Università degli Studi di Trieste – Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria delle Telecomunicazioni

RATE COMPATIBLE PUNCTURATION OF LDPC CODES

Laureando: NOSCHESE Matteo

MODERN COMMUNICATION SYSTEM REQUIREMENTS

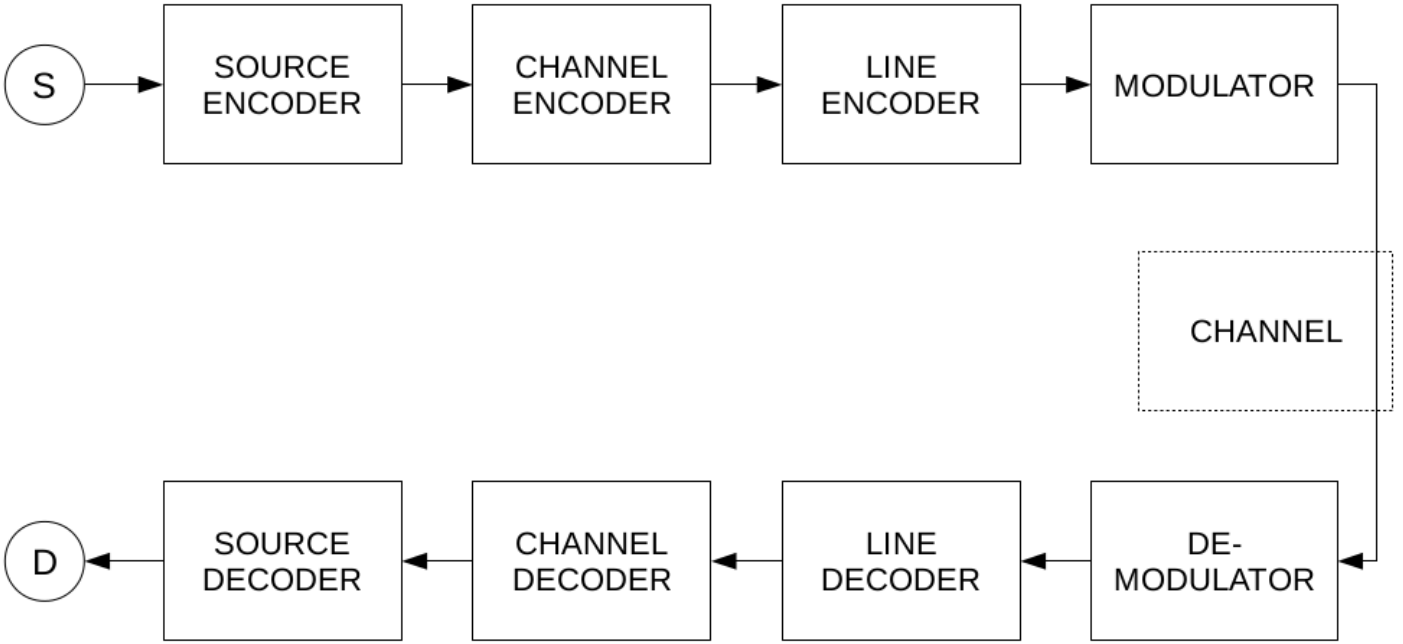
- High transmission (baud) rate
  - Efficient use of bandwidth
- Extended mobility
  - Wireless networks
- Multiple access
  - Growing number of users
- Reduced power consumption for battery powered devices

ADAPTIVE SYSTEMS

Communication systems that can *dynamically* modify their parameters to adapt to the current transmission scenario.

- Modulation.
- Channel coding technique.
- Transmission power.
- Adaptive antennas (smart antennas).

MODERN DIGITAL COMMUNICATION SYSTEM



CHANNEL MODELS

Mathematical models introduced to study the effects of the channel on the transmitted signal and aid the design of efficient transmission techniques. Some are listed here.

- Binary Symmetric Channel (BSC).
- Binary Erasure Channel (BEC).
- Additional White Gaussian Noise Channel (AWGN).
- Fading (multipath) channel.

CHANNEL CAPACITY

$$C = \max_{p_X(x)} \left(I(X,Y) \right)$$

- Defined by Shannon as the *greatest possible mutual information between the transmitter and the receiver.*
- Measured in information bits per symbol.
- For a binary modulation, the channel capacity is the maximum rate of the channel code that can be employed without violating Shannon's constraint.

CHANNEL CODING

The techniques employed to “protect” the information from impairments introduced by the transmission are collectively referred to as *channel coding*.

Regardless of the structured of the employed code, channel coding involves adding *redundancy* to the source coded information.

There are several different classifications for channel coding techniques.

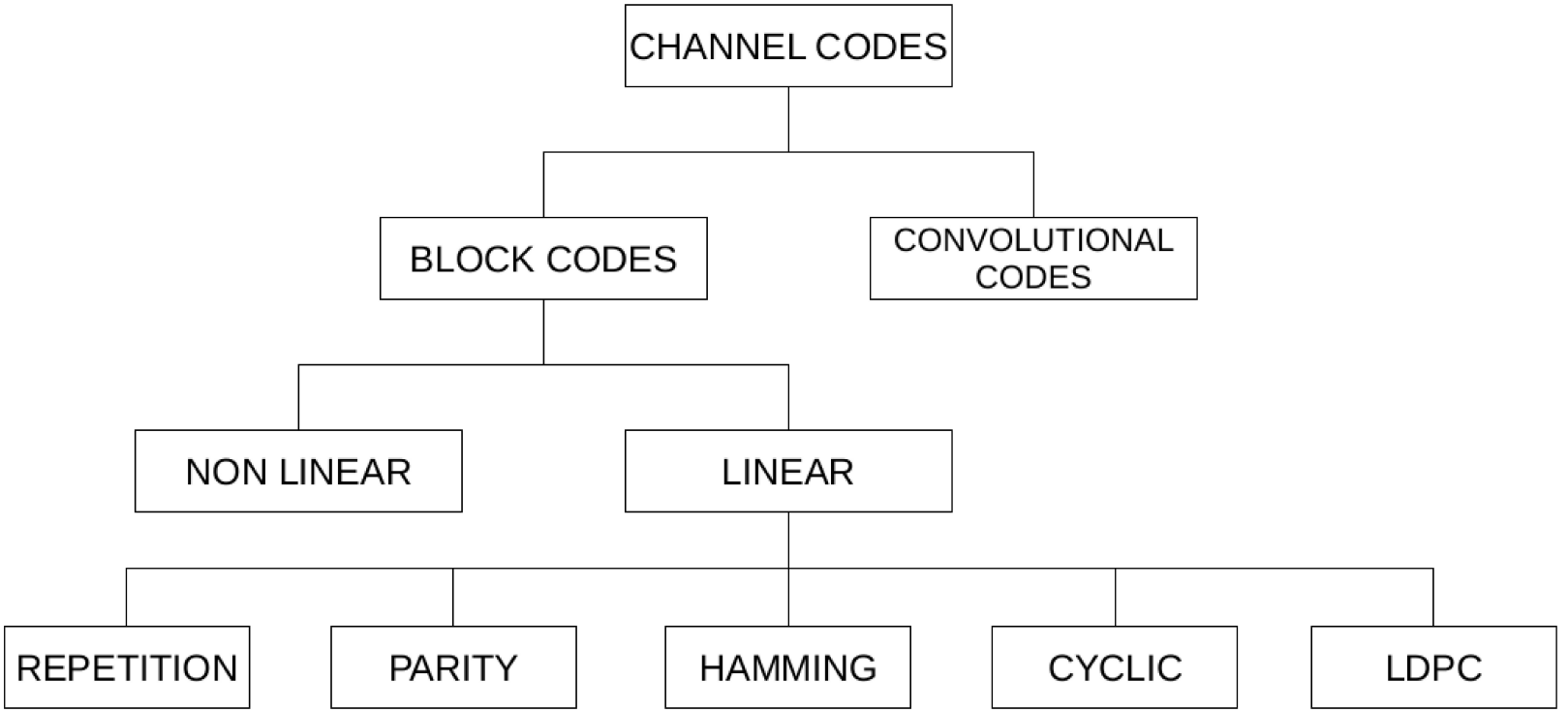
CHANNEL CODING

- Error Detecting (ED) Codes: add a “small” amount of redundancy to *reveal* errors in transmission. If an error is detected, a retransmission is usually requested.
- Error Correcting (EC) Codes: add a “large” amount of redundancy, so the receiver can attempt to *correct* errors without the need of a retransmission.
- Hybrid Codes: have *mixed* capabilities, can correct “light” errors and reveal “heavy” errors.

CHANNEL CODING

- Block Codes: the source information is divided in blocks of *k* bits. To each block, *q* bits of redundancy are added.
- Convolutional Codes: the source information is treated as a continuous stream. Bits are combined into output streams by shift registers and adders.
  - Turbo codes: a *feedback* is added to increase the performances of the code and avoid low-weight output sequences.

CHANNEL CODING – A CLASSIFICATION



AUTOMATIC REPEAT REQUEST (ARQ)

In an ARQ scenario, a feedback channel is used to notify the status of decoding. Whenever a block is received, a response is sent back to the transmitter.

- An ACK is sent when the block is received correctly.
- A NACK is sent when one or more errors are detected. In this case, a retransmission of the corrupted information is usually requested.

ED codes are employed to perform error detection. In Hybrid ARQ, mixed ED/FEC techniques are used.

ARQ TECHNIQUES

- Stop and Wait
- Go Back N
- Selective Repeat
- Hybrid ARQ
  - Type 1
  - Type 2
  - Soft combining
  - Incremental redundancy (Type 3)

RATE ADAPTABILITY

Rate adaptability can be realized with several pairs of encoders and decoders, one pair for each desired code rate.

- However, this is highly undesirable due to the high amount of added complexity.
- Rate adaptability can also be realized by *puncturing* a low rate channel code, resulting in only one encoder and decoder.

CODE PUNCTURATION

Code puncturation is defined as *the process of removing some of the parity bits from the encoded codeword*. The removed bits are not transmitted on the channel, resulting in a higher code rate.

- A specific method of puncturation can be univocally defined by listing the positions to be punctured.
- Equivalently, a puncturation *pattern* can be given: a binary sequence where discarded positions are marked with a 0 and unpunctured positions are marked with a 1.

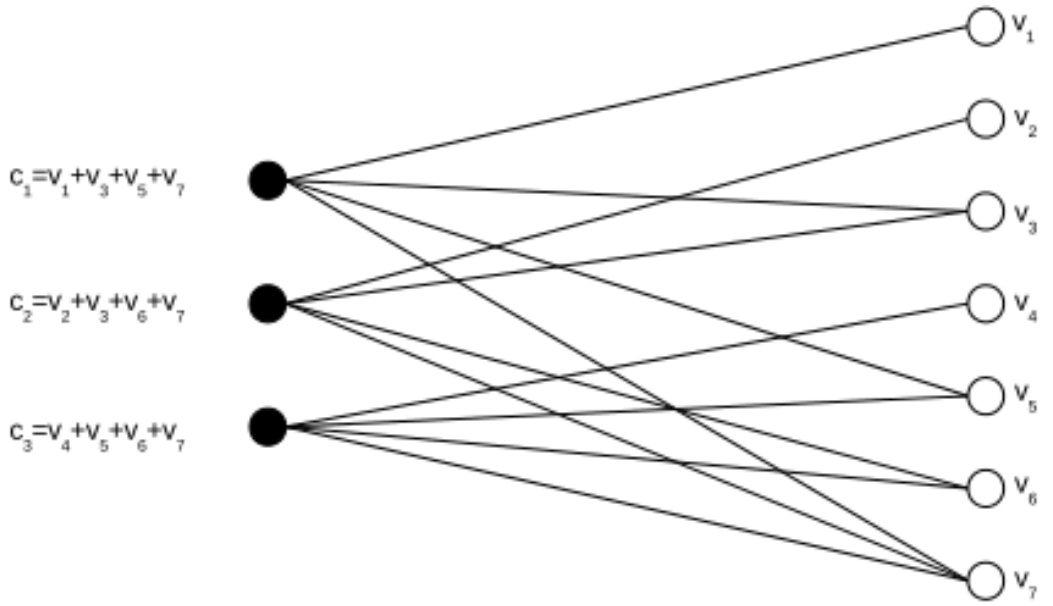
LDPC CODES DEFINED

- Introduced by Robert Gallagher in 1960
- Block codes based on *sparse* parity matrix
- Rediscovered in the 1990s
- Capacity achieving performances
- Low computational complexity
- Included in IEEE802.11n, 802.16e (Wi-MAX), 10G-BaseT Ethernet and Digital Video Broadcasting (DVB)

TANNER GRAPH

A graphical representation of the parity matrix using a *bipartite* graph

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



TANNER GRAPH

There is a *biunivocal* correspondence between a parity matrix and the associated Tanned graph.

- Each check node is associated with a parity equation.
- Each variable node is associated with a bit of the codeword.
- Each edge is associated to a 1 in the parity matrix: edges connect check nodes to the variable nodes involved in the check node's equation.

TANNER GRAPH

- A Tanner graph (code) is *regular* if all check nodes have the same degree d_c and all the variable nodes have the same degree d_v.
- Two ways to specify degree distribution:
  - Node – perspective
  - Edge – perspective
- Decoding can be performed on a Tanner graph with the *Belief-Propagation* algorithm.

THE BELIEF PROPAGATION ALGORITHM

The Belief Propagation (or message-passing) algorithm is most commonly employed for LDPC decoding.

- Under belief-propagation decoding, variable nodes and check nodes exchange ”messages” between each other *iteratively*.
- A check node gets messages from the variable nodes it is connected to (”*neighbours*”), processes the messages and sends the result to back to its neighbouring variable nodes.
- Similarly, a variable node receives messages from its check nodes and returns the processed message back to them.

THE BELIEF PROPAGATION ALGORITHM - EXAMPLE

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

Consider the regular block code defined by the above H matrix: a possible valid codeword of H is X = [10010101]. Suppose the Y = [11010101] vector is received: the corresponding error vector is E = [01000000].

The following tables describe the first iteration of the algorithm.

THE BELIEF PROPAGATION ALGORITHM - EXAMPLE

Check nodes evolution (Hard decision)

Check Node	Activities				
c_1	receive	$v_2 \to 1$	$v_4 \to 1$	$v_5 \to 0$	$v_8 \to 1$
	send	$0 \to v_2$	$0 \to v_4$	$1 \to v_5$	$0 \to v_8$
c_2	receive	$v_2 \to 1$	$v_3 \to 1$	$v_6 \to 0$	$c_8 \to 1$
	send	$0 \to v_1$	$0 \to v_2$	$1 \to v_3$	$0 \to v_6$
c_3	receive	$v_3 \to 0$	$v_6 \to 1$	$v_7 \to 0$	$c_8 \to 1$
	send	$0 \to v_3$	$1 \to v_6$	$0 \to v_7$	$1 \to v_8$
c_4	receive	$v_1 \to 0$	$v_4 \to 1$	$v_5 \to 0$	$c_7 \to 0$
	send	$1 \to v_1$	$1 \to v_4$	$0 \to v_5$	$0 \to v_7$

THE BELIEF PROPAGATION ALGORITHM - EXAMPLE

Variable nodes evolution (Hard decision)

Variable Node	y_i	Messages from check nodes		decision
v_1	1	$c_2 \to 0$	$c_4 \to 1$	1
v_2	1	$c_1 \to 0$	$c_2 \to 0$	0
v_3	0	$c_2 \to 1$	$c_3 \to 0$	0
v_4	1	$c_1 \to 0$	$c_4 \to 1$	1
v_5	0	$c_1 \to 1$	$c_4 \to 0$	0
v_6	1	$c_2 \to 0$	$c_3 \to 1$	1
v_7	0	$c_3 \to 0$	$c_4 \to 0$	0
v_8	1	$c_1 \to 1$	$c_3 \to 1$	1

NODE-PERSPECTIVE DEGREE DISTRIBUTION

Normalized fraction of degree-i variable nodes:

$$L_i = \frac{n_i}{\sum_{i=1}^{d_v} n_i} \qquad L(x) = \sum_{i=1}^{d_v} L_i x^i$$

Normalized fraction of degree-j check nodes:

$$R_j = \frac{m_j}{\sum_{j=1}^{d_c} m_j} \qquad R(x) = \sum_{j=1}^{d_j} R_j x^j$$

EDGE-PERSPECTIVE DEGREE DISTRIBUTION

In this case, the distribution specifies the normalized fractions of edges that connect to a degree-i variable node, and the fractions of edges that connect to a degree-j check node.

$$\lambda(x) = \sum_{i=2}^{d_l} \lambda_i x^{i-1}$$

$$\rho(x) = \sum_{j=2}^{d_r} \rho_j x^{j-1}$$

EQUIVALENT PARITY MATRIX

Puncturing a parity bit is equivalent to eliminating a variable node from the Tanner graph associated to the mother code.

One of the check equations associated with the punctured bit is “subtracted” from the parity matrix H. More specifically, it is subtracted from all the check equations associated with the punctured bit, including itself.

H is defined over GF(2), hence subtraction is equivalent to modulo-2 sum.

EQUIVALENT PARITY MATRIX

$$H = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$\hat{H} = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

IMPLICATIONS OF THE ALL-ZERO CODEWORD

It is assumed that the channel-coded information is a sequence of 0s, and that a BPSK modulation is employed on an AWGN channel. The resulting channel output probability distribution is:

$$f_u(u) = \frac{1}{\sqrt{4\pi m_u}} e^{-\frac{(u-m_u)^2}{4m_u}}$$

$$E[LLR] = \frac{2}{\sigma_N^2} = m_u \qquad \sigma_{LLR}^2 = \frac{4}{\sigma_N^2} = 2m_u \qquad LLR = \log \frac{f_y(y|x=+1)}{f_y(y|x=-1)}$$

BLOCK ERROR PROBABILITY

The probability of a block error at the L[th] iteration of the belief propagation algorithm is:

$$P_L = \sum_{i=2}^{d_l} \lambda'_i Q\left(\sqrt{\frac{s + it_L}{2}}\right)$$

where

$$t_L = f\left(s, t_{L-1}\right)$$

is the mean update equation for the L[th] iteration of the belief propagation algorithm.

BLOCK ERROR PROBABILITY

The following quantity is defined:

$$\Delta t = f(s,t) - t = s - (i_1 - 2) - 4\log\lambda_{i1} - 4\sum_{i=2}^{d_r}\rho_j \log(j-1) + O(t^{-1})$$

Δt is the rate of increase of the function *f*().

The Q function is monotonically decreasing: hence, maximizing Δt minimizes the block error probability.

This can be expressed as a cost function to minimize.

THE COST FUNCTION

$$f\left(\hat{\lambda}_{i_1}, \rho\left(\hat{x}\right)\right) = \log \hat{\lambda}_{i_1} + \sum_{j=2}^{d_r} \hat{\rho}_j \log\left(j-1\right) - c_0 \hat{i}_1$$

$\hat{\lambda}_{i_1}$ First non-zero coefficient of $\hat{\lambda}(x)$

i_1 Index of $\hat{\lambda}_{i_1}$

c_0 Integer depending on the code length

$\hat{\lambda}(x)\hat{\rho}(x)$ Edge-perspective degree distributions for the equivalent punctured parity matrix

THE PATTERN GENERATION ALGORITHM (1)

1. The number of bit to puncture is $p = n\left(1 - \frac{R_0}{R_1}\right)$
   1. *n* is the mother length block code, R_0 is the mother length rate, and $R_1 > R_0$ is the rate to achieve with puncturation.
2. Initialize a list L of *p* integers indicating bit positions. At this point, all the values of L are equal to 0, indicating no punctured bits.
3. Generate a puncturation pattern for each bit position that is not punctured yet (that is, not listed in L).
4. Compute the equivalent parity matrix for each puncturation pattern generated this way.

THE PATTERN GENERATION ALGORITHM (2)

5. Compute the edge-perspective variable and check node degree distributions.

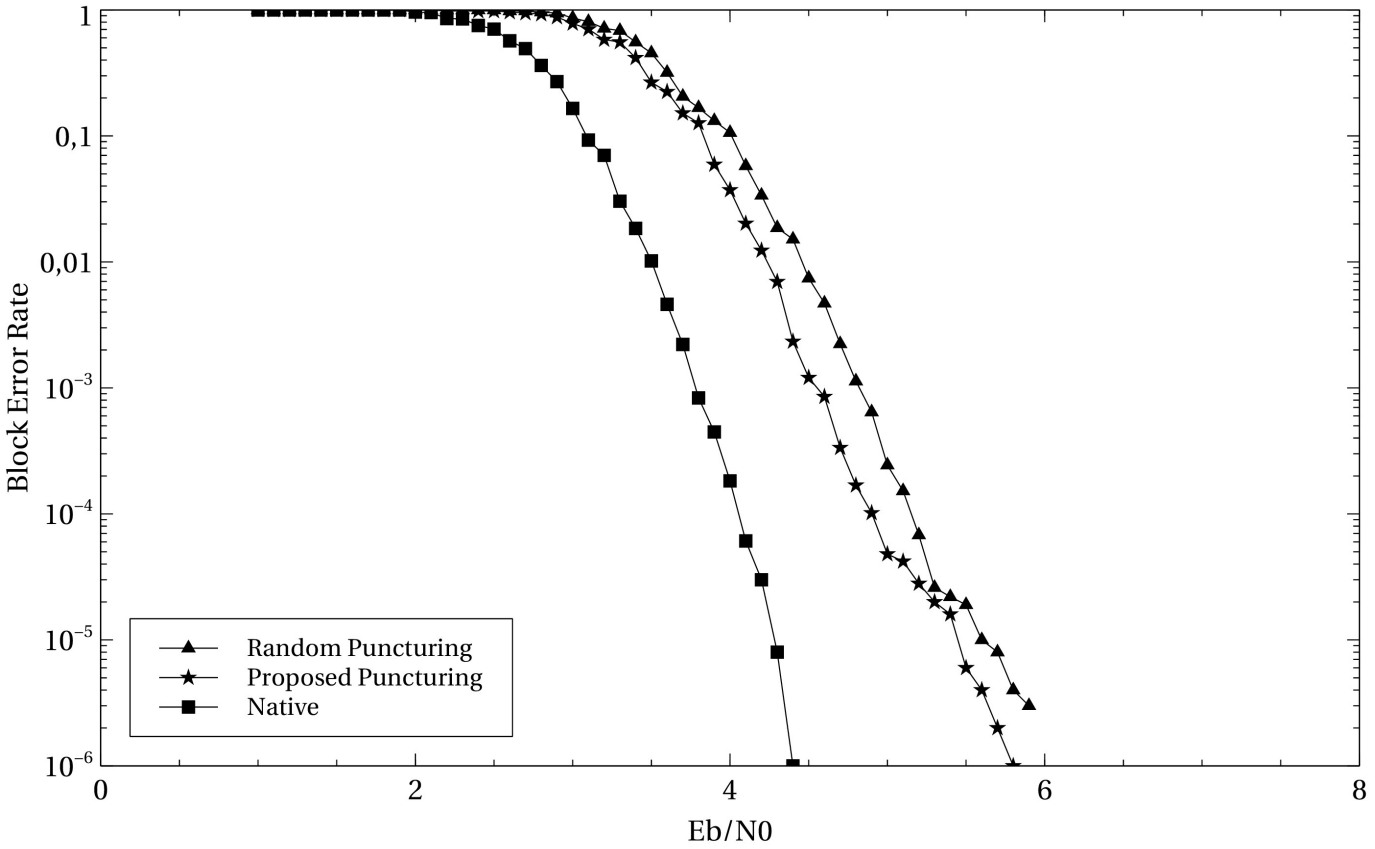
6. Compute the cost function for each pattern.

7. The position yielding the lowest value of the cost function is chosen and added to L. If two or more positions yield the same value, the lowest position is chosen.

8. If all *p* positions of L have been set (that is, they are different from 0), the algorithm ends. Otherwise, return to step 3.

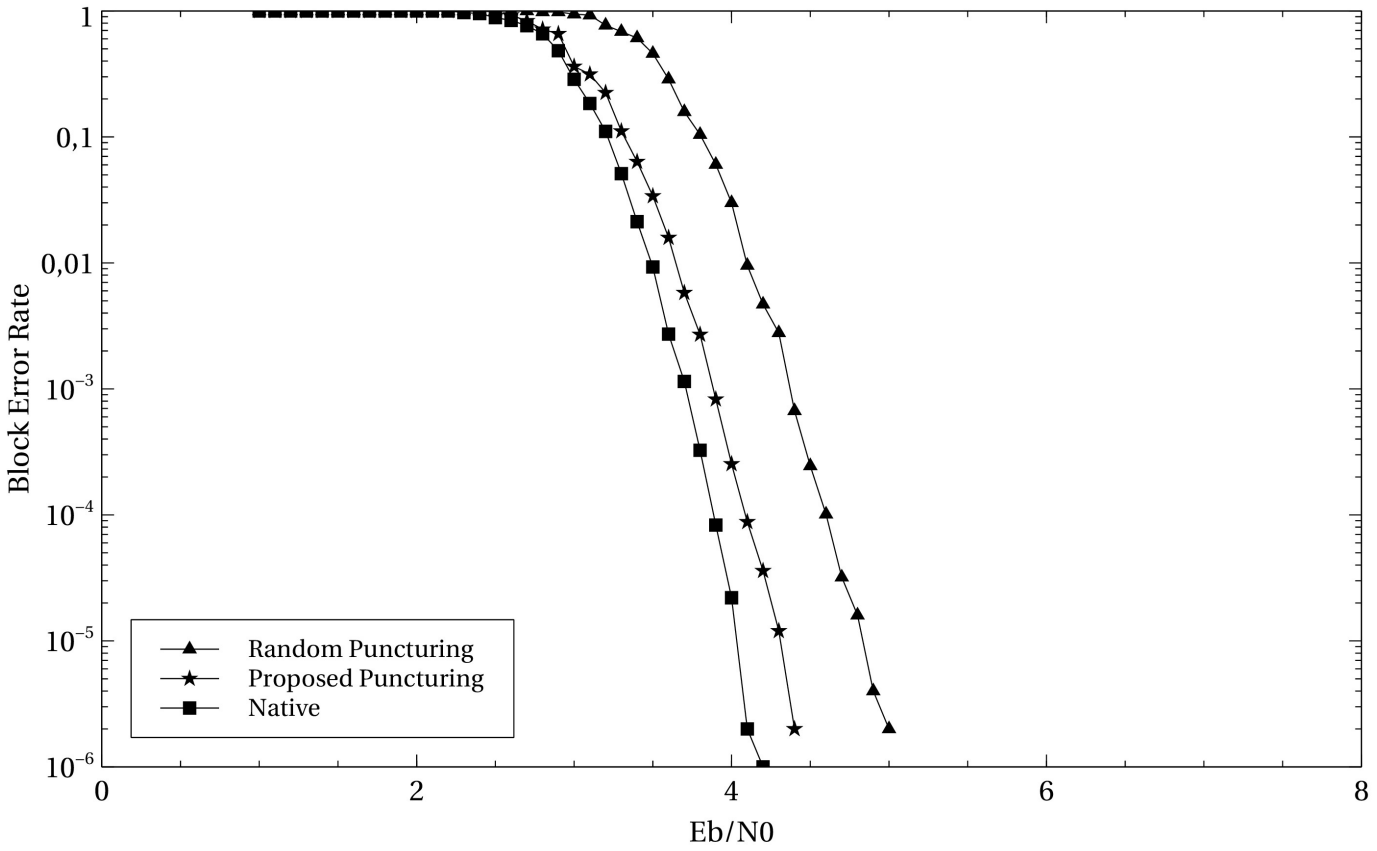
SOME SIMULATION RESULTS

Block Error Rate for the regular (3,6), n=1000 code



SOME SIMULATION RESULTS

Block Error Rate for the regular (3,6), n=2000 code



SOFTWARE IMPLEMENTATION

- The pattern generation algorithm and the simulator have been written in C++ with the aid of the IT++ libraries.
- The source code have been compiled on a Linux system using g++ as the compiler of choice.
- Parameters can be passed by command line or read from an XML-like configuration file.
- The results are logged to a text file and/or to a CSV file, for easy manipulation of the data using a spreadsheet software. The verbosity of the output if also configurable.

FURTHER DEVELOPMENT

- Consider higher order modulations such as QPSK or 8-PSK.
- Consider different values of c_0 in the cost function.
- Simulate the performances of the punctured code on different channel models.
- Apply the proposed algorithm to non-regular mother codes.
- Modify the algorithm to randomize the bit position choice when more than one position yield the same value of the cost function.

CONCLUSIONS

- The class of LDPC codes have been formally defined.
- A structural analysis based on the work of Richardson and Urbanke was proposed.
- A compact, computationally light cost function was devised. The function can be used to determine the “quality” of a puncturation pattern.
- A rate compatible pattern generation algorithm was implemented using the cost function and applied to a number of LDPC codes.
- Simulation were run to validate the results.

THANKS FOR YOUR ATTENTION