LDPC

Low Density Parity Check Codes: an introduction

Pierangelo Migliorati

DII - University of Brescia - Italy

HISTORY

- Proposed by Gallager, in 1962 …

- Their importance remained neglected for over 35 years, due to the limited ability of computer simulations at that time

- In the 1999, McKay and Neal, studying turbo codes, introduced a "new" class of block codes: LDPC.

  Soon (☹) they understood that these new codes were already been introduced by Gallager, in 1962 !!!

- Nowadays, LDPC are used in many telecommunication systems (e.g., mobile phones, …, …, …)

Linear Block Codes

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

BACKGROUND: PARITY CHECK MATRIX

Codeword (of linear block codes) constraints are often written in a matrix form, so:

$$Hy^T=0$$

For example:

$$\underbrace{\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&1&1&0&0&1\end{bmatrix}}_{H}\begin{bmatrix}c_1\\c_2\\c_3\\c_4\\c_5\\c_6\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

The matrix H is called *Parity Check Matrix (is the transposed of H^T).*

ENCODING

How can I decide the value of parity bits in the codeword ?
Still considering the previous example, we can write:

$$\begin{cases} c_4 = c_1 \oplus c_2 \\ c_5 = c_2 \oplus c_3 \\ c_6 = c_1 \oplus c_2 \oplus c_3 \end{cases}$$

In the matrix form:

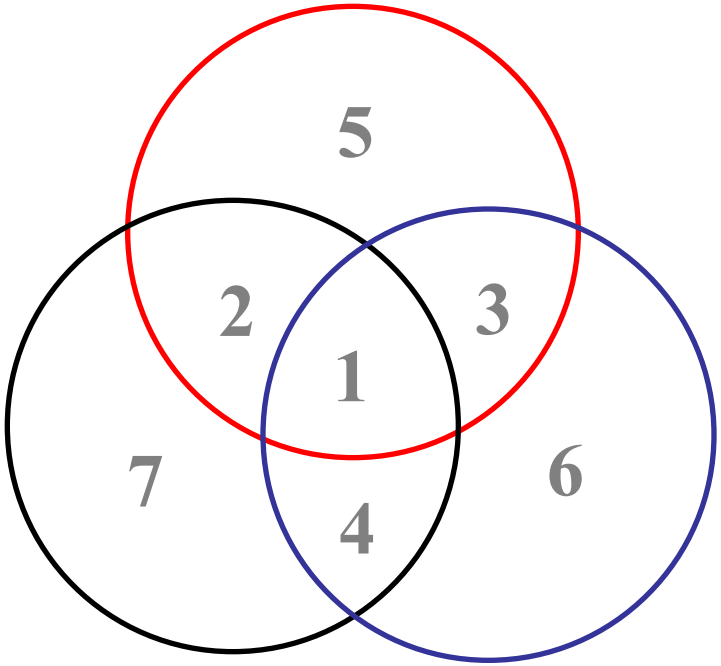
$$|c_1\ c_2\ c_3\ c_4\ c_5\ c_6| \quad = \quad |u_1\ u_2 u_3| \ . \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \longrightarrow G$$

Matrix G is called *Generator Matrix*.
Labeling message bits $\mathbf{u}=[u_1\ u_2\ \ldots\ u_k]$, we can obtain the codeword (starting binary message **u**) in this way: $\boxed{c=\mathbf{u}G}$

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

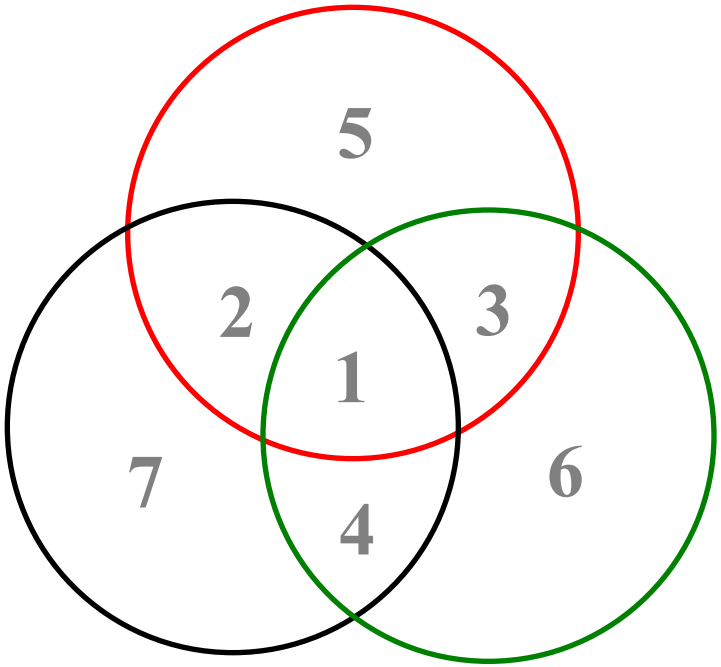


Example: (7,4) Hamming Code

- 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
0 0 0 1	0 1 1
0 0 1 0	1 1 0
0 0 1 1	1 0 1
0 1 0 0	1 0 1
0 1 0 1	1 1 0
0 1 1 0	0 1 1
0 1 1 1	0 0 0
1 0 0 0	1 1 1
1 0 0 1	1 0 0
1 0 1 0	0 0 1
1 0 1 1	0 1 0
1 1 0 0	0 1 0
1 1 0 1	0 0 1
1 1 1 0	1 0 0
1 1 1 1	1 1 1

Hamming Code – Parity Checks



1	2	3	4	5	6	7
1	1	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	1	0	0	1

Hamming Code: Matrix Perspective

- Parity check matrix H

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

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

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

- Generator matrix G

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

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

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

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

Parity-Check Equations

- Parity-check matrix implies system of linear equations.

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

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

TANNER GRAPHS

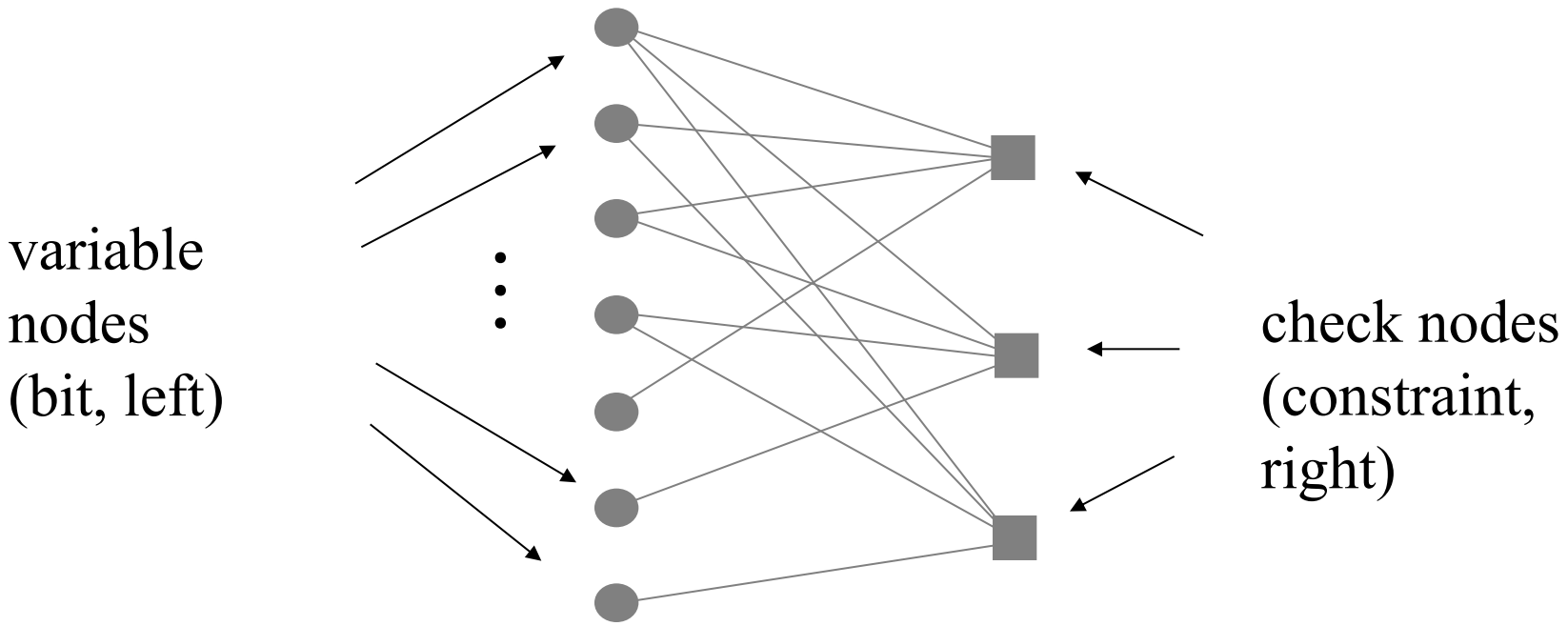
- LBC codes are often represented in graphical form by a *Tanner Graph*

- The Tanner graph consists of two set of vertices:
  - n vertices for the codeword bits (bit nodes)
  - $m = n\text{-}k$ vertices for the parity check equations (check nodes)

- Bit node j is connected to check node $i \Leftrightarrow h_{ji}$ of H is equal to 1

- This graph shows the relationship between codeword bits and Parity Check Equations (PCE)

Tanner Graph Terminology



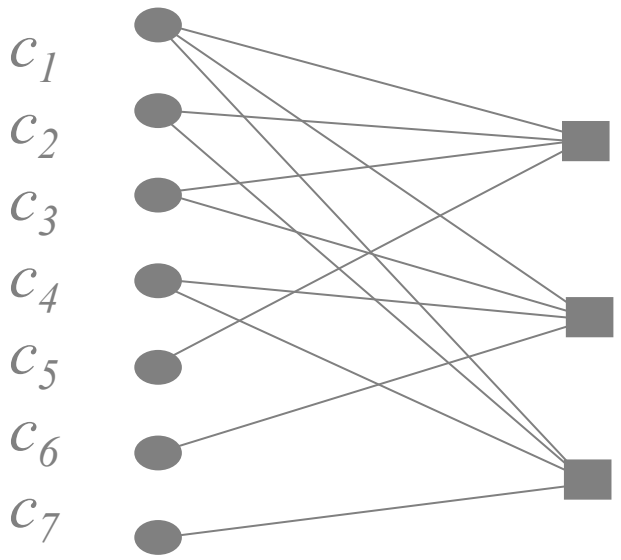
The degree of a node is the number of edges connected to it.

Hamming Code: Tanner Graph

- Bi-partite graph

representing parity-check equations

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



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

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

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

LDPC

LDPC codes are Linear Block Codes with a parity check matrix that contains only a very, very, very, small number of "1"s

- The sparseness of H guarantees an increase "ONLY" linear with the code length of the SOFT decoding complexity (instead of exponential, as in the more general case !!!)

- Classic block codes are not so different with respect to LDPC codes (H sparse). If a block code has a sparse H → it's a LDPC code … but it's not easy to find a sparse H for any existing traditional block code (e.g., BCH, RS, …)

- What's therefore the biggest difference between classic block codes and LDPC codes ?

  → *Decoding Complexity !!!*

  _ Classic block codes are decoded with ML HARD-Decoding algorithms; LDPC codes are SOFT-Decoded, iteratively

LDPC …

$$H = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0
\end{bmatrix} \tag{7.1}$$

There are five ones in each row and three in each column. Each equation of parity involves five bits of code and each bit intervenes only in three rules of parity.
It can be verified that only 10 of the 12 lines are linearly independent (even if for greater symmetry all the lines are used in the decoding), so the code size is N = 20, K = 10.

DEFINITIONS

- A LDPC code is called *regular* if H (m,n) contains:

a) a fixed number $w_c >= 3$ of '1s' for each column

b) a fixed number w_r of '1s' for each row, where:

$$m \cdot w_r = n \cdot w_c$$

$$w_c << m = n - k$$

(k = n° of message bits)

- A LDPC code is called *irregular* if H (m,n) doesn't contain a constant number of '1s' for each column and row

TANNER GRAPHS (2)

- LDPC codes are often represented in graphical form by a *Tanner Graph*

- The Tanner graph consists of two set of vertices:
  - n vertices for the codeword bits (bit nodes)
  - $m = n\text{-}k$ vertices for the parity check equations (check nodes)

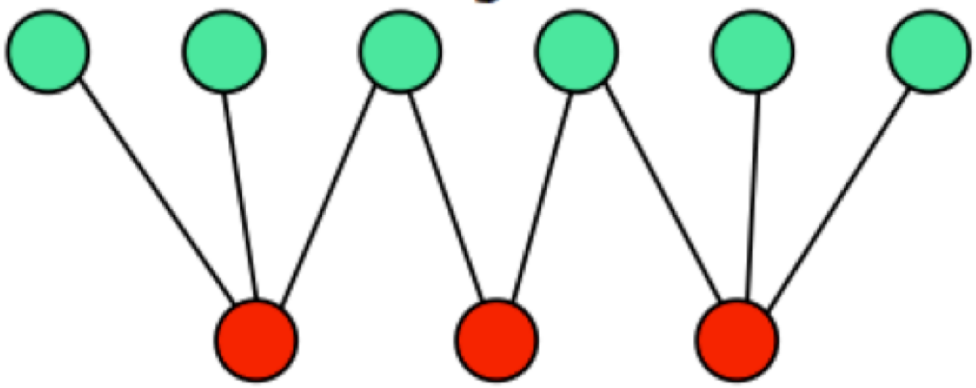
- Bit node j is connected to check node $i \Leftrightarrow h_{ji}$ of H is equal to 1

- This graph shows the relationship between codeword bits and Parity Check Equations (PCE)

Tanner Graph of a Binary Linear Code

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

Message nodes



Check nodes

Column ->

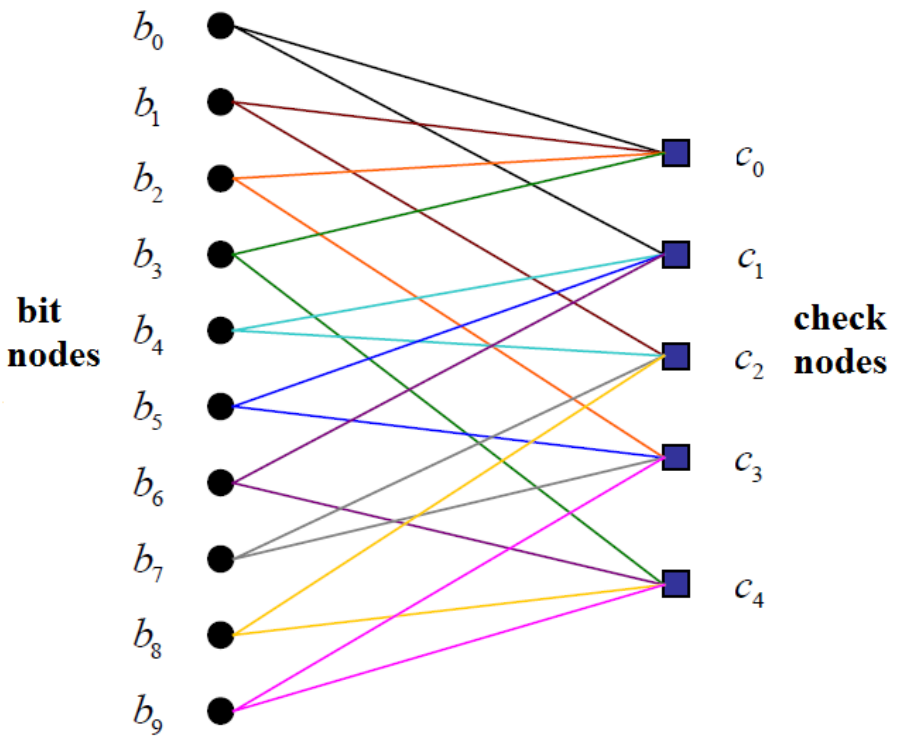
message nodes

Row ->

check nodes

TANNER GRAPH – An EXAMPLE

$$\mathbf{H}_{(n-k)\times n} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}_{5\times 10}$$

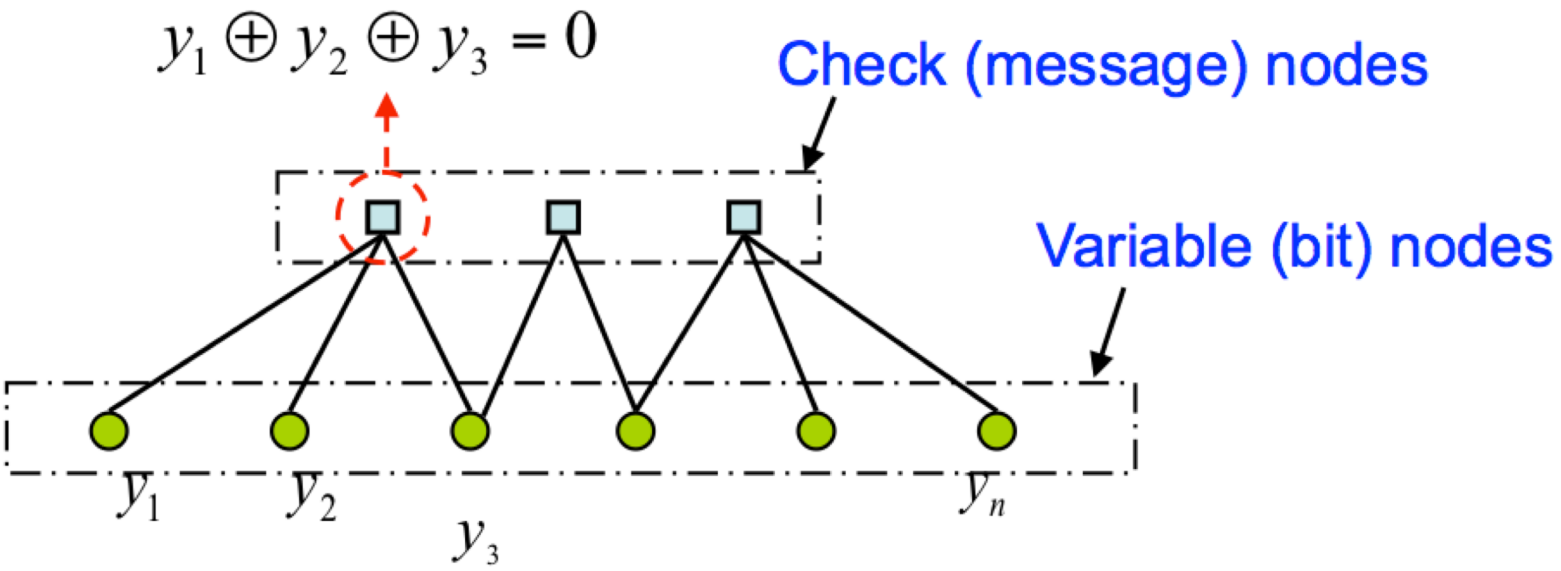


Let's see that bit nodes b_0 b_1 b_2 and b_3 are connected to check node c_1, indeed h_{00} h_{01} h_{02} and h_{03} are equal to 1.

We can observe that this graph is regular: every bit node has 2 links, while every check node has 4 links ⇨ LDPC code is therefore regular (w_c=2, w_r=4)

Low-Density Parity-Check Codes

- Defined by random *sparse* graphs (Tanner graphs)



Simple iterative decoding: message-passing algorithm

DECODING OF LDPC CODES

- The class of decoding algorithms used to decode LDPC codes are collectively indicated as *Message-Passing* algorithms (or *iterative decoding* algorithms)

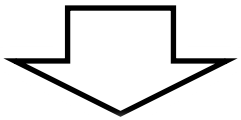
- Different message-passing algorithms are named for the different type of messages passed, or for the type of operation performed at the nodes

- If messages are:
  I. binary → bit-flipping decoding (hard decision)
  II. probabilities → *belief propagation* decoding (soft decision)

BIT-FLIPPING DECODING

- The bit-flipping algorithm is a hard-decision message-passing algorithm for LDPC codes
- How does it work ?

  We know that we can use the information in the bit nodes in order to verify multiple parity check equations !!!
- And so there are two steps:
  1. Bit nodes send the info (i.e., their value) to the check nodes
  2. Check nodes verify Parity Check Equations, and transmit results to bit nodes
- Analysing iteratively bit nodes and PCE, the algorithm inverts the value of probable wrong bits (i.e., *bit flipping*)



In this way it's possible, at the end of any iteration, to find all the correct values of the codeword !!!

BIT-FLIPPING DECODING: EXAMPLE 1 (1)

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

$$\mathbf{c} = [1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1]$$

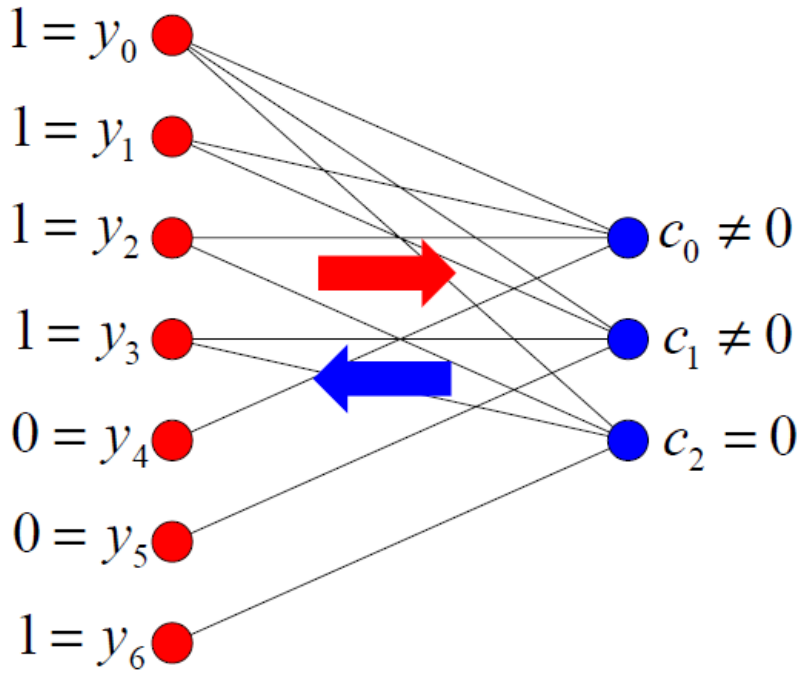
$$\mathbf{y} = \mathbf{c} + [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\mathbf{y} = [1 \quad (1) \quad 1 \quad 1 \quad 0 \quad 0 \quad 1]$$

Suppose we have this parity check matrix H … we transmit a codeword *c* through a noisy channel, and the decoder receives *y*

How could be detected that the second bit is wrong ?

BIT-FLIPPING DECODING: EXAMPLE 1 (2)



Observing Tanner graph we can see that the parity check equation c_2 is the only correct → we can think that wrong bits aren't that linked to node c_2!

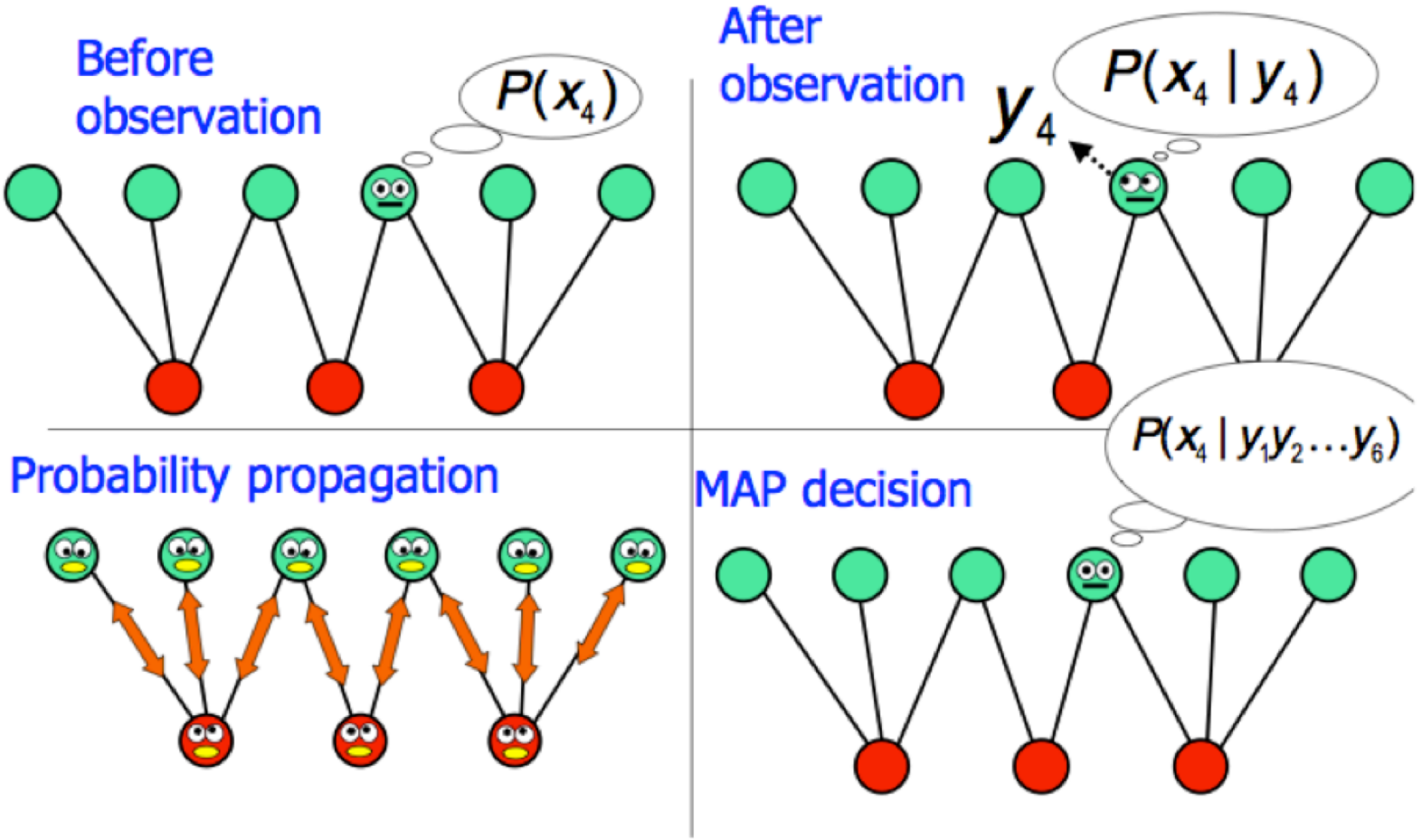
And so the decoder checks only y_1, y_4 and y_5 inverting iteratively (flipping) these bits …

… in this case, at the first iteration the decoder discovers that y_1 is (could be !) the wrong bit !

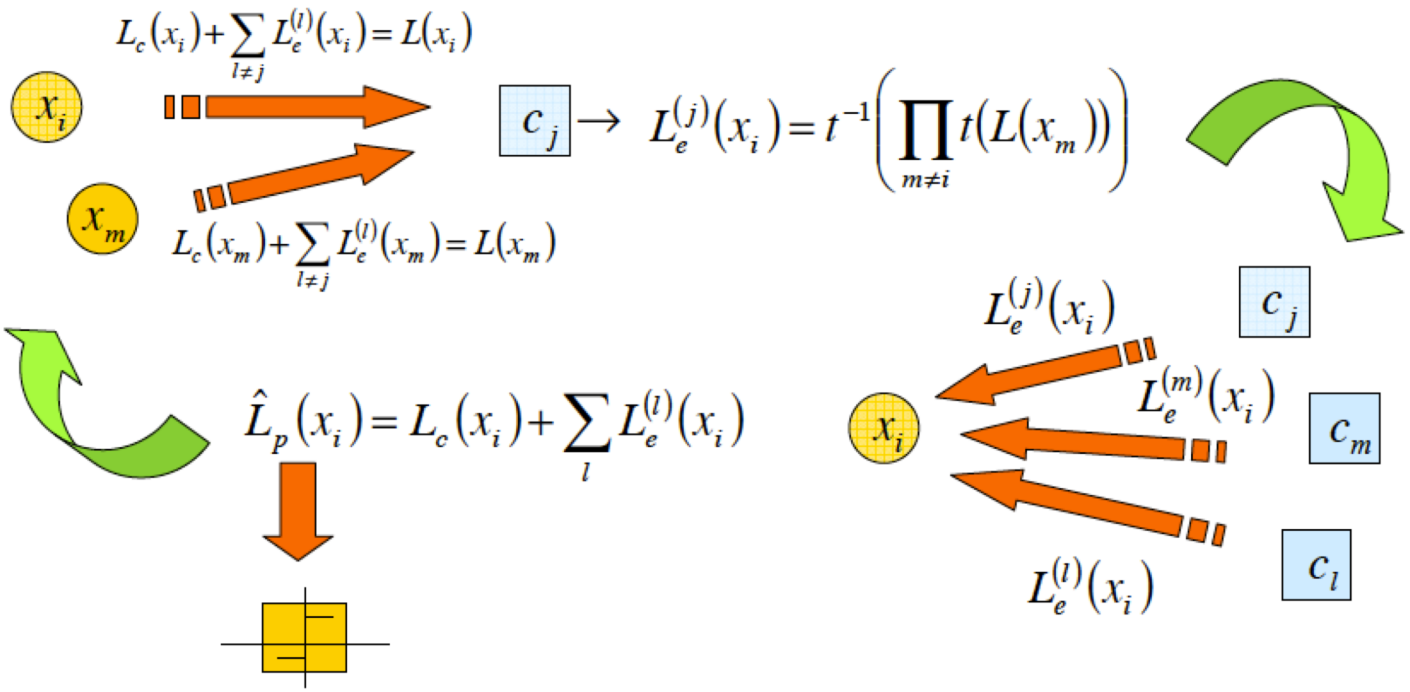
SUM-PRODUCT ALGORITHM (Soft Decoding)

- It's a *belief propagation* algorithm
- We know that the additive noise is not a discrete value …
- … we can model it as a white Gaussian process
- So this algorithm get conditional probability distributions of bit node values !
- Matematical computations are not dealt in detail …

Message Passing … (soft dec.)



Codici LDPC: decodifica (2)



PERFORMANCE

- For large dimensions of H we can have very good performance
- LDPC codes can show better performance than turbo codes
- Irregular LDPC codes → faster convergence

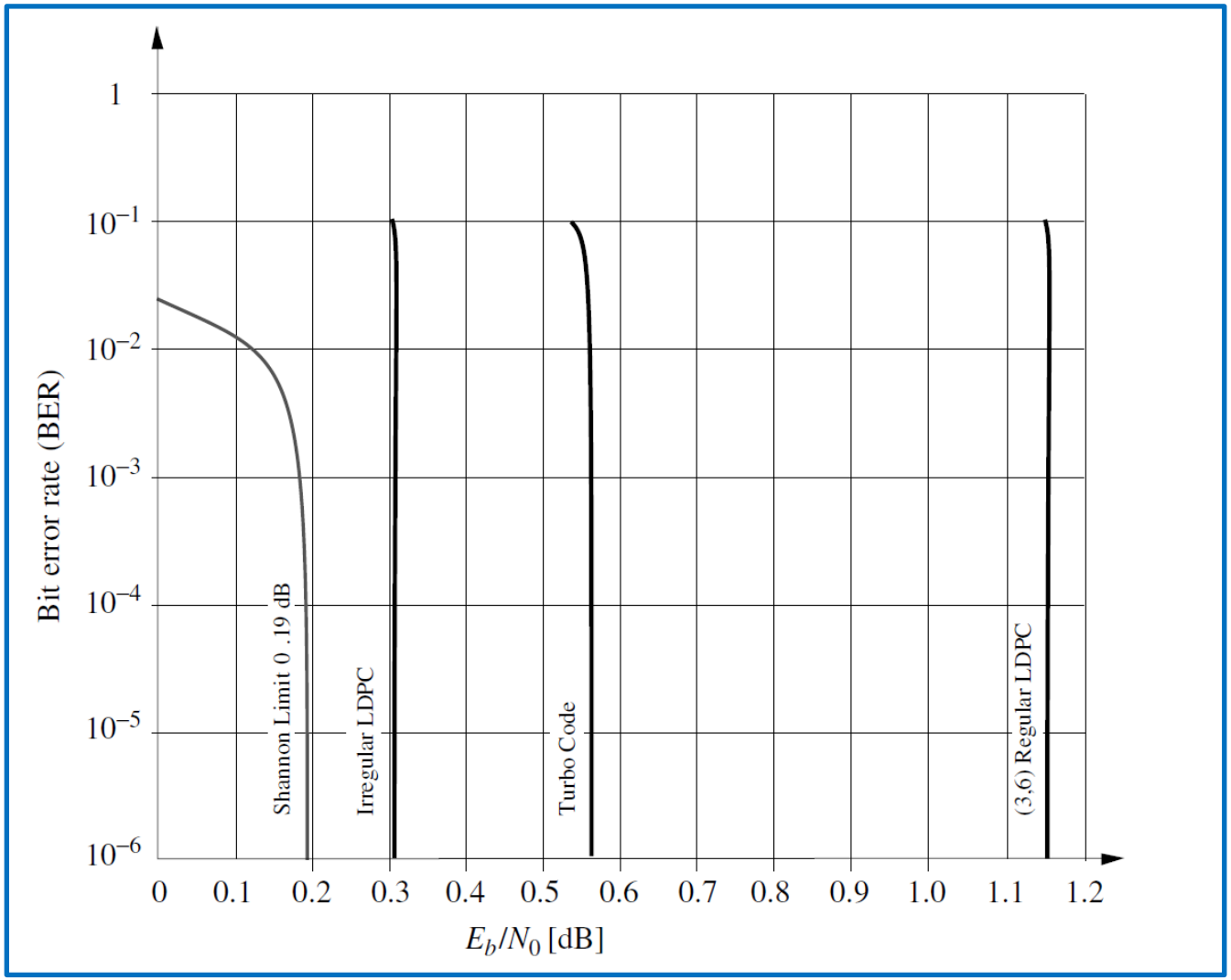
- LDPC codes can approach the Shannon limit to within small fractions of a decibel !!!

Shannon's theorem:

$$\text{if } R < C \Rightarrow P_e < \varepsilon$$

R = transmission rate, C = channel capacity,
P_e = error probability

LDPC: performance



R=1/2, n = 10^6

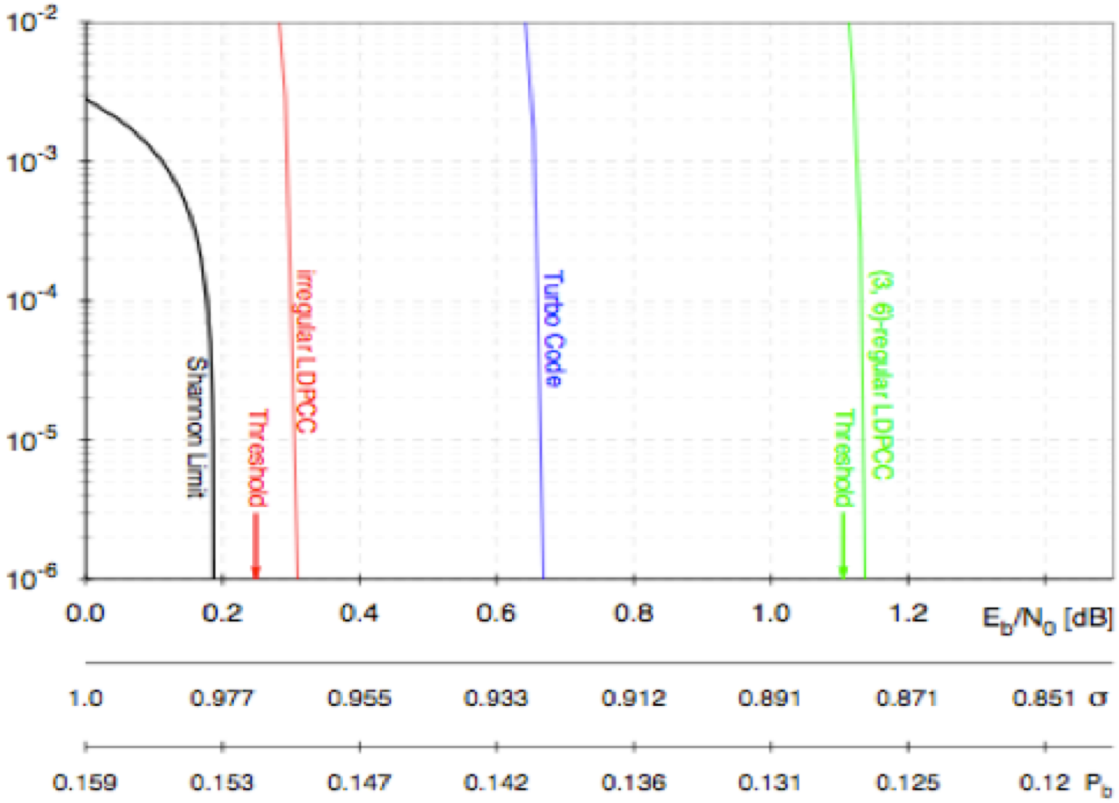


Figure 1: Comparison of $(3, 6)$-regular LDPCC, turbo code, and optimized irregular LDPCC. All codes are of length 10^6 and of rate one-half. The bit error rate for the AWGNC is shown as a function of E_b/N_0 (in dB), the standard deviation σ, as well as the raw input bit error probability P_b.

WHERE'S THE SECRET (if any …) ?

- In the matrix H there are few "1"s → minimum distance is therefore very small …

- Many "1"s → decoding algorithms will be much more complex

- Shannon's theorem says that long and random codes could achieve capacity limits … the problem was:
  "how can be generated and decoded these codes ?"

  → LDPC codes give us a possible (practical) solution …

LDPC CODES vs TURBO CODES

- Advantages ☺
  1. It's not necessary to know *the noise* parameters
  2. It's possible to stop the iteration process since every parity check equations is observed
  3. Better performance (for irregular and long LDPC codes)
  4. Only one decoder (Turbo codes need two decoders…)
  5. Smaller iteration decoding complexity

- Disadvantages ☹
  1. LDPC codes need more iterations
  2. The construction process of LDPC code is more complicated than the construction process of a Turbo code

TECHNOLOGY

- DVB-S2 – 2005 → digital television broadcast standard
- Wi-MAX (IEEE – 80216e) – 2005 → wireless communications standard
- 10 GBase-T Ethernet (802.3an) – 2006 → computer network standard
- Wi-Fi (IEEE 820.11n) – 2007
- DVB-T2 – 2008
- G.hn (ITU G.9960) – 2009 → home network technology
- DVB-C2 – 2010
- They're used in every advanced technique for errors control
- July 2010: was created a new SoC (System on a Chip) device based on LDPC code → hard disk capacity of 320 GB

DVB = Digital Video Broadcasting (S = satellite – T = terrestrial – C = cable – 2 = second generation)
ITU = Telecommunication Standardization Section

Conclusions

- It is now possible to closely approach the Shannon limit by using Turbo and LDPC codes
- Binary capacity approaching codes can be combined with higher order modulations
- These codes are making (quickly) their way into standards
  - Binary turbo: UMTS, cdma2000, …
  - Duo-binary turbo: DVB-RCS, 802.16, …
  - LDPC: DVB-S2 standard, …
- Software for simulating turbo and LDPC codes can be found, e.g., at: www.iterativesolutions.com

FUTURE: ONLY LDPC CODES ???

- History of coding started in the 1948 (Shannon)
- After many years, LDPC codes are able to approximate the Shannon limit within (about) 0.6 – 0.8 dB
  - It' difficult to think that LDPC codes can be replaced from other technologies

… but …

also in the 1971 people thougth that RS (Reed-Solomon) and convolutional codes were practically insuperable …
… also a workshop in Florida (1971) is remembered for the expression:

ENCODING IS DEAD !!!

Now we all know they were wrong !!!
… and maybe still us … ☺


"That's
all
folks!"