
# Theory and Design of

# Turbo and Related Codes

*Lecture 6*

*Jossy Sayir & Gottfried Lechner*

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

Kplus
Kompetenzzentren-Programm

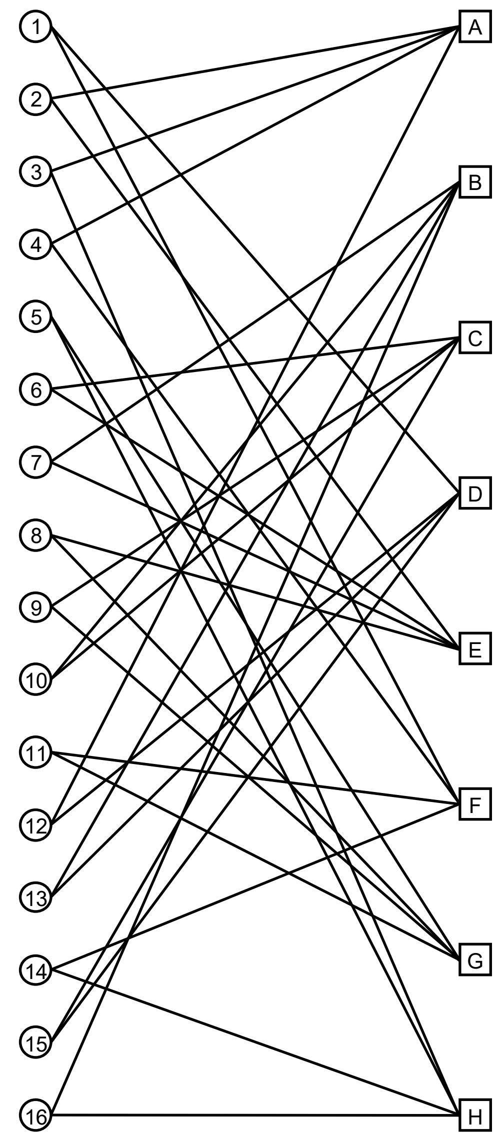
# Concentration theorems

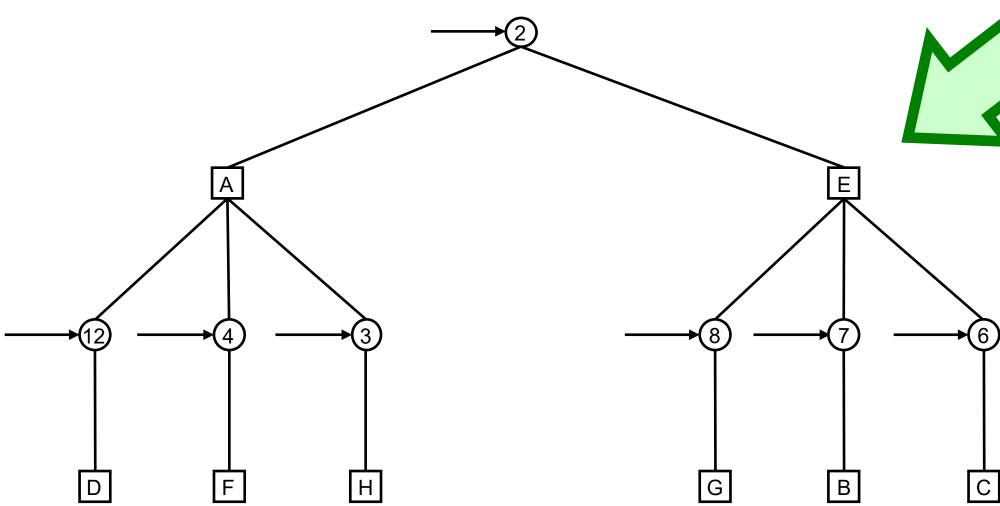
(Luby, Mitzenmacher, Shokrollahi, Spielman / Richardson & Urbanke)

- We will only consider expected decoding behavior over all graphs of a given degree distribution (not the behavior for one specific graph/code)
- Concentration: the probability that the performance of a specific code diverge by $\varepsilon$ from the expected performance over all graphs converges to 0 exponentially in the code length $N$ → expected performance suffices
- Cycle-free behavior: for any iteration number $n_{it}$, one can choose a code length $N$ so that the expected performance at iteration $n_{it}$ is as near as desired to the decoder performance under the tree assumption → tree behavior suffices for $N \to \infty$

# Reminder: Matrix → Graph → Tree

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

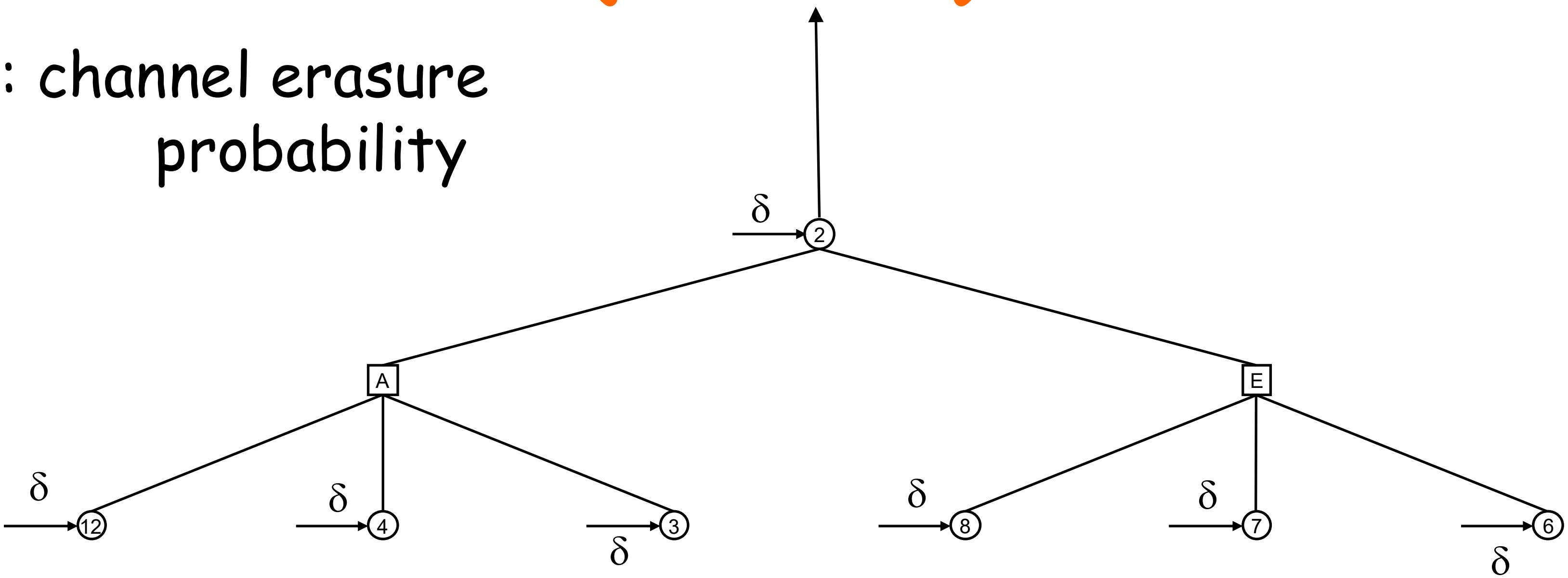


# Tree Decoding Performance for the BEC

P(erasure) = ?

δ: channel erasure probability



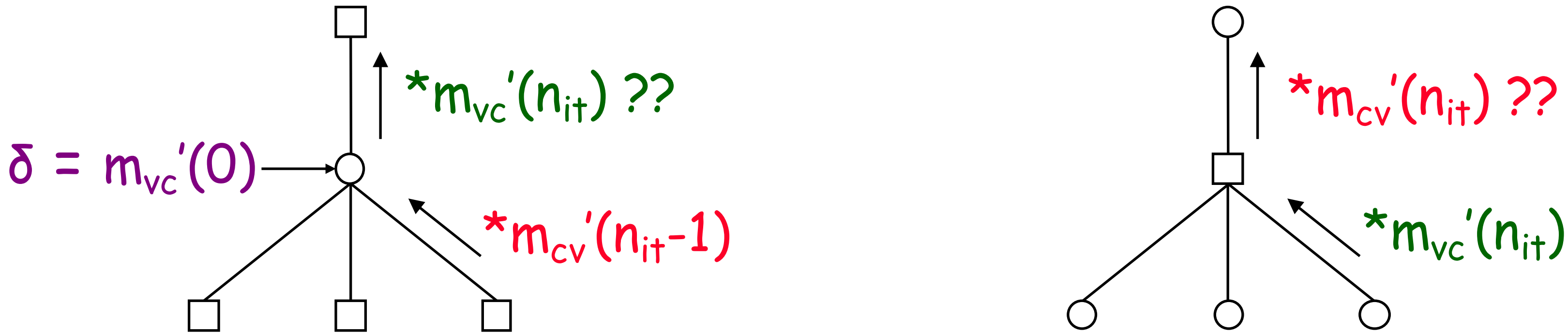
# "Simulating Simulating"
(Tom Richardson, ISIT 2004, Chicago)

- Simulation: run message-passing decoding for codewords transmitted over a binary erasure channel and measure the resulting error probability

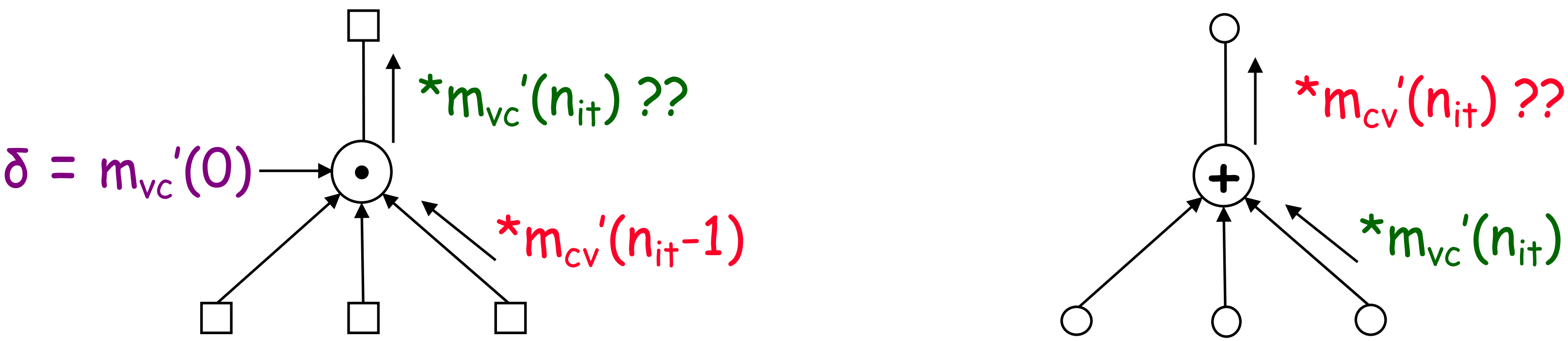
- Simulating simulation: compute probability distributions of messages passing through the decoder

- Instead of $m_{vc}$ and $m_{cv}$, compute $*m_{vc} = P(m_{vc} = 0,1,\Delta)$ and $*m_{cv} = P(m_{cv} = 0,1,\Delta)$

# Combining Erasure Probabilities




Equivalently:

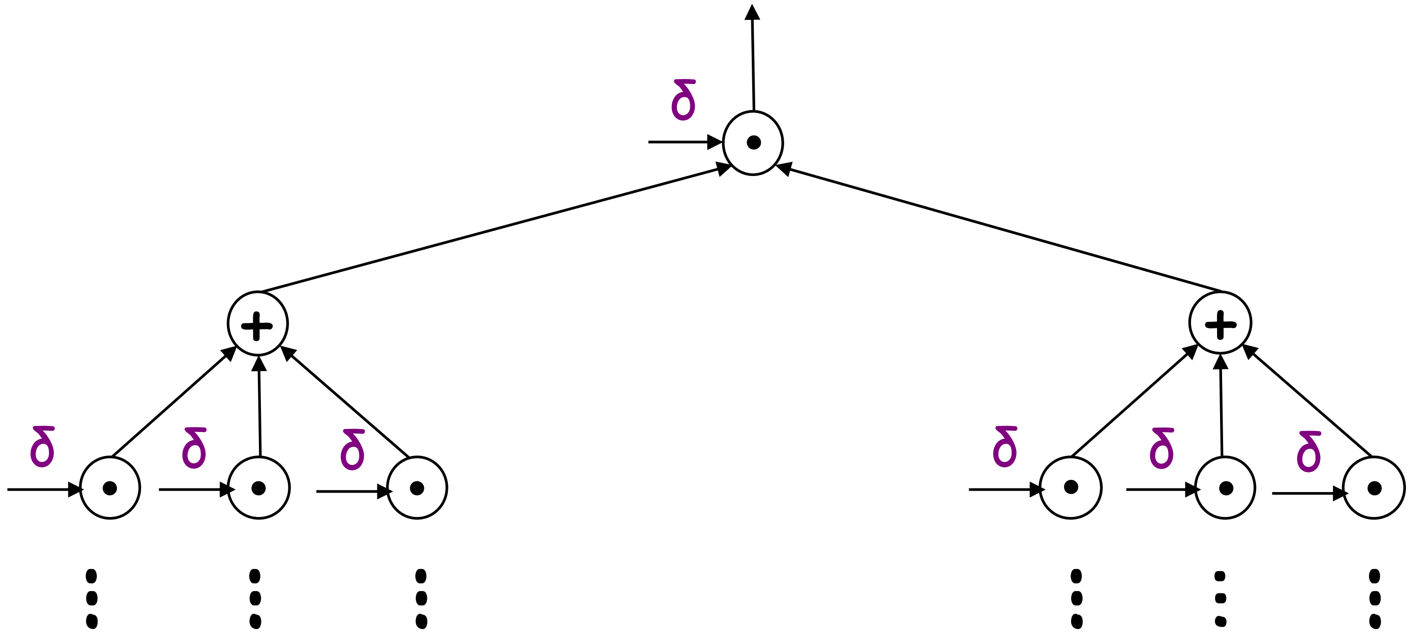



AND - operation

OR - operation

# AND-OR Tree

$P(1) = ?$




Let's simplify notation:

$p_0 = \delta = m_{vc}'(0)$

$q_k = m_{cv}'(k)$ ,

$p_k = m_{vc}'(k)$

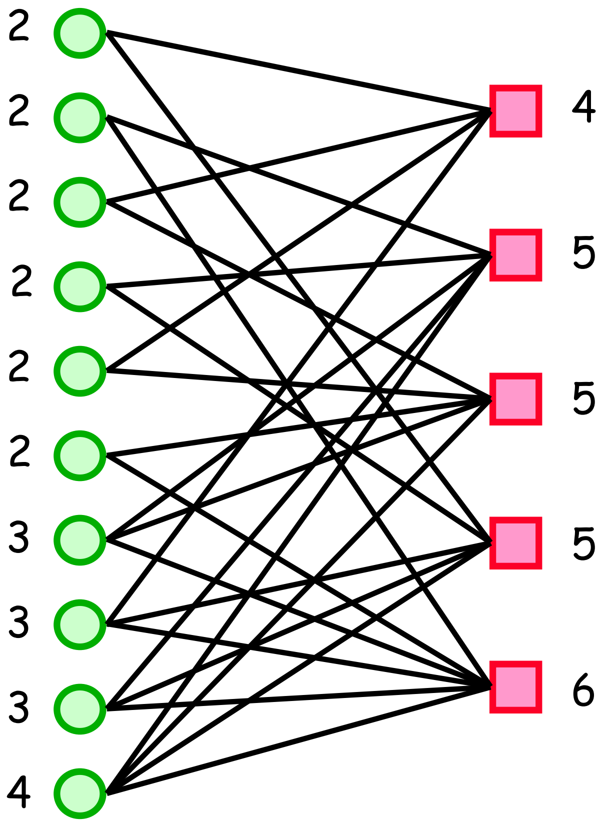
# Solution

$$q_k = 1 - (1 - p_{k-1})^{d_c - 1}$$

$$p_k = p_0 \, q_k^{d_v - 1}$$

$$p_k = p_0 \left(1 - (1 - p_{k-1})^{d_c - 1}\right)^{d_v - 1}$$

# Irregular Graphs




$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} 4 \\ 5 \\ 5 \\ 5 \\ 6 \end{matrix}$$

2 2 2 2 2 2 3 3 3 4

$$(d_{v,1}, d_{v,2}, d_{v,3}, d_{v,4}, d_{v,5}, d_{v,6}, d_{v,7}, d_{v,8}, d_{v,9}, d_{v,10}) = (2, 2, 2, 2, 2, 2, 3, 3, 3, 4)$$

$$(d_{c,1}, d_{c,2}, d_{c,3}, d_{c,4}, d_{c,5}) = (4, 5, 5, 5, 6)$$

# Degree Polynomials

$\ell_i$: proportion of left (variable) nodes with degree $i$

$r_i$: proportion of right (check) nodes with degree $i$

In our example, $\ell_2 = 6/10$, $\ell_3 = 3/10$, $\ell_4 = 1/10$
$r_4 = 1/5$, $r_5 = 3/5$, $r_6 = 1/5$

$\lambda_i$: proportion of edges incident to left nodes with degree $i$

$\rho_i$: proportion of edges incident to right nodes with degree $i$

In our example, $\lambda_2 = 12/25$, $\lambda_3 = 9/25$, $\lambda_4 = 4/25$
$\rho_4 = 4/25$, $\rho_5 = 15/25$, $\rho_6 = 6/25$

# Degree Polynomials (continued)

Total number of edges: #(edges) = $\sum_i i\, r_i = \sum_i i\, \ell_i$

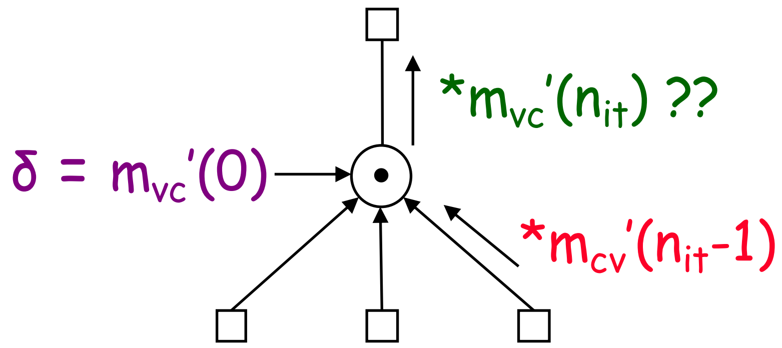
$\lambda_i = i¢\ell_i$ / #(edges)

$\rho_i = i¢r_i$ / #(edges)

$$\lambda(x) = \sum_{i=2}^{d_{v,max}} \lambda_i x^{i-1} \qquad \rho(x) = \sum_{i=2}^{d_{c,max}} \rho_i x^{i-1}$$

$$R = 1 - \frac{\int_0^1 \lambda(x)dx}{\int_0^1 \rho(x)dx}$$

# Recursion for irregular graphs




AND - operation

$d_v$ is a random variable!!

OR-operation is an AND-operation on NOT($m'_{vc}$)

$$p_{k+1} = p_0\, \lambda(1 - \rho(1 - p_k))$$

# Code Optimization

- Linear Programming
- Capacity-achieving power series
- Taylor Expansion

# Fixed Points

- A fixed point is where $p^* = p_0 \lambda(1 - \rho(1 - p^*))$

- Iterative decoding will get stuck if there exists a fixed point $p^*$ such that $0 < p^* \leq p_0$

Iterative Decoding will be successful if
$p < p_0 \lambda(1 - \rho(1 - p))$ for all $0 < p \leq p_0$

- Dual Equation: $q^* > \rho(1 - p_0 \lambda(1 - q^*))$

# Linear Programming

- For a given $p_0$, maximize the rate R
- Choose $\lambda(x)$ and $\rho(x)$ to maximize R so that there are no fixed points
- We can pick either $\lambda(x)$ or $\rho(x)$ and vary the other
- We obtain a linear programming problem under a continuum of conditions
- Discretization of the conditions yields numerical optimization technique

# Achieving Capacity?

- Linear optimization for finite-length polynomials gives excellent results in practice
- The resulting codes do not “achieve” capacity (i.e., no guaranteed success probability for $R = 1 - p_0$)

# Capacity-Achieving Power Series

- Define: $H(D) = 1 + 1/2 + 1/3 + 1/4 + \ldots + 1/D$
- Define:
- Define: