AN INTRODUCTION TO ERROR CORRECTING CODES
Part 3

Jack Keil Wolf

ECE 154 C

Spring 2010

Introduction to LDPC Codes

- **These codes were invented by Gallager in his Ph.D. dissertation at M.I.T. in 1960.**
- **They were ignored for many years since they were thought to be impractical.**
- **But with present day technology they are very practical.**
- **Their performance is similar to turbo codes but they may have some implementation advantages.**

Outline: Some Questions

- What is a parity check code?
- What is an LDPC code?
- What is a message passing decoder for LDPC codes?
- What is the performance of these codes?

What is a Parity Check Code?

- **A binary parity check code is a block code: i.e., a collection of binary vectors of fixed length n.**

- **The symbols in the code satisfy r parity check equations of the form:**

$$x_a \oplus x_b \oplus x_c \oplus \ldots \oplus x_z = 0$$

where $\oplus$ means modulo 2 addition and

$$x_a, x_b, x_c, \ldots, x_z$$

are the code symbols in the equation.

- **Each codeword of length n can contain (n-r)=k information digits and r check digits.**

What is a Parity Check Matrix?

- **A parity check matrix is an r-row by n-column binary matrix. Remember k=n-r.**
- **The rows represent the equations and the columns represent the digits in the code word.**
- **There is a 1 in the i-th row and j-th column if and only if the i-th code digit is contained in the j-th equation.**

Example: Hamming Code with n=7, k=4, and r=3

- For a code word of the form $c_1, c_2, c_3, c_4, c_5, c_6, c_7$, the equations are:

$$c_1 \oplus c_2 \oplus c_3 \oplus c_5 = 0$$
$$c_1 \oplus c_2 \oplus c_4 \oplus c_6 = 0$$
$$c_1 \oplus c_3 \oplus c_4 \oplus c_7 = 0.$$

- The parity check matrix for this code is then:

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

- Note that c_1 is contained in all three equations while c_2 is contained in only the first two equations.

What is an LDPC Code?

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

The Equations for A Simple LDPC Code with n=12

$$c_3 \oplus c_6 \oplus c_7 \oplus c_8 = 0$$
$$c_1 \oplus c_2 \oplus c_5 \oplus c_{12} = 0$$
$$c_4 \oplus c_9 \oplus c_{10} \oplus c_{11} = 0$$
$$c_2 \oplus c_6 \oplus c_7 \oplus c_{10} = 0$$
$$c_1 \oplus c_3 \oplus c_8 \oplus c_{11} = 0$$
$$c_4 \oplus c_5 \oplus c_9 \oplus c_{12} = 0$$
$$c_1 \oplus c_4 \oplus c_5 \oplus c_7 = 0$$
$$c_6 \oplus c_8 \oplus c_{11} \oplus c_{12} = 0$$
$$c_2 \oplus c_3 \oplus c_9 \oplus c_{10} = 0.$$

- **There are actually only 7 independent equations so there are 7 parity digits**.

The Parity Check Matrix for the Simple LDPC Code

$c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8\ c_9\ c_{10}\ c_{11}\ c_{12}$

0	0	1	0	0	1	1	1	0	0	0	0	$c_3 \oplus c_6 \oplus c_7 \oplus c_8 = 0$
1	1	0	0	1	0	0	0	0	0	0	1	$c_1 \oplus c_2 \oplus c_5 \oplus c_{12} = 0$
0	0	0	1	0	0	0	0	1	1	1	0	$c_4 \oplus c_9 \oplus c_{10} \oplus c_{11} = 0$
0	1	0	0	0	1	1	0	0	1	0	0	$c_2 \oplus c_6 \oplus c_7 \oplus c_{10} = 0$
1	0	1	0	0	0	0	1	0	0	1	0	$c_1 \oplus c_3 \oplus c_8 \oplus c_{11} = 0$
0	0	0	1	1	0	0	0	1	0	0	1	$c_4 \oplus c_5 \oplus c_9 \oplus c_{12} = 0$
1	0	0	1	1	0	1	0	0	0	0	0	$c_1 \oplus c_4 \oplus c_5 \oplus c_7 = 0$
0	0	0	0	0	1	0	1	0	0	1	1	$c_6 \oplus c_8 \oplus c_{11} \oplus c_{12} = 0$
0	1	1	0	0	0	0	0	1	1	0	0	$c_2 \oplus c_3 \oplus c_9 \oplus c_{10} = 0$

The Parity Check Matrix for the Simple LDPC Code

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

- **Note that each code symbol is contained in 3 equations and each equation involves 4 code symbols.**

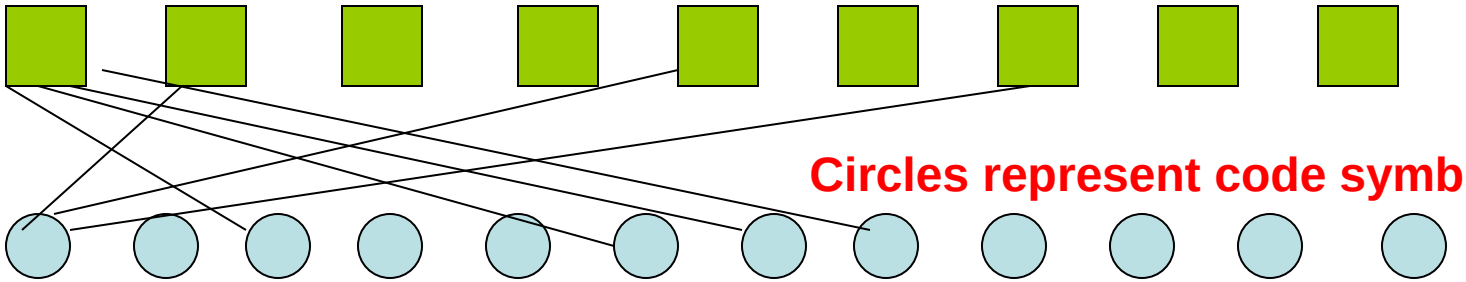
A Graphical Description of LDPC Codes

- **Decoding of LDPC codes is best understood by a graphical description.**
- **The graph has two types of nodes: bit nodes and parity nodes.**
- **Each bit node represents a code symbol and each parity node represents a parity equation.**
- **There is a line drawn between a bit node and a parity node if and only if that bit is involved in that parity equation.**

The Graph for the Simple LDPC Code

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

Squares represent parity equations.

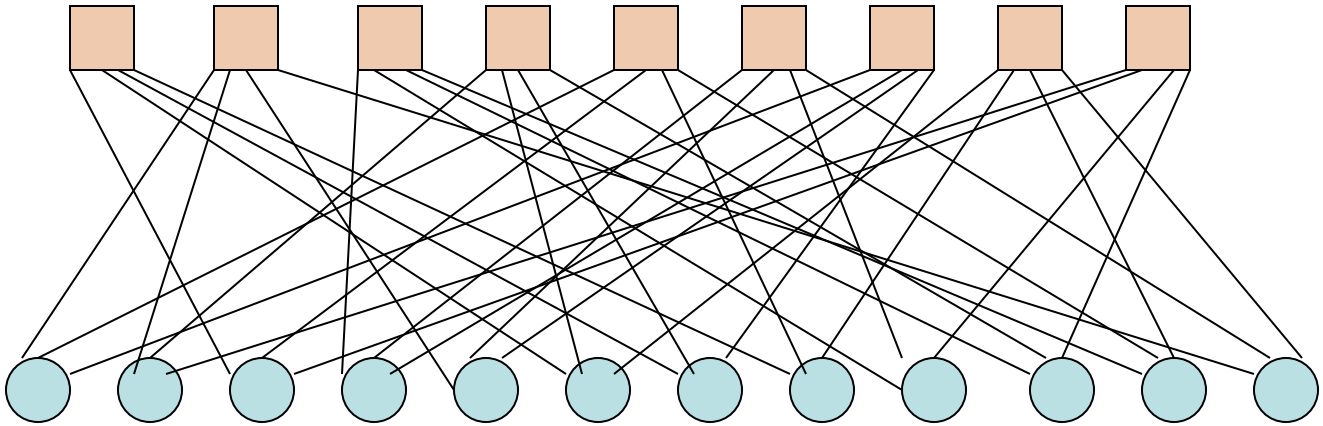


Circles represent code symbols.

Only the lines corresponding to the 1st row and 1st column are shown.

Entire Graph for the Simple LDPC Code

- **Note that each bit node has 3 lines connecting it to parity nodes and each parity node has 4 lines connecting it to bit nodes.**



Decoding of LDPC Codes by Message Passing on the Graph

- **Decoding is accomplished by passing messages along the lines of the graph.**

- **The messages on the lines that connect to the i-th bit node, c_i, are estimates of $Pr[c_i = 1]$ (or some equivalent information).**

- **At the nodes the various estimates are combined in a particular way.**

Decoding of LDPC Codes by Message Passing on the Graph

- **Each bit node is furnished an initial estimate of the probability it is a 1 from the soft output of the channel.**

- **The bit node broadcasts this initial estimate to the parity nodes on the lines connected to that bit node.**

- **But each parity node must make new estimates for the bits involved in that parity equation and send these new estimates (on the lines) back to the bit nodes.**

Estimation of Probabilities by Parity Nodes

- **Each parity node knows that there are an even number of 1's in the bits connected to that node.**
- **But the parity node has received estimates of the probability that each bit node connected to it is a 1.**
- **The parity node sends a new estimate to the i-th bit node based upon all the other probabilities furnished to it.**

Estimation of Probabilities by Parity Nodes

- For example, consider the parity node corresponding to the equation $c_3 \oplus c_6 \oplus c_7 \oplus c_8 = 0$.

- This parity node has the estimates p_3, p_6, p_7, and p_8 corresponding to the bit nodes c_3, c_6, c_7, and c_8, where p_i is an estimate for $Pr[c_i=1]$.

- The new estimate for the bit node c_3 is:

$$p'_3=p_6(1-p_7)(1-p_8)+ p_7(1-p_6)(1-p_8)+ p_8(1-p_6)(1-p_7)+ p_6p_7p_8$$

and for the other nodes:

$$p'_6=p_3(1-p_7)(1-p_8)+ p_7(1-p_3)(1-p_8)+ p_8(1-p_3)(1-p_7)+ p_3p_7p_8$$
$$p'_7=p_6(1-p_3)(1-p_8)+ p_3(1-p_6)(1-p_8)+ p_8(1-p_3)(1-p_6)+ p_3p_6p_8$$
$$p'_8=p_6(1-p_7)(1-p_3)+ p_7(1-p_6)(1-p_3)+ p_3(1-p_6)(1-p_7)+ p_3p_6p_7$$

Estimation of Probabilities by Bit Nodes

- But the bit nodes are provided different estimates of Pr[c=1] by the channel and by each of the parity nodes connected to it.
- It no longer broadcasts a single estimate but sends different estimates to each parity equation.
- The new estimate sent to each parity node is obtained by combining all other current estimates.
- That is, in determining the new estimate sent to a parity node, it ignores the estimate received from that parity node.

Estimation of Probabilities by Bit Nodes

- **The new estimate sent to each parity node is equal to the normalized product of the other estimates.**
- **The proper normalization is a detail which will be discussed later.**
- **If instead of passing estimates of Pr[c=1] we pass estimates of log {Pr[c=1]/Pr[c=0]} where Pr[c=0] = 1 - Pr[c=1], we merely need to add the appropriate terms.**
- **The channel estimate is always used in all estimates passed to the parity node.**

Estimation of Probabilities by Bit Nodes

- **The following table illustrates how estimates are combined by a bit node involved in 3 parity equations A, B, and C.**

Estimate received from channel:	p_{ch}
Estimate received from parity node A:	p_A
Estimate received from parity node B:	p_B
Estimate received from parity node C:	p_C
New estimate sent to parity node A:	$K\, p_{ch}\, p_B\, p_C$
New estimate sent to parity node B:	$K\, p_{ch} p_A\, p_C$
New estimate sent to parity node C:	$K\, p_{ch} p_A\, p_B$

The Rest of the Decoding Algorithm

- **The process now repeats: parity nodes passing messages to bit nodes and bit nodes passing messages to parity nodes.**

- **At the last step, a final estimate is computed at each bit node by computing the normalized product of all of its estimates.**

- **Then a hard decision is made on each bit by comparing the final estimate with the threshold 0.5.**

Final Estimate Made by Bit Nodes

- **The following table illustrates how the final estimate is made by a bit node involved in 3 parity equations A, B, and C.**

Estimate received from channel:	p_{ch}
Estimate received from parity node A:	p_A
Estimate received from parity node B:	p_B
Estimate received from parity node C:	p_C
FINAL ESTIMATE:	$K\, p_{ch} p_A\, p_B\, p_C$

Decoding of Simple Example

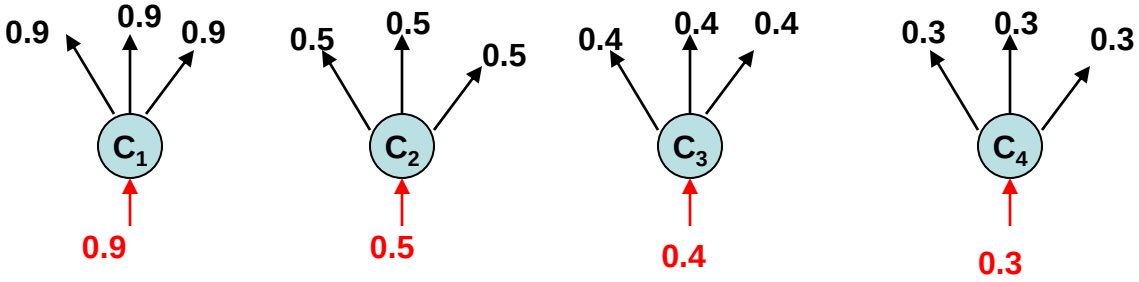
- **Suppose the following Pr[C_i=1], i=1, 2, …, 12 are obtained from channel:**

  0.9 0.5 0.4 0.3 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9

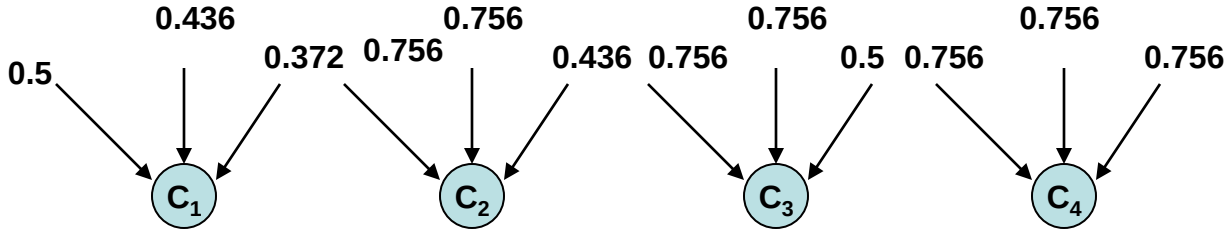
- **We now watch the decoder decode.**

Decoding of Simple Example: First 4 Bit Nodes Only

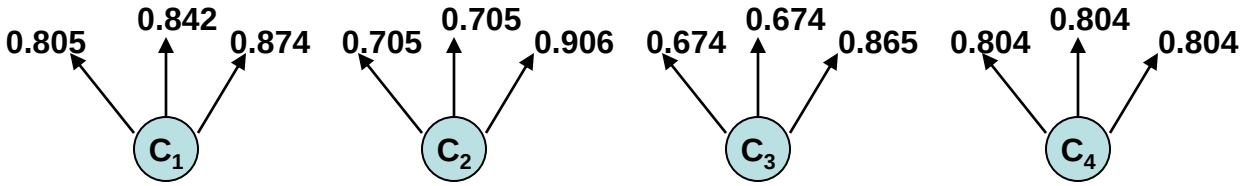
- **Initial broadcast from first 4 bit nodes:**



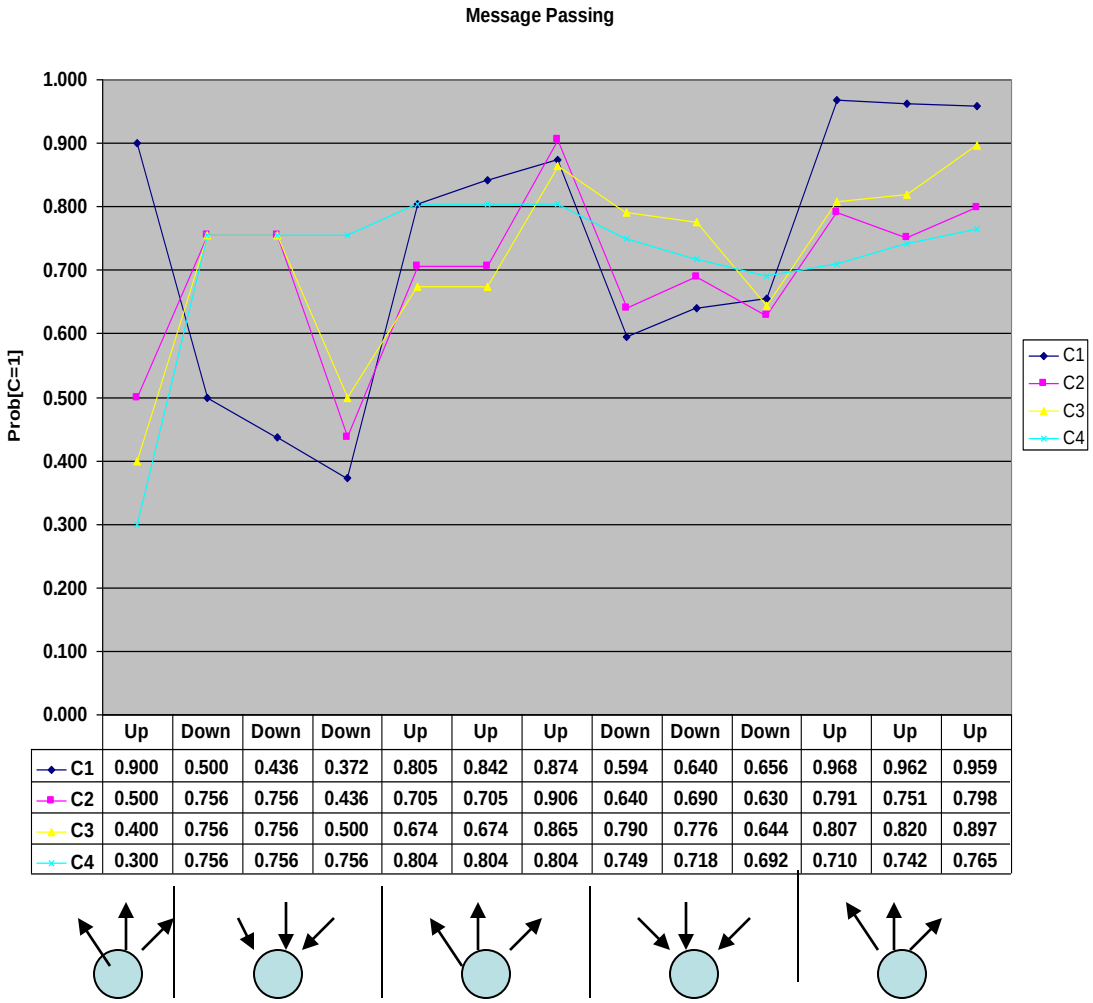
- **Transmission from parity nodes to these 4 bit nodes:**



- **Next transmission from the first 4 bit nodes:**



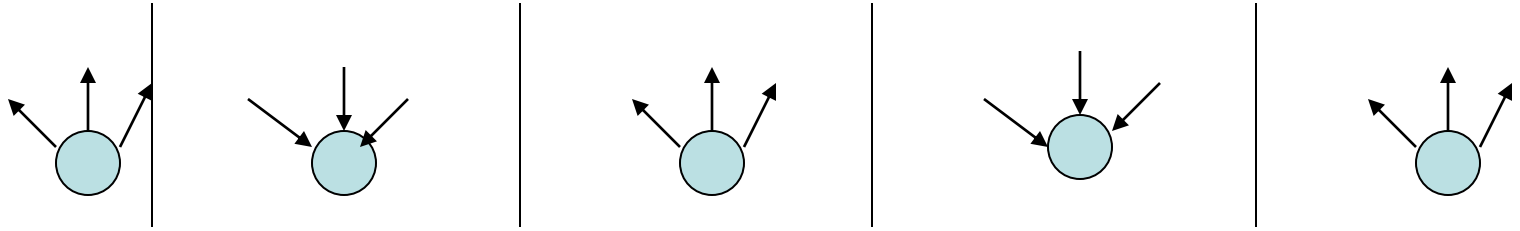
Message Passing for First 4 Bit Nodes for More Iterations



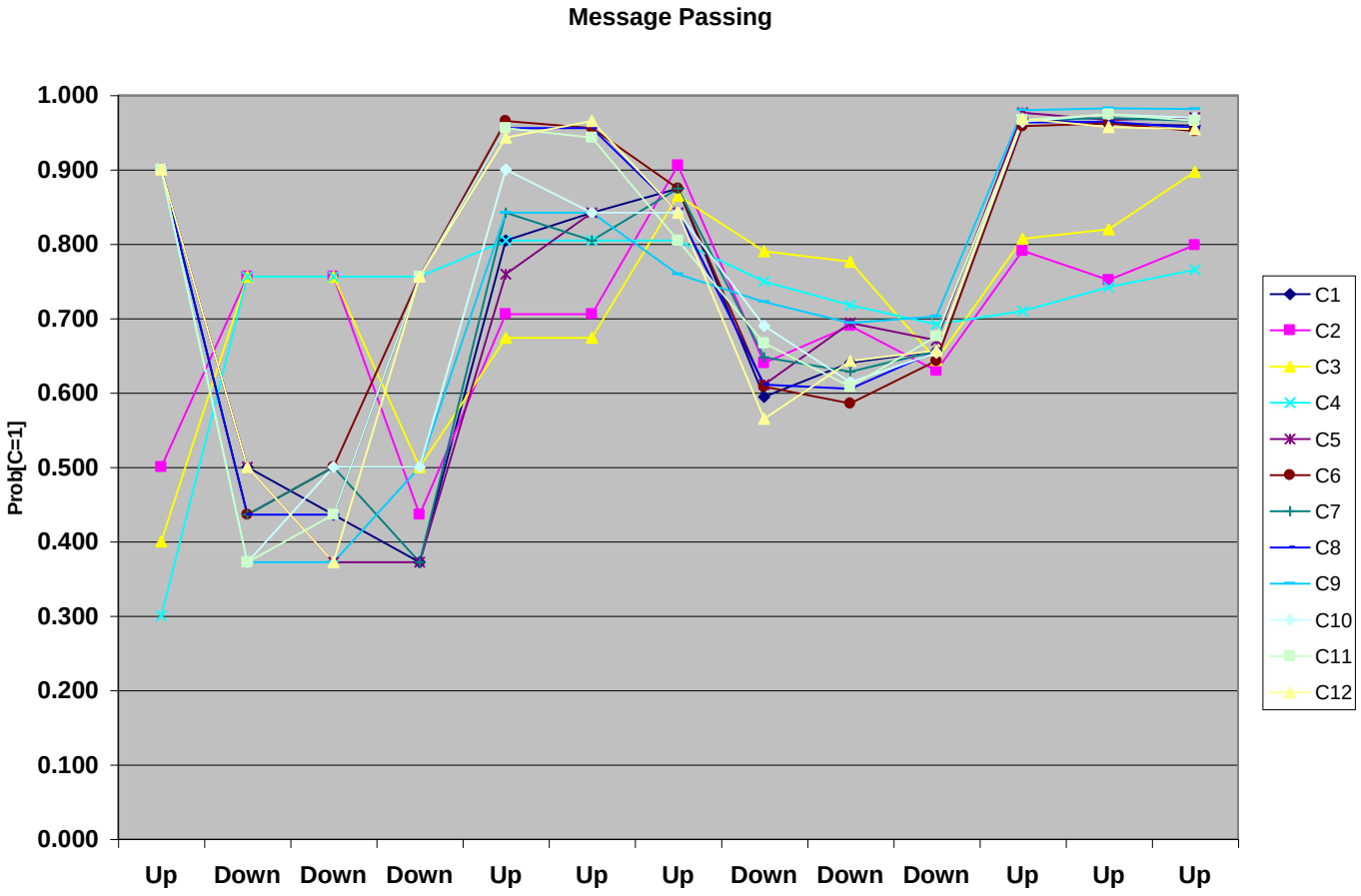
	Up	Down	Down	Down	Up	Up	Up	Down	Down	Down	Up	Up	Up
C1	0.900	0.500	0.436	0.372	0.805	0.842	0.874	0.594	0.640	0.656	0.968	0.962	0.959
C2	0.500	0.756	0.756	0.436	0.705	0.705	0.906	0.640	0.690	0.630	0.791	0.751	0.798
C3	0.400	0.756	0.756	0.500	0.674	0.674	0.865	0.790	0.776	0.644	0.807	0.820	0.897
C4	0.300	0.756	0.756	0.756	0.804	0.804	0.804	0.749	0.718	0.692	0.710	0.742	0.765

Messages Passed To and From All 12 Bit Nodes

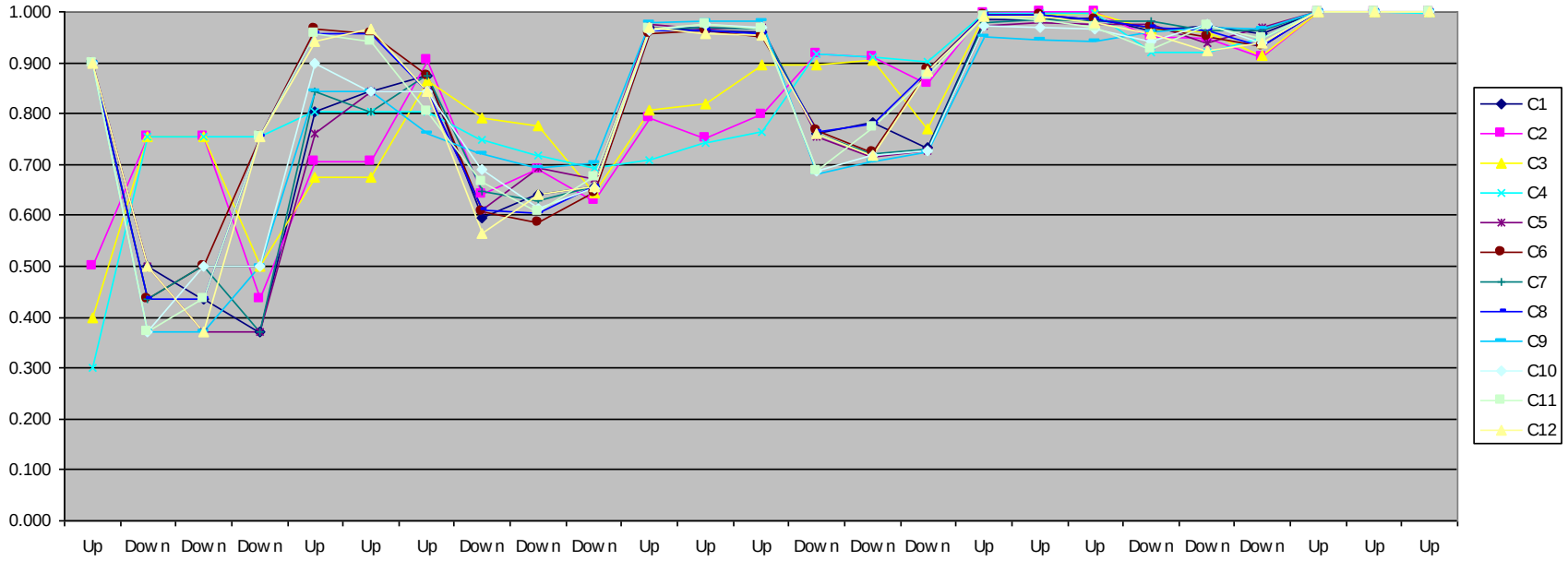
	Up	Down	Down	Down	Up	Up	Up	Down	Down	Down	Up	Up	Up
C1	0.900	0.500	0.436	0.372	0.805	0.842	0.874	0.594	0.640	0.656	0.968	0.962	0.959
C2	0.500	0.756	0.756	0.436	0.705	0.705	0.906	0.640	0.690	0.630	0.791	0.751	0.798
C3	0.400	0.756	0.756	0.500	0.674	0.674	0.865	0.790	0.776	0.644	0.807	0.820	0.897
C4	0.300	0.756	0.756	0.756	0.804	0.804	0.804	0.749	0.718	0.692	0.710	0.742	0.765
C5	0.900	0.500	0.372	0.372	0.759	0.842	0.842	0.611	0.694	0.671	0.976	0.966	0.970
C6	0.900	0.436	0.500	0.756	0.965	0.956	0.874	0.608	0.586	0.643	0.958	0.962	0.952
C7	0.900	0.436	0.500	0.372	0.842	0.805	0.874	0.647	0.628	0.656	0.967	0.969	0.965
C8	0.900	0.436	0.436	0.756	0.956	0.956	0.843	0.611	0.605	0.656	0.963	0.964	0.956
C9	0.900	0.372	0.372	0.500	0.842	0.842	0.759	0.722	0.694	0.703	0.980	0.982	0.981
C10	0.900	0.372	0.500	0.500	0.900	0.842	0.842	0.690	0.614	0.654	0.964	0.974	0.970
C11	0.900	0.372	0.436	0.756	0.956	0.943	0.805	0.667	0.608	0.676	0.967	0.974	0.965
C12	0.900	0.500	0.372	0.756	0.943	0.965	0.842	0.565	0.642	0.657	0.969	0.957	0.955



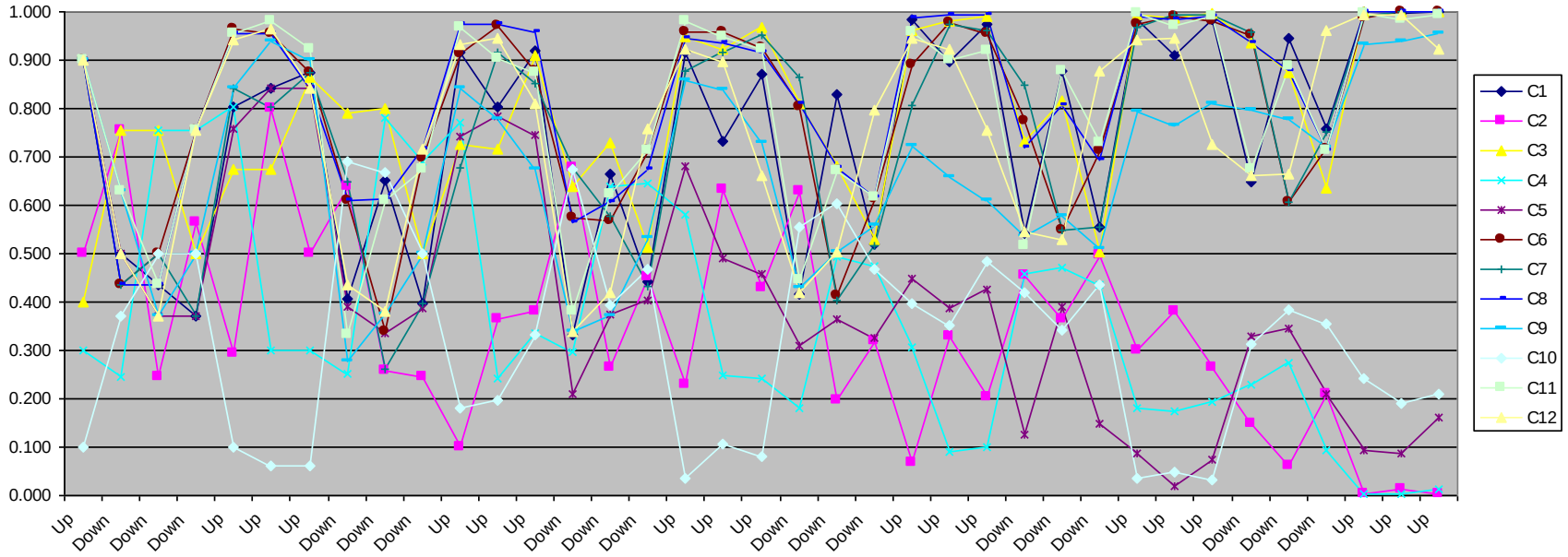
Messages Passed To and From All 12 Bit Nodes



More Iterations
All 12 Bit Nodes



More Interesting Example
All 12 Bit Nodes



Computation at Bit Nodes

- **If estimates of probabilities are statistically independent you should multiply them.**

- **But you need to normalize the product. Otherwise the product is smaller than every single estimate.**

- **For example, with three independent estimates all equal to 0.9, the unnormalized product is:**

$$(0.9)^3 = 0.729$$

where the correct normalized product is:

$$(0.9)^3 \; / \; [(0.1)^3 + (0.9)^3] = 0.9986$$

Derivation of Correct Normalization

- Assume we have 3 independent estimates, p_a, p_b, and p_c from which we compute the new estimate p' from the formula:

$$p' = K\, p_a\, p_b\, p_c.$$

- But the same normalization must hold for (1-p'):

$$(1-p') = K(1- p_a)(1- p_b)(1- p_c)$$

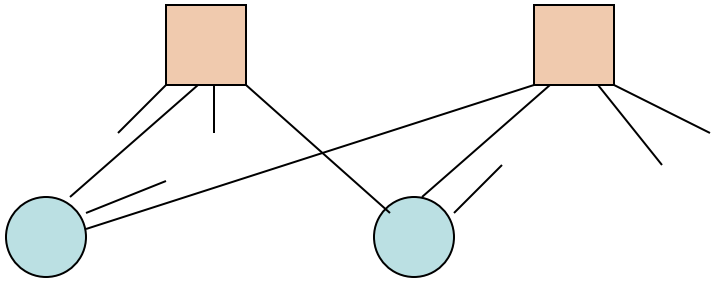
- From the first equation $(1-p') = 1- K\, p_a\, p_b\, p_c$.

- Setting $(1- K\, p_a\, p_b\, p_c)$ equal to $K(1- p_a)(1- p_b)(1- p_c)$ and solving for K we obtain:

$$K = 1 / [(1- p_a)(1- p_b)(1- p_c) + p_a\, p_b\, p_c]$$

Assumption of Independence

- **Note that in our example, parts of the graph looks like:**



- **This is called a cycle of length 4.**
- **Cycles cause estimates to be dependent and our combining formulas are incorrect.**
- **As a result short cycles should be avoided in the design of codes.**

Computation at Parity Nodes

- **When a parity equation involves many bits, an alternative formula is used.**

- **Details are omitted here but can be found in the literature.**

Rate of a Regular LDPC Code

- **Assume a LDPC is designed where:**

  (1) every bit is in J parity checks, and
  (2) every parity check checks K bits.

- **Since the number of 1's in a parity check matrix is the same whether we count by rows or columns, we have**

  J (# of columns) = K (# of rows)
  or J (n) = K (n-k).

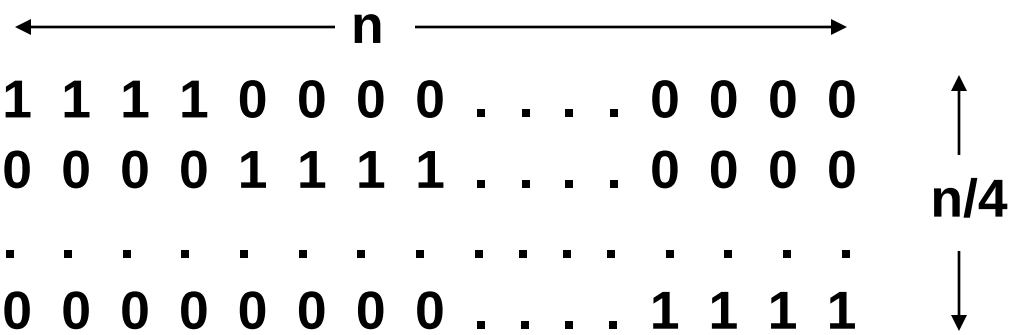
- **Solving for k / n, we have k/n = (1- J / K), the rate of the code.**

- **Higher rate codes can be obtained by puncturing lower rate codes.**

Design of a Parity Matrix for a Regular LDPC Code

- **The following procedure was suggested by Gallager. We illustrate it for a code with J = 3 and K = 4 .**

1. Construct the first n/4 rows as follows:



2. Construct the next n/4 rows by permuting the columns of the first n/4 rows.

3. Repeat 2 using another permutation of the columns.

Irregular LDPC Codes

- **Irregular LDPC codes have a variable number of 1's in the rows and in the columns.**
- **The optimal distributions for the rows and the columns are found by a technique called density evolution.**
- **Irregular LDPC codes perform better than regular LDPC codes.**
- **The basic idea is to give greater protection to some digits and to have some of the parity equations give more reliable information to give the decoding a jump start .**

Paper on Irregular LDPC Codes Luby et al (ISIT 1998)

ISIT 1998, Cambridge, MA, USA, August 16 – August 21

Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation

Michael G. Luby[1]
International Computer Science Institute
Berkeley, CA
Email luby@icsi.berkeley.edu

Michael Mitzenmacher
Digital Systems Research Center
Palo Alto, CA
Email michaelm@pa.dec.com

M. Amin Shokrollahi
International Computer Science Institute
Berkeley, CA
Email amin@icsi.berkeley.edu

Daniel A. Spielman
Department of Mathematics, M.I.T.
Cambridge, MA
Email spielman@math.mit.edu

Abstract — **We construct new families of low-density parity-check codes, which we call *irregular codes*. When decoded using belief propagation, our codes can correct more errors than previously known low-density codes. Our improved performance comes from using codes based on irregular random bipartite graphs, based on the work of [2]. Previously studied low-density codes have been derived from regular bipartite graphs. Initial experimental results for our irregular codes suggest that, with improvements, irregular codes may be able to match turbo code performance.**

I. Introduction

We have constructed new families of low-density codes, which we call *irregular codes*. The terminology comes from the fact that the parity-check matrices of our codes yield highly irregular bipartite graphs. These codes have significantly improved performance over previously known codes of this type. Using belief propagation, they can correct a substantially higher number of errors, albeit at the expense of a slightly slower running time. Further information can be found in an extended version of this paper, available as TR-97-044 of the International Computer Science Institute in Berkeley.

II. Irregular Graphs: Intuition and Example

We offer some intuition as to why using irregular graphs should improve performance. Consider trying to build a regular low-density code that transmits at a fixed rate. From the point of view of a message node, it is best to have high degree, since the more information it gets from its check nodes, the more accurately it can judge what its correct value should be. In contrast, from the point of view of a check node, it is best to have low degree, since the lower the degree of a check node, the more valuable the information it can transmit to its neighbors. These two competing requirements must be appropriately balanced. Previous work has shown that for regular graphs, low degree graphs yield the best performance [1]. If one allows irregular graphs, however, there is significantly more flexibility in balancing these competing requirements. Message nodes with high degree will tend to their correct value quickly. These nodes then provide good information to the check nodes, which subsequently provide better information to lower degree message nodes. Irregular graph constructions thus lead to a wave effect, where high degree message nodes tend to get corrected first, and then message nodes with slightly smaller degree, and so on down the line.

[1] Parts of this research were done while at Digital Systems Research Center. Research partially supported by NSF operating grant NCR-9416101.

Code Rate 1/2	
Left Degrees	$\lambda_3 = 0.44506$, $\lambda_4 = 0.26704$, $\lambda_6 = 0.14835$, $\lambda_{17} = 0.07854$, $\lambda_{33} = 0.04046$, $\lambda_{64} = 0.02055$
Right Degrees	$\rho_7 = 0.35282$, $\rho_8 = 0.29548$, $\rho_{10} = 0.10225$, $\rho_{20} = 0.18321$, $\rho_{64} = 0.04179$, $\rho_{65} = 0.02445$

Table 1: Sample code parameters.

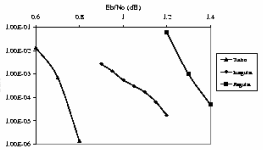


Figure 1: From left to right, rate 1/2 turbo codes, irregular codes, and regular codes.

This intuition (which we observe in our experiments) unfortunately does not provide clues as to how to construct appropriate irregular graphs. Moreover, because belief propagation is not yet well understood mathematically, creating the proper irregular graphs appears a daunting challenge. We meet this challenge by using irregular graphs that have been proven to be effective for erasure codes that function in a similar manner [2]. In the area of erasure codes, the mathematical framework has been established to both design irregular graphs and prove their effectiveness.

We provide an example of the irregular graphs used in Table 1. In the table, λ_i (ρ_i) denotes the fraction of nodes of degree i on the left (right) hand side of the graph. Note that given a vector λ and ρ one can construct a graph with (approximately) the correct node fractions for any number of nodes.

References

[1] D. J. C. MacKay and R. M. Neal, "Good Error Correcting Codes Based on Very Sparse Matrices," available from http://wol.ra.phy.cam.ac.uk/mackay.

[2] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical Loss-Resilient Codes," *Proc. 29th Symp. on Theory of Computing*, 1997, pp. 150–159.

Paper on Irregular LDPC Codes Luby et al (ISIT 1998)

- Code Rate ½

- Left degrees

  Λ_3 =.44506 λ_5 =. 26704
  Λ_9 =.14835 λ_{17}=. 07854
  λ_{33}=.04046 λ_{65}=. 02055

- Right degrees

  ρ_7 =.38282 ρ_8 =. 29548
  ρ_{19}=.10225 ρ_{20}=. 18321
  ρ_{84}=.04179 ρ_{85}=. 02445

Code Rate 1/2	
Left Degrees	$\lambda_3 = 0.44506$, $\lambda_5 = 0.26704$,
	$\lambda_9 = 0.14835$, $\lambda_{17} = 0.07854$,
	$\lambda_{33} = 0.04046$, $\lambda_{65} = 0.02055$
Right Degrees	$\rho_7 = 0.35282$, $\rho_8 = 0.29548$,
	$\rho_{19} = 0.10225$, $\rho_{20} = 0.18321$,
	$\rho_{84} = 0.04179$, $\rho_{85} = 0.02445$

Table 1: Sample code parameters.

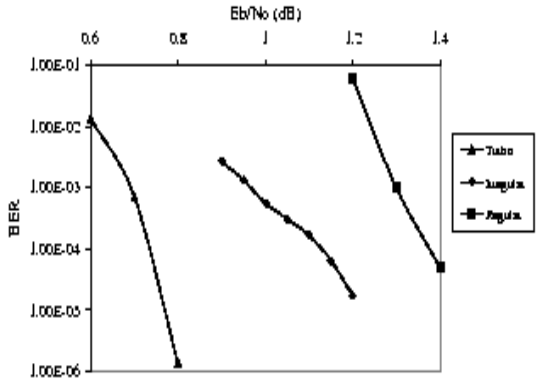
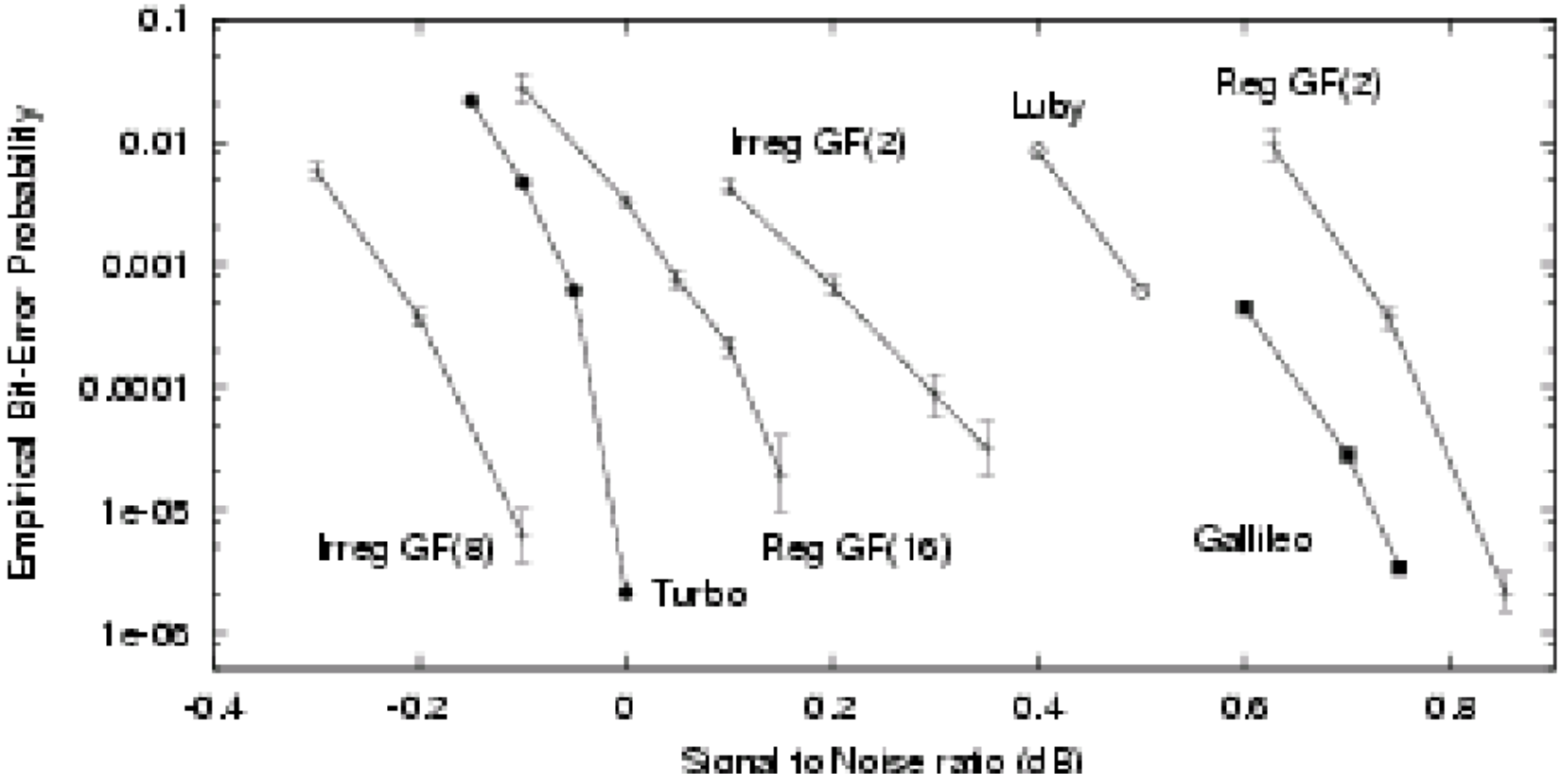


Figure 1: From left to right, rate 1/2 turbo codes, irregular codes, and regular codes.

From MacKay's Website



From MacKay's Website

- The figure shows the performance of various codes with rate 1/4 over the Gaussian Channel. From left to right:
- Irregular low density parity check code over GF(8), blocklength 48000 bits (Davey and MacKay, 1999);
- JPL turbo code (JPL, 1996) blocklength 65536;
- Regular LDPC over GF(16), blocklength 24448 bits (Davey and MacKay, 1998);
- Irregular binary LDPC, blocklength 16000 bits (Davey, 1999);
- M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi and D.A. Spielman's (1998) irregular binary LDPC, blocklength 64000 bits;
- JPL's code for Galileo: a concatenated code based on constraint length 15, rate 1/4 convolutional code (in 1992, this was the best known code of rate 1/4); blocklength about 64,000 bits;
- Regular binary LDPC: blocklength 40000 bits (MacKay, 1999).

Conclusions

- **The inherent parallelism in decoding LDPC codes suggests their use in high data rate systems.**
- **A comparison of LDPC codes and turbo codes is complicated and depends on many issues: e.g., block length, channel model, etc.**
- **LDPC codes are well worthwhile investigating. Some issues to be resolved are:**
  - **Performance for channel models of interest**
  - **Optimization of irregular LDPC codes (for channels of interest).**
  - **Implementation in VLSI.**
  - **Patent issues.**