LDPC Codes
(Advanced Signal Processing Seminar)

Sebastian Tschiatschek

Signal Processing and Speech Communication Laboratory

Nov 29, 2010

Outline

Introduction

"LDPC Codes" ...

- stands for *Low Density Parity Check Codes* (LDPC codes are also known as *Gallager codes* in honor of Rober G. Gallager),
- were invented by Gallager 1962 (and then forgotten for many years),
- are linear codes with very sparse parity check matrices,
- typically use large block-lengths,
- can approach the channel capacity for many standard channels (for large block lengths),
- can be *efficiently* decoded by iterative schemes.

Applications

LDPC codes are used in applications...

- requiring reliable and efficient information transfer,
- or with constrained return-channel and data-corrupting noise.

Specifically,

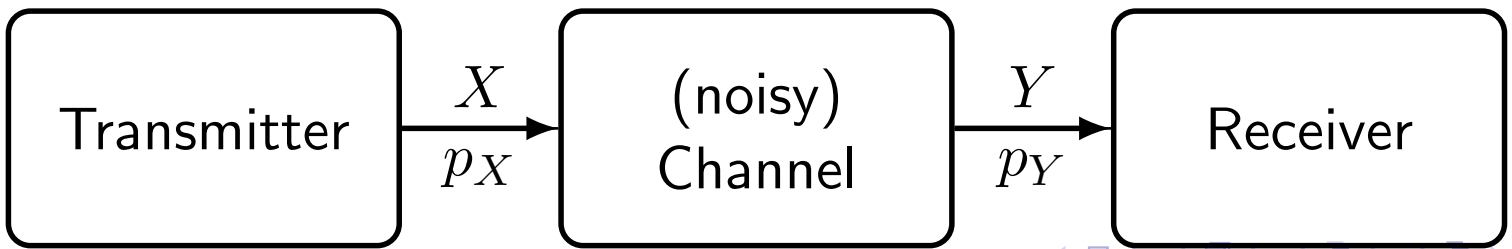
- FEC scheme for DVB-S2 (digital television over satellite transmission; block-lengths of 64.800 [normal] or 16.200 bits [short]; code-rates from $\frac{1}{4}$ to $\frac{9}{10}$),
- 10GBase-T-Ethernet (10 gigabits per second over twisted-pair cables; block-lengths of 2048 bits),
- ...

Channel Capcity C

- *Tightest upper bound on the amount of information* that can be reliably transmitted over a communications channel.
- By the noisy-channel coding theorem it is the *limiting rate that can be achieved with arbitrarily small error probability*. This theorem also tells us, that we need *long block-lengths for capacity-approaching codes*.
- Formally,

$$C = \sup_{p_X} I(X;Y),$$

where $I(X;Y)$ is the mutual information of the random variables X and Y.



Channel Capcity C, contd.

Examples (Capacity in bits per channel use)

- Binary symmetric channel (BSC):

$$C_{\text{BSC}} = 1 - h(\epsilon),$$

  where $h(\epsilon)$ is the binary entropy function.

- Binary erasure channel (BEC):

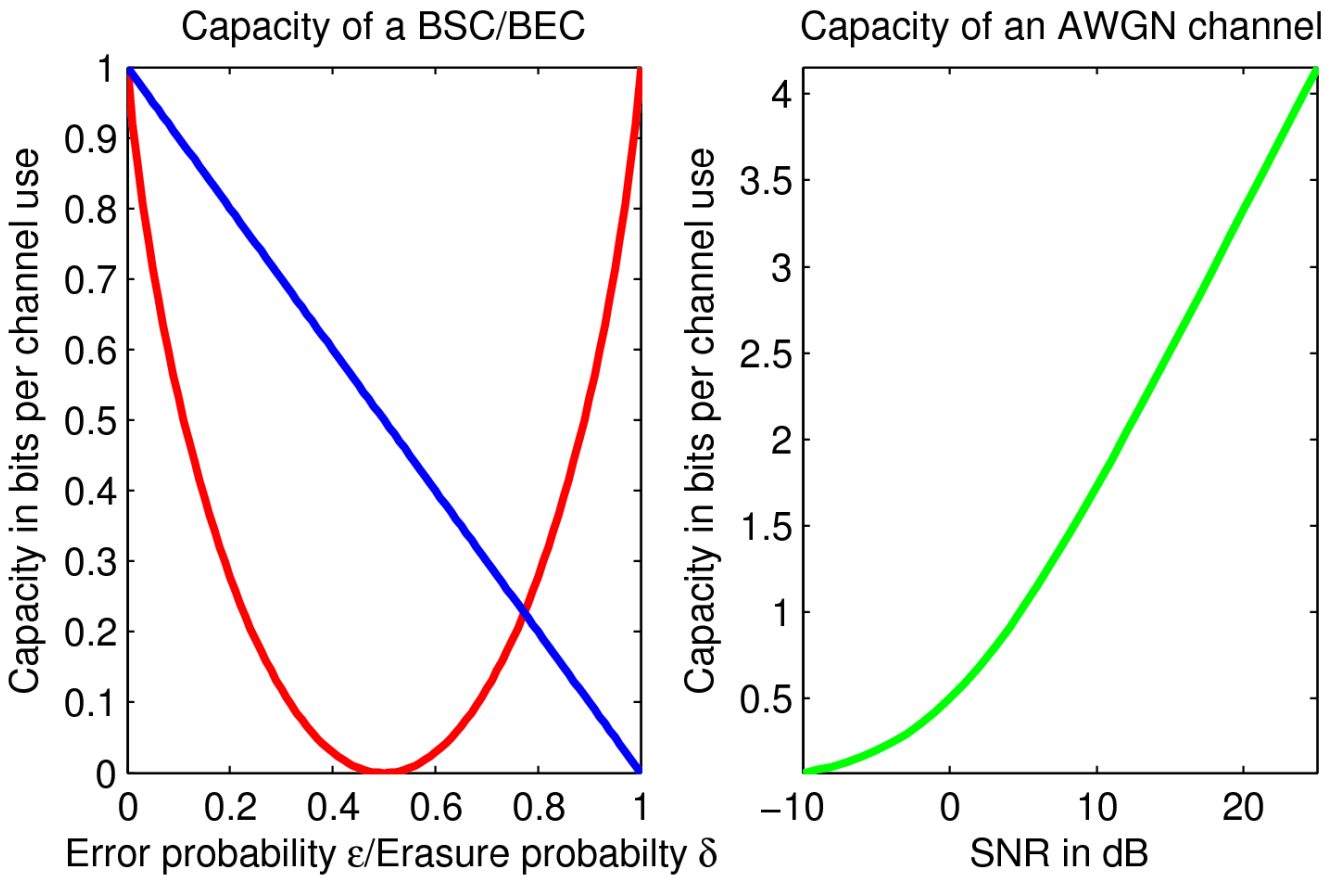
$$C_{\text{BEC}} = 1 - \delta,$$

  where δ is the probabilty of erasure.

- AWGN channel:

$$C_{\text{AWGN}} = \frac{1}{2} \log_2 (1 + \text{SNR})$$

Channel Capcity C, contd.



Capabilities of Coding

Rating of Codes [3]

- Very good codes: Achieve arbitrary small probability of error at any communication rate up to the channel capacity (such codes exist by the noisy-channel coding theorem).
- Good codes: Achieve arbitrary small probability of error at nonzero communication rates up to some maximum rate (that can be less than the channel capacity).
- Bad codes: Achieve arbitrary small probability of error only by decreasing the information rate to zero.
- Practical codes: Can be encoded and decoded in time and space polynomial in the blocklength.

Some codes are only good for some specific channel.

History [3]

- Noisy-channel coding theorem (1948): doesn't tell us how to create capacity-approaching codes (only existence shown)
- 1948: very good cyclic codes exist (nonconstructively).
- 1982: explicit algebraic construction of very good codes for certain channels
- But: no practical decoding algorithm is known for these codes (the general linear decoding problem is NP-complete)
- today: practical, constructive codes are very good codes but costly to decode (although the cost is adequate)

Outline

LDPC Parity Check Matrices [2]

LDPC codes use *very sparse* parity check matrices simply *to make iterative message passing algorithms work well*.

Figure: Sparse parity-check matrix with $N = 20.000$ columns of weight 3 and $M = 10.000$ rows of weight 6 [2].

LDPC Parity Check Matrices, contd. [5]

regular LDPC codes: each row and each column has a fixed small Hamming weight w_{row} and w_{col}, respectively.
irregular LDPC codes: w_{row} and w_{col} are defined to be the average Hamming weight of rows and cols, respectively.

Note on the code rate

Counting the nonzero entries in the $m \times n$ parity check matrix reveals, that

$$n \cdot w_{\text{col}} = m \cdot w_{\text{row}}.$$

Hence, the dimension k of the code is
$\dim(\mathbb{F}^n) - \dim(\text{Ker } \mathbf{G}^t) \geq n - m$ and thus the rate r satisfies

$$r = \frac{k}{n} \geq \frac{n-m}{m} = 1 - \frac{w_{\text{col}}}{w_{\text{row}}}.$$

Construction of LDPC Parity Check Matrices [5], [6]

Intuitive method (more sophisticated methods exist):

1. start with an empty parity check matrix,
2. place the nonzero entries at random (constraint to the condition on the Hamming weights) and
3. modify the resulting matrix to obtain decoding graphs with short cycles.

Matrices with a *girth*, i.e. the length of the shortest cycle in the graph, of length 4 or 6 are usually discarded as they decrease the performance of decoders for two reasons:

- they prevent the used sum-product algorithm from converging and
- the independence of extrinsic information is affected.

Construction of LDPC Parity Check Matrices, contd.

Choice of block-lengths and column weights [2]

- Long block-lengths lead to improved performance (clear by Shannons prove).
- Using an optimal decoder, the best performance would be obtained by using codes closest to random (i.e. large column weight), but we end up with bad decoder performance (using the sum-product algorithm).
- Selecting the optimal (small) column weight is still a heuristic and a tradeoff between performance and decoding complexity.
- Too many columns of weight 2 result in poorer codes (the minimum distance of codes of weight 2 grows logarithmically with the block lengths while for codes with weights ≥ 3 it grows linearly with the code length [4]).

Revision: From Parity Check Matrix to Factor Graph [5]

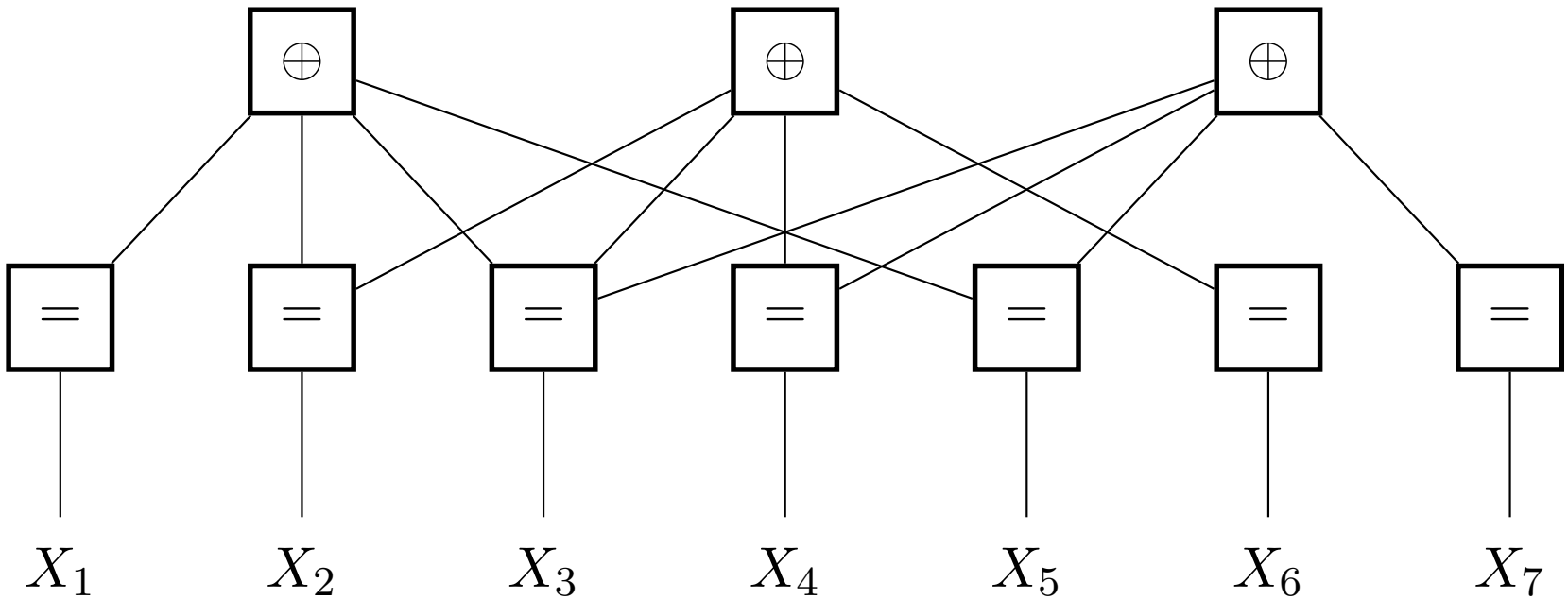
Consider the $(7, 4, 3)$ Hamming code defined by the parity check matrix

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

(length of codeword $n = 7$ bits, length of symbols $k = 4$ bits, minimum hamming weight between each pair of codewords $d = 3$).
Conversion to a factor graph:

1. there is a parity check node for every row of $\mathbf{H}$,
2. there is an equality check node for every column of $\mathbf{H}$,
3. paritiy check i is connected to equality node j iff $H_{ij} = 1$.

Revision: From Parity Check Matrix to FG, contd. [5]



Outline

Iterative Decoding [2]

Considering the construction of LDPC codes two main questions arise:

Theoretical effectiveness

How well would LDPC codes work using the best possible algorithm for decoding?

Details skipped because of time reasons. Short answer: LDPC codes can be very good codes for appropriately chosen parity check matrices and certain channels.

Practical effectiveness

How good can we decode LDPC codes using practical algorithms and how much performance do we loose thereby?

Answered partially during the rest of this representation.

Decoding Problem [2]

Let $\mathbf{G}$ be a generater matrix for the considered code consistent with the parity check matrix $\mathbf{H}$, i.e. $\mathbf{H}\mathbf{G}^t = \mathbf{0}$.
Transmitting a message $\mathbf{t}$ encoded by the code and distorted by noise $\mathbf{n}$ on the channel results in a received symbol

$$\mathbf{r} = \mathbf{G}^t\mathbf{t} + \mathbf{n}.$$

The problem is now to calculate the message $\mathbf{t}'$ as

$$\mathbf{t}' = \arg\max_{\mathbf{x}} P(\mathbf{r}|\mathbf{x}),$$

i.e. the message that leads most likely to the received data.
$\Rightarrow$ This problem can be efficiently solved using the Sum-Product-Algorithm.

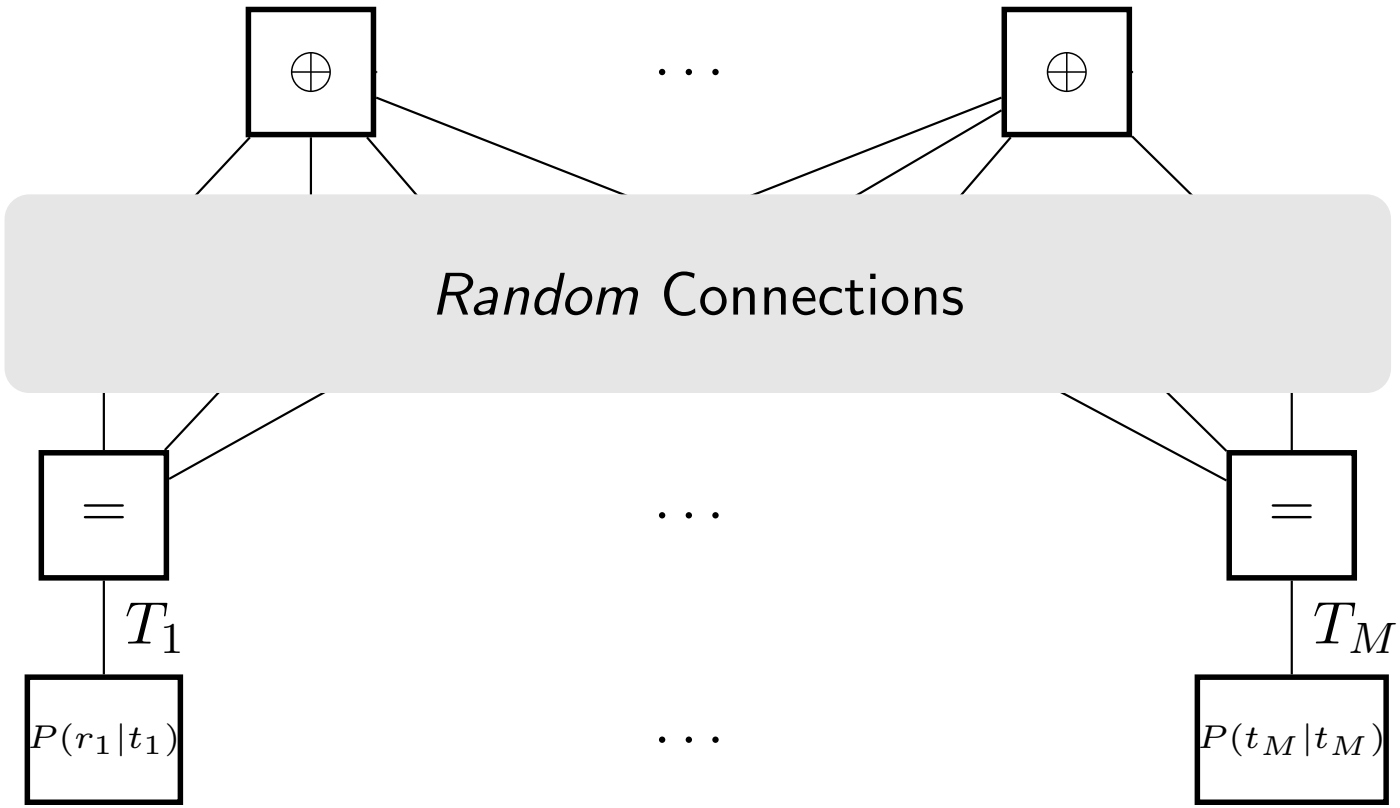
Decoding Problem, contd. [2]

There are two approaches to the decoding problem that both can be viewed as factor graphs:

1. Codeword decoding viewpoint
2. Syndrome decoding viewpoint

Both viewpoints are essentially equivalent and give the same results (up to introduced numerical errors).
We assume a *memoryless* channel and that the distribution of the *bit probabilities* of an encoded symbol is *separable*.

Decoding Problem, Codeword Decoding VP [2]



Decoding Problem, Syndrome Decoding VP [2]

Assume now that we have a *fixed noise model* (given by the channel model):

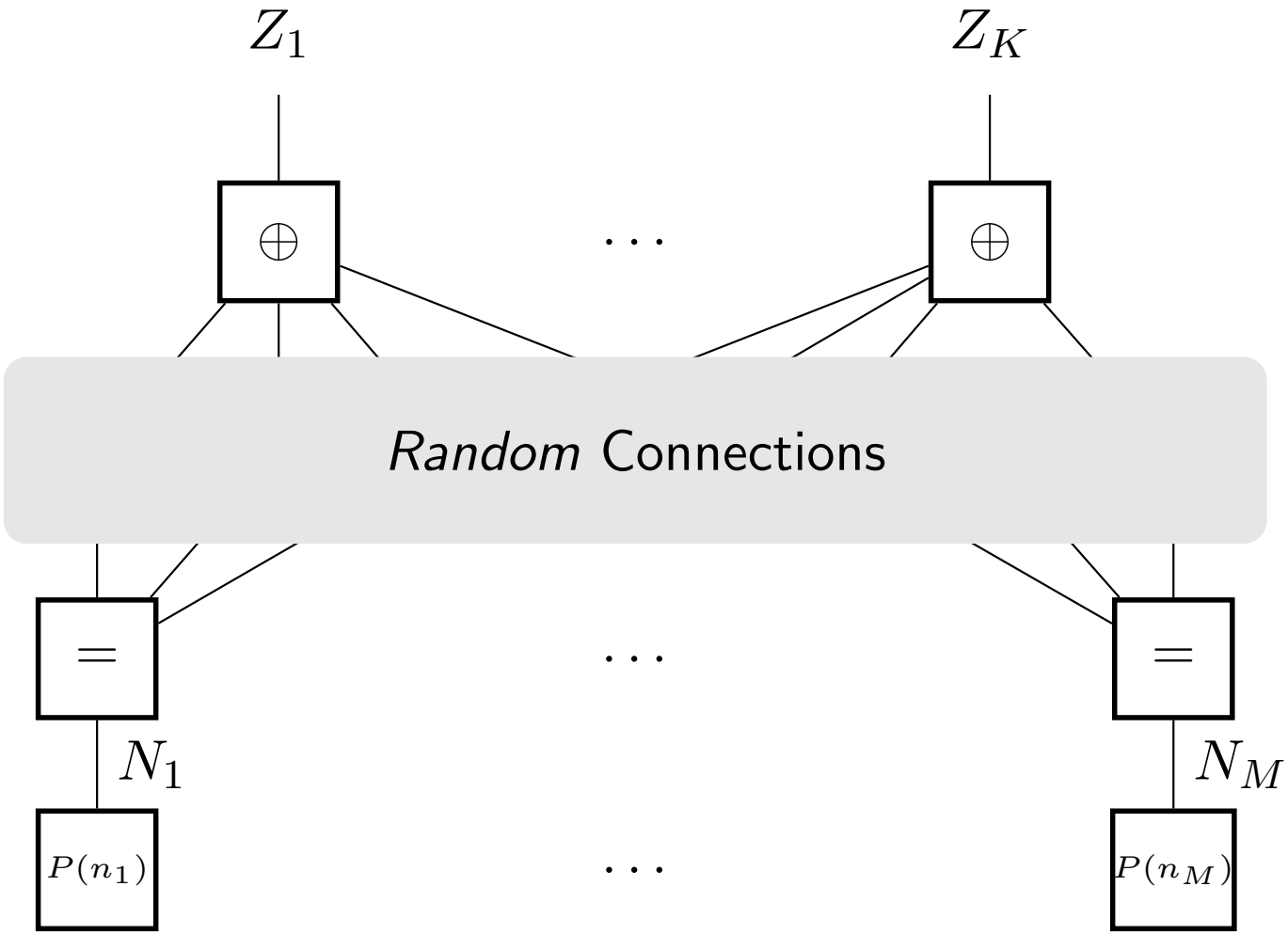
- the transmitted data is *altered by an additive noise vector* $\mathbf{n}$,
- $\mathbf{n}$ is distributed according to $P(\mathbf{n})$ (this distribution can be derived from the channel properties).

Multiplication of the equation for the received signal with the parity check matrix $\mathbf{H}$ from the left yields

$$\mathbf{Hr} = \mathbf{Hn}.$$

Hence, decoding resorts to finding the most probable noise vector such that $\mathbf{Hn}$ equals the syndrome $\mathbf{z} = \mathbf{Hr}$.

Decoding Problem, Syndrome Decoding VP contd. [2]



Decoding with the Sum-Product-Algorithm [2]

We focus on the syndrome decoding viewpoint, i.e. we want to find $\mathbf{n}$ such that

$$P(\mathbf{n}|\mathbf{z})$$

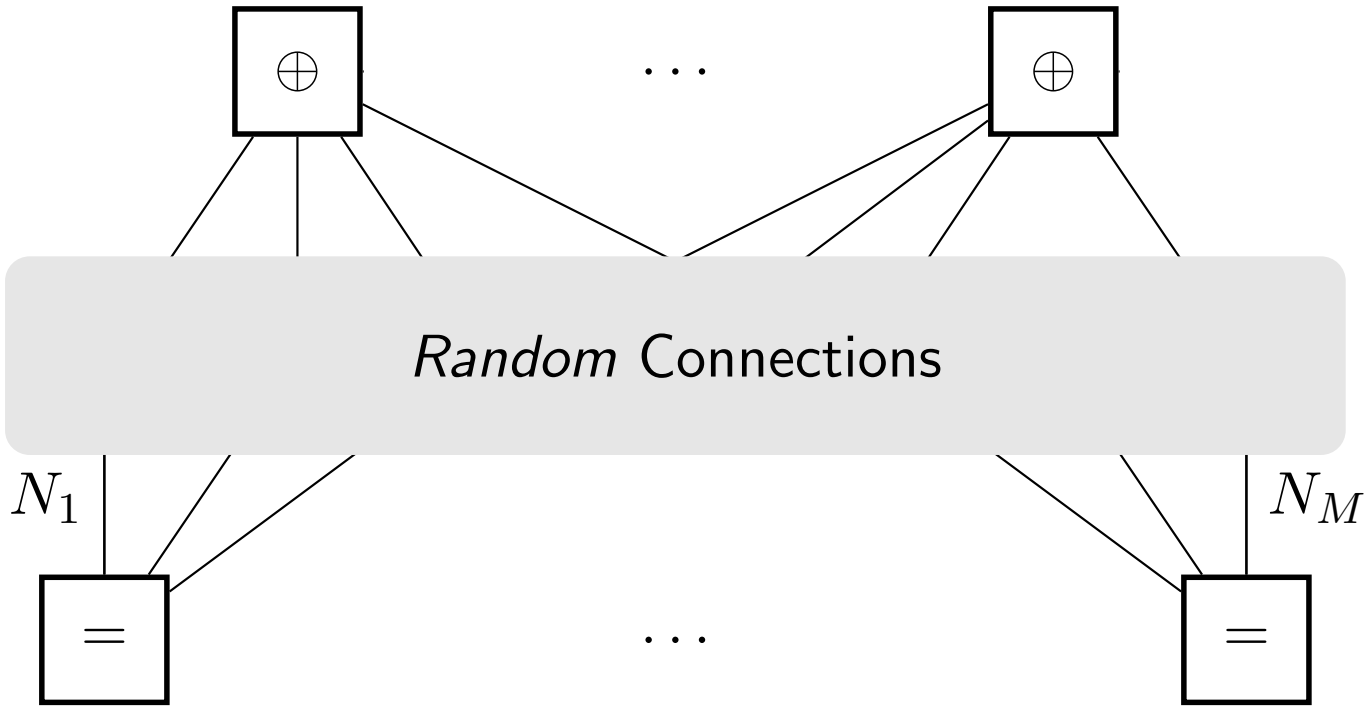
is maximal.
Working out the sum-product algorithm for solving the above problem results in a 3 *step* algorithm:

1. Initialization (only once)
2. Horizontal Step (repeated)
3. Vertical Step (repeated)

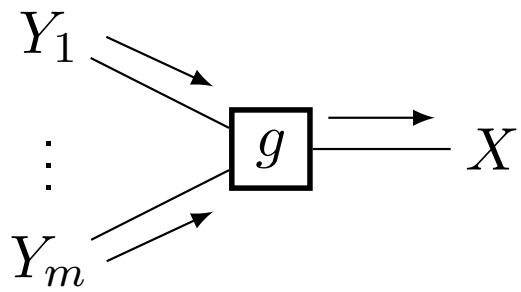
For simplicity we absorb some edges/nodes into the corresponding factors.

Decoding with the Sum-Product-Algorithm, contd. [2]

Finally considered factor graph:



Reminder: Message Passing, Sum-Product Rule [5]



Sum-Product Rule

The message out of some node $g(x, y_1, \ldots, y_m)$ along the edge X is the function

$$\vec{\mu}_X(x) := \sum_{y_1} \ldots \sum_{y_m} g(x, y_1, \ldots, y_m) \vec{\mu}_{Y_1}(y_1) \cdots \vec{\mu}_{Y_m}(y_m),$$

where $\vec{\mu}_{Y_l}$ is the message that arrives at $g()$ along the edge Y_l.

Reminder: Message Passing, Sum-Product Rule, contd. [5]

Further,

- the message out of a terminal node is the function itself.
- messages out of *open* half edges are carrying neutral messages.
- marginals can be computed by multiplying (the correct) messages, i.e.

$$P(X = x) = \overrightarrow{\mu}_X(x) \cdot \overleftarrow{\mu}_X(x),$$

where X is the variable of an edge.

Note: Switch of notation on the slides (to follow MacKays' book [2])

Decoding with the SP-Algorithm, Quantities [2]

We need to define some quantities:

- $\mathcal{N}(i) := \{j | H_{ij} = 1\}$, i.e. $\mathcal{N}(i)$ is the set of bits that participate in parity check i (ith row of $\mathbf{H}$)
- $\mathcal{M}(j) := \{i | H_{ij} = 1\}$, i.e. $\mathcal{M}(j)$ is the set of checks in which bit j participiates (jth column of $\mathbf{H}$)

The algorithm will pass messages along the edges:

- q_{ij}^x is the belief that bit j of $\mathbf{n}$ has the value x given the information obtained by checks other than check i.
- r_{ij}^x is the probability of check i being satisfied if bit j of $\mathbf{n}$ is considered fixed at $\mathbf{x}$ and the other bits have a seperable distribution given by the probabilites $\{q_{ij'} | j' \in \mathcal{N}(i) \setminus j\}$

Decoding with the SP-Algorithm, Initialization [2]

Let $p_j^0 = P(n_j = 0)$ be the prior probability that bit n_j is zero, then

$$p_j^1 = P(n_j = 1) = 1 - p_j^0.$$

For a binary symmetric channel with bit error probability ϵ

$$p_j^1 = \epsilon$$

and for a binary input Gaussian channel with real output p_j^1 will be intialized to the normalized likelihood.
Then, set

$$q_{ij}^0 = p_j^0,\ q_{ij}^1 = p_j^1 \qquad \forall i, j : H_{ij} = 1.$$

Decoding with the SP-Algorithm, Horizontal Step [2]

Horizontal refers to the parity check matrix $\mathbf{H}$, i.e. a row of $\mathbf{H}$. Run through checks i and for all $j \in \mathcal{N}(i)$ the quantities r_{ij}^0 and r_{ij}^1 are updated:

- $$r_{ij}^0 = \sum_{\{n_{j'}|j' \in \mathcal{N}(i)\backslash j\}} P(z_i|n_j = 0, \{n_{j'}|j' \in \mathcal{N}(i)\backslash j\}) \prod_{\{j' \in \mathcal{N}(i)\backslash j\}} q_{ij'}^{n_{j'}}$$

- $$r_{ij}^1 = \sum_{\{n_{j'}|j' \in \mathcal{N}(i)\backslash j\}} P(z_i|n_j = 1, \{n_{j'}|j' \in \mathcal{N}(i)\backslash j\}) \prod_{\{j' \in \mathcal{N}(i)\backslash j\}} q_{ij'}^{n_{j'}}$$

Conditional probabilities are either zero or one, depending on $\mathbf{x}$ satisfying the considered parity check or not.

Decoding with the SP-Algorithm, Vertical Step [2]

Using the quantities r_{ij}^0 and r_{ij}^1 update the belfies q_{ij}^0 and q_{ij}^1:

- $$q_{ij}^0 = \alpha_{ij} p_j^0 \prod_{i' \in \mathcal{M}(j) \backslash i} r_{i'j}^0$$
- $$q_{ij}^1 = \alpha_{ij} p_j^1 \prod_{i' \in \mathcal{M}(j) \backslash i} r_{i'j}^1,$$

where α_{mn} is a scaling factor such that $q_{ij}^0 + q_{ij}^1 = 1$.

Decoding with the SP-Algorithm, Vertical Step, contd. [2]

From the above beliefs we can compute the bit probabilities:

- $$q_j^0 = \alpha_n p_n^0 \prod_{i \in \mathcal{M}(j)} r_{ij}^0$$
- $$q_j^1 = \alpha_n p_n^1 \prod_{i \in \mathcal{M}(j)} r_{ij}^1.$$

These beliefs are then used in the final step (or to decide if another iteration is required or not).

Decoding with the SP-Algorithm, Final Step/Abort [2]

After every iteration, set $\hat{n}_j$ to 1 if $q_j^1 > 0.5$ and to zero otherwise. Then check if

$$\mathbf{H}\hat{\mathbf{n}} = \mathbf{z}.$$

If this is true, accept $\hat{\mathbf{n}}$ as the found noise vector, otherwise continue iterating until some maximum number of iterations is reached. If no valid noise vector was found by then, mark the block as a failure.

Outline

Performance of LDPC Codes [2]

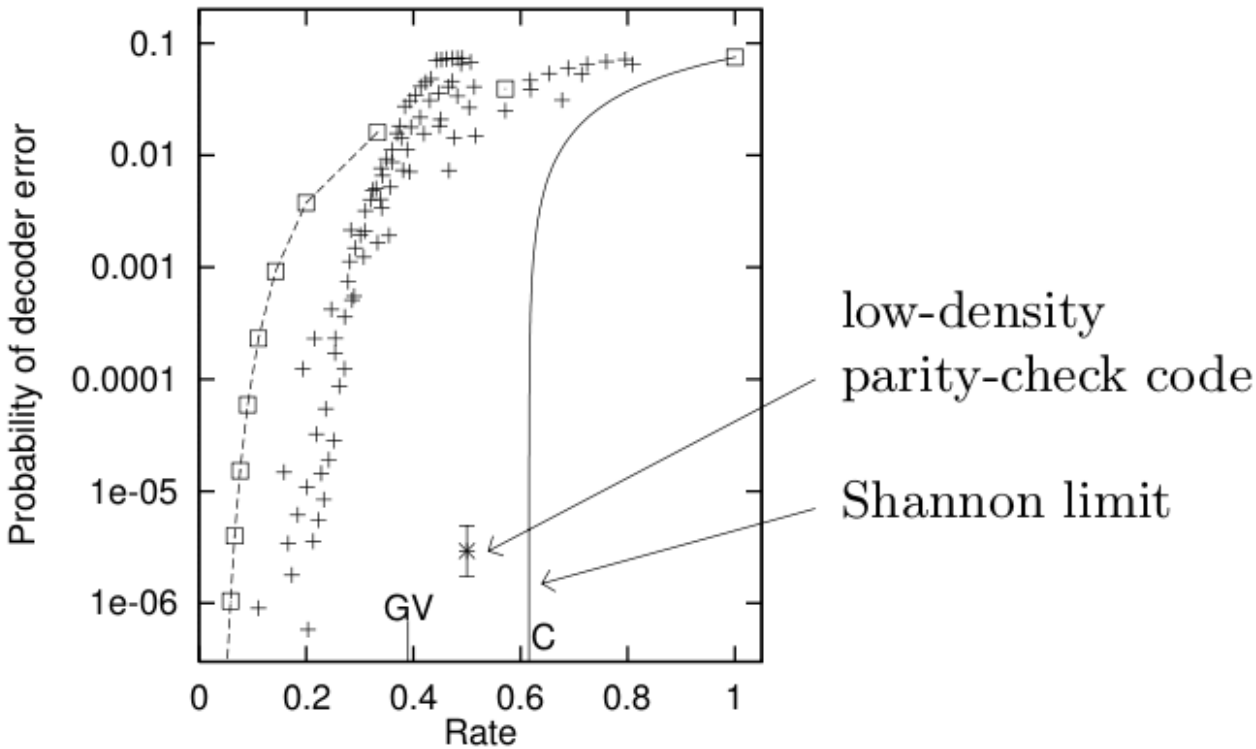


Figure: Error probability of an LDPC code for binary symmetric channel with error probability of 7.5 % compared with algebraic codes. Squares: repetition codes and Hamming (7,4) code; other points: Reed-Muller and BCH codes.

Performance of LDPC Codes, contd. [2]

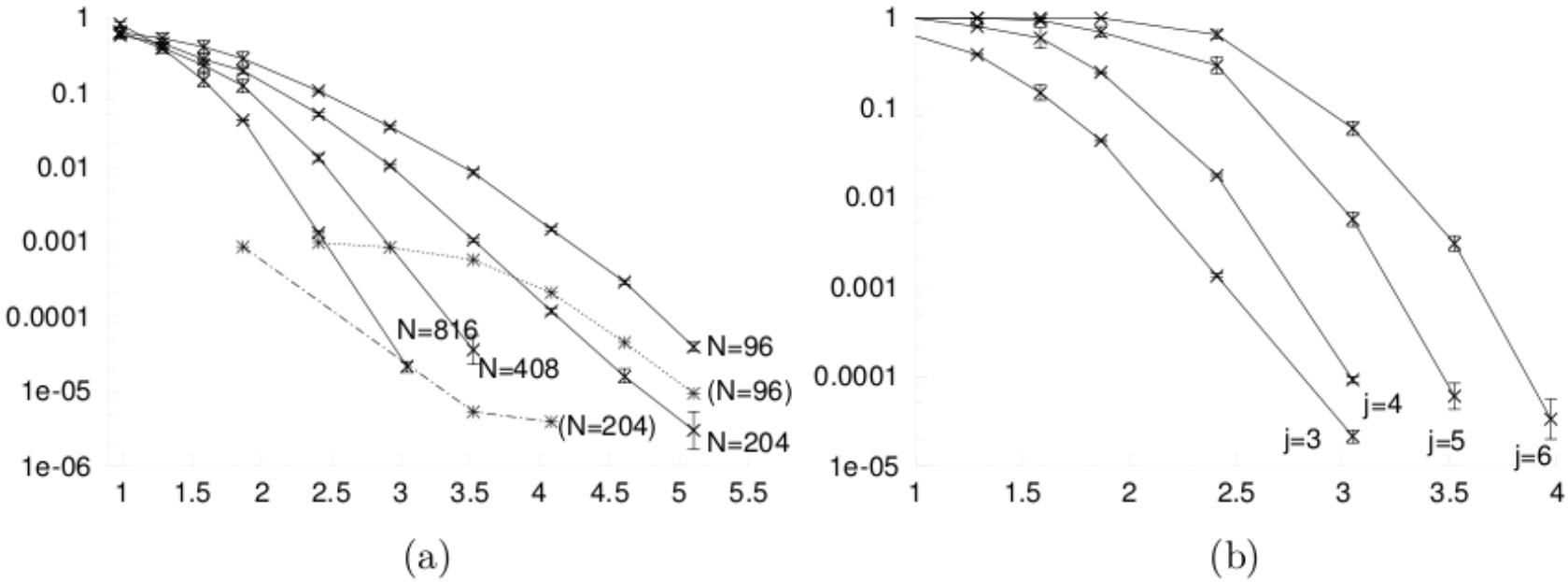


Figure: Performance of rate-1/2 Gallager codes on the Gaussian channel for different configurations of the used LDPC code. The plots show the block error probability versus the SNR E_b/N_0. (a) Dependence on block-length N for codes with column weight 3 and row weight 6. The dashed line is the frequency of undetected errors. (b) Dependence on the column weight j for codes of fixed length $N = 816$.

Outline

Example - Image Transmission, Encoding [2]

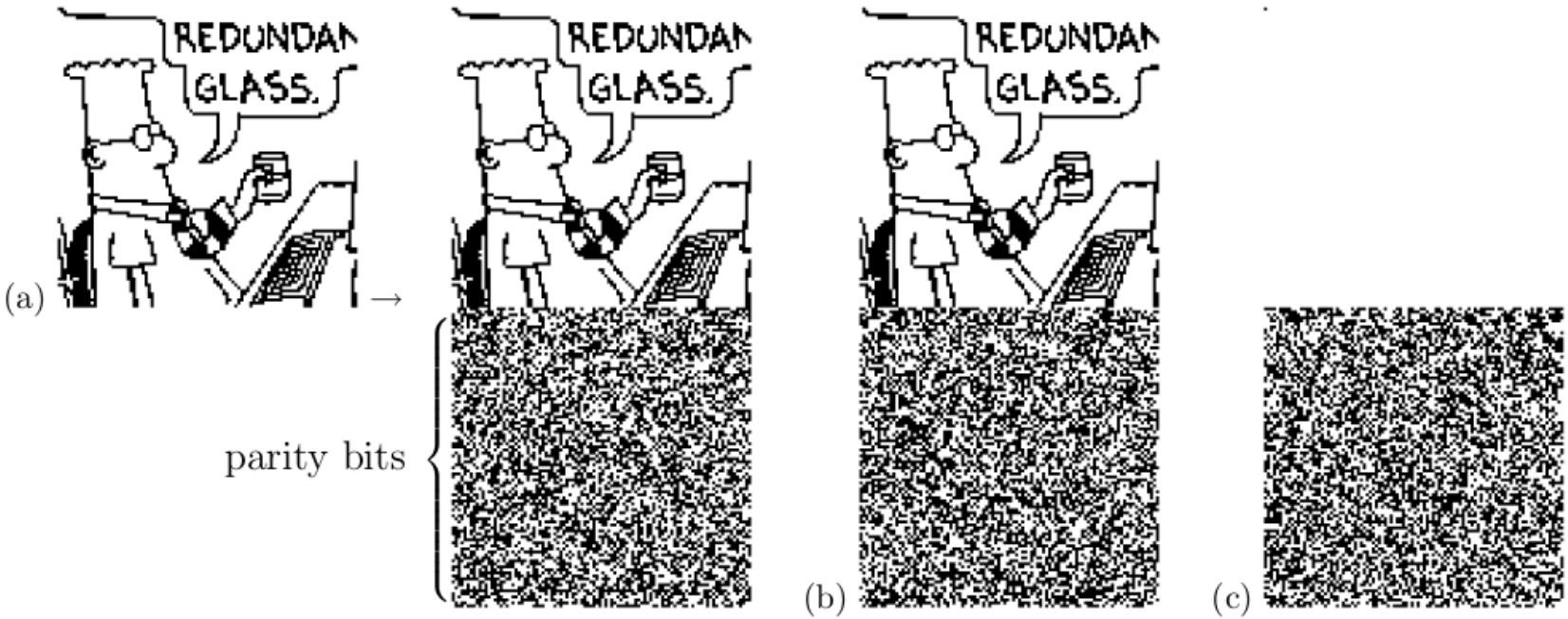


Figure: Encoding data using a rate-1/2 Gallager code with a 10.000×20.000 parity check-matrix with column weight 1. (a) Code generates vectors with 10.000 source and 10.000 parity check bits. (b) Source sequence with altered first bit. (c) Modulo-2 difference of original vector and altered vector.

Example - Image Transmission, Decoding [2]

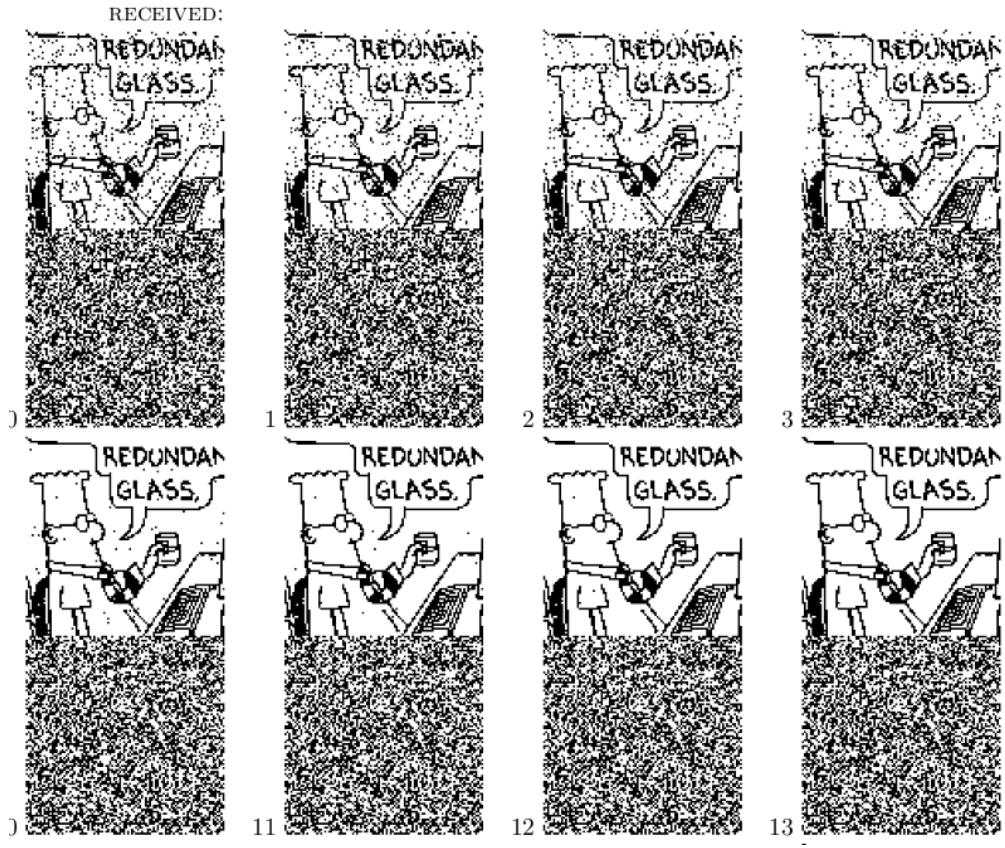


Figure: Iterative probabilistic decoding after transmittion of the above image over a BSC with error probability of 7.5 %. Image shows the best guess after the shown number of decoding iterations. Final decoding is error free.

Matlab Examples

Maybe in January – if I get another timeslot and there is interest.

Outline

Current Research

- Scheduling: Serial versus parallel scheduling (up to two times faster)
- Generation of LDPC parity check matrices (with long cycles; allowing very good codes as block-lengths goes to infinity)
- Improving decoding speed
- Decoding error using fixed point architectures
- Memory efficient decoding
- Column-weight 2 LDPC codes (lower complexity of decoder, potential on partial response channels)
- ...

Outline

Conclusions

LDPC codes ...

- are based on sparse parity check matrices,
- use these sparse matrices to make decoding feasbile.
- are typically decoded using the sum-product algorithm.
- can approach the channel capacity when using long block-lengths.
- are practical codes and can be good (or even very good) codes though.
- are used for reliable forward error correction.
- still come with lots of technical difficulties and open research questions.

Thanks

Thank you for your attention!
Any questions?

General Linear Decoding Problem [2]

Find the maximum-likelihood source vector $\mathbf{s}$ in the equation

$$\mathbf{G}^t\mathbf{s} + \mathbf{n} = \mathbf{r},$$

where $\mathbf{G}$ is the generator matrix, $\mathbf{n}$ is a noise vector and $\mathbf{r}$ is the received vector.

References

X. Hu, E. Eleftheriou, and D.-M. Arnold,

"Regular and Irregular Progressive Edge-Growth Tanner Graphs,"

IEEE Trans. Inform. Theory, vol. 51, pp. 386–398, 2003

D.J.C. MacKay,

"Information Theory, Inference, and Learning Algorithms,"

Cambridge University Press, Hardback, Sep 2003

D.J.C. MacKay,

"Good Error-Correcting Codes Based on Very Sparse Matrices,"

IEEE Tran. Inform. Theory, vol. 45, no. 2, Mar. 1999

G. Malema, and M. Liebelt,

"High Girth Column-Weight-Two LDPC Codes Based on Distance Graphs,"

EURASIP Journal on Wireless Communications and Networking, vol. 2007, article 48158, 5 pages

H.A. Loelliger,

"An Introduction to Factor Graphs,"

IEEE Signal Processing Magazine, 2004

J. Fan,Y. Xiao, and K. Kim,

"Design LDPC Codes without Cycles of Length 4 and 6,"

Research Letters in Communications, vol. 2008, article 354137, 5 pages