



Forschungszentrum Telekommunikation Wien
[Telecommunications Research Center Vienna]


Theory and Design of Turbo and Related Codes

Lecture 4

Gottfried Lechner & Jossy Sayir

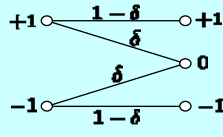
<http://userver.ftw.at/~jossy/turbo/index.html>



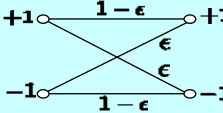
What we have learned last time... 

Channels:

Binary Erasure Channel

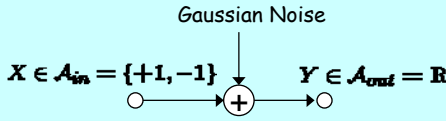



Binary Symmetric Channel



BIAWGN

Gaussian Noise

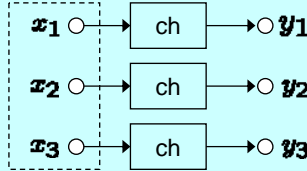


© ftw. 2005 

What we have learned last time...



Log-Likelihood Ratios



$$\begin{aligned}
 L(X_1|Y = \underline{y}) &= \log \frac{P(X_1 = +1|Y = \underline{y})}{P(X_1 = -1|Y = \underline{y})} = \log \frac{P(X_1 = +1)}{P(X_1 = -1)} + \log \frac{P(Y = \underline{y}|X_1 = +1)}{P(Y = \underline{y}|X_1 = -1)} \\
 &= \underbrace{\log \frac{P(X_1 = +1)}{P(X_1 = -1)}}_{\text{a-priori L-value}} + \underbrace{\log \frac{P(Y_1 = y_1|X_1 = +1)}{P(Y_1 = y_1|X_1 = -1)}}_{\text{channel L-value}} + \underbrace{\log \frac{P(Y_{[1]} = y_{[1]}|X_1 = +1)}{P(Y_{[1]} = y_{[1]}|X_1 = -1)}}_{\text{extrinsic L-value}}
 \end{aligned}$$

$$L_{X_1,app}(\underline{y}) = L_{X_1,apri} + L_{X_1,ch}(y_1) + L_{X_1,extr}(y_{[1]})$$

© ftw. 2005



What we have learned last time...



Decoding for Repeat Codes

$$x_1 = x_2 = x_3$$

$$L(X_1|Y = \underline{y}) = L_{X_1,ch}(y_1) + L_{X_1,intr}(y_1) + \sum_{i \neq 1} L_{X_i,intr}(y_i)$$

Decoding for Single Parity-Check Codes

$$x_1 \cdot x_2 \cdot x_3 = +1$$

$$L(X_1|Y = \underline{y}) = L_{X_1,ch}(y_1) + 2 \cdot \tanh^{-1} \left[\prod_{i \neq 1} \tanh \frac{L_{X_i,intr}(y_i)}{2} \right]$$

© ftw. 2005



Inventors of Factor Graphs



<http://www.highteducation.org>

R. M. Tanner,
"A Recursive Approach to Low Complexity Codes"
IEEE Trans. Inform. Theory, Sep. 1981.



Photo by Jessy Savir

B.J. Frey, F.R. Kschischang, and H.-A. Loeliger,
"Factor Graphs and the Sum-Product Algorithm,"
IEEE Trans. Inform. Theory, Feb. 2001.



http://www.isee.org/organizations/history_center/gedec/forney.html

G. David Forney, Jr.,
"Codes on Graphs: Normal Realizations"
IEEE Trans. Inform. Theory, Feb. 2001.

© ftw. 2005



Why Factor Graphs?



- They allow a **graphical representation** of functional dependencies
- Many algorithms can be easily explained and understood by factor graphs (similar to trellis representation and Viterbi decoding)

© ftw. 2005



Graphical Representation



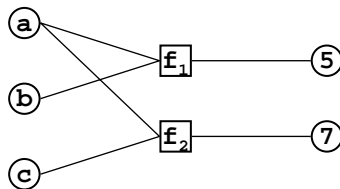
system of equations

$$\begin{aligned} f_1 &: a + b = 5 \\ f_2 &: a + c = 7 \end{aligned}$$

variables: a, b, c, 5, 7
represented by
variable nodes

functions: f_1, f_2
represented by
function nodes

dependencies
represented by
edges



© ftw. 2005



LDPC Code

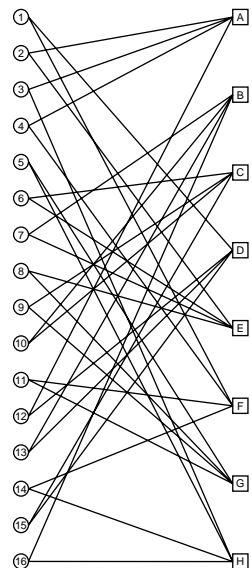


$$x \cdot H^T = 0$$

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

- variable node (column of H)
- function node (row of H)
- edge (a 1 of H)

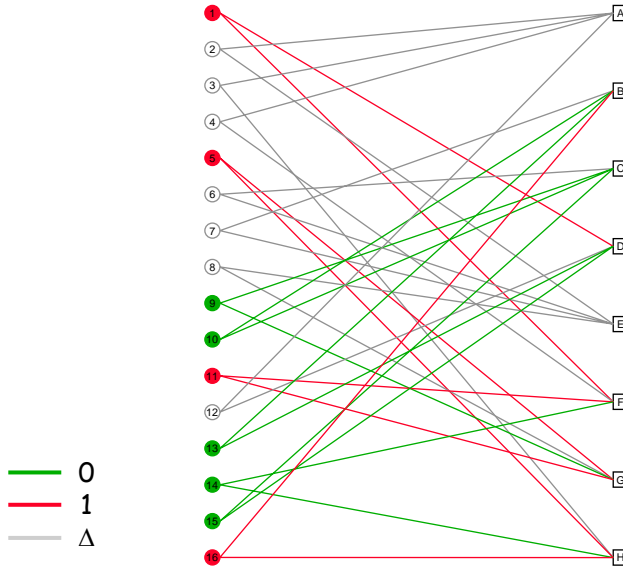
The degree of a node is the number of its connecting edges.



© ftw. 2005



Decoding with Factor Graphs on the BEC



© ftw. 2005

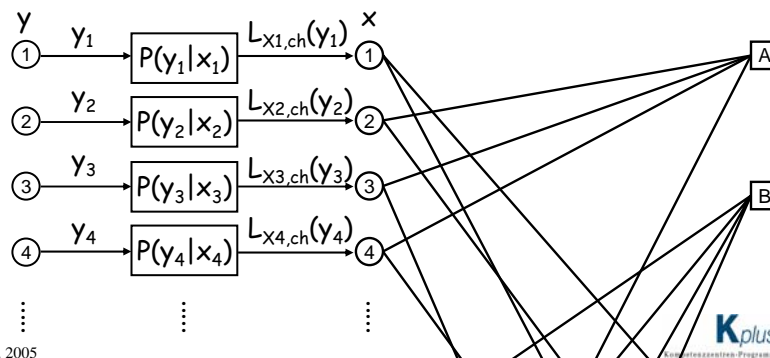


Where is the Channel?



We don't know x but only an erroneous observation y which is the output of the channel.

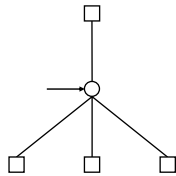
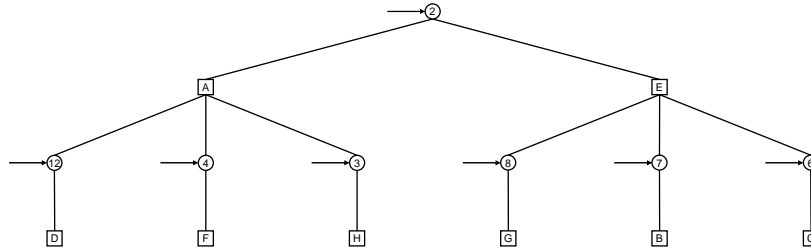
Therefore, just include the channel in the factor graph.
(Assumption: zero a-priori L-value, memoryless channel)



© ftw. 2005

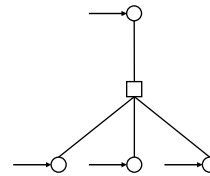


"Tree" Perspective



basic elements

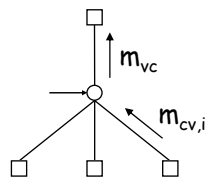
Do you know these codes?



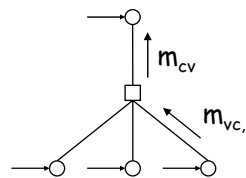
© ftw. 2005

K_{plus}
Kompetenzzentren-Programm

Computation for the BEC



Repeat Code



Single Parity-Check Code

$$m_{vc} = \begin{cases} 0, 1 & \text{if at least one } m_{cv,i} \text{ is no erasure} \\ & \text{or the node itself is no erasure} \\ \Delta & \text{if all } m_{cv,i} \text{ are erasures and the node} \\ & \text{itself is an erasure} \end{cases}$$

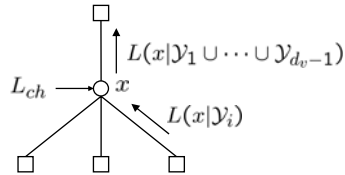
$$m_{cv} = \begin{cases} \sum_i m_{vc,i} & \text{if no } m_{vc,i} \text{ is an erasure} \\ \Delta & \text{if at least one } m_{vc,i} \text{ is an erasure} \end{cases}$$

© ftw. 2005

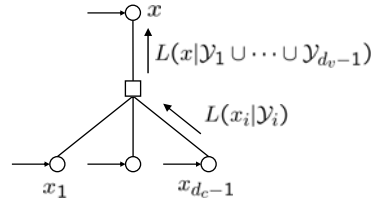
K_{plus}
Kompetenzzentren-Programm

General Computation

Instead of messages 0, 1, Δ we transmit L-values over the edges of the graph.



Repeat Code



Single Parity-Check Code

We can reuse the results from lecture 2 for the repeat and single parity-check code! (Assumption: zero a-priori L-value)

$$\begin{aligned} L(x|Y_1 \cup \dots \cup Y_{d_v-1}) &= \\ &= L_{ch} + \sum_i L(x|Y_i) \end{aligned}$$

$$\begin{aligned} L(x|Y_1 \cup \dots \cup Y_{d_v-1}) &= \\ &= 2 \cdot \tanh^{-1} \left[\prod_i \tanh \frac{L(x_i|Y_i)}{2} \right] \end{aligned}$$

General Computation vs. BEC

$$\begin{aligned} L(x|Y_1 \cup \dots \cup Y_{d_v-1}) &= \\ &= L_{ch} + \sum_i L(x|Y_i) \end{aligned}$$

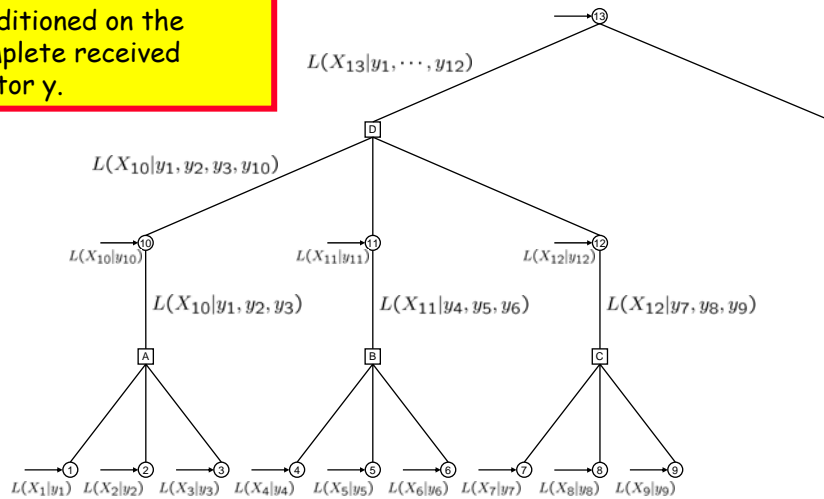
$$\begin{aligned} L(x|Y_1 \cup \dots \cup Y_{d_v-1}) &= \\ &= 2 \cdot \tanh^{-1} \left[\prod_i \tanh \frac{L(x_i|Y_i)}{2} \right] \end{aligned}$$

L-values when transmitting over a BEC are only $-\infty$, 0 and $+\infty$.

Inserting these values in the general formulas leads to the computation we used for the BEC.

Equivalence to ML Symbol Decoding

The result is the L-value of the desired variable conditioned on the complete received vector y .



© ftw. 2005


Kompetenzzentren-Programm

Complexity of "Tree-Decoding"

- We could redraw the graph as a tree for every variable node and use the message passing algorithm on the tree to decode.
- This requires redrawing the graph N times.
- For every graph we have to process all variable nodes and all check nodes. This leads to a complexity proportional to N^2 .

© ftw. 2005


Kompetenzzentren-Programm

Sum-Product Algorithm



- We can do better - like we did when decoding on the BEC as before - by decoding all variable nodes in parallel.
This algorithm is called **Sum-Product algorithm (SPA)**.
- We use flooding scheduling, i.e. we process all variable nodes, then all check nodes, ...
- We stop the algorithm when all check nodes are satisfied or a maximum number of iterations is reached.

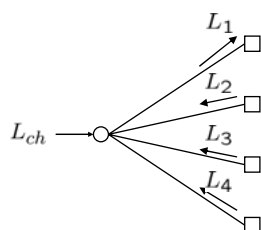
© ftw. 2005

K_{plus}
Kompetenzzentren-Programm

Computations at the Nodes as in the Tree

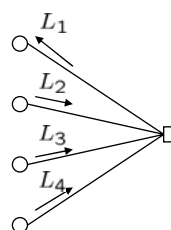


variable nodes



$$L_i = L_{ch} + \sum_{j \neq i} L_j$$

check nodes

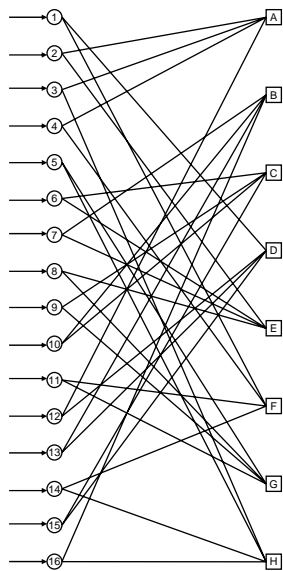


$$L_i = 2 \cdot \tanh^{-1} \left[\prod_{j \neq i} \tanh \frac{L_j}{2} \right]$$

© ftw. 2005

K_{plus}
Kompetenzzentren-Programm

Steps of the SPA



Step 0: receive L-values from the channel

Step 1: process variable nodes

Step 2: interleave messages

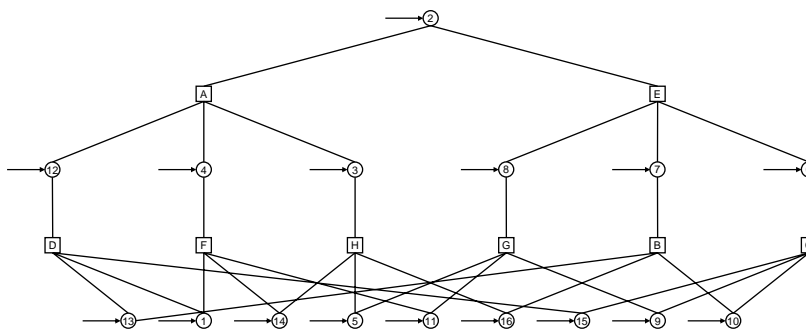
Step 3: process check nodes

Step 4: deinterleave messages

© ftw. 2005

K_{plus}
Kompetenzzentrum-Programm

"Tree" Perspective in Practice



When using random codes with finite block length, the tree perspective is only true for a few steps.

© ftw. 2005

K_{plus}
Kompetenzzentrum-Programm

Bad News

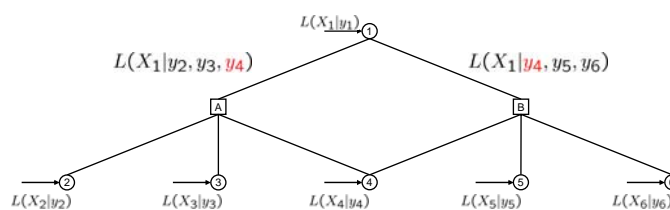


- The tree assumption for random LDPC codes is only fulfilled if the block length goes to infinity. For finite length codes, the graph usually contains closed loops (**cycles**).
- These cycles cause **correlations** between the messages and the assumption of independent observations is not fulfilled.
- Conclusion: The Sum-Product algorithm is **suboptimal**.

© ftw. 2005

K_{plus}
Kompetenzzentrum-Programm

Cycles



Messages arriving at the variable node are not conditioned on independent observations.

The computations at the variable nodes are not correct.

© ftw. 2005

K_{plus}
Kompetenzzentrum-Programm

Good News

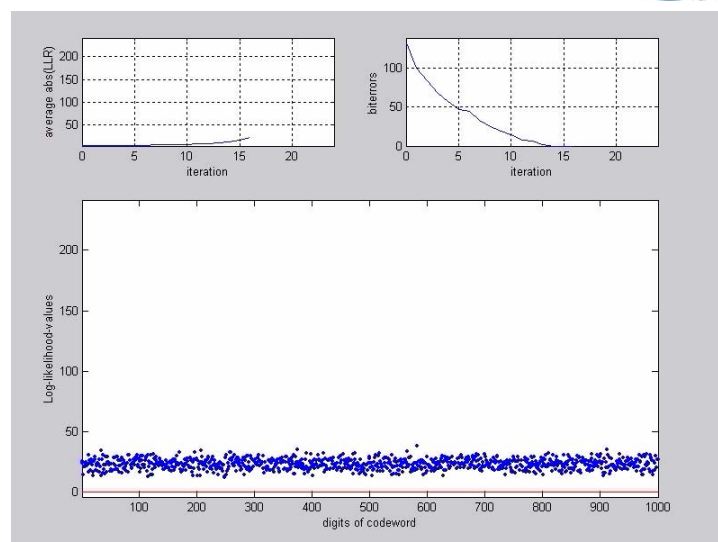


- The Sum-Product algorithm works quite well on graph with cycles if the cycles are not too short.
- In practice the performance of the Sum-Product algorithm is very close to the optimal decoder if the block length is sufficiently large.

© ftw. 2005



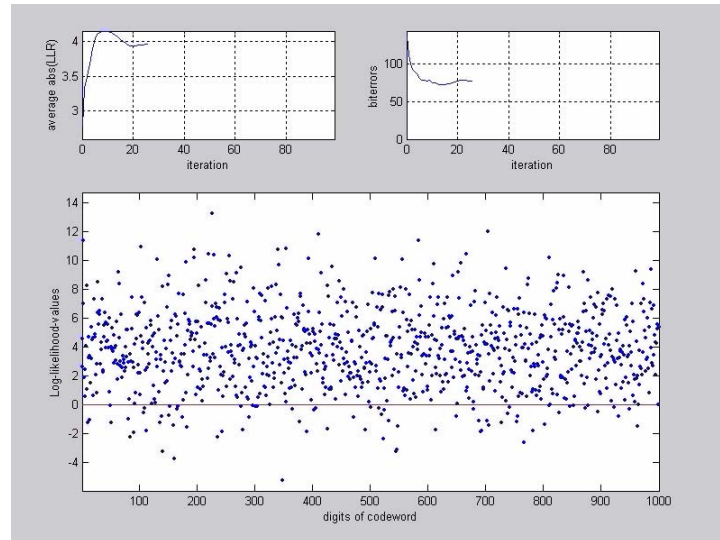
Successful Decoding



© ftw. 2005



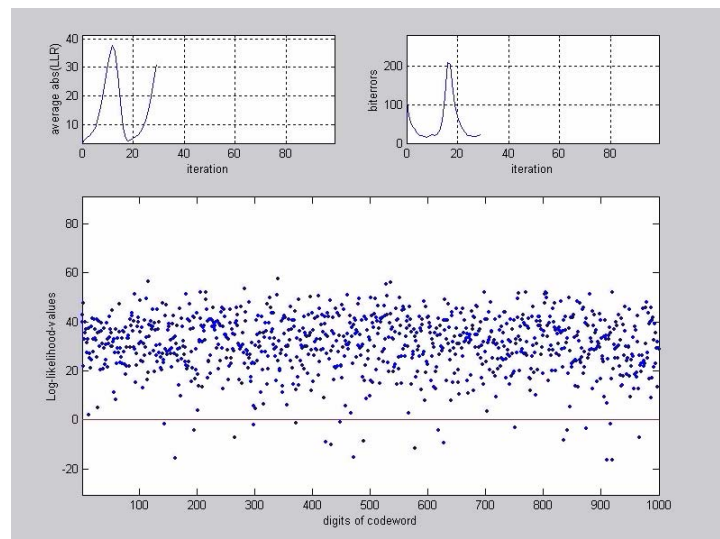
Unsuccessful Decoding



© ftw. 2005



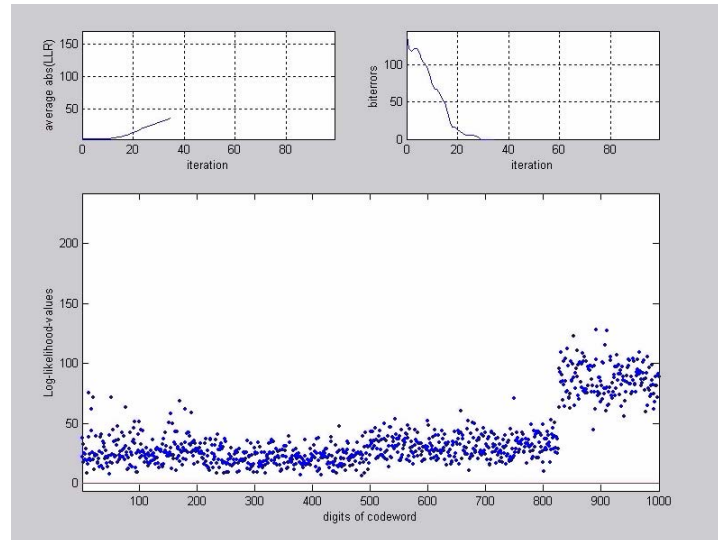
Oscillations



© ftw. 2005



Irregular Codes



© ftw. 2005



Summary



- Factor graphs represent variables, functions and their dependencies
- The factor graph representation of a parity-check matrix can be used for decoding by passing messages along the edges.
- This is optimal if the graph is cycle-free (a tree).
- The algorithm work well if there are cycles that are not too short.
- **Recommended literature:**
H.A. Loeliger, "An Introduction to Factor Graphs", IEEE Signal Processing Magazine, January 2004, pp. 28-41

© ftw. 2005

