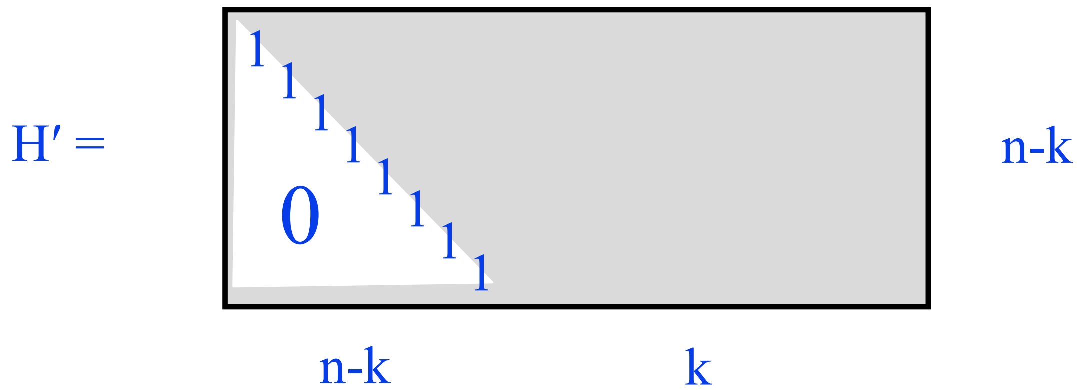
$$\Lambda(x) = nx^j$$
$$\mathrm{P}(x) = \frac{jn}{k}x^k$$

$$\lambda(x) = x^{j-1}$$
$$\rho(x) = x^{k-1}$$

$$R(\lambda, \rho) = 1 - \frac{1/k}{1/j} = 1 - \frac{j}{k}$$

Encoding LDPC Codes

- Convert H into equivalent upper triangular form H′



(e.g., by Gaussian elimination and column swapping – complexity ~ O(n^3))

- This is a “pre-processing” step only.

Encoding LDPC Codes

- Set $c_{n\text{-}k+1},\ldots,c_n$ equal to the data bits $x_1,\ldots,x_k$.
- Solve for parities c_ℓ, ℓ=1,…, n-k, in reverse order; i.e., starting with ℓ=n-k, compute

$$c_l = -\sum_{j=l+1}^{n-k} H_{l,j}c_j - \sum_{j=l+1}^{k} H_{l,j-n+k}x_j$$

(complexity ~O(n^2))

- Another general encoding technique based upon "approximate lower triangulation" has complexity no more than O(n^2), with the constant coefficient small enough to allow practical encoding for block lengths on the order of n=10^5.

Linear Encoding Complexity

- It has been shown that "optimized" ensembles of irregular LDPC codes can be encoded with preprocessing complexity at most O($n^{3/2}$), and subsequent complexity ~O(n).
- It has been shown that a necessary condition for the ensemble of (λ, ρ)-irregular LDPC codes to be linear-time encodable is

$$\lambda'(0)\rho'(1) > 1$$

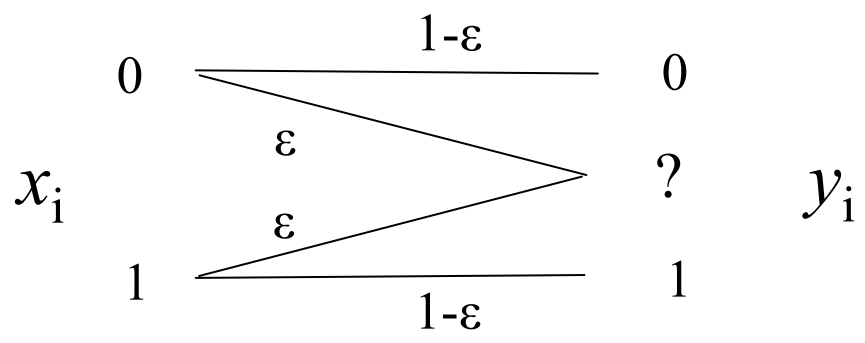
- Alternatively, LDPC code ensembles with additional "structure" have linear encoding complexity, such as "irregular repeat-accumulate (IRA)" codes.

Decoding of LDPC Codes

- Gallager introduced the idea of iterative, message-passing decoding of LDPC codes.
- The idea is to iteratively share the results of local node decoding by passing them along edges of the Tanner graph.
- We will first demonstrate this decoding method for the binary erasure channel BEC(ε).
- The performance and optimization of LDPC codes for the BEC will tell us a lot about other channels, too.

Decoding for the BEC

- Recall: Binary erasure channel, BEC(ε)



$$x = (x_1, x_2, \dots , x_n) \qquad \text{transmitted codeword}$$
$$y = (y_1, y_2, \dots , y_n) \qquad \text{received word}$$

- **Note:** if $y_i \in \{0,1\}$, then $x_i = y_i$.

Optimal Block Decoding - BEC

- Maximum *a posteriori* (MAP) **block** decoding rule minimizes **block** error probability:

$$\hat{x}^{MAP}(y) = \arg\max_{x \in C} P_{X|Y}(x \mid y)$$

- Assume that codewords are transmitted equiprobably.

$$\hat{x}^{MAP}(y) = \arg\max_{x \in C} P_{Y|X}(y \mid x)$$

- If the (non-empty) set X(y) of codewords *compatible* with y contains only one codeword x, then

$$\hat{x}^{MAP}(y) = x$$

- If X(y) contains more than one codeword, then declare a block erasure.

Optimal Bit Decoding - BEC

- Maximum *a posteriori* (MAP) **bit** decoding rule minimizes **bit** error probability:

$$\hat{x}_i^{MAP}(y) = \arg\max_{b \in \{0,1\}} P_{X_i|Y}(b \mid y)$$

$$= \arg\max_{b \in \{0,1\}} \sum_{\substack{x \in C \\ x_i = b}} P_{X|Y}(x \mid y)$$

- Assume that codewords are transmitted equiprobably.
- If **every** codeword $x \in X(y)$ satisfies $x_i = b$, then set

$$\hat{x}^{MAP}(y) = b$$

- Otherwise, declare a bit erasure in position i.

MAP Decoding Complexity

- Let $E \subseteq \{1,\ldots,n\}$ denote the positions of erasures in y, and let F denote its complement in $\{1,\ldots,n\}$.
- Let w_E and w_F denote the corresponding sub-words of word w.
- Let H_E and H_F denote the corresponding submatrices of the parity check matrix H.
- Then X(y), the set of codewords compatible with y, satisfies

$$\mathrm{X}(y) = \left\{\, x \in C \mid x_F = y_F \text{ and } H_E x_E^{\mathrm{T}} = H_F y_F^{\mathrm{T}} \,\right\}$$

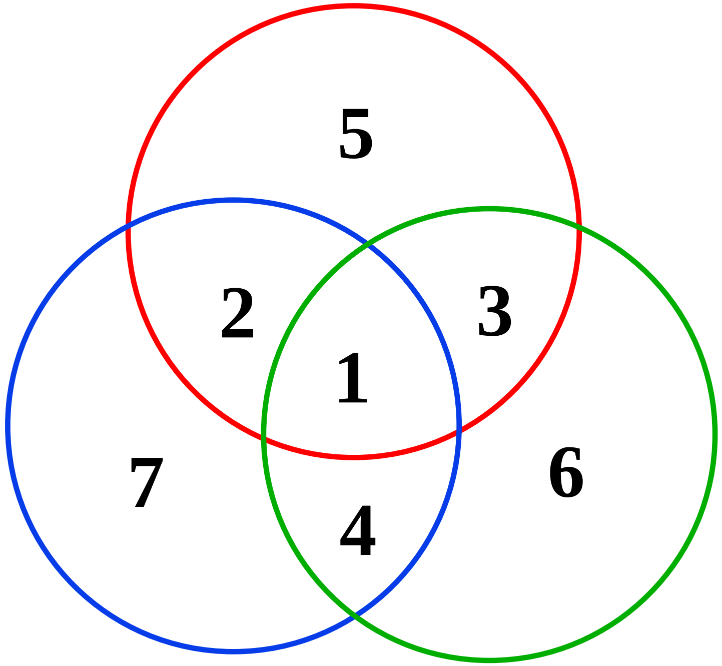
- So, optimal (MAP) decoding can be done by solving a set of linear equations, requiring complexity at most O(n^3).
- For large blocklength n, this can be prohibitive!

Simpler Decoding

- We now describe an alternative decoding procedure that can be implemented very simply.
- It is a "local" decoding technique that tries to fill in erasures "one parity-check equation at a time."
- We will illustrate it using a very simple and familiar linear code, the (7,4) Hamming code.
- We'll compare its performance to that of optimal bit-wise decoding.
- Then, we'll reformulate it as a "message-passing" decoding algorithm and apply it to LDPC codes.

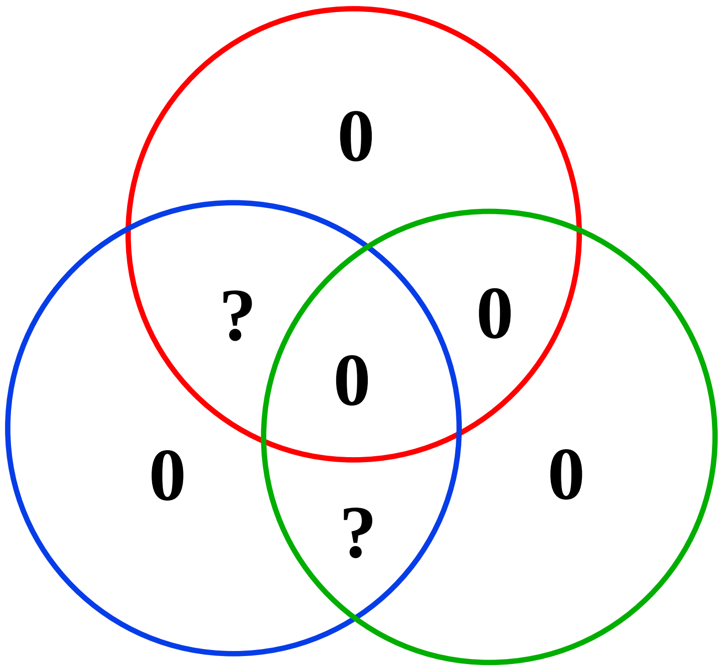
Local Decoding of Erasures

- $d_{min} = 3$, so any two erasures can be uniquely filled to get a codeword.
- Decoding can be done *locally*: Given any pattern of one or two erasures, there will always be a parity-check (circle) involving exactly one erasure.
- The parity-check represented by the circle can be used to fill in the erased bit.
- This leaves at most one more erasure. Any parity-check (circle) involving it can be used to fill it in.



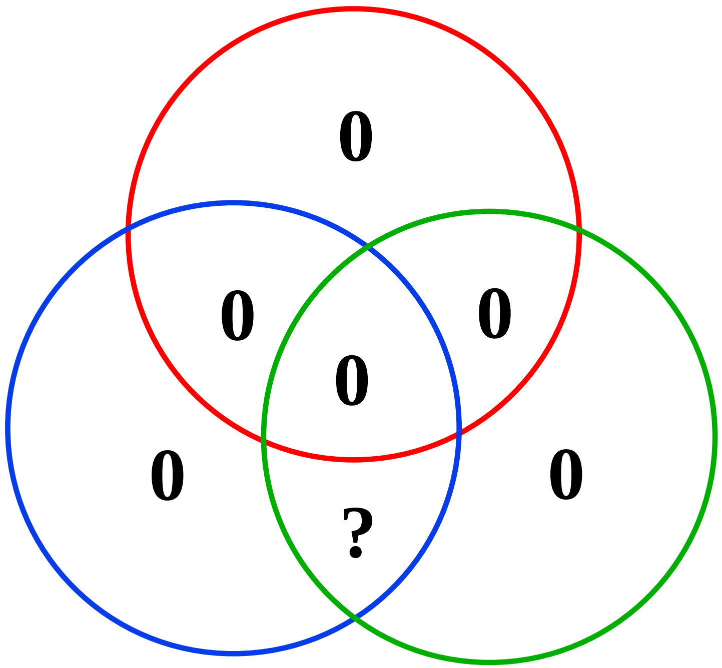
Local Decoding - Example

- All-0's codeword transmitted.
- Two erasures as shown.
- Start with either the red parity or green parity circle.
- The red parity circle requires that the erased symbol inside it be 0.



Local Decoding -Example

- Next, the green parity circle or the blue parity circle can be selected.
- Either one requires that the remaining erased symbol be 0.

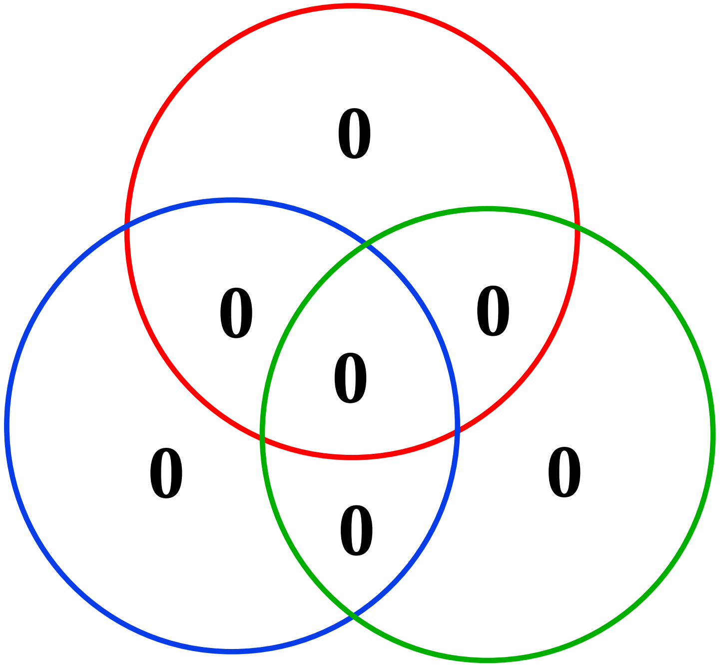


Local Decoding -Example

- Estimated codeword:

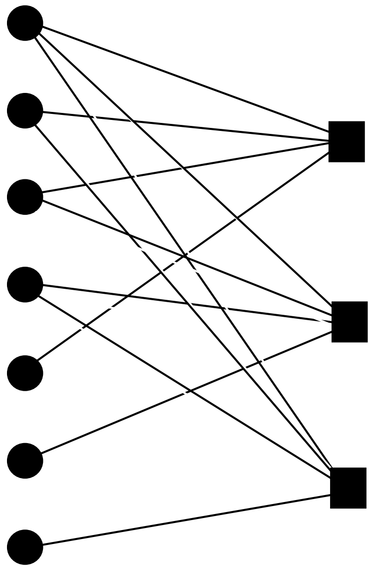
  [0 0 0 0 0 0 0]

- Decoding successful!!
- This procedure would have worked no matter which codeword was transmitted.

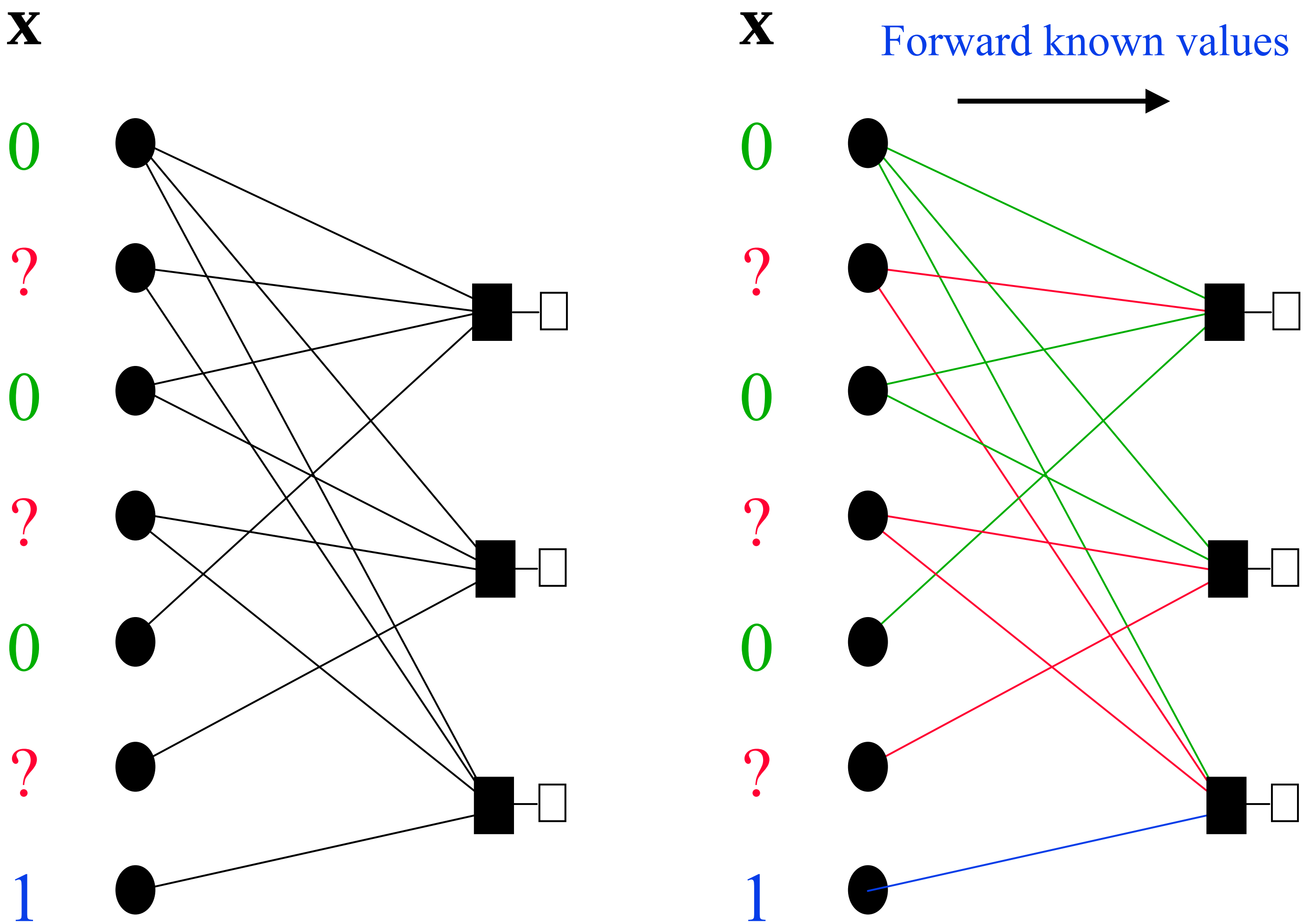


Decoding with the Tanner Graph: an a-Peeling Decoder

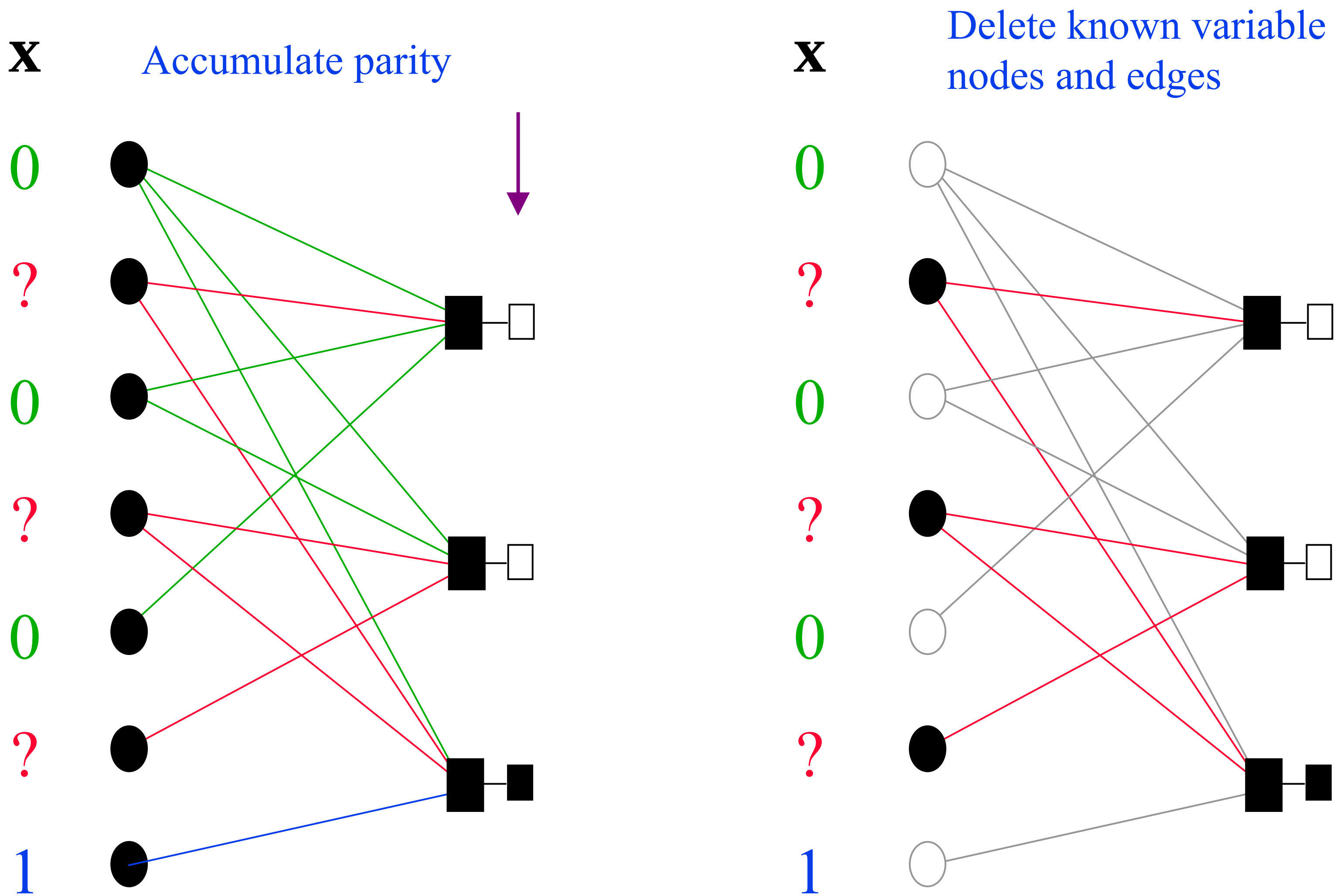
- Initialization:
  - Forward known variable node values along outgoing edges
  - Accumulate forwarded values at check nodes and “record” the parity
  - Delete known variable nodes and all outgoing edges



Peeling Decoder – Initialization



Peeling Decoder - Initialization



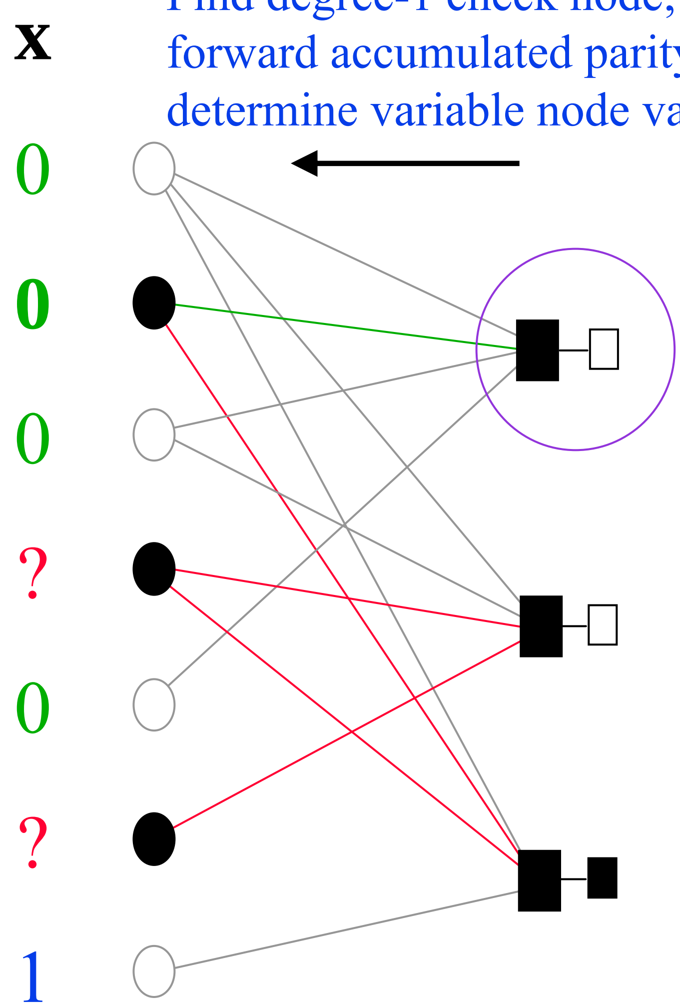
Decoding with the Tanner Graph: an a-Peeling Decoder

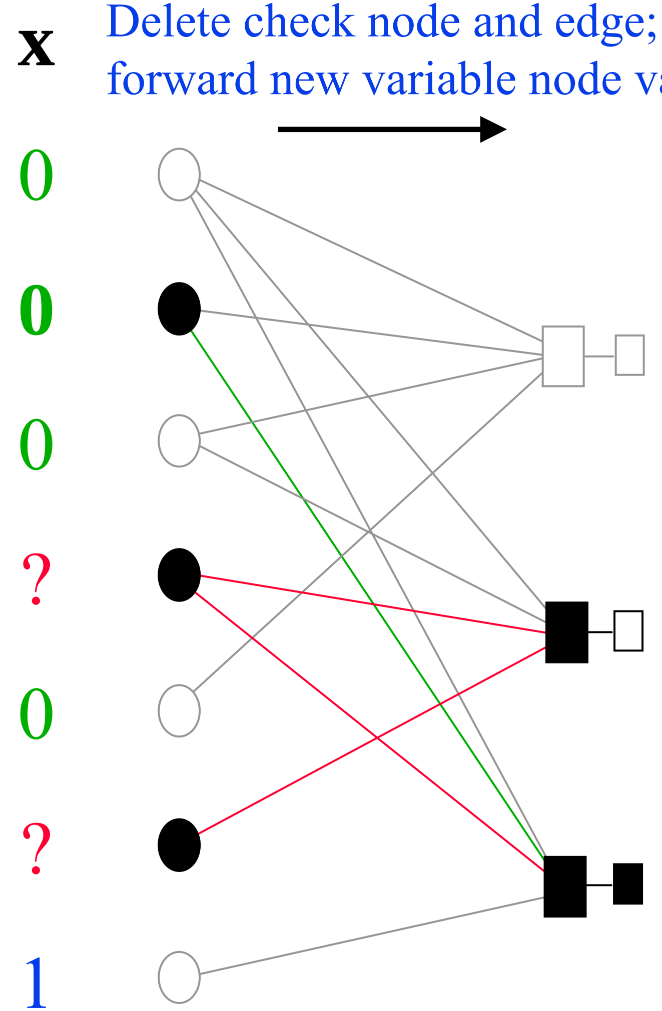
- Decoding step:
  - Select, if possible, a check node with one edge remaining; forward its parity, thereby determining the connected variable node
  - Delete the check node and its outgoing edge
  - Follow procedure in the initialization process at the known variable node
- Termination
  - If remaining graph is empty, the codeword is determined
  - If decoding step gets stuck, declare decoding failure

Peeling Decoder – Step 1

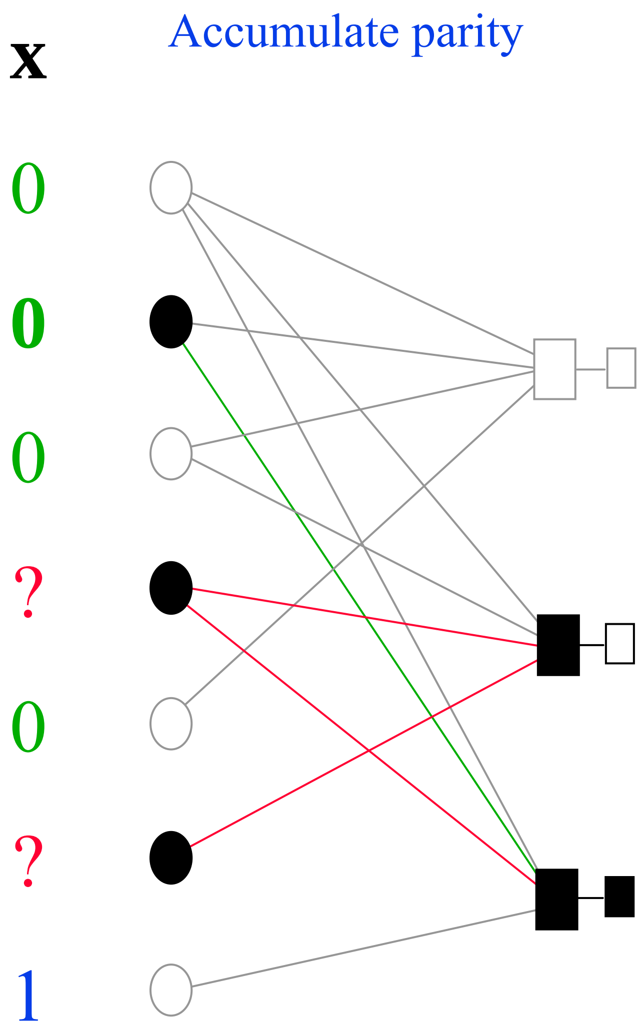
Find degree-1 check node; forward accumulated parity; determine variable node value

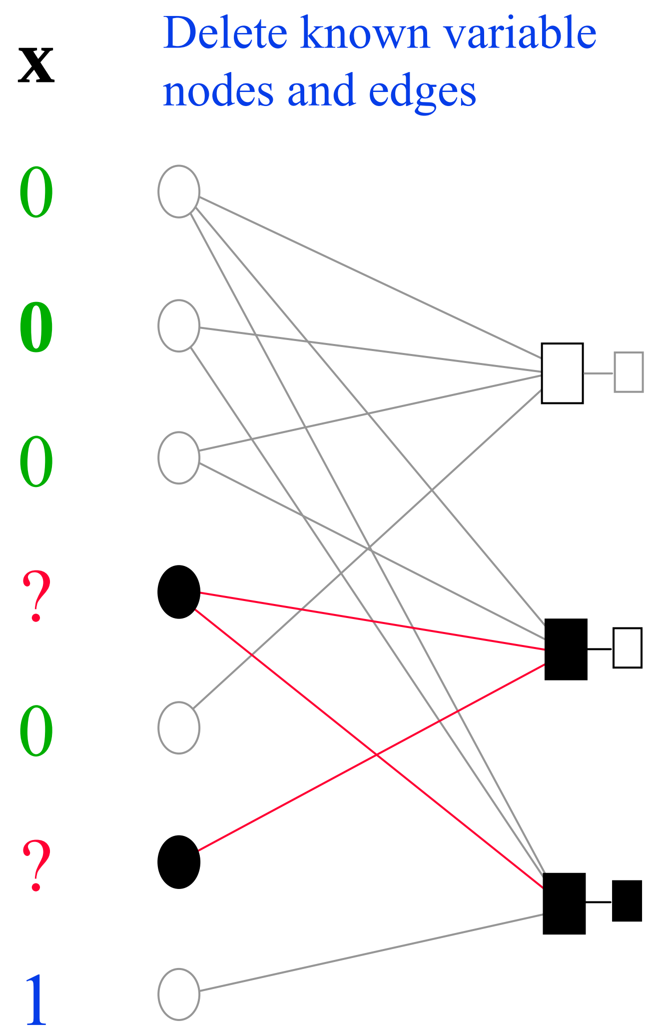
Delete check node and edge; forward new variable node value





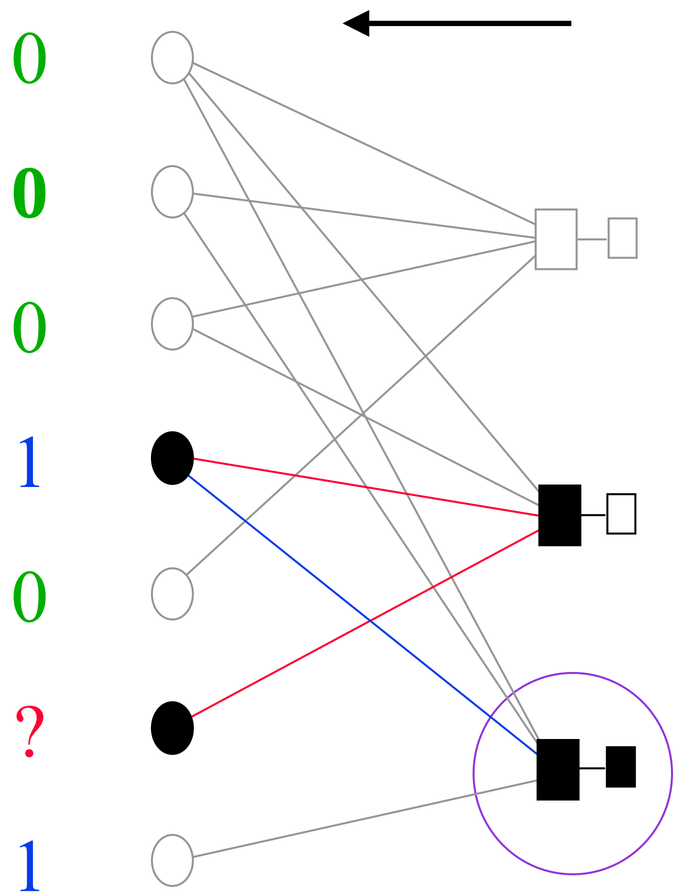
Peeling Decoder – Step 1



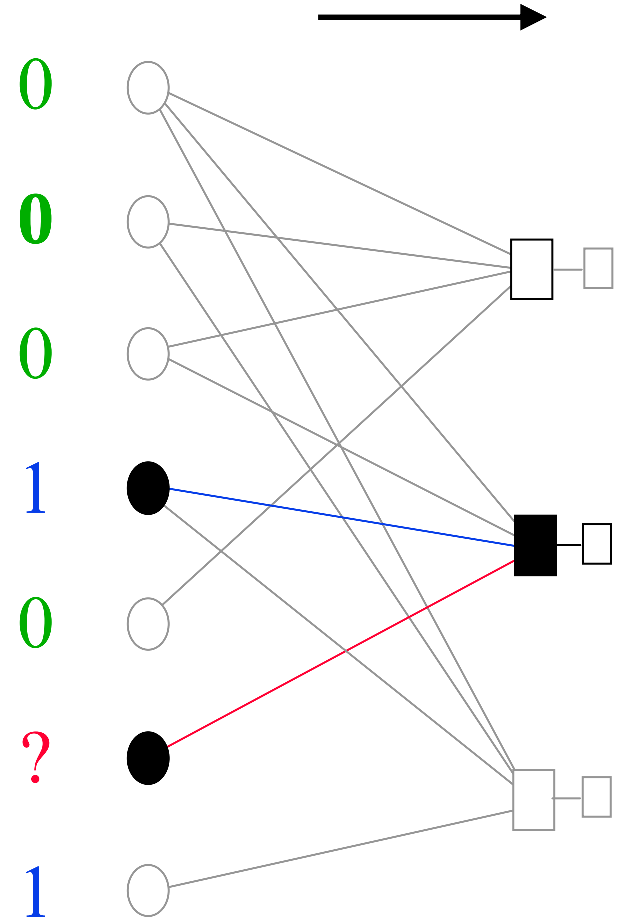


Peeling Decoder – Step 2

Find degree-1 check node;
forward accumulated parity;
determine variable node value

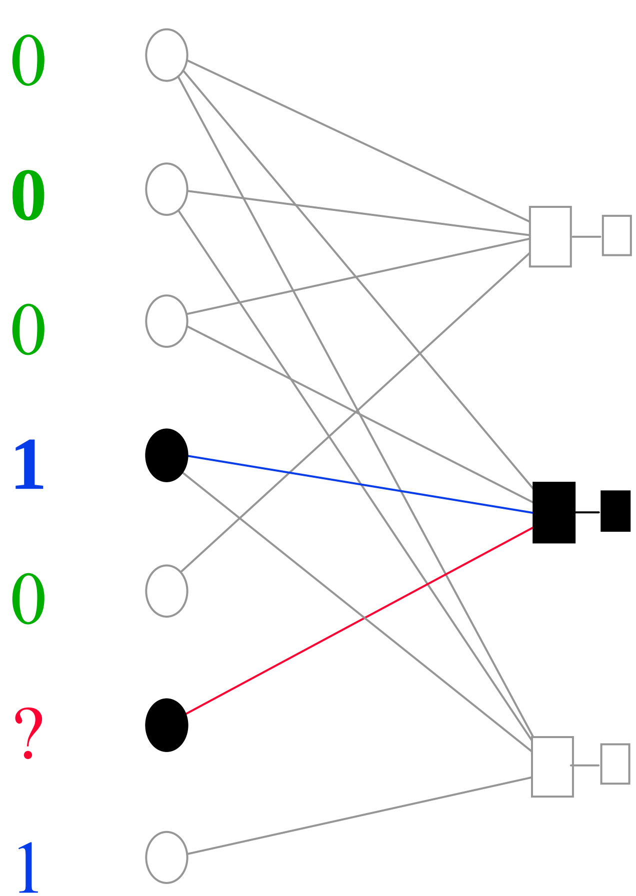


Delete check node and edge;
forward new variable node value

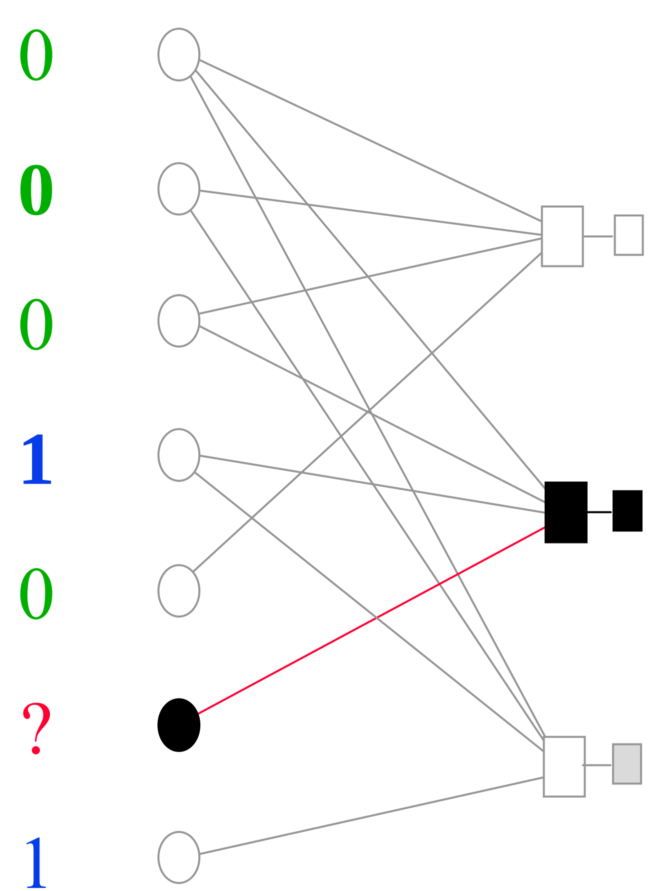


Peeling Decoder – Step 2

Accumulate parity

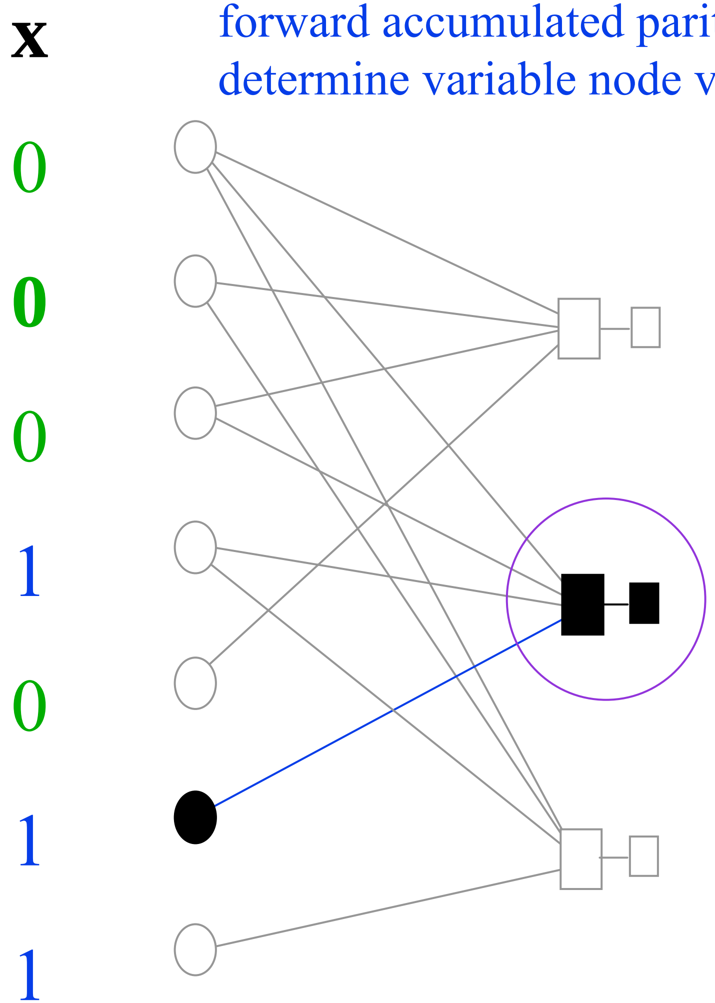


Delete known variable nodes and edges

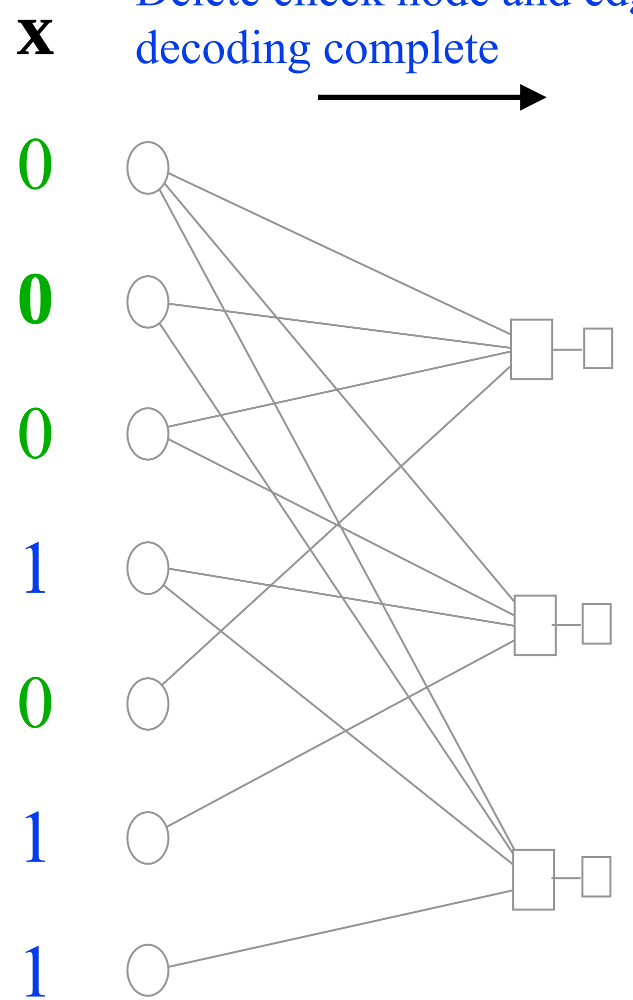


Peeling Decoder – Step 3

Find degree-1 check node;
forward accumulated parity;
determine variable node value

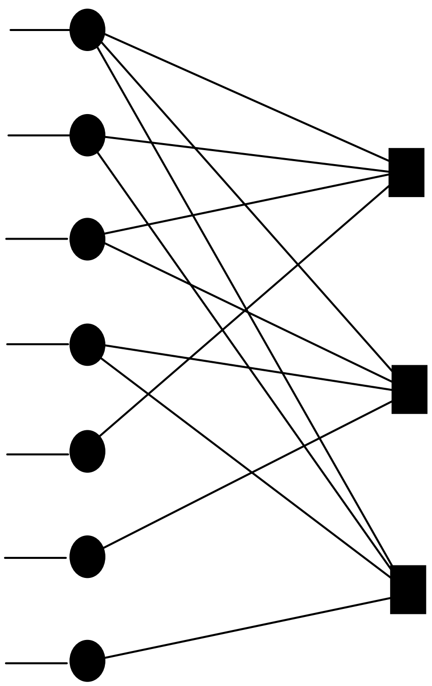


Delete check node and edge;
decoding complete

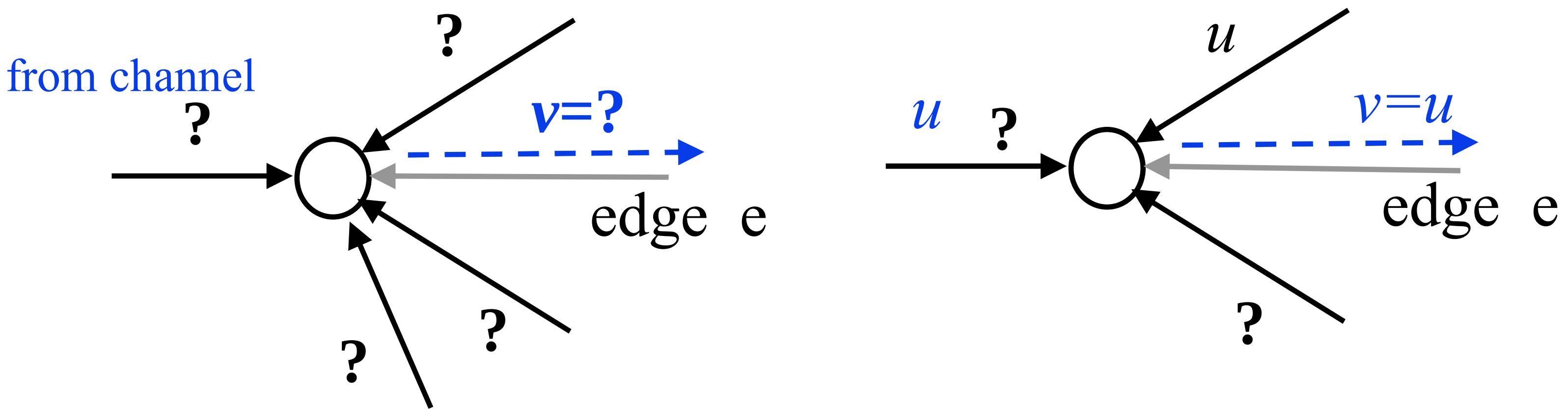


Message-Passing Decoding

- The local decoding procedure can be described in terms of an iterative, "message-passing" algorithm in which all variable nodes and all check nodes in parallel iteratively pass messages along their adjacent edges.
- The values of the code bits are updated accordingly.
- The algorithm continues until all erasures are filled in, or until the completion of a specified number of iterations.



Variable-to-Check Node Message



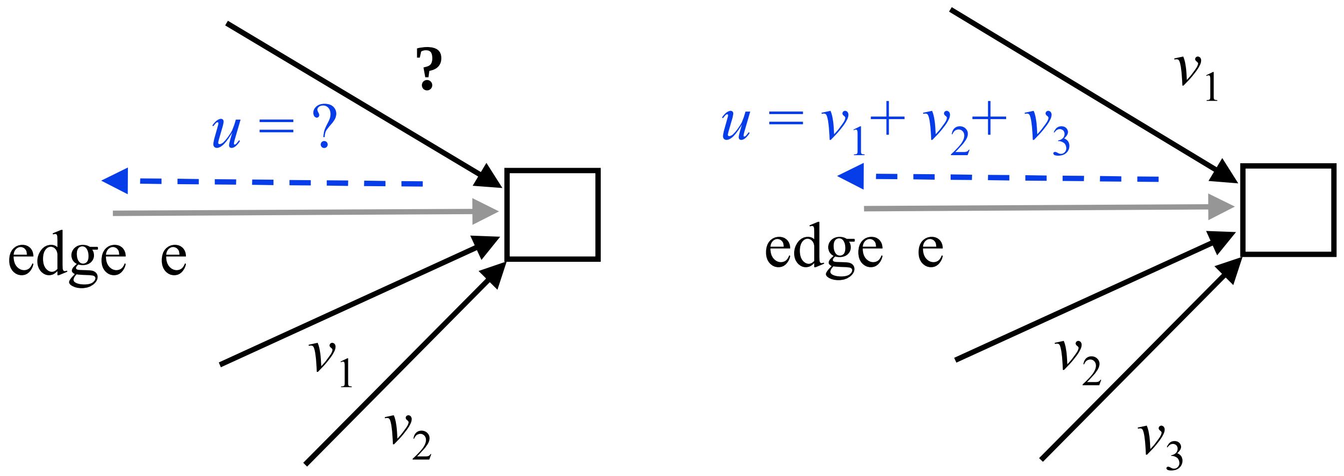
Variable-to-check message on edge e

If *all* other incoming messages are ?, send message v = ?

If *any* other incoming message u is 0 or 1, send $v=u$ and, if the bit was an erasure, fill it with u, too.

(Note that there are no errors on the BEC, so a message that is 0 or 1 must be correct. Messages cannot be inconsistent.)

Check-to-Variable Node Message

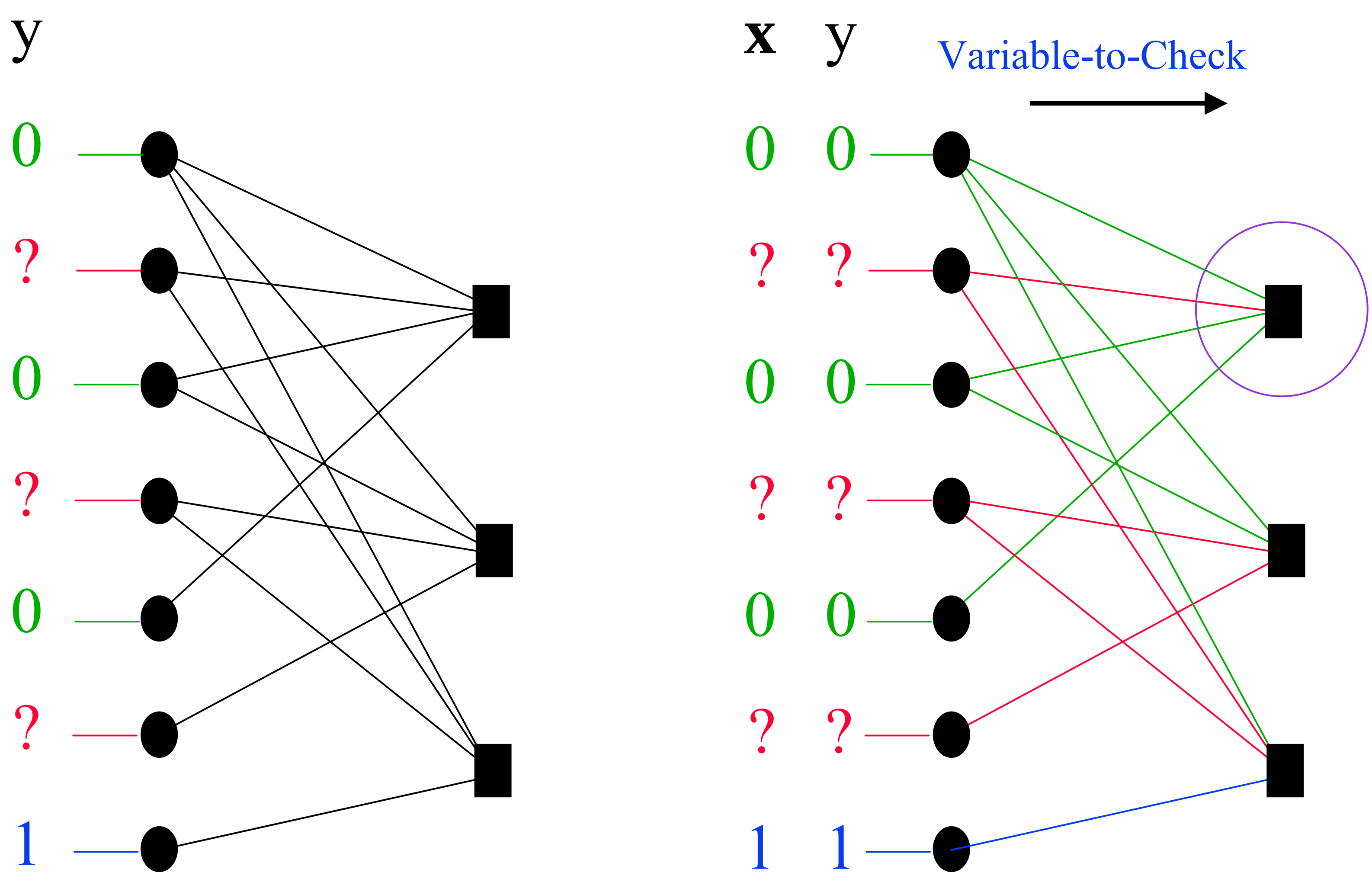


Check-to-variable message on edge e

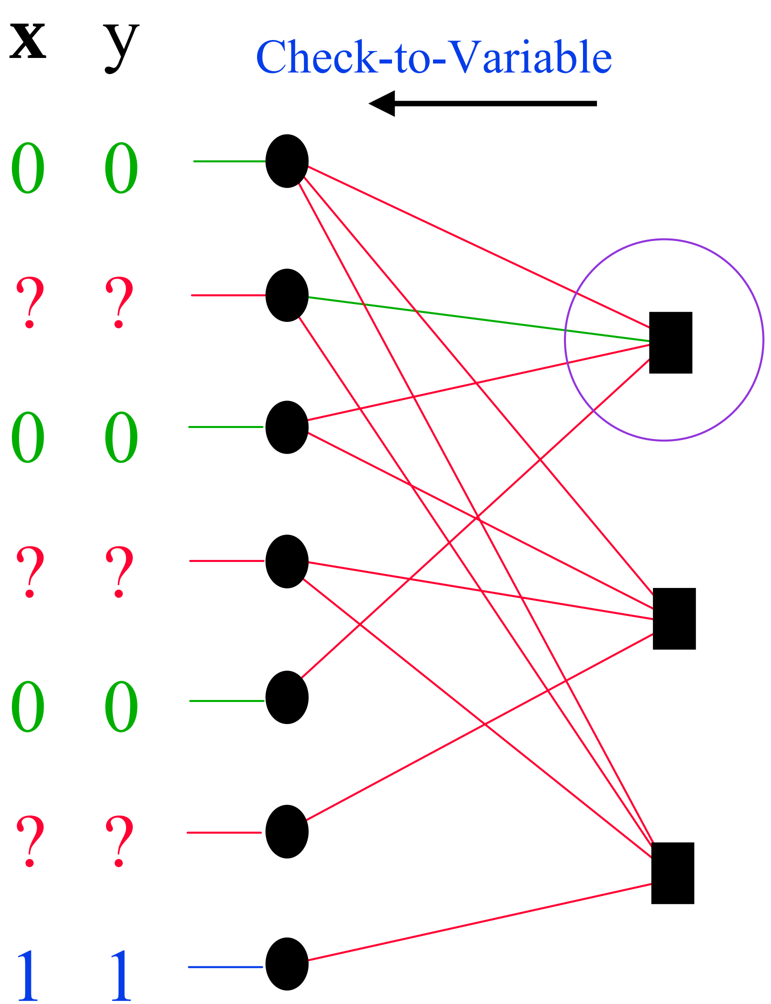
If *any* other incoming message is ?, send $u = ?$

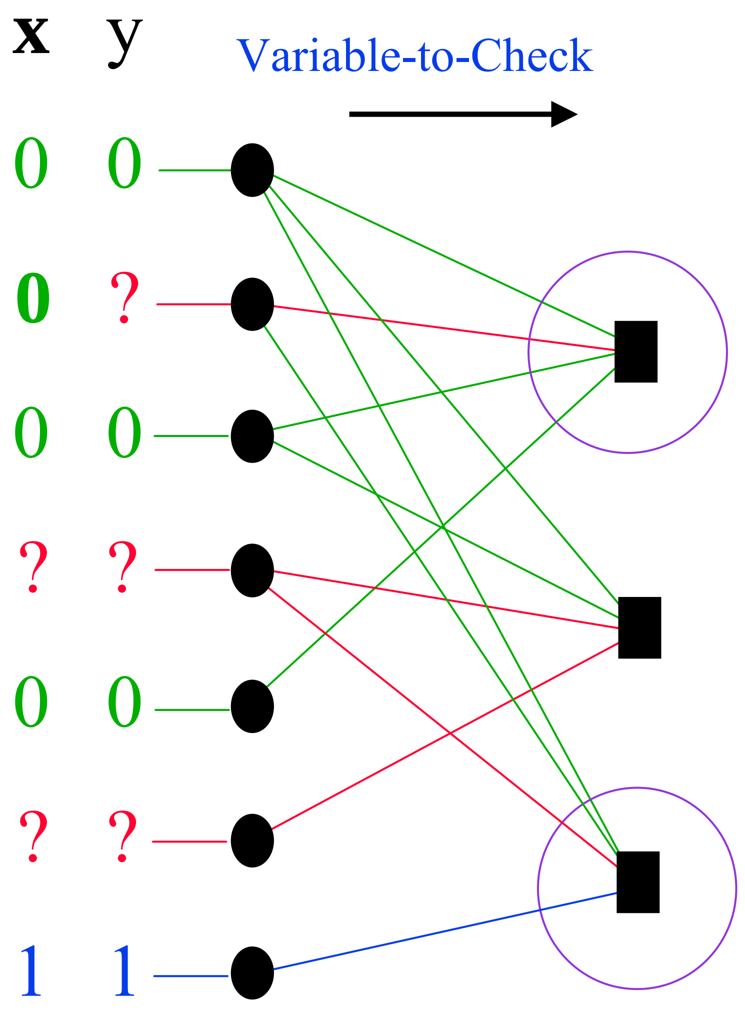
If *all* other incoming messages are in {0,1}, send the XOR of them, $u = v_1 + v_2 + v_3$.

Message-Passing Example – Initialization

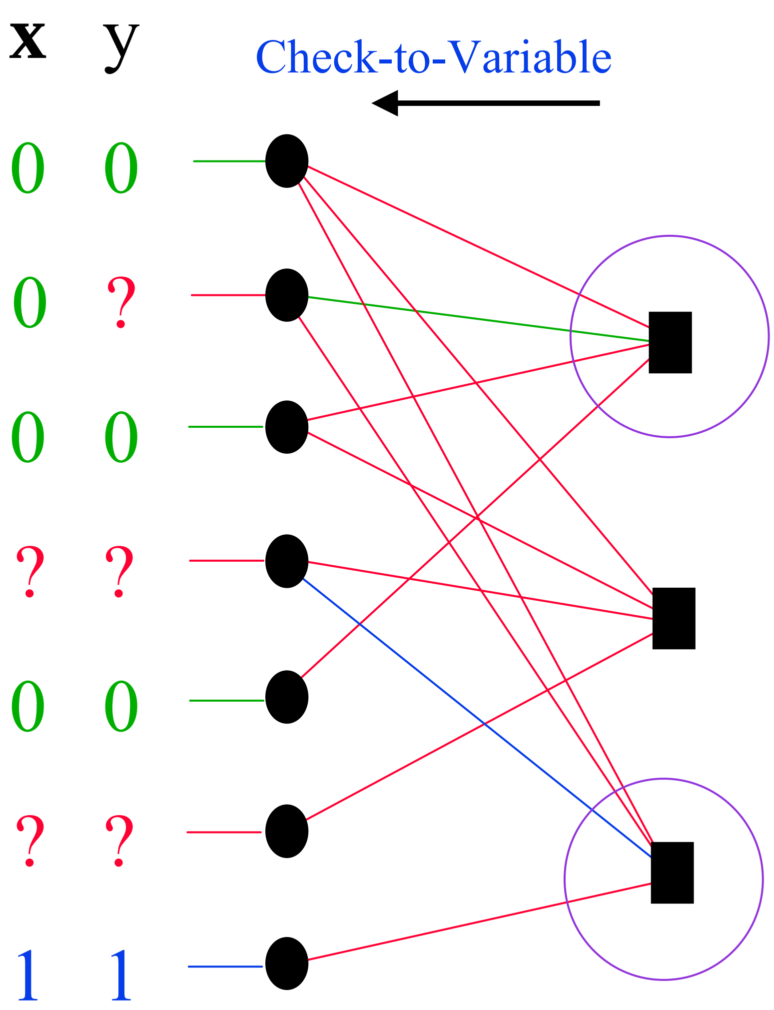


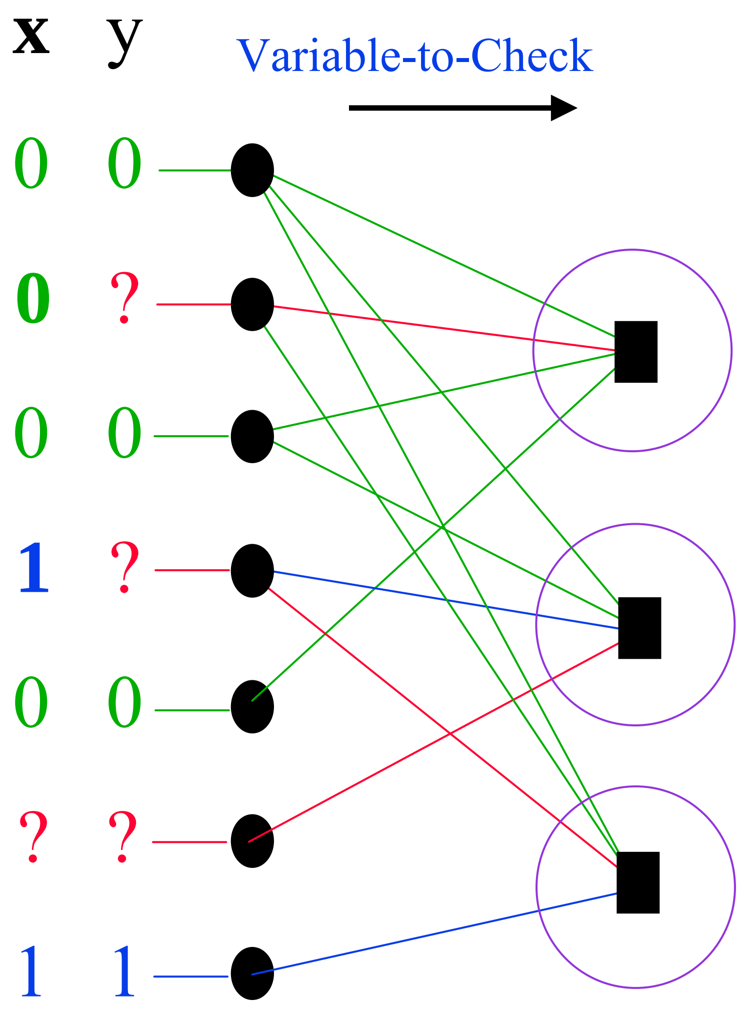
Message-Passing Example – Round 1



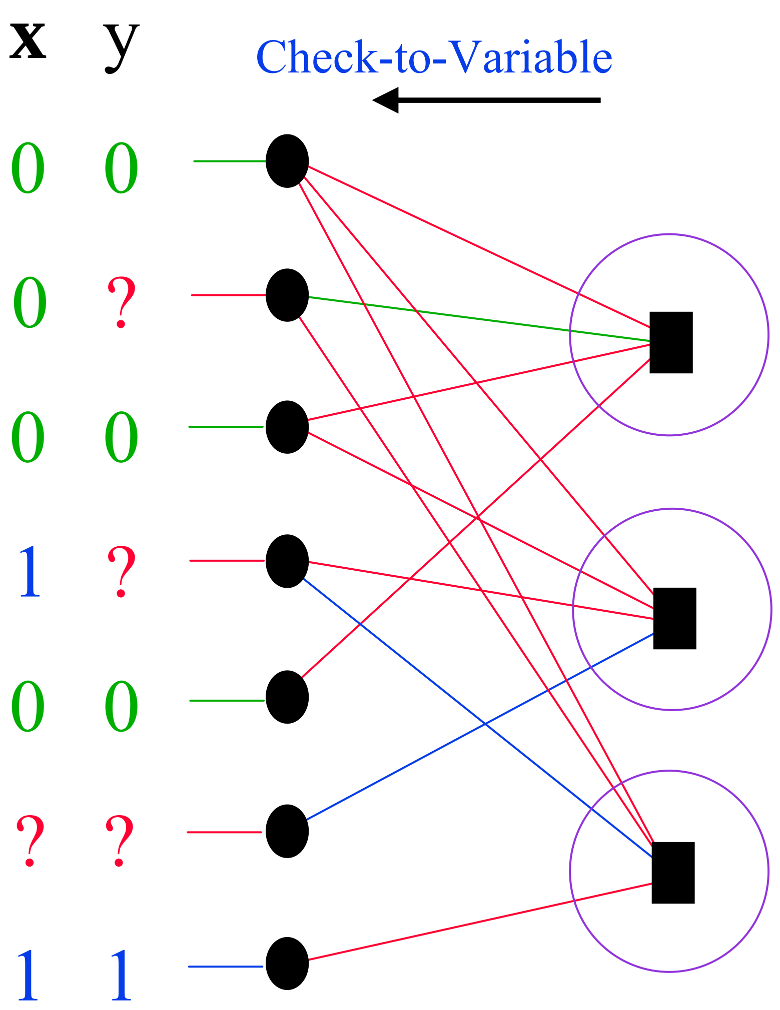


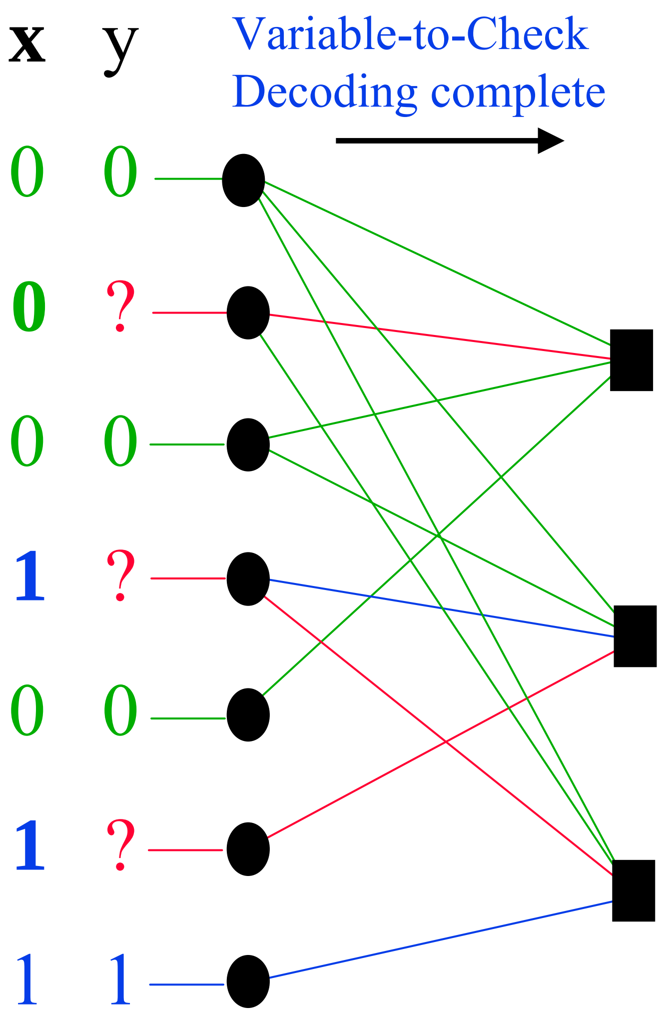
Message-Passing Example – Round 2





Message-Passing Example – Round 3

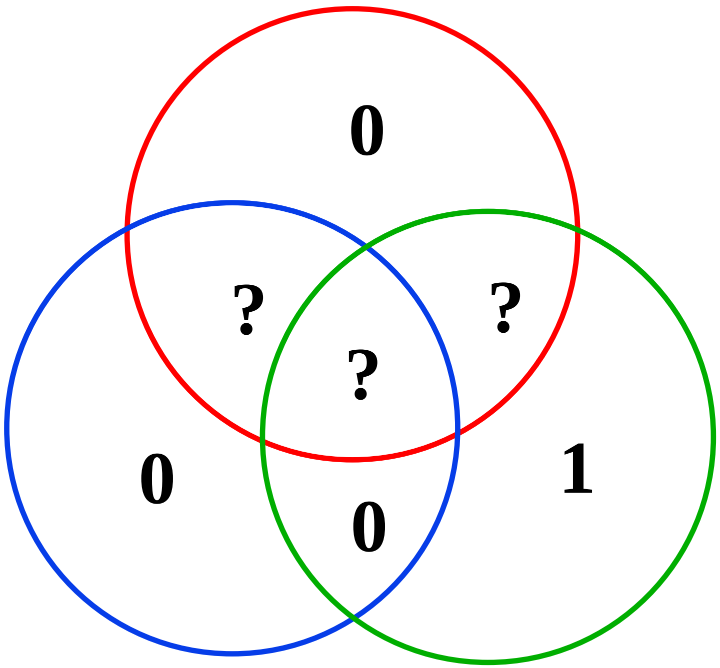




Sub-optimality of Message-Passing Decoder

Hamming code: decoding of 3 erasures

- There are 7 patterns of 3 erasures that correspond to the support of a weight-3 codeword. These can not be decoded by any decoder!
- The other 28 patterns of 3 erasures can be uniquely filled in by the optimal decoder.
- We just saw a pattern of 3 erasures that was corrected by the local decoder. Are there any that it cannot?
- Test: ? ? ? 0 0 1 0



Sub-optimality of Message-Passing Decoder

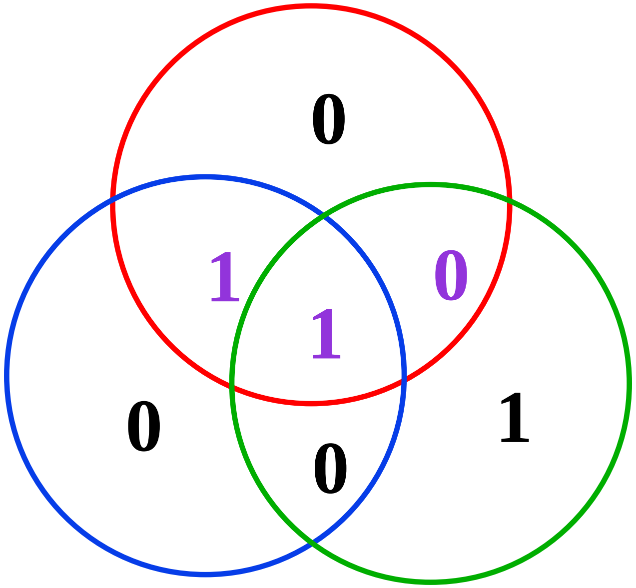
- Test: ? ? ? 0 0 1 0

- There is a unique way to fill the erasures and get a codeword:

  1 1 0 0 0 1 0

  The optimal decoder would find it.

- But every parity-check has at least 2 erasures, so local decoding will not work!

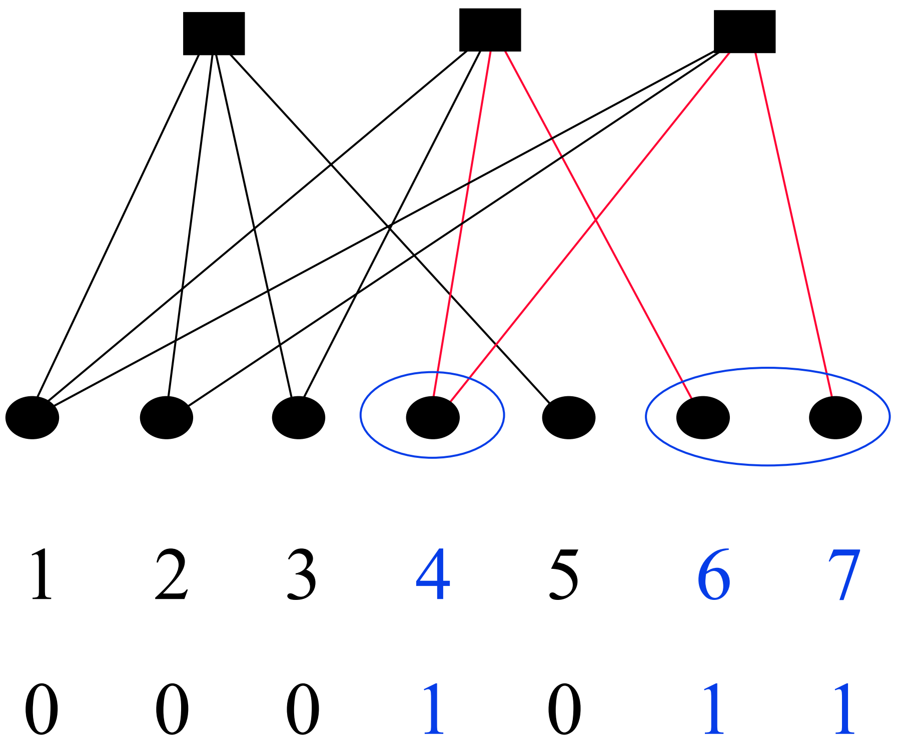


Stopping Sets

- A stopping set is a subset S of the variable nodes such that every check node connected to S is connected to S at least twice.
- The empty set is a stopping set (trivially).
- The support set (i.e., the positions of 1's) of any codeword is a stopping set (parity condition).
- A stopping set need not be the support of a codeword.

Stopping Sets

- Example 1: (7,4) Hamming code

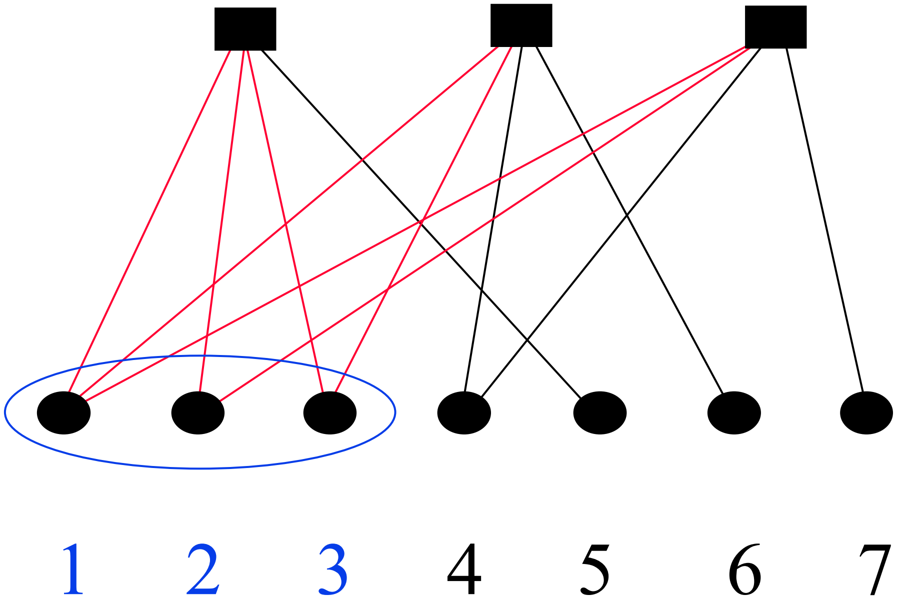


Codeword support set

S={4,6,7}

Stopping Sets

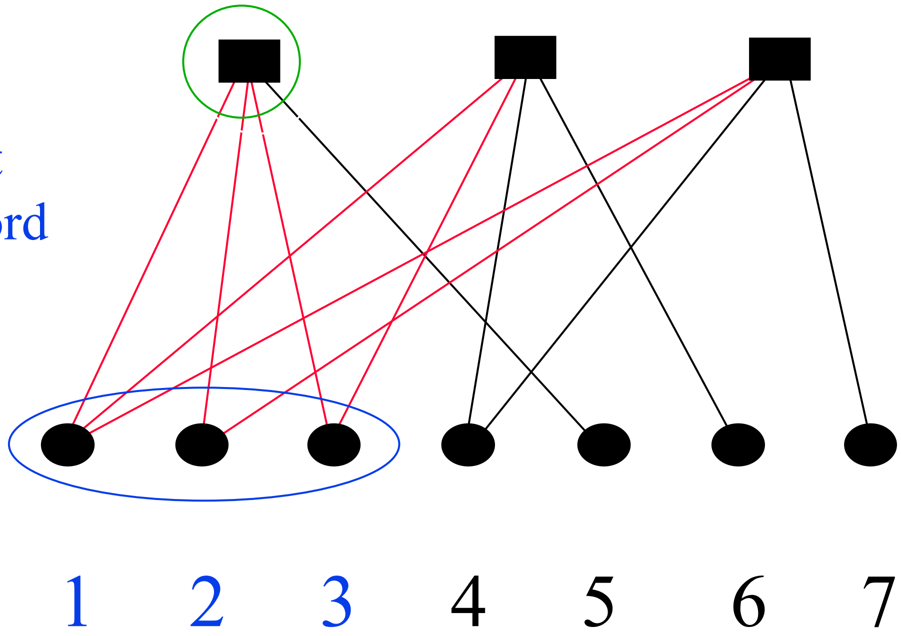
- Example 2: (7,4) Hamming code



Stopping Sets

- Example 2: (7,4) Hamming code

Not the support
set of a codeword
S={1,2,3}



Stopping Set Properties

- Every set of variable nodes contains a largest stopping set (since the union of stopping sets is also a stopping set).
- The message-passing decoder needs a check node with at most one edge connected to an erasure to proceed.
- So, if the remaining erasures form a stopping set, the decoder must stop.
- Let E be the initial set of erasures. When the message-passing decoder stops, the remaining set of erasures is the largest stopping set S in E.
  - If S is empty, the codeword has been recovered.
  - If not, the decoder has failed.

Suboptimality of Message-Passing Decoder

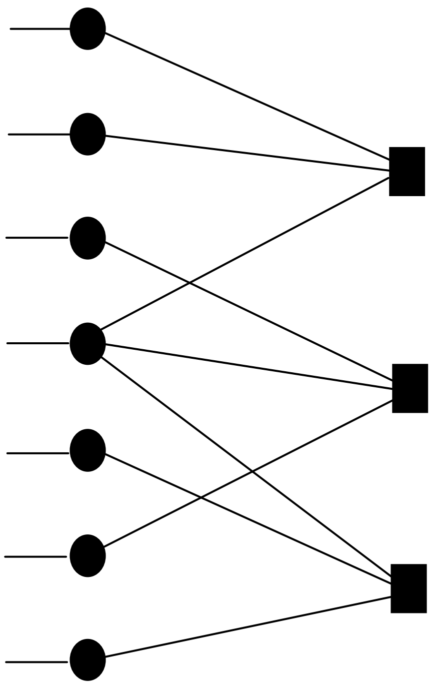
- An optimal (MAP) decoder for a code C on the BEC fails if and only if the set of erased variables includes the support set of a codeword.
- The message-passing decoder fails if and only the set of erased variables includes a non-empty stopping set.
- Conclusion: Message-passing may fail where optimal decoding succeeds!!

Message-passing is suboptimal!!

Comments on Message-Passing Decoding

- Bad news:
  - Message-passing decoding on a Tanner graph is not always optimal...
- Good news:
  - For any code C, there is a parity-check matrix on whose Tanner graph message-passing is optimal, e.g., the matrix of codewords of the dual code $C^{\perp}$.
- Bad news:
  - That Tanner graph may be very dense, so even message-passing decoding is too complex.

Another (7,4) Code



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

R=4/7 $d_{\min}$=2

All stopping sets contain codeword supports.

Message-passing decoder on this graph is optimal!

(Cycle-free Tanner graph implies this.)

Comments on Message-Passing Decoding

- Good news:
  - If a Tanner graph is cycle-free, the message-passing decoder is optimal!
- Bad news:
  - Binary linear codes with cycle-free Tanner graphs are necessarily weak...
- Good news:
  - The Tanner graph of a long LDPC code behaves almost like a cycle-free graph!

Analysis of LDPC Codes on BEC

- In the spirit of Shannon, we can analyze the performance of message-passing decoding on ensembles of LDPC codes with specified degree distributions (λ,ρ).
- The results of the analysis allow us to design LDPC codes that transmit reliably with MP decoding at rates approaching the Shannon capacity of the BEC.
- In fact, sequences of LDPC codes have been designed that actually achieve the Shannon capacity.
- The analysis can assume the all-0's codeword is sent.

Key Results - 1

- Concentration
  - With high probability, the performance of ℓ rounds of MP decoding on a randomly selected (n, λ, ρ) code converges to the ensemble average performance as the length $n\rightarrow\infty$.
- Convergence to cycle-free performance
  - The average performance of ℓ rounds of MP decoding on the (n, λ, ρ) ensemble converges to the performance on a graph with no cycles of length $\leq 2\ell$ as the length $n\rightarrow\infty$.

Key Results - 2

- Computing the cycle-free performance
  - The cycle-free performance can be computed by a tractable algorithm – density evolution.

- Threshold calculation
  - There is a threshold probability $p^*(\lambda,\rho)$ such that, for channel erasure probability $\varepsilon < p^*(\lambda,\rho)$, the cycle-free error probability approaches 0 as the number of iterations $\ell \rightarrow \infty$.

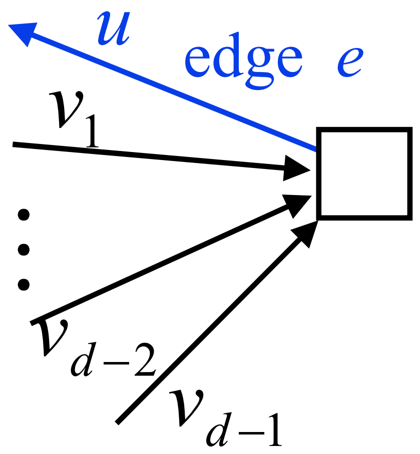
Asymptotic Performance Analysis

- We assume a cycle-free (λ,ρ) Tanner graph.
- Let $p_0 = \varepsilon$, the channel erasure probability.
- We find a recursion formula for p_ℓ , the probability that a randomly chosen edge carries a variable-to-check erasure message in round ℓ.
- We then find the largest ε such that p_ℓ converges to 0, as $\ell \rightarrow \infty$. This value is called the threshold.
- This procedure is called "density evolution" analysis.

Density Evolution-1

- Consider a check node of degree d with independent incoming messages.

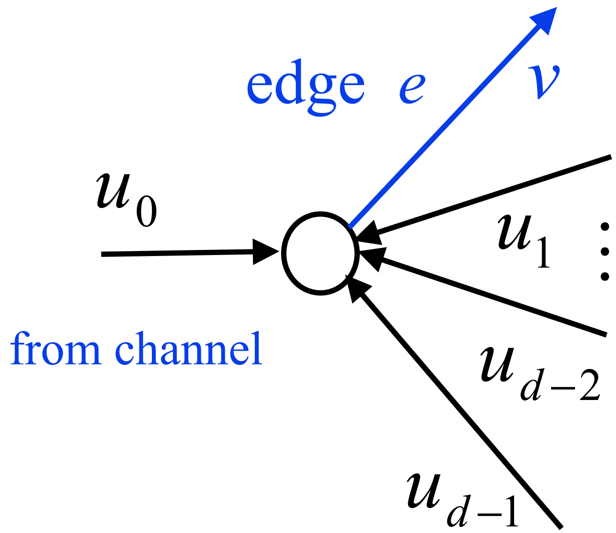
$$\Pr(u=?) = \Pr(v_i = ?, \text{for some } i = 1,\ldots,d-1)$$
$$= 1 - \Pr(v_i \neq ?, \text{for all } i = 1,\ldots,d-1)$$
$$= 1-(1-p_{\ell-1})^{d-1}$$



- The probability that edge e connects to a check node of degree d is ρ_d, so

$$\Pr(u=?) = \sum_{d=1}^{d_c} \rho_d \left(1-(1-p_{\ell-1})^{d-1}\right)$$
$$= 1 - \sum_{d=1}^{d_c} \rho_d (1-p_{\ell-1})^{d-1}$$
$$= 1 - \rho(1-p_{\ell-1})$$

Density Evolution-2



- Consider a variable node of degree d with independent incoming messages.

$$\Pr(v=?) = \Pr(u_0=?)\Pr(u_i=?, \text{for all } i=1,\ldots,d-1)$$
$$= p_0\left[1-\rho(1-p_{\ell-1})\right]^{d-1}$$

- The probability that edge e connects to a variable node of degree d is λ_d, so

$$\Pr(v=?) = \sum_{d=1}^{d_v}\lambda_d p_0\left[1-\rho(1-p_{\ell-1})\right]^{d-1}$$
$$= p_0\lambda(1-\rho(1-p_{\ell-1}))$$

$$\boxed{p_\ell = p_0\lambda\,(1-\rho(1-p_{\ell-1}))}$$

Threshold Property

$$\boxed{p_\ell = p_0 \lambda\left(1 - \rho(1 - p_{\ell-1})\right)}$$

- There is a threshold probability $p^*(\lambda, \rho)$ such that

if

$$p_0 = \varepsilon < p^*(\lambda, \rho),$$

then

$$\lim_{\ell \to \infty} p_\ell \to 0.$$

Threshold Interpretation

- Operationally, this means that using a code drawn from the ensemble of length-n LDPC codes with degree distribution pair (λ, ρ), we can transmit as reliably as desired over the BEC(ε) channel if

$$\varepsilon < p^*(\lambda, \rho),$$

for sufficiently large block length n .

Computing the Threshold

- Define $f(p,x) = p\,\lambda\,(1-\rho(1-x))$
- The threshold $p^*(\lambda, \rho)$ is the largest probability p such that

$$f(p,x) - x < 0$$

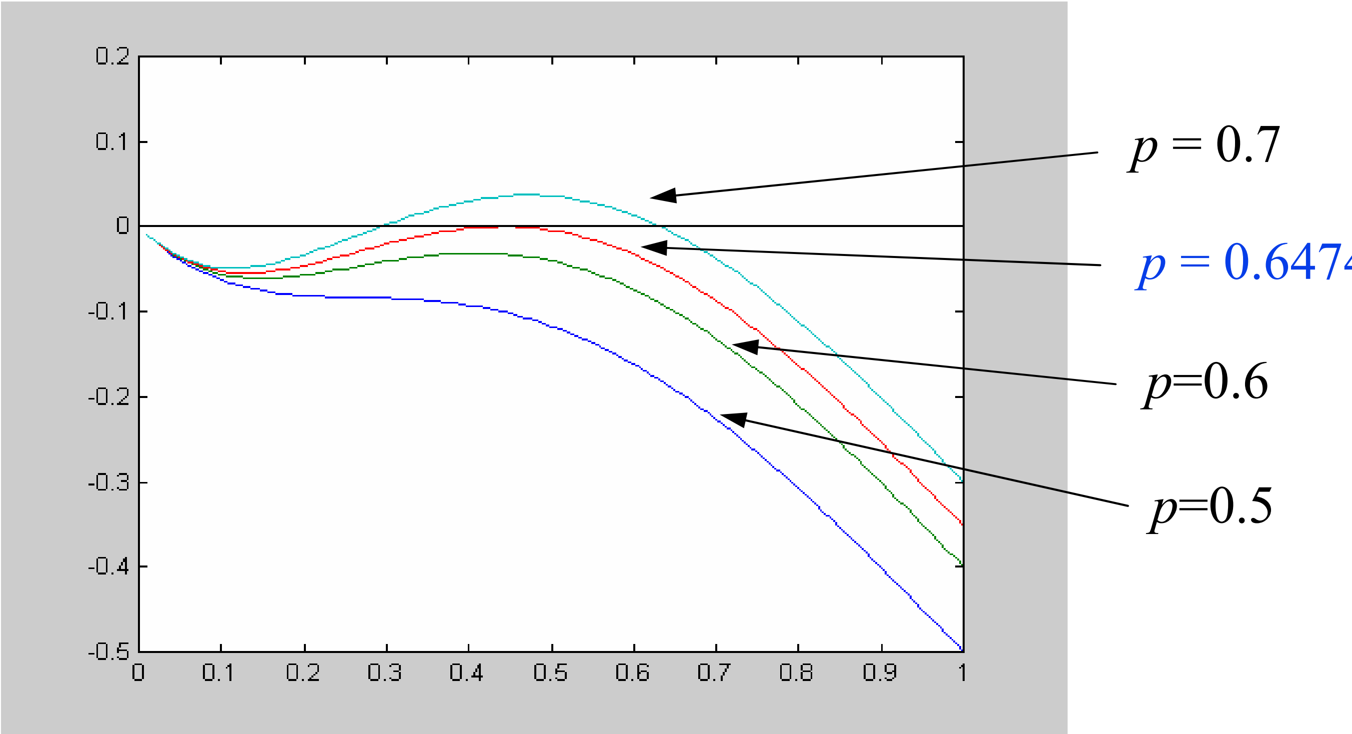
on the interval $x \in (0,1]$.

- This leads to a graphical interpretation of the threshold $p^*(\lambda, \rho)$

Graphical Determination of the Threshold

- Example: $(j,k)=(3,4)$

$$f(x,p)-x = p\left(1-(1-x)^3\right)^2 - x \qquad p^* \approx 0.6474$$



(j,k)-Regular LDPC Code Thresholds

- There is a closed form expression for thresholds of (j,k)-regular LDPC codes.
- Examples:

(j,k)	R	p^{Sh}	$p^*(j,k)$
(3,4)	1/4	¾=0.75	≈0.6474
(3,5)	2/5	3/5=0.6	≈0.5176
(3,6)	1/2	½=0.5	≈0.4294
(4,6)	1/3	⅔≈0.67	≈0.5061
(4,8)	1/2	½=0.5	≈0.3834

$$p^*(3,4) = \frac{3125}{3672 + 252\sqrt{21}}$$
$$\approx 0.647426$$

Degree Distribution Optimization

- Two approaches:
  - Fix design rate R(λ,ρ) and find degree distributions $\lambda(x)$, $\rho(x)$ to maximize the threshold $p^*(\lambda,\rho)$.
  - Fix the threshold p^*, and find degree distributions $\lambda(x)$, $\rho(x)$ to maximize the rate R(λ,ρ).
- For the latter, we can:
  - start with a specific $\rho(x)$ and optimize $\lambda(x)$;
  - then, for the optimal $\lambda(x)$, find the optimal check distribution;
  - ping-pong back and forth until satisfied with the results.

Variable Degree Distribution Optimization

- Fix a check degree distribution $\rho(x)$ and threshold ε.
- Fix maximum variable degree $l_{\max}$.
- Define $$g(x,\lambda_2,\ldots,\lambda_{l_{\max}}) = \varepsilon\lambda(1-\rho(1-x))-x$$ $$= \varepsilon\sum_{i\geq 2}\lambda_i(1-\rho(1-x))^{i-1}-x$$
- Use linear programming to find

$$\max_{\lambda}\left\{\sum_{i=2}^{l_{\max}}(\lambda_i/i)\,\middle|\,\lambda_i\geq 0;\sum_{i=2}^{l_{\max}}\lambda_i=1;\, g\leq 0 \text{ for } x\in[0,1]\right\}$$

- Since the rate R(λ,ρ) is an increasing function of λ_i/i, this maximizes the design rate.

Practical Optimization

- In practice, good performance is found for a check degree distribution of the form:

$$\rho(x) = ax^{r-1} + (1-a)x^{r}$$

- Example 1: ι_{max} = 8, r =6, design rate ½

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

$$\rho(x) = x^{5}$$

- Rate: R(λ,ρ) ≈ 0.5004
- Threshold: $p^*(\lambda,\rho) \approx 0.4810$

Bound on the Threshold

- Taylor series analysis yields the general upper bound:

$$p^*(\lambda,\rho) \leq \frac{1}{\lambda'(0)\rho'(1)}.$$

- For previous example with $p^*(\lambda,\rho) \approx 0.4810$, the upper bound gives:

$$\frac{1}{\lambda'(0)\rho'(1)} = \frac{1}{(0.409)\cdot 5} \leq 0.4890$$

EXIT Chart Analysis

- Extrinsic information transfer (EXIT) charts provide a nice graphical depiction of density evolution and MP decoding [tenBrink,1999]
- Rewrite the density evolution recursion as:

$$
\begin{aligned}
f(x,p) &= p\lambda(1-\rho(1-x)) \\
&= \mathrm{v}_p(\mathrm{c}(x))
\end{aligned}
$$

where

$$
v_p(x) = p\lambda(x)
$$

$$
c(x) = 1-\rho(1-x)
$$

EXIT Chart Analysis

- Recall that the MP convergence condition was

$$f(x, p) < x, \text{ for all } x \in (0,1)$$

- Since $\lambda(x)$ is invertible, the condition becomes

$$c(x) < \mathrm{v}_p^{-1}(x), \text{ for all } x \in (0,1)$$

- Graphically, this says that the curve for $c(x)$ must lie below the curve for $\mathrm{v}_p^{-1}(x)$ for all $p < p^*$.

EXIT Chart Example

- Example: (3,4)-regular LDPC code, p^*=0.6474

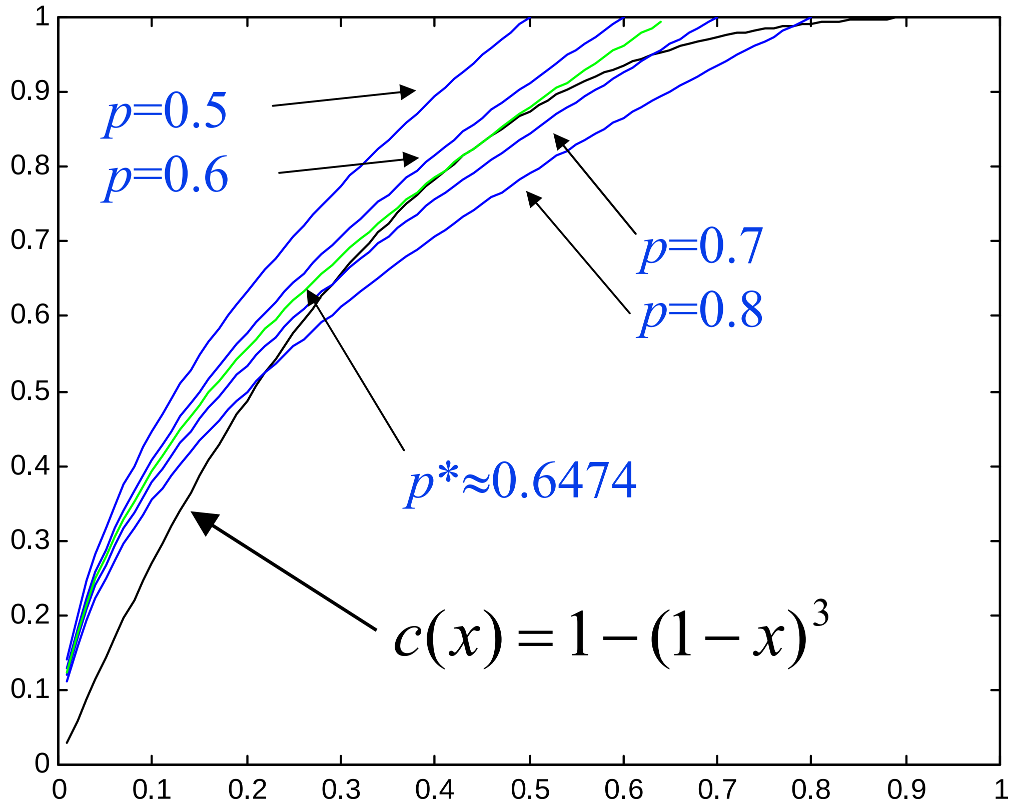
$$\lambda(x) = x^2 \qquad \rho(x) = x^3$$

$$\begin{aligned} v_p(x) &= p\lambda(x) \\ &= px^2 \end{aligned} \qquad \begin{aligned} c(x) &= 1-\rho(1-x) \\ &= 1-(1-x)^3 \end{aligned}$$

$$v_p^{-1}(x) = \left(\frac{x}{p}\right)^{\frac{1}{2}}$$

EXIT Chart Example

- Example: (3,4)-regular LDPC code, $p^*=0.6474$



$$v_p^{-1}(x)=\left(\frac{x}{p}\right)^{\frac{1}{2}}$$

for various values of initial erasure probability p

EXIT Charts and Density Evolution

- EXIT charts can be used to visualize density evolution.
  - Assume initial fraction of erasure messages $p_0=p$.
  - The fraction of erasures emitted successively by check node q_i and by variable nodes and p_i are obtained by successively applying $c(x)$ and $v_p(x)$.

$$q_1 = c(p_0)$$

$$p_1 = v_p(q_1) = v_p(c(p_0)) \qquad [\text{note}: v_p^{-1}(p_1) = q_1]$$

$$q_2 = c(p_1)$$

$$p_2 = v_p(q_2) = v_p(c(p_1)) \qquad [\text{note}: v_p^{-1}(p_2) = q_2]$$

EXIT Charts and Density Evolution

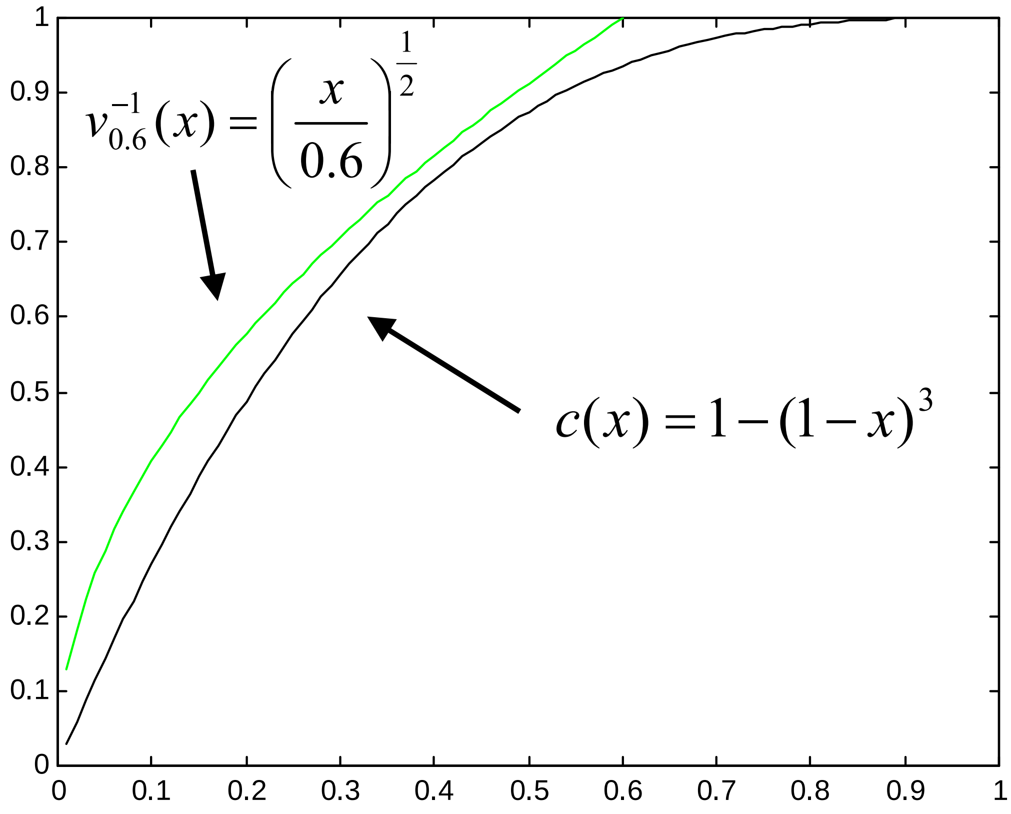
- Graphically, this computation describes a staircase function.

- If $p < p^*$, there is a "tunnel" between $v_p^{-1}(x)$ and $c(x)$ through which the staircase descends to ground level, i.e., no erasures.

- If $p > p^*$, the tunnel closes, stopping the staircase descent at a positive fraction of errors.

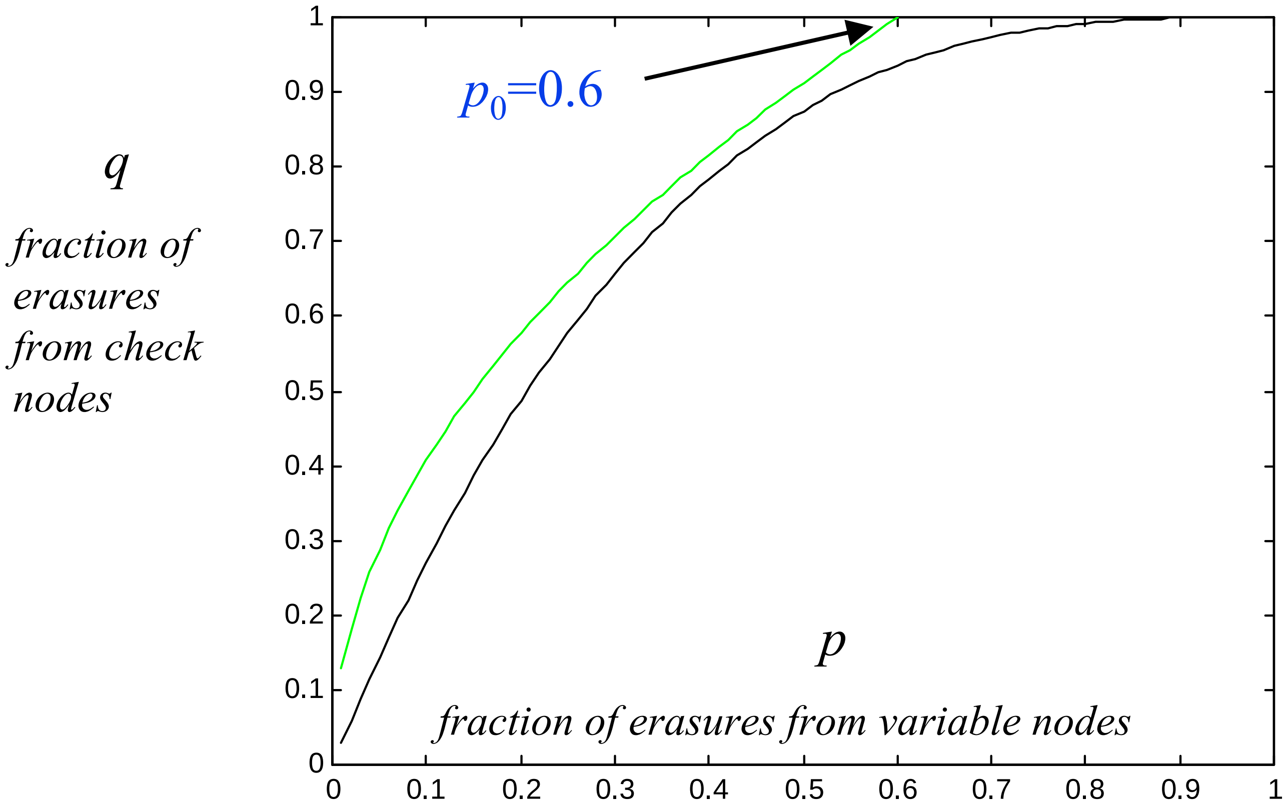
Density Evolution Visualization - 1

- Example: (3,4)-regular LDPC code, p=0.6



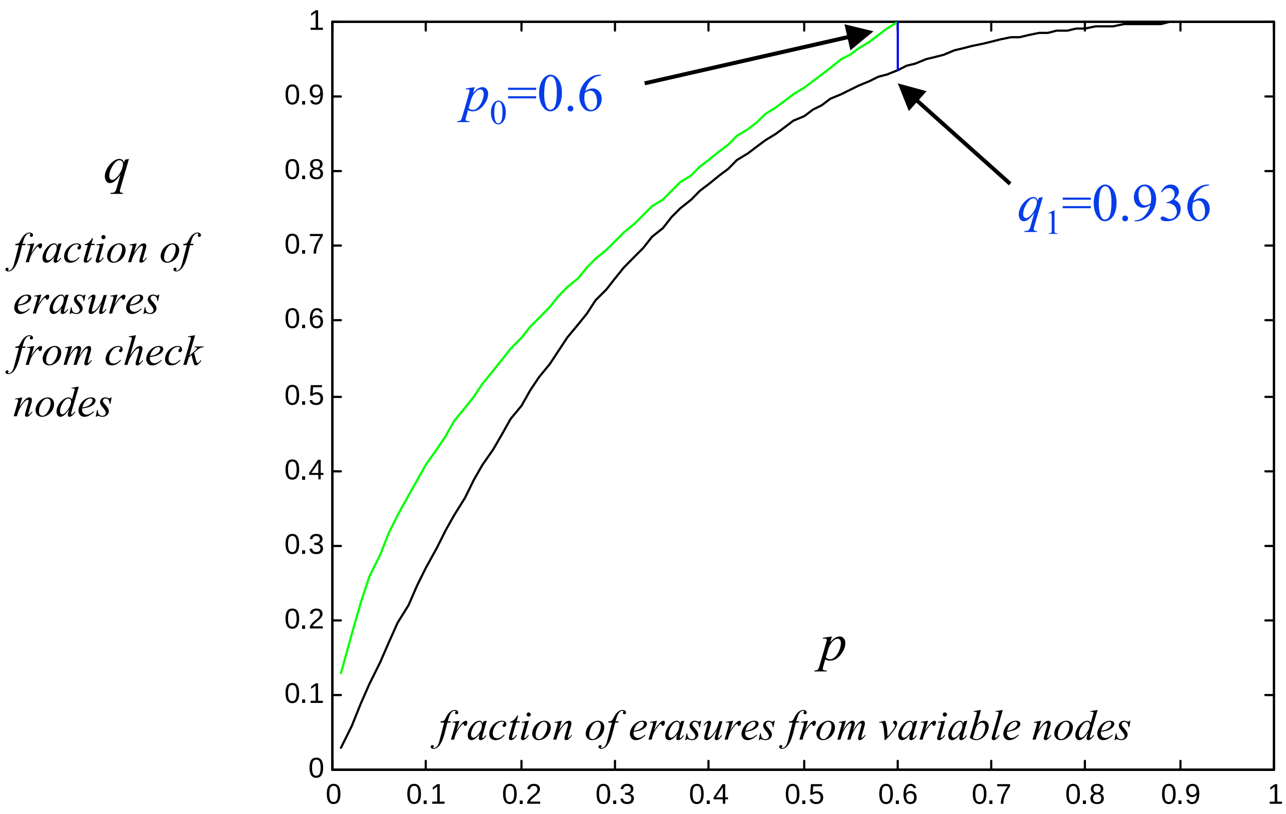
Density Evolution Visualization-2

- Example: (3,4)-regular LDPC code



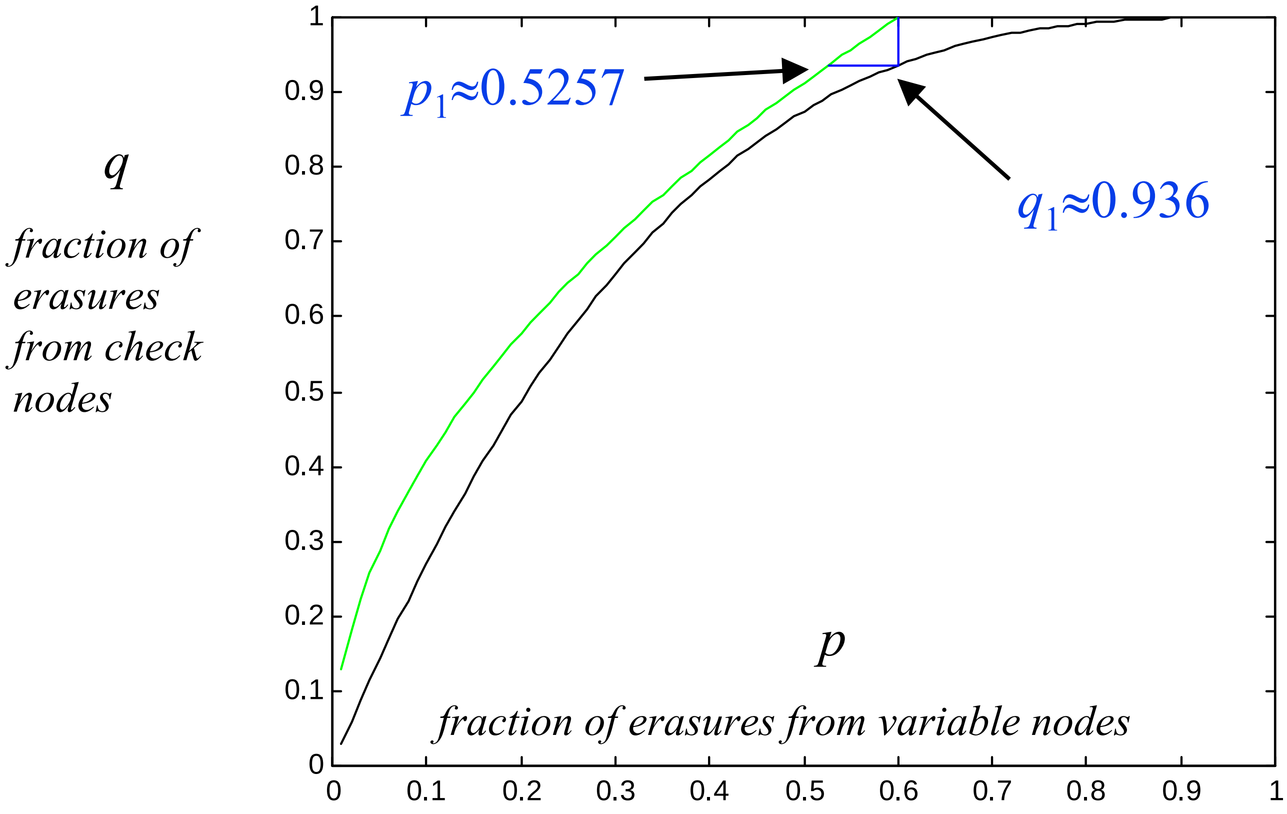
Density Evolution Visualization

- Example: (3,4)-regular LDPC code



Density Evolution Visualization

- Example: (3,4)-regular LDPC code

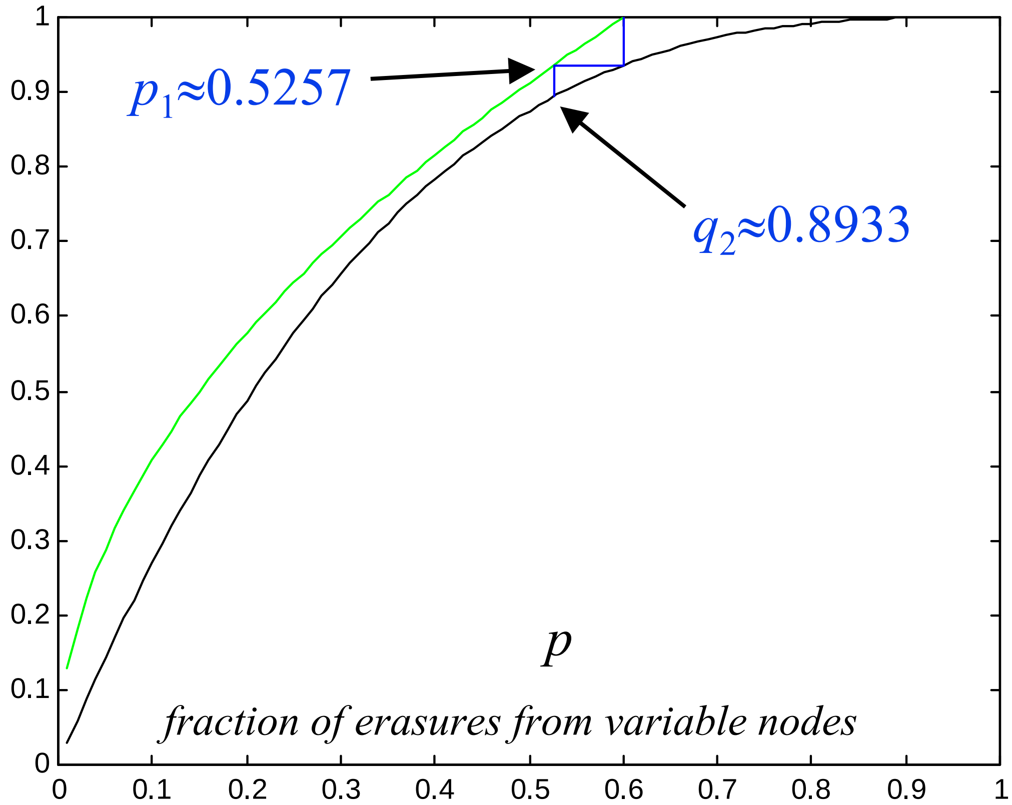


Density Evolution Visualization

- Example: (3,4)-regular LDPC code

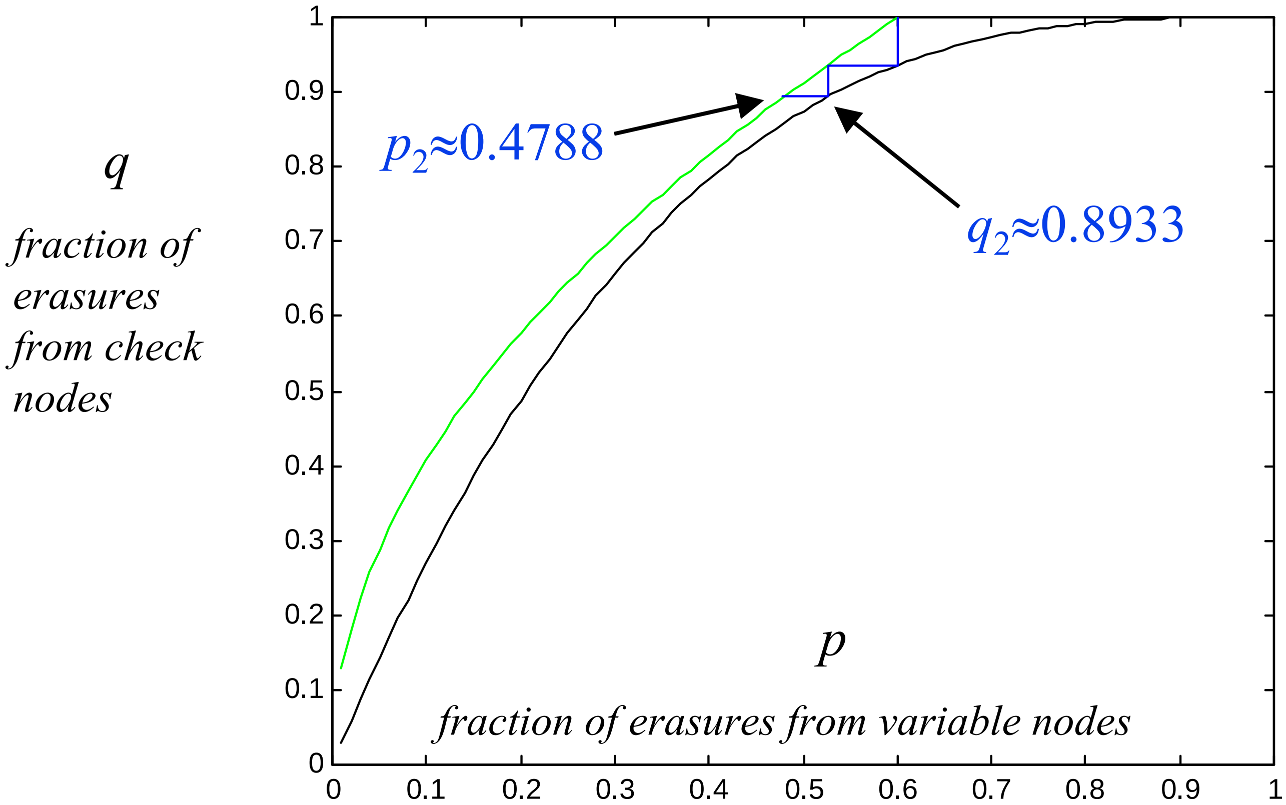
q

fraction of erasures from check nodes



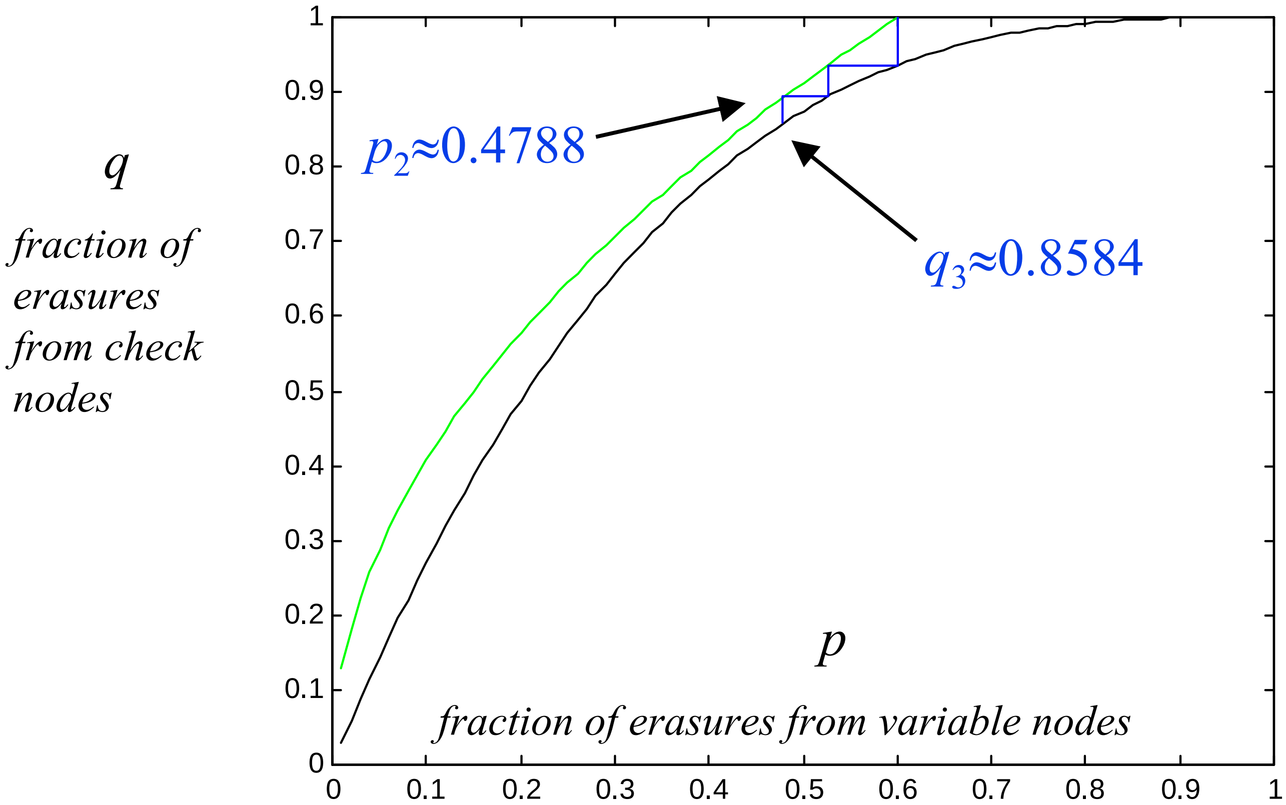
Density Evolution Visualization

- Example: (3,4)-regular LDPC code



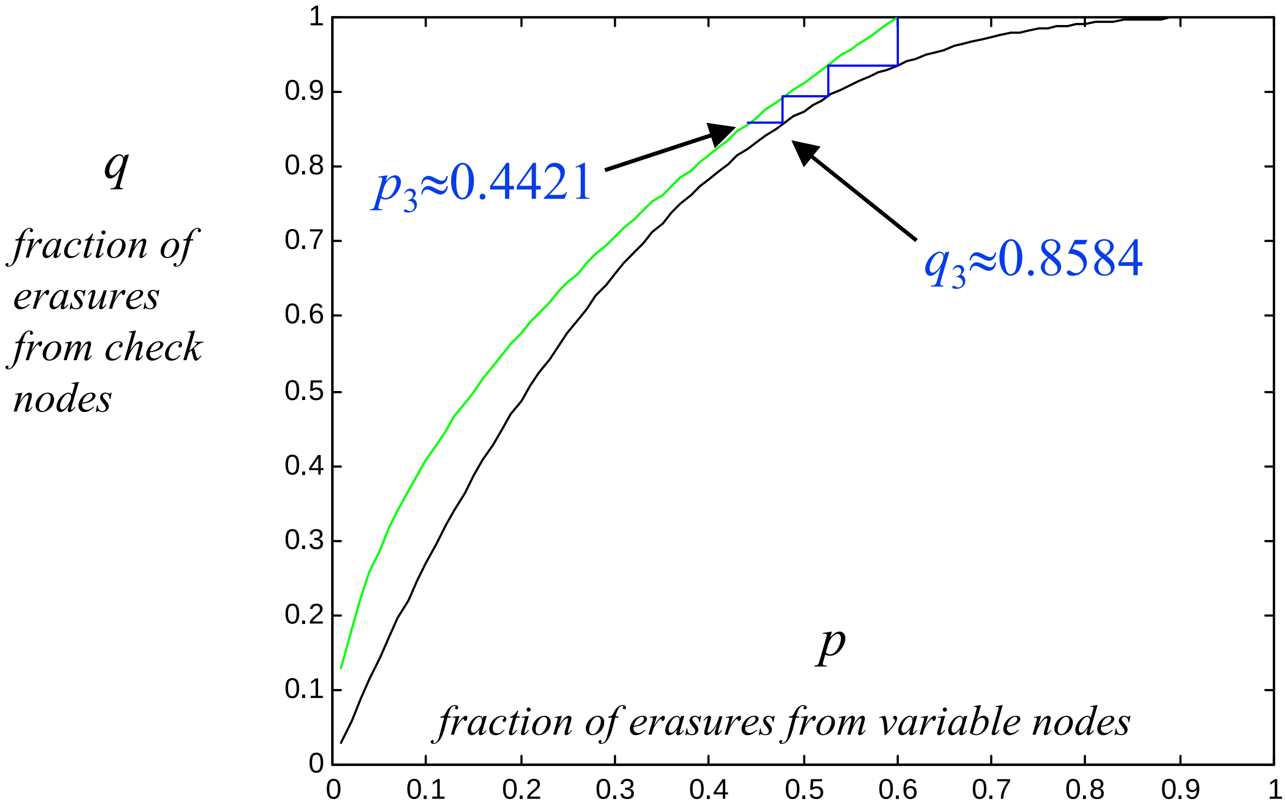
Density Evolution Visualization

- Example: (3,4)-regular LDPC code



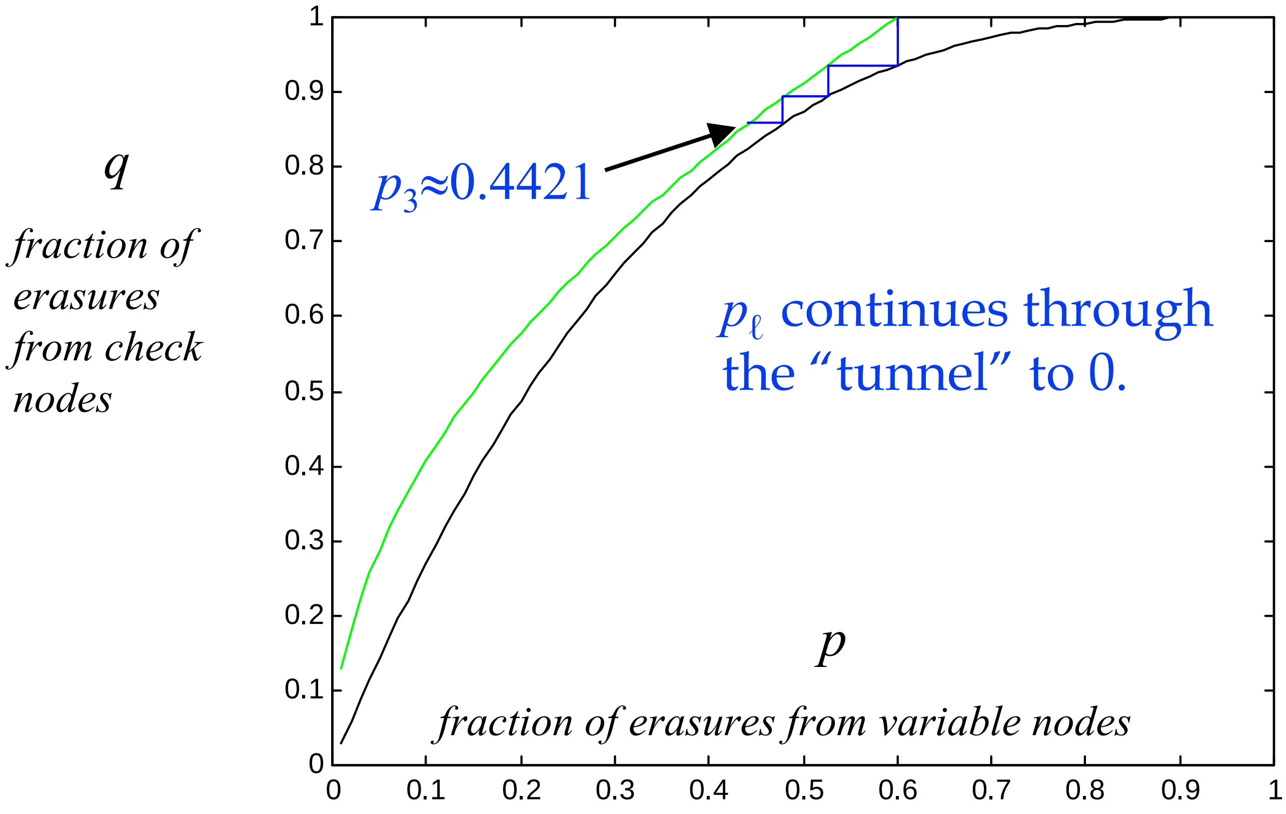
Density Evolution Visualization

- Example: (3,4)-regular LDPC code



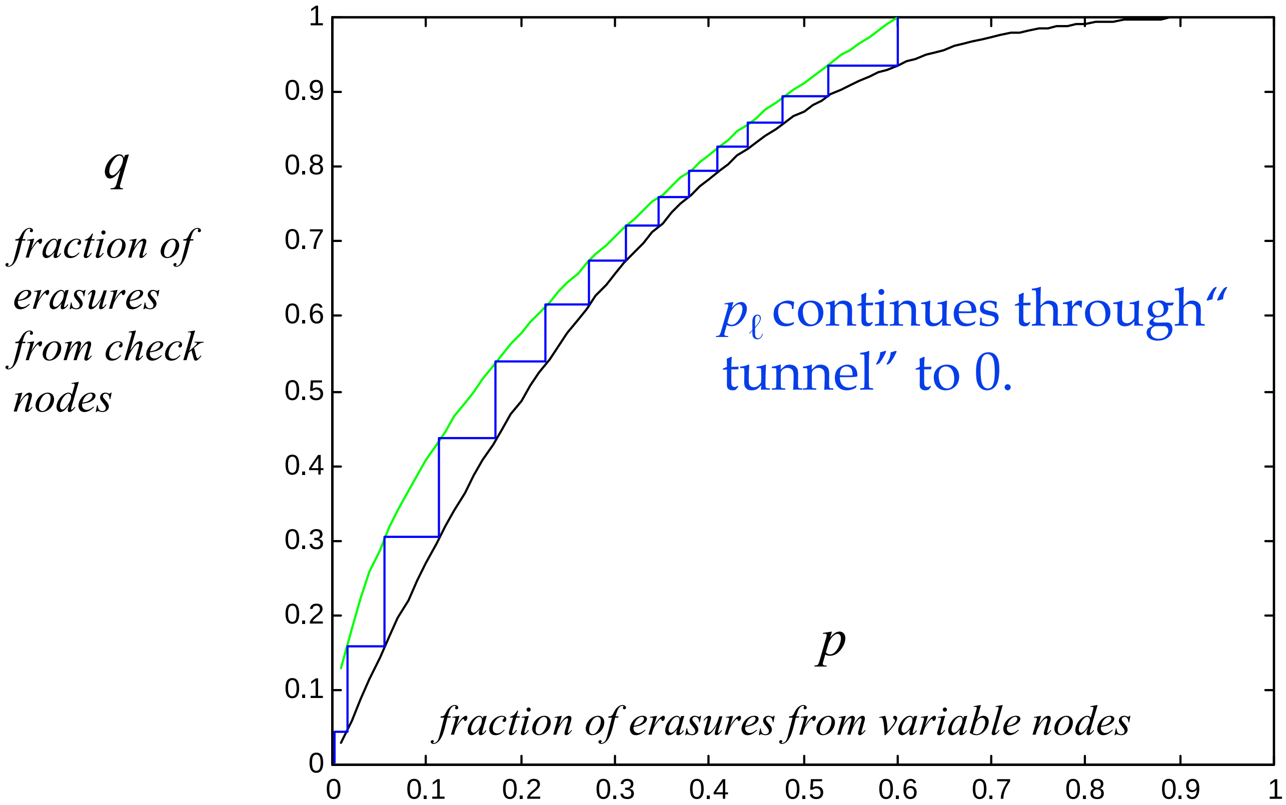
Density Evolution Visualization

- Example: (3,4)-regular LDPC code



Density Evolution Visualization

- Example: (3,4)-regular LDPC code



Matching Condition

- For capacity-achieving sequences of LDPC codes for the BEC, the EXIT chart curves must match.
- This is called the matching condition.
- Such sequences have been developed:
    - Tornado codes
    - Right-regular LDPC codes
    - Accumulate-Repeat-Accumulate codes

Decoding for Other Channels

- We now consider analysis and design of LDPC codes for BSC(p) and BiAWGN(σ) channels. We call p and σ the "channel parameter" for these two channels, respectively.
- Many concepts, results, and design methods have natural (but non-trivial) extensions to these channels.
- The messages are probability mass functions or log-likelihood ratios.
- The message-passing paradigm at variable and check nodes will be applied.
- The decoding method is called "belief propagation" or BP, for short.

Belief Propagation

- Consider transmission of binary inputs $X \in \{\pm 1\}$ over a memoryless channel using linear code C.
- Assume codewords are transmitted equiprobably.
- Then $$\hat{x}_i^{MAP}(y) = \arg\max_{x_i \in \{\pm 1\}} P_{X_i|Y}(x_i \mid y)$$

$$= \operatorname*{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} P_{X|Y}(x \mid y)$$

$$= \operatorname*{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} P_{Y|X}(y \mid x) P_X(x)$$

$$= \operatorname*{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \left(\prod_{j=1}^{n} P_{Y_j|X_j}(y_j \mid x_j) \right) \cdot f_C(x)$$

where $f_C(x)$ is the indicator function for C.

Belief Propagation

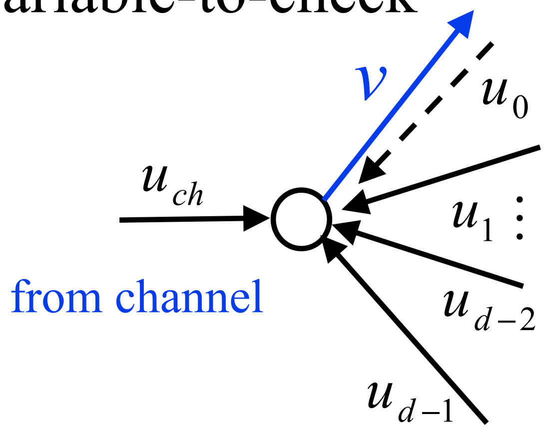
- For codes with cycle-free Tanner graphs, there is a message-passing approach to bit-wise MAP decoding.
- The messages are essentially conditional bit distributions, denoted $u = [u(1), u(-1)]$.
- The initial messages presented by the channel to the variable nodes are of the form

$$u_{ch,i} = [u_{ch,i}(1), u_{ch,i}(-1)] = [p_{Y_i|X_i}(y_i \mid 1), p_{Y_i|X_i}(y_i \mid -1)]$$

- The variable-to-check and check-to-variable message updates are determined by the "sum-product" update rule.
- The BEC decoder can be formulated as a BP decoder.

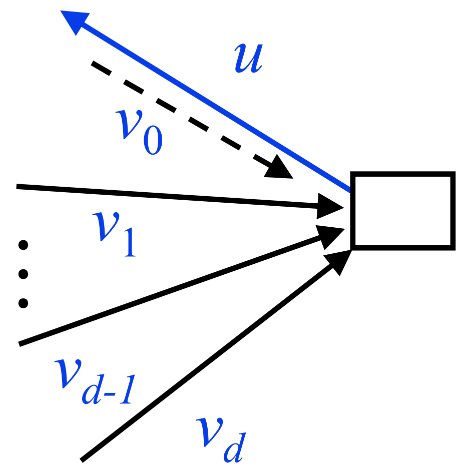
Sum-Product Update Rule

- Variable-to-check



$$v(b) = u_{ch} \prod_{k=1}^{d-1} u_k(b), \text{ for } b \in \{\pm 1\}$$

- Check-to-variable

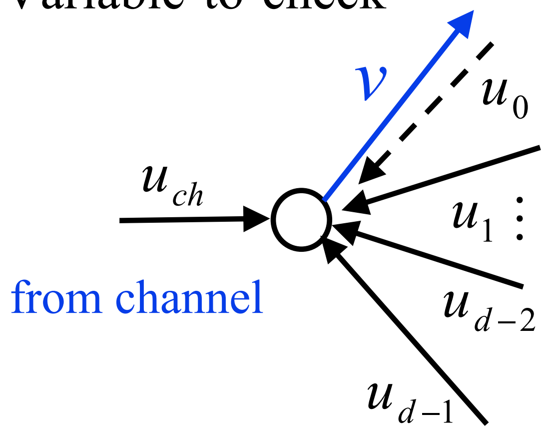


$$u(b) = \sum_{\{x_1, x_2, \ldots, x_{d-1}\}} f(b, x_1, x_2, \ldots, x_{d-1}) \prod_{k=1}^{d-1} v_k(x_k),$$

where f is the parity-check indicator function.

Variable Node Update - Heuristic

- Variable-to-check



$$v(b) = u_{ch} \prod_{k=1}^{d-1} u_k(b)\,,\ \text{for } b \in \{\pm 1\}$$

Suppose incoming messages $u_0, u_1, \ldots, u_{d-1}$ from check nodes 0,1, ..., d-1 and message u_{ch} from the channel are independent estimates of $[P(x = 1), P(x = -1)]$.

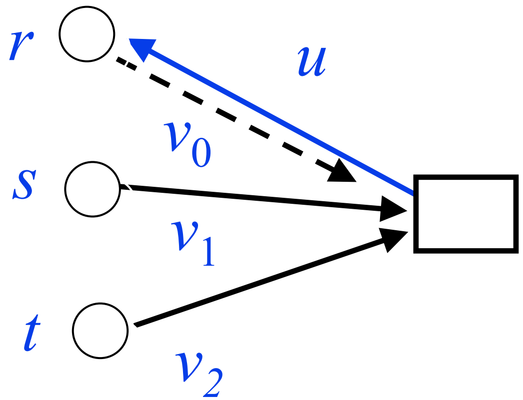
Then, a reasonable estimate to send to check node 0 based upon the other estimates would be the product of those estimates (suitably normalized).

$$\hat{P}(x = b) = P_{ch}(x = b) \prod_{k=1}^{d-1} P_k(x = b)$$

We do not use the "intrinsic information" u_0 provided by check node 0. The estimate v represents "extrinsic information".

Check-Node Update - Heuristic

$$u(b) = \sum_{\{x_1, x_2, \ldots, x_{d-1}\}} f(b, x_1, x_2, \ldots, x_{d-1}) \prod_{k=1}^{d-1} v_k(x_k),$$



Parity-check node equation: $r \oplus s \oplus t = 0$

Over {-1,1}, this translates to: $r \cdot s \cdot t = 1$

$P(r{=}1) = P(s = 1, t = 1) + P(s = \text{-}1, t = \text{-}1)$

$= P(s = 1)P(t = 1) + P(s = \text{-}1)P(t = \text{-}1)$

[by independence assumption]

Similarly

$P(r = \text{-}1) = P(s = 1, t = \text{-}1) + P(s = \text{-}1, t = 1)$

$= P(s = 1)P(t = \text{-}1) + P(s = \text{-}1)P(t = 1)$

Log-Likelihood Formulation

- The sum-product update is simplified using log-likelihoods
- For message *u*, define

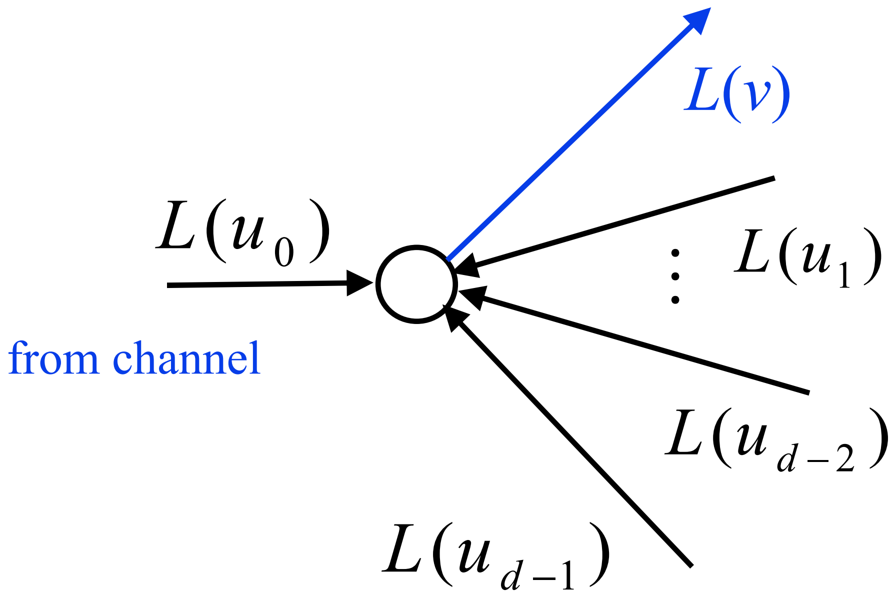
$$L(u) = \log \frac{u(1)}{u(-1)}$$

- Note that

$$u(1) = \frac{e^{L(u)}}{1+e^{L(u)}} \quad \text{and} \quad u(-1) = \frac{1}{1+e^{L(u)}}$$

Log-Likelihood Formulation – Variable Node

- The variable-to-check update rule then takes the form:

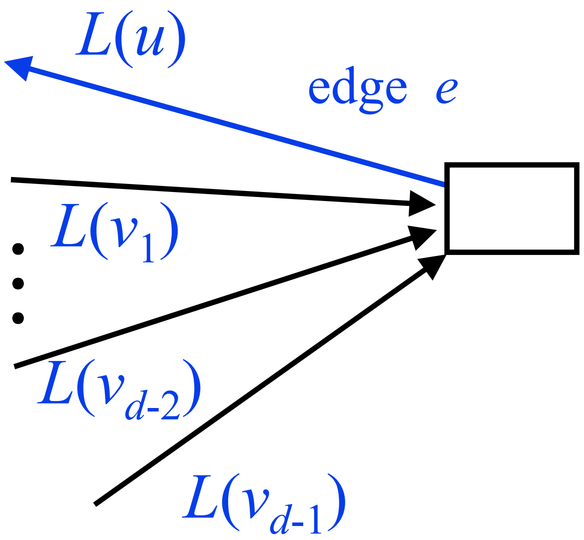


$$L(v) = \sum_{k=0}^{d-1} L(u_k)$$

Log-Likelihood Formulation – Check Node

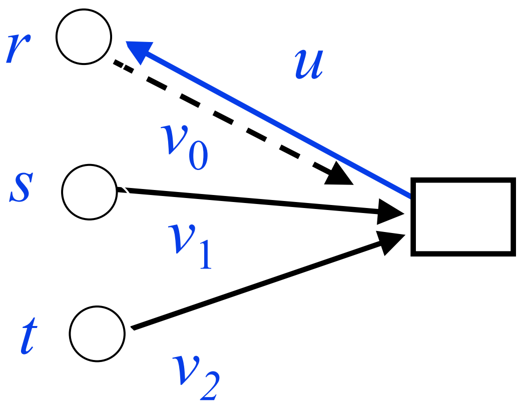
- The check-to-variable update rule then takes the form:

$$L(u) = 2\tanh^{-1}\left(\prod_{k=1}^{d-1} \tanh\frac{L(v_k)}{2}\right)$$



Log-Likelihood Formulation – Check Node

- To see this, consider the special case of a degree 3 check node.
- It is easy to verify that



$$P_r - Q_r = (P_s - Q_s)(P_t - Q_t)$$

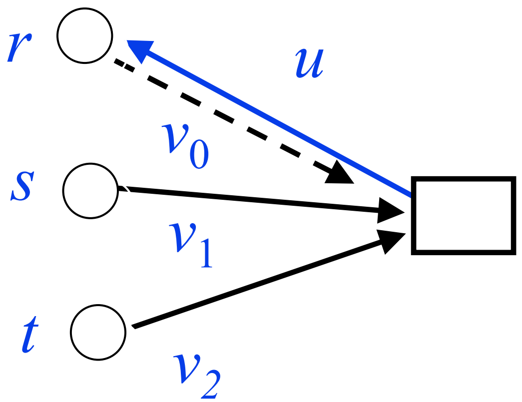
where

$$P_a = P(a=1) \text{ and } Q_a = P(a=-1), \text{ for node } a$$

- This can be generalized to a check node of any degree by a simple inductive argument.

Log-Likelihood Formulation – Check Node

- Translating to log-likelihood ratios, this becomes



$$\frac{e^{L(u)}-1}{e^{L(u)}+1}=\frac{e^{L(v_1)}-1}{e^{L(v_1)}+1}\cdot\frac{e^{L(v_2)}-1}{e^{L(v_2)}+1}$$

- Noting that

$$\frac{e^{L(a)}-1}{e^{L(a)}+1}=\frac{e^{\frac{L(a)}{2}}-e^{-\frac{L(a)}{2}}}{e^{\frac{L(a)}{2}}+e^{-\frac{L(a)}{2}}}=\tanh\left(\frac{L(a)}{2}\right)$$

we conclude

$$\tanh\left(\frac{L(u)}{2}\right)=\tanh\left(\frac{L(v_1)}{2}\right)\tanh\left(\frac{L(v_2)}{2}\right)$$

Key Results -1

- Concentration
  - With high probability, the performance of ℓ rounds of BP decoding on a randomly selected (n, λ, ρ) code converges to the ensemble average performance as the length $n\rightarrow\infty$.
- Convergence to cycle-free performance
  - The average performance of ℓ rounds of MP decoding on the (n, λ, ρ) ensemble converges to the performance on a graph with no cycles of length $\leq 2\ell$ as the length $n\rightarrow\infty$.

Key Results -2

- Computing the cycle-free performance
  - The cycle-free performance can be computed by a somewhat more complex, but still tractable, algorithm – density evolution.

- Threshold calculation
  - There is a threshold channel parameter $p^*(\lambda,\rho)$ such that, for any "better" channel parameter p, the cycle-free error probability approaches 0 as the number of iterations $\ell \rightarrow \infty$.

Density Evolution (AWGN)

- Assume the all-1's sequence is transmitted
- The density evolution algorithm computes the probability distribution or density of LLR messages after each round of BP decoding.
- Let P_0 denote the initial LLR message density. It depends on the channel parameter σ.
- Let P_ℓ denote the density after ℓ iterations.
- The density evolution equation for a (λ,ρ) degree distribution pair is:

$$P_\ell = P_0 \otimes \lambda(\Gamma^{-1}(\rho(\Gamma(P_{\ell-1}))))$$

Density Evolution

$$\boxed{P_\ell = P_0 \otimes \lambda(\Gamma^{-1}(\rho(\Gamma(P_{\ell-1}))))}$$

- Here $\otimes$ denotes convolution of densities and Γ is interpreted as an invertible operator on probability densities.
- We interpret $\lambda(P)$ and $\rho(P)$ as operations on densities:

$$\lambda(P) = \sum_{i \geq 2} \lambda_i (P)^{\otimes(i-1)} \text{ and } \rho(P) = \sum_{i \geq 2} \rho_i (P)^{\otimes(i-1)}$$

- The fraction of incorrect (i.e., negative) messages after ℓ iterations is:

$$\int_{-\infty}^{0} P_\ell(z)dz$$

Threshold

$$\boxed{P_\ell = P_0 \otimes \lambda(\Gamma^{-1}(\rho(\Gamma(P_{\ell-1}))))}$$

- The threshold σ* is the maximum σ such that

$$\lim_{\ell \to \infty} \int_{-\infty}^{0} P_\ell(z)\,dz = 0.$$

- Operationally, this represents the minimum SNR such that a code drawn from the (λ, ρ) ensemble will ensure reliable transmission as the block length approaches infinity.

Degree Distribution Optimization

- For a given rate, the objective is to optimize $\lambda(x)$ and $\rho(x)$ for the best threshold p^*.
- The maximum left and right degrees are fixed.
- For some channels, the optimization procedure is not trivial, but there are some techniques that can be applied in practice.

Thresholds - (j,k)-Regular

- BSC(p)

(j,k)	R	$p^*(j,k)$	p^{Sh}
(3,4)	0.25	0.167	0.215
(4,6)	0.333	0.116	0.174
(3,5)	0.4	0.113	0.146
(3,6)	0.5	0.084	0.11
(4,8)	0.5	0.076	0.11

- BiAWGN(σ)

(j,k)	R	σ^*	σ^{Sh}
(3,4)	0.25	1.26	1.549
(4,6)	0.333	1.01	1.295
(3,5)	0.4	1.0	1.148
(3,6)	0.5	0.88	0.979
(4,8)	0.5	0.83	0.979

Thresholds and Optimized Irregular Codes

BiAWGN   Rate R=½

$\sigma^{\mathrm{Sh}} = 0.979$

λ_{max}	σ^*
15	0.9622
20	0.9646
30	0.9690
40	0.9718

d_v	15	20	30	50
λ_2^*	0.24446	0.23261	0.21306	0.18379
λ_2	0.23802	0.21991	0.19606	0.17120
λ_3	0.20997	0.23328	0.24039	0.21053
λ_4	0.03492	0.02058		0.00273
λ_5	0.12015			
λ_6		0.08543	0.00228	
λ_7	0.01587	0.06540	0.05516	0.00009
λ_8		0.04767	0.16602	0.15269
λ_9		0.01912	0.04088	0.09227
λ_{10}			0.01064	0.02802
λ_{14}	0.00480			
λ_{15}	0.37627			0.01206
λ_{19}		0.08064		
λ_{20}		0.22798		
λ_{28}			0.00221	
λ_{30}			0.28636	0.07212
λ_{50}				0.25830
ρ_8	0.98013	0.64854	0.00749	
ρ_9	0.01987	0.34747	0.99101	0.33620
ρ_{10}		0.00399	0.00150	0.08883
ρ_{11}				0.57497
σ^*	0.9622	0.9649	0.9690	0.9718
$(\frac{E_b}{N_0})^*_{dB}$	0.3347	0.3104	0.2735	0.2485
p^*	0.1493	0.1500	0.1510	0.1517

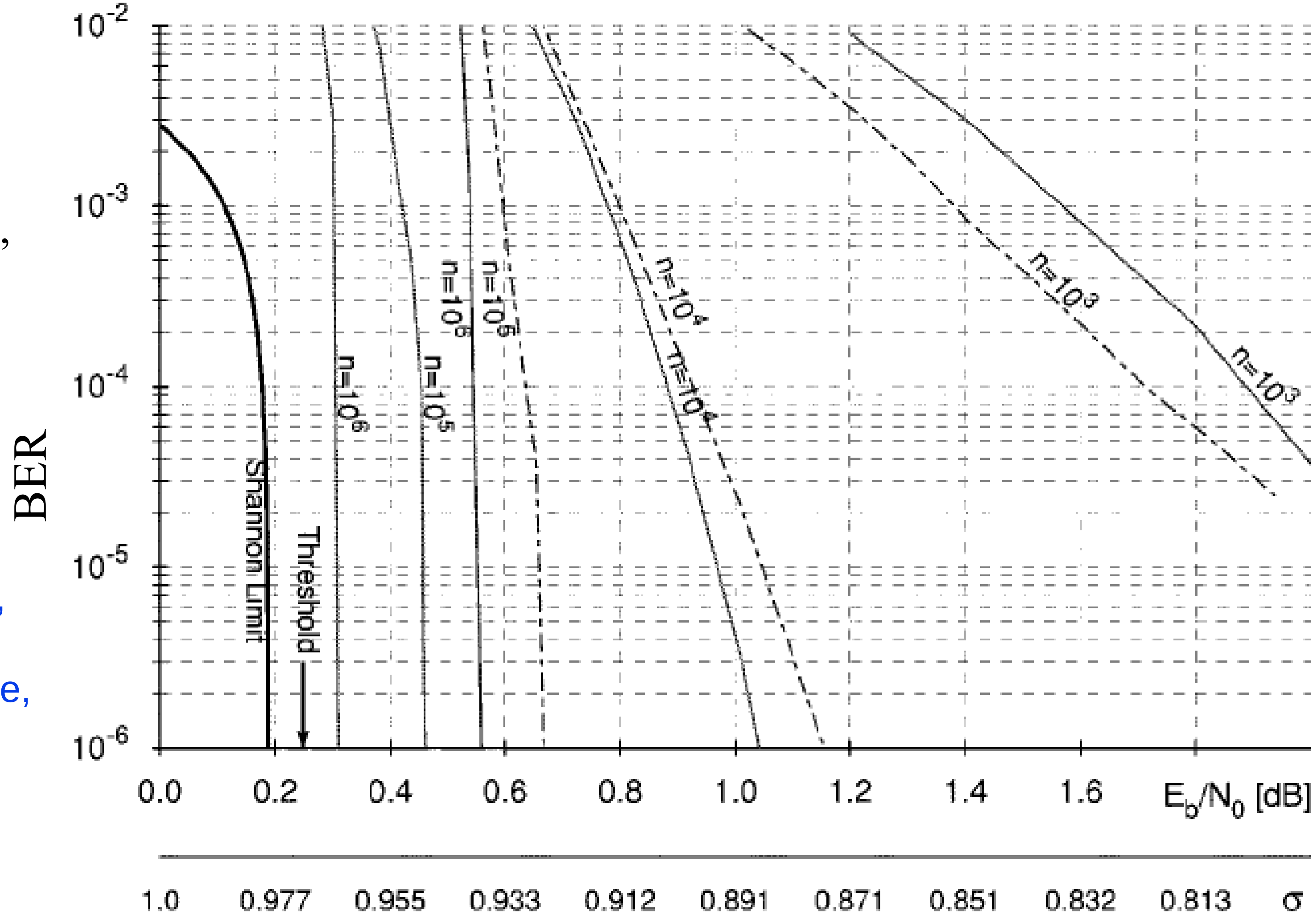
Irregular Code vs. Turbo Codes

AWGN

R=1/2

n =10^3, 10^4,

10^5, 10^6



Richardson, Shokrollahi, and Urbanke, 2001

Density Evolution

- Density evolution must track probability distributions/densities of the log-likelihood ratio messages.
- A "discretized" version of the sum-product algorithm, and associated "discretized" density evolution, speeds code design considerably.
- This design method has produced rate ½ LDPC ensembles with thresholds within 0.0045dB of the Shannon limit on the AWGN channel!
- A rate 1/2 code with block length 10^7 provided BER of 10^{-6} within 0.04 dB of the Shannon limit!

Some Really Good LDPC Codes

Chung, et al., 2001.

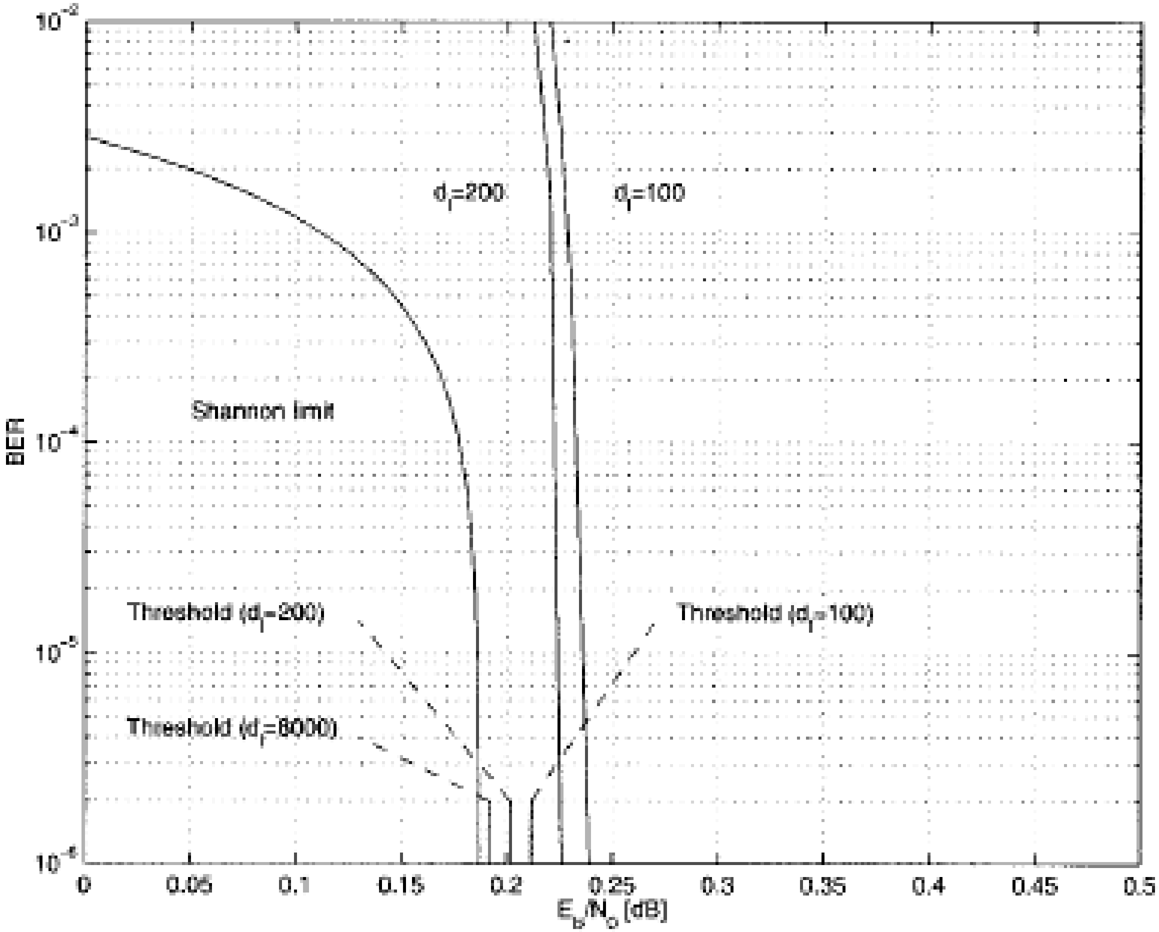
TABLE II
GOOD RATE-1/2 CODES WITH $d_l = 100, 200, 8000$

d_l	100		200		8000	
	x	λ_x	x	λ_x	x	λ_x
	2	0.170031	2	0.153425	2	0.096294
	3	0.160460	3	0.147526	3	0.095393
	6	0.112837	6	0.041539	6	0.033599
	7	0.047489	7	0.147551	7	0.091918
	10	0.011481	18	0.047938	15	0.031642
	11	0.091537	19	0.119555	20	0.086563
	26	0.152978	55	0.036379	50	0.093896
	27	0.036131	56	0.126714	70	0.006035
	100	0.217056	200	0.179373	100	0.018375
					150	0.086919
					400	0.089018
					900	0.057176
					2000	0.085816
					3000	0.006163
					6000	0.003028
					8000	0.118165
ρ_{av}	10.9375		12.0000		18.5000	
σ	0.97592		0.97704		0.9781869	
SNR_{norm}	0.0247		0.0147		0.00450	

0.0045dB from Shannon limit!

Good Code Performance

Chung, et al., 2001.



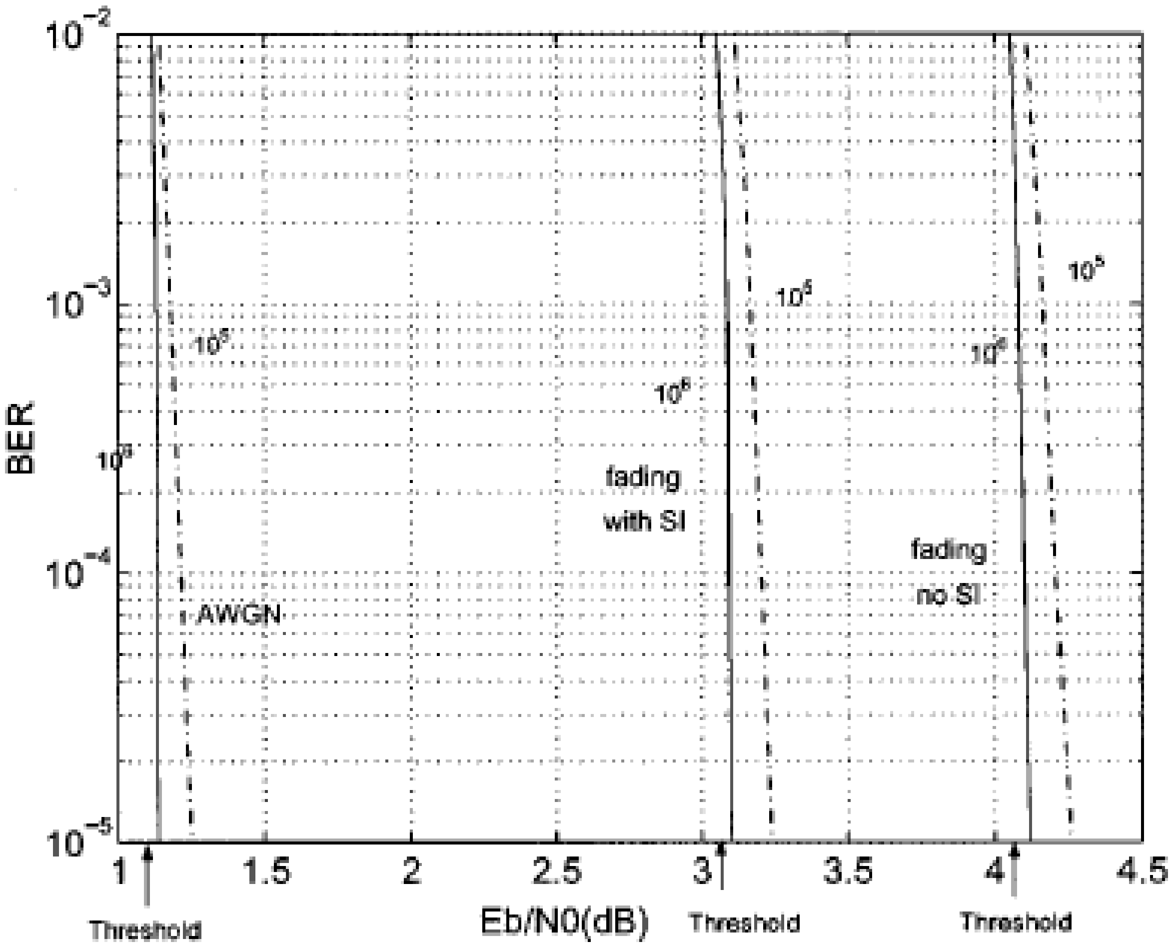
Applications of LDPC Codes

- The performance benefits that LDPC codes offer on the BEC, BSC, and AWGN channels have been shown empirically (and sometimes analytically) to extend to many other channels, including
  - Fading channels
  - Channels with memory
  - Coded modulation for bandwidth-limited channels
  - MIMO Systems

Rayleigh Fading Channels

Hou, et al.
2001

R=1/2,
(3,6)



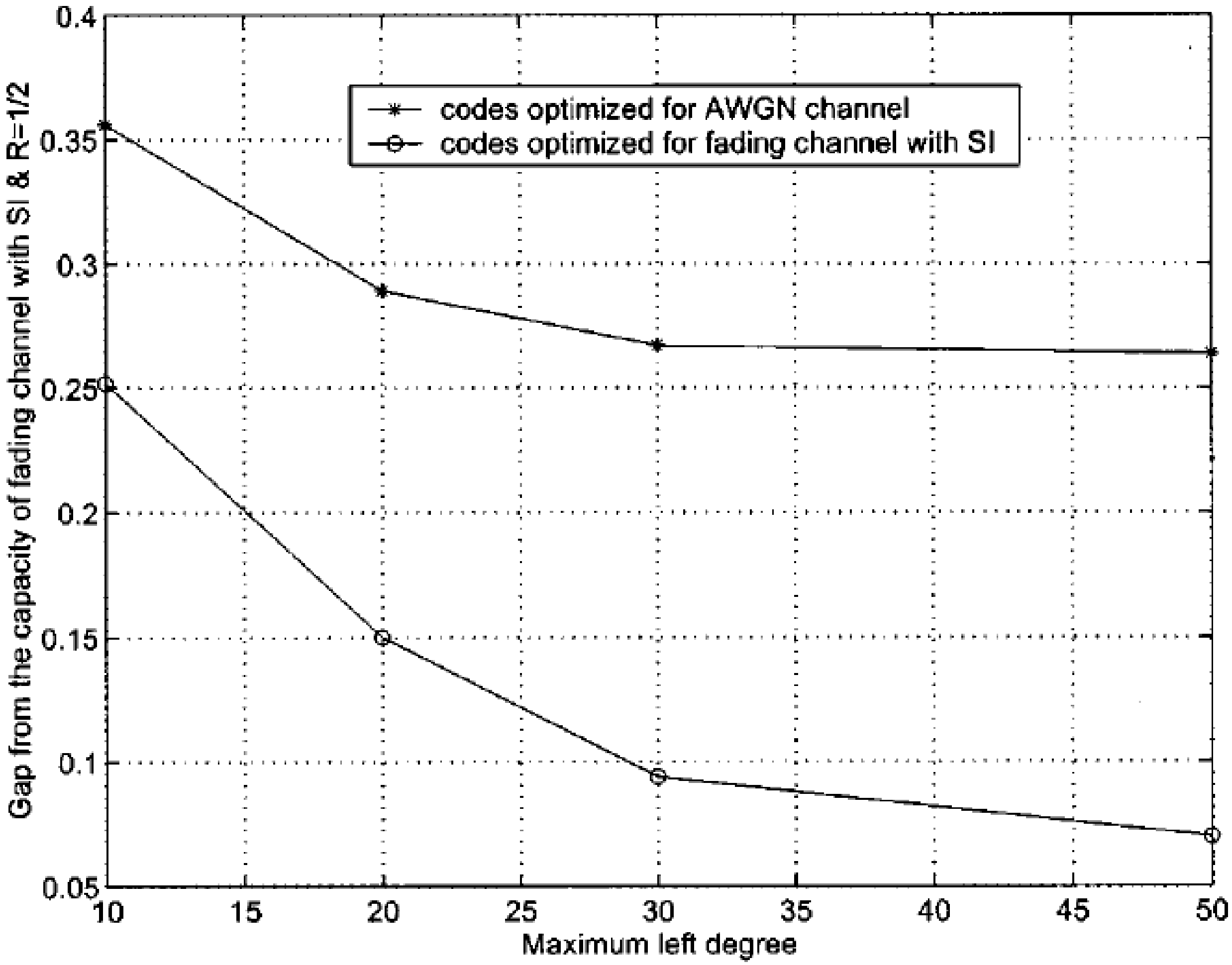
Rayleigh Fading Channels

TABLE I

GOOD DEGREE DISTRIBUTION PAIRS OF RATE-$1/2$ FOR THE UNCORRELATED RAYLEIGH FADING CHANNELS WITH SI AND WITH CONSTRAINTS ON THE MAXIMAL LEFT DEGREES $d_{l\,\max} = 10, 20, 30$ AND 50. FOR EACH DISTRIBUTION PAIR THE NOISE THRESHOLD VALUE σ^* AND THE CORRESPONDING $(E_b/N_0)^*$ (dB) ARE GIVEN. THE MAXIMAL VALUE OF λ_2 SATISFYING CONDITION (23), λ_2^*, IS GIVEN FOR $\sigma = \sigma^*$ AND THE GIVEN $\rho'(1)$. NOTE THAT THE CAPACITY FOR THIS CHANNEL AT CODE RATE $1/2$ IS 1.830 dB

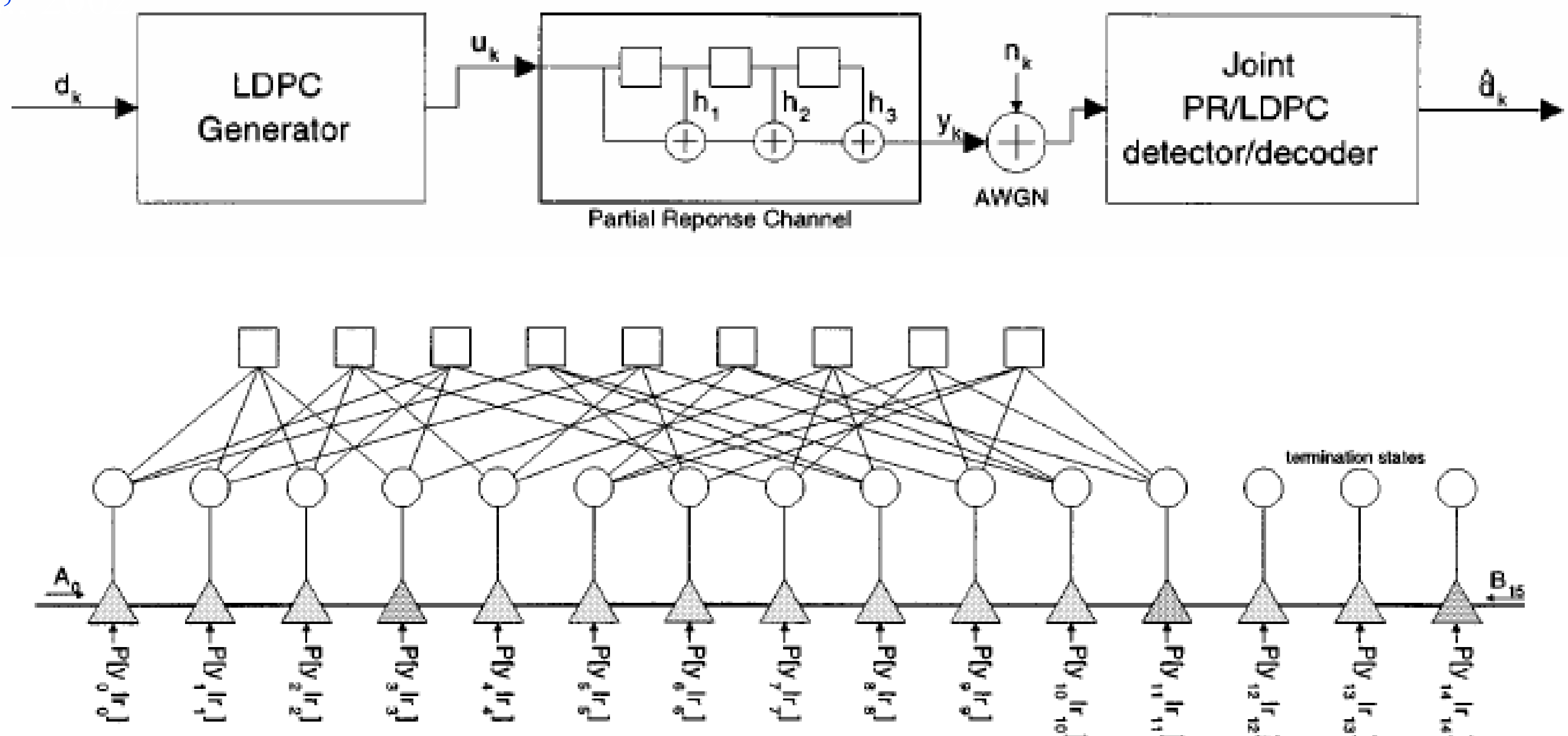
d_{lmax}	10	20	30	50
λ_2^*	0.300932	0.253856	0.229439	0.204885
λ_2	0.292439	0.246544	0.220033	0.194255
λ_3	0.253636	0.230609	0.222611	0.206322
λ_4	0.060454	0.002045	0.000100	0.000111
λ_6		0.046487	0.000919	
λ_7		0.150161	0.069962	0.092232
λ_8		0.035344	0.201925	0.111427
λ_9	0.031610			0.014172
λ_{10}	0.361861			
λ_{15}			0.000531	0.113788
λ_{19}		0.004812		
λ_{20}		0.283998		
λ_{29}			0.001791	
λ_{30}			0.282128	0.001514
λ_{49}				0.003503
λ_{50}				0.262676
ρ_6	0.007254			
ρ_7	0.979220	0.000952		
ρ_8	0.013526	0.951871	0.254080	
ρ_9		0.047177	0.739388	0.346906
ρ_{10}			0.006532	0.645429
ρ_{11}				0.007665
σ^*	0.7869	0.7962	0.8013	0.8035
$\left(\frac{E_b}{N_0}\right)^*$ dB	2.082	1.980	1.924	1.900

Rayleigh Fading Channels

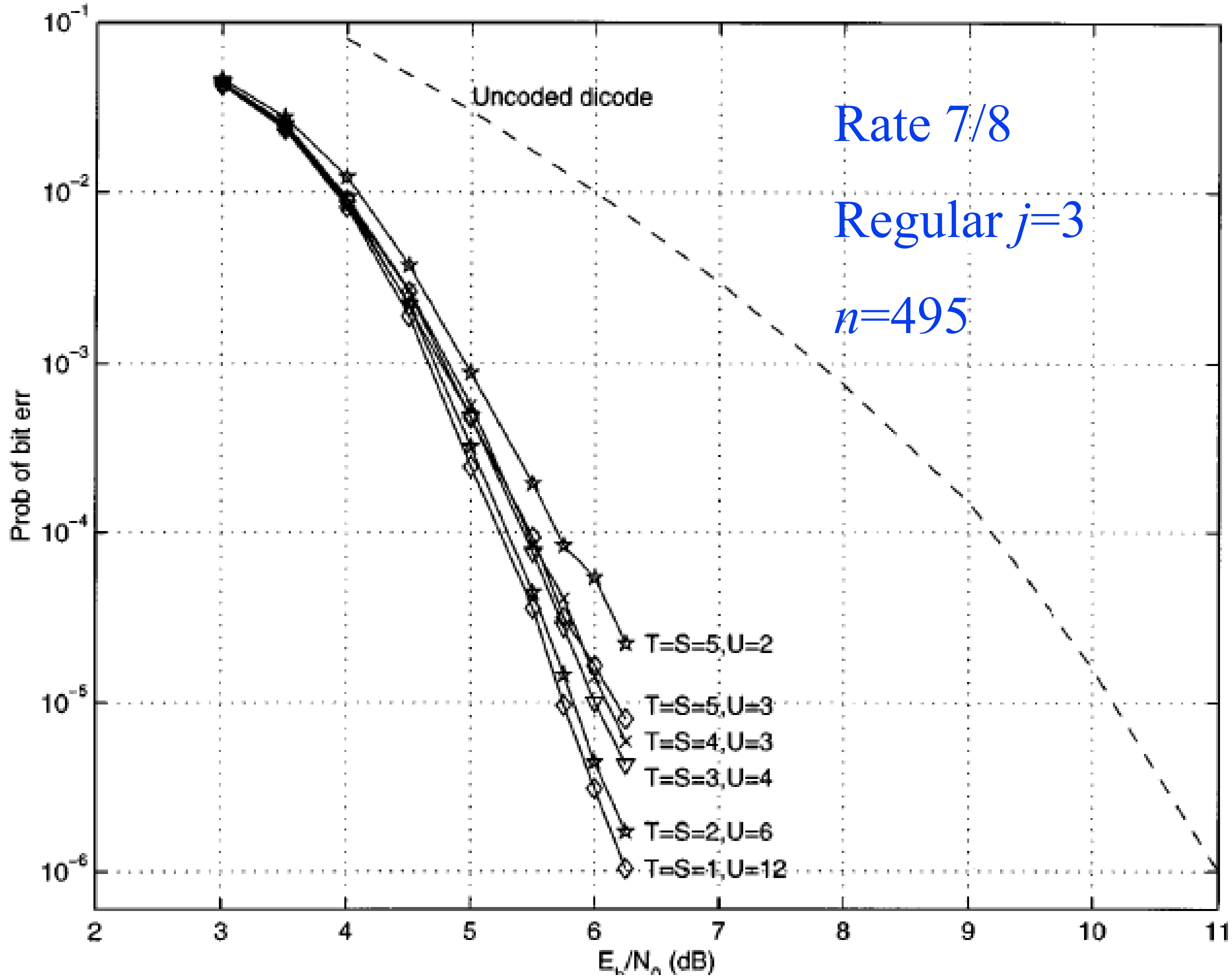


Partial-Response Channels

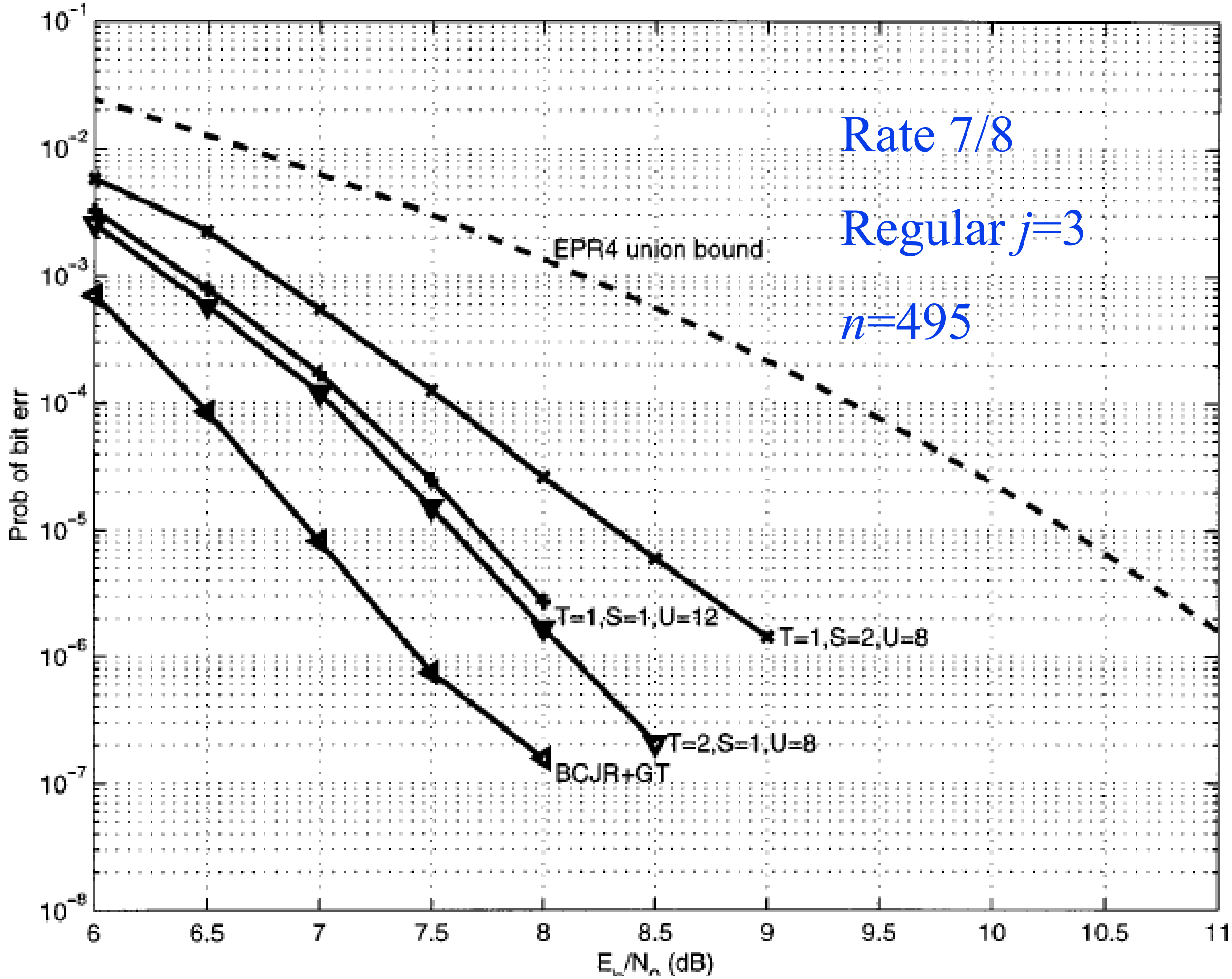
Kurkoski, et al., 2002



Dicode (1-D) Channel Results



EPR4 ($1+D-D^2-D^3$) Channel Results



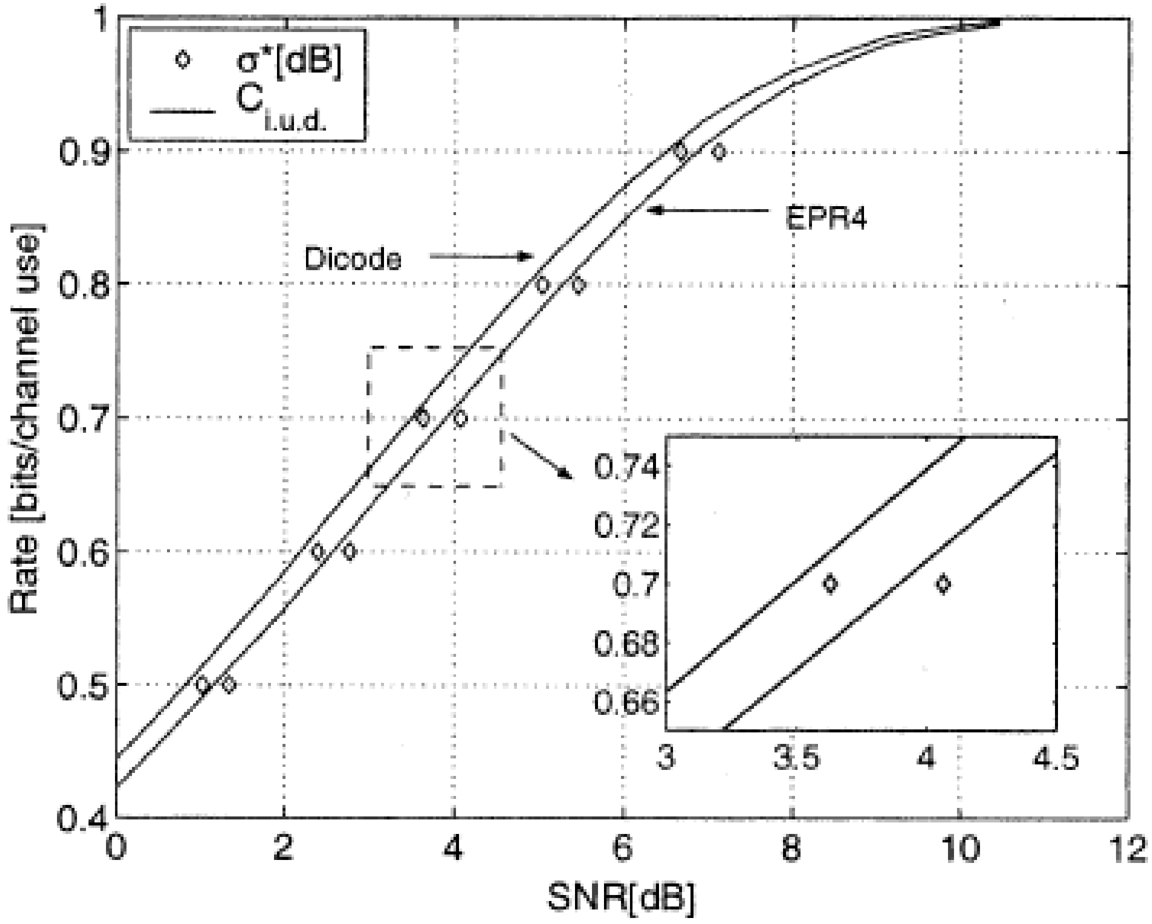
Optimized Codes for Partial Response

Varnica and Kavcic, 2003

TABLE I
GOOD DEGREE SEQUENCES FOR THE DICODE AND THE EPR4 CHANNEL

Dicode channel $r=0.7$			EPR4 channel $r=0.7$		
i	λ_i	ρ_i	i	λ_i	ρ_i
2	0.2032	0.0004	2	0.2022	
3	0.2298	0.0002	3	0.2244	
5	0.1397		4	0.0269	
6	0.0077		6	0.0417	
15		0.6252	10		0.1934
16		0.3588	11	0.0048	0.1023
30		0.0154	12	0.0007	
47	0.1271		14		0.0658
48	0.2925		23		0.3138
			24		0.3247
			42	0.1576	
			43	0.3157	
			50	0.0260	
threshold $\sigma^*=0.6586$			threshold $\sigma^*=0.6269$		
distance to $C_{i.u.d}$ 0.14dB			distance to $C_{i.u.d}$ 0.15dB		

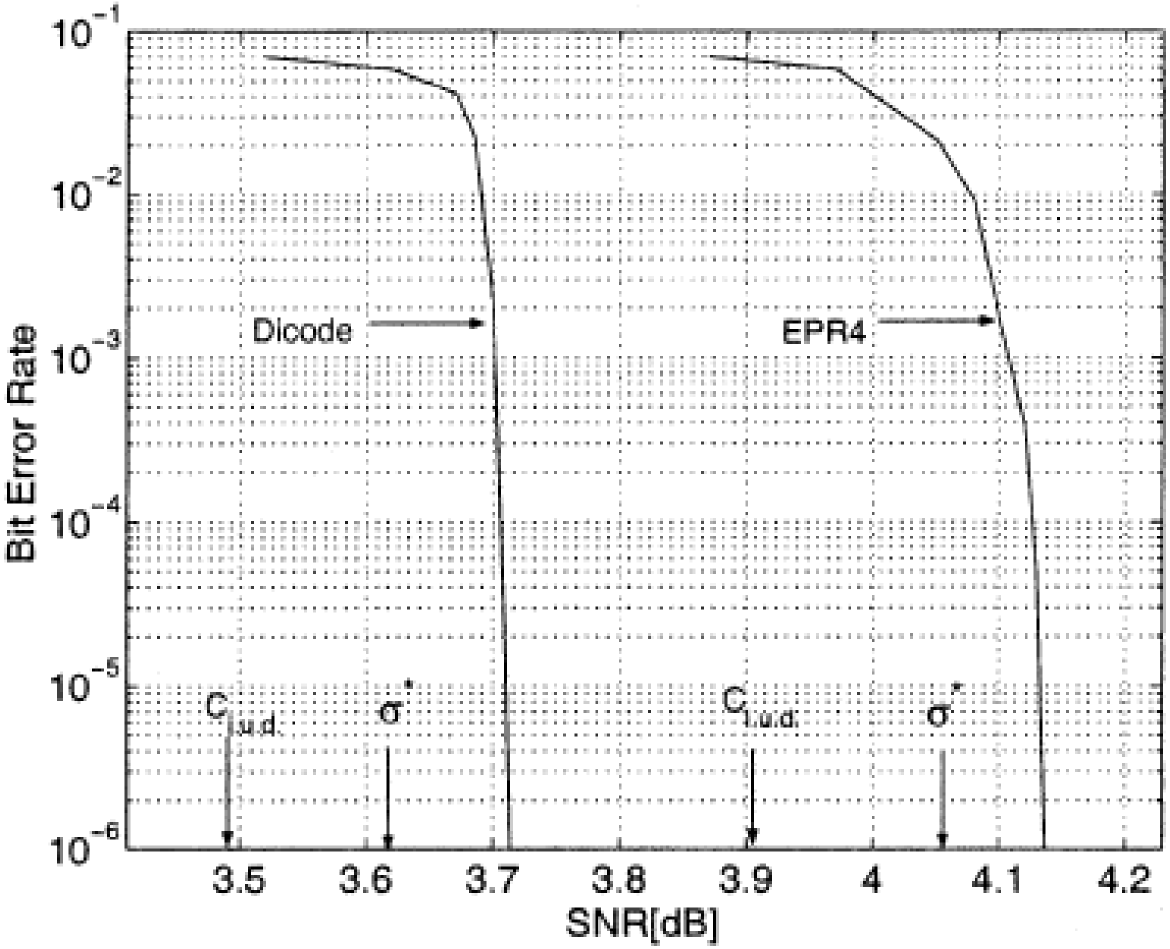
Optimized Codes for Partial Response



Optimized Codes for Partial Response

R=0.7

n=10^6



Some Basic References

- R.G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963 (Sc.D. MIT, 1960).
- T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press (Preliminary version, May 12, 2007)
- Special Issue on Codes on Graphs and Iterative Algorithms, *IEEE Transactions on Information Theory*, February 2001.
- S-Y Chung, et al., “On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit,” *IEEE Commun. Letters*, vol. 5, no. 2, pp. 58-60, February 2001.

Additional References

- J. Hou, P.H. Siegel, and L.B. Milstein, “ Performance analysis and code optimization of low density parity-check codes on Rayleigh fading channels,” *IEEE J. Select. Areas Commun.*, Issue on *The Turbo Principle: From Theory to Practice I*, vol. 19, no. 5, pp. 924-934, May 2001.

- B.M. Kurkoski, P.H. Siegel, and J.K. Wolf, “Joint message passing decoding of LCPC coded partial-response channels,” *IEEE Trans. Inform. Theory,* vol. 48, no. 6, pp. 1410-1422, June 2002. (See also B.M. Kurkoski, P.H. Siegel, and J.K. Wolf, “Correction to “Joint message passing decoding of LCPC coded partial-response channels”,” *IEEE Trans. Inform. Theory,* vol. 49, no. 8, p. 2076, August 2003.)

- N. Varnica and A. Kavcic, “Optimized LDPC codes for partial response channels,” *IEEE Communications Letters*, vol. 7, pp. 168-170, April 2003.

Concluding Remarks

- LDPC codes are very powerful codes with enormous practical potential, founded upon deep and rich theory.
- There continue to be important advances in all of the key aspects of LDPC code design, analysis, and implementation.
- LDPC codes are now finding their way into many applications:
  - Satellite broadcast
  - Cellular wireless
  - Data storage
  - And many more …