Unequal Error Protection Turbo and LDPC codes

Neele von Deetzen

School of Engineering and Science
Jacobs University Bremen
Germany

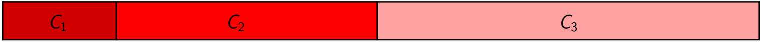
Summer Academy 2007
1.7.-14.7.2007

Overview

- Unequal Error Protection (UEP)
- UEP convolutional and Turbo codes
  - Pruning vs. puncturing
- Bit-irregular UEP LDPC codes
  - Irregular bit-node profile
  - Behaviour and optimisation target
  - Algorithm
- Bit-irregular UEP LDPC codes for higher order constellations
  - Motivation
  - Extended density evolution
- Summary

Unequal Error Protection (UEP)

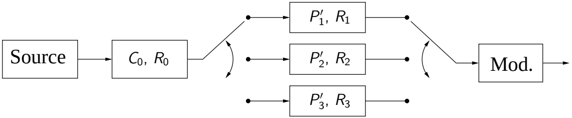
- Data of different importance from source encoder



- Multimedia:
  - Header
  - DC components, magnitude information
  - AC components, high resolution data, position information
- Different protection classes/levels provided by
  - Physical layer: modulation, bit loading, power loading, ...
  - Network layer: network coding, protocols, ...
  - Channel coding: variable rate codes, local properties, ...

Conventional UEP Convolutional Codes

- Goal: a code family with several code rates $R_i = \frac{k_i}{n_i}$
- Well-known: Puncturing
- Increased code rate by excluding code bits from transmission



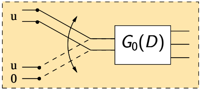
Decoding

- Punctured positions are known to the receiver
- Content is unknown, 0 and 1 are assumed equally likely
- No contribution to decoding

What about reducing the code rate?

New Approach: Pruning

- Reduced code rate by inserting known bits into the info sequence
- Pruning pattern: Insert bits periodically in certain patterns ($\leftrightarrow$ puncturing pattern)
- Fixing bits means pruning paths from the trellis $\longrightarrow$ sub-codes

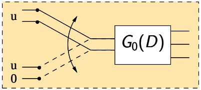


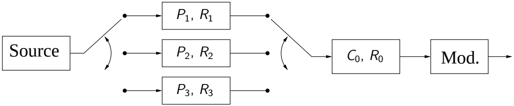
Decoding

- Receiver knows pruned positions and their content
- A-priori knowledge of pruned positions is infinite!
- Turbo codes: Increased extrinsic information!

New Approach: Pruning

- Reduced code rate by inserting known bits into the info sequence
- Pruning pattern: Insert bits periodically in certain patterns (↔ puncturing pattern)
- Fixing bits means pruning paths from the trellis → sub-codes





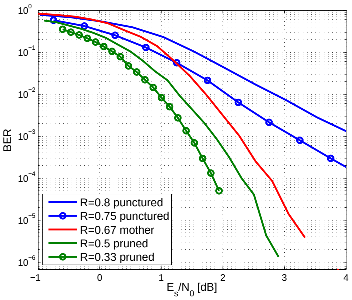
Decoding

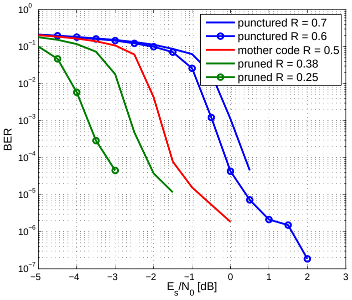
- Receiver knows pruned positions and their content
- A-priori knowledge of pruned positions is infinite!
- Turbo codes: Increased extrinsic information!

Time-Variant Pruning - Results

- Convolutional (NSC) mother code, $R = 2/3$, N $= 200$
- Punctured: $R = 3/4$ and $R = 4/5$
- Pruned: $R = 1/2$ and $R = 1/3$

- Turbo code with (RSC) mother code, $R = 0.5$, N $= 1000$
- Punctured: $R = 0.6$ and $R = 0.7$
- Pruned: $R = 0.25$ and $R = 0.38$





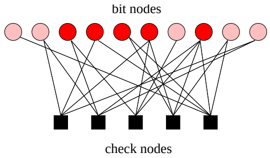
Bit-Irregular UEP LDPC Codes

LDPC codes

- Block codes with sparse parity-check matrix H
- Irregular bit-node profile $\tilde{\lambda}(x)$
- Group bit nodes into protection classes
- Define and optimise bit-node profile for each class $\rightarrow \tilde{\lambda}^{(k)}(x)$

$$\begin{aligned}\tilde{\lambda}(x) &= \tilde{\lambda}_2 \cdot x + \tilde{\lambda}_3 \cdot x^2 + \ldots \\ &= \tilde{\lambda}_2^{(1)} \cdot x + \tilde{\lambda}_2^{(2)} \cdot x + \tilde{\lambda}_3^{(1)} \cdot x^2 + \tilde{\lambda}_3^{(2)} \cdot x^2 + \ldots \\ &= \tilde{\lambda}^{(1)}(x) + \tilde{\lambda}^{(2)}(x)\end{aligned}$$

- Common check-node profile $\tilde{\rho}(x)$ assumed for all classes



Inherent and enhanced UEP of LDPC Codes

Inherent UEP

- Highly connected nodes are protected better!
- Most connected bit nodes assigned to most important class
- Maximise average connection degree of bit nodes in each class

Enhanced UEP

By density evolution, one can show that a class' error probability is minimised by

- maximising the average connection degree $d_{v_{av}}^{(k)}$ and
- maximising the minimum degree $d_{v_{min}}^{(k)}$ of its bit nodes.

Optimisation strategy

Optimisation is done class after class, most important class first!

Objective function and constraints

- Objective functions: $\max d_{v_{av}}^{(k)}$ and $\max d_{v_{min}}^{(k)}$
- Constraints:
  - $C1$ Rate constraint
  - $C2$ Proportion distribution constraints
  - $C3$ Convergence constraint
  - $C4$ Stability condition
  - $C5$ Minimum bit-node degree constraint
  - $C6$ Previous classes constraints

Used optimisation method

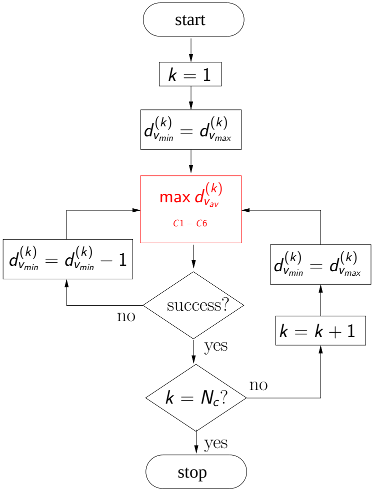
- Linear programming (LIPSOL, simplex)
- Single objective function
- Equality/inequality constraints, lower/upper bounds

Hierarchical Optimisation Algorithm

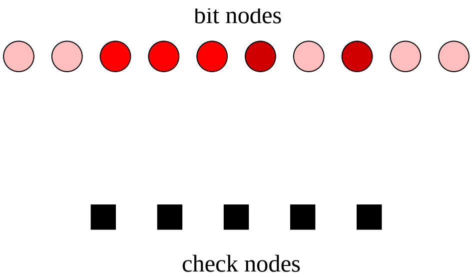
- Given parameters: σ^2 (SNR), $\tilde{\rho}(x)$, N_c, R
- For each class k, starting with the most important class
  - Initialisation $d_{v_{min}}^{(k)} = d_{v_{max}}^{(k)}$
  - While optimisation failure
    - Maximise average connection degree of class k:

      $$\max d_{v_{av}}^{(k)}$$

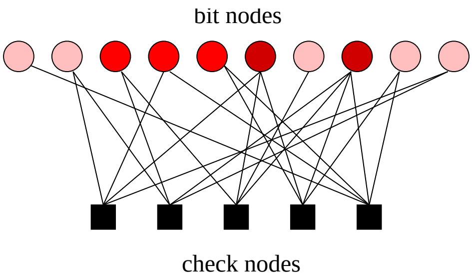
      fulfilling constraints $C1$-$C6$
    - $d_{v_{min}}^{(k)} = d_{v_{min}}^{(k)} - 1$
  - End
- End



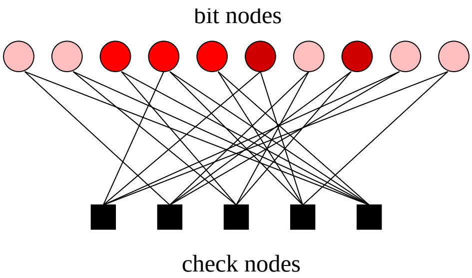
Hierarchical Optimisation Algorithm



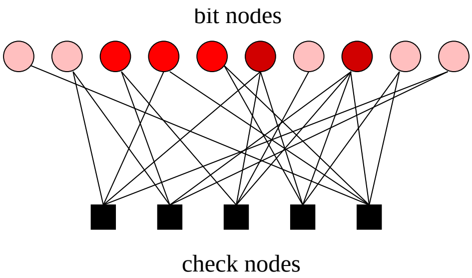
Hierarchical Optimisation Algorithm



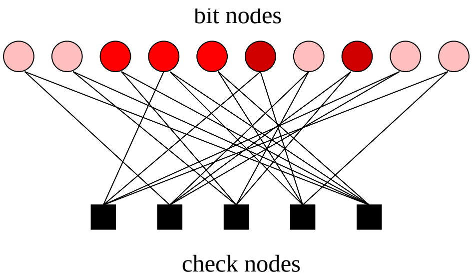
Hierarchical Optimisation Algorithm



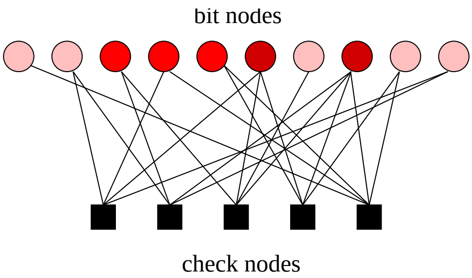
Hierarchical Optimisation Algorithm



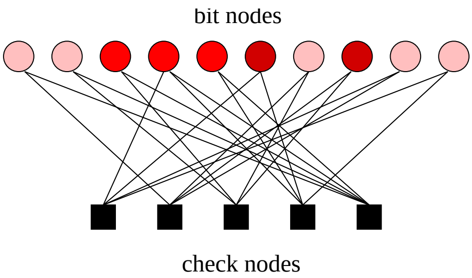
Hierarchical Optimisation Algorithm



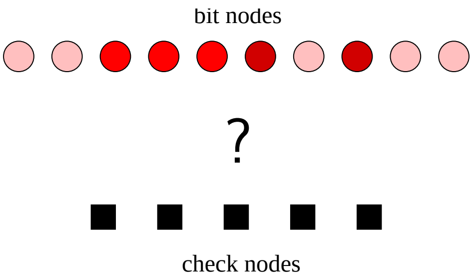
Hierarchical Optimisation Algorithm



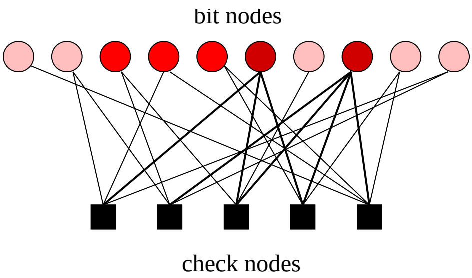
Hierarchical Optimisation Algorithm



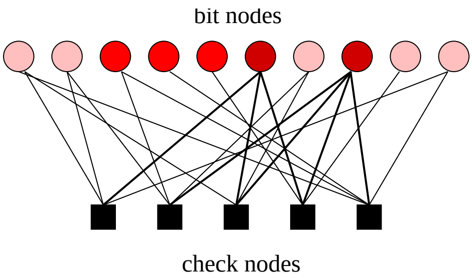
Hierarchical Optimisation Algorithm



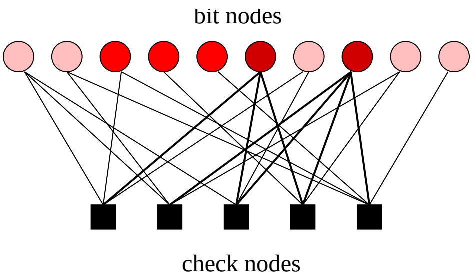
Hierarchical Optimisation Algorithm



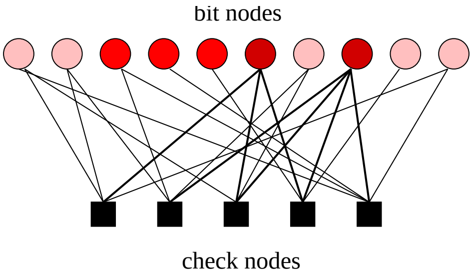
Hierarchical Optimisation Algorithm



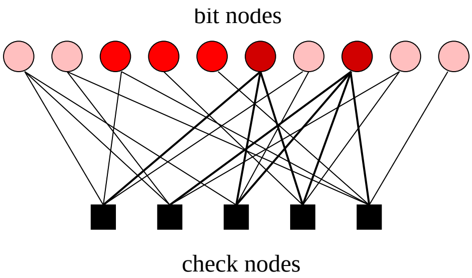
Hierarchical Optimisation Algorithm



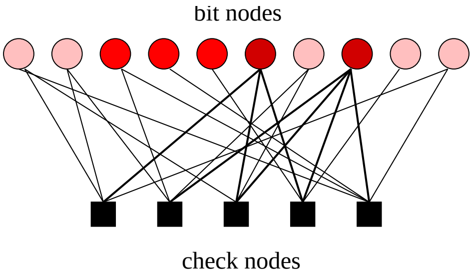
Hierarchical Optimisation Algorithm



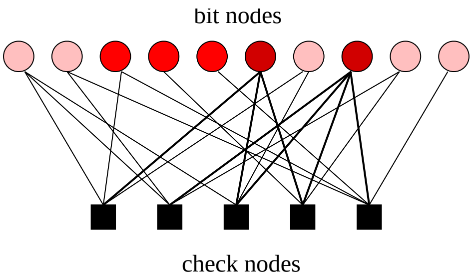
Hierarchical Optimisation Algorithm



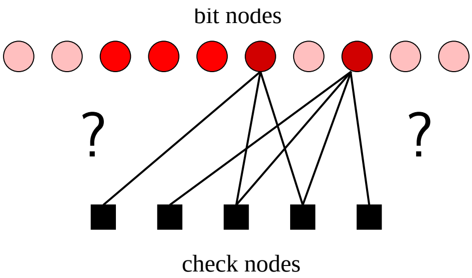
Hierarchical Optimisation Algorithm



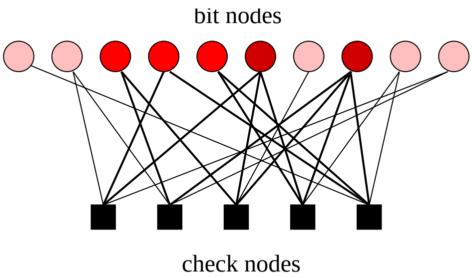
Hierarchical Optimisation Algorithm



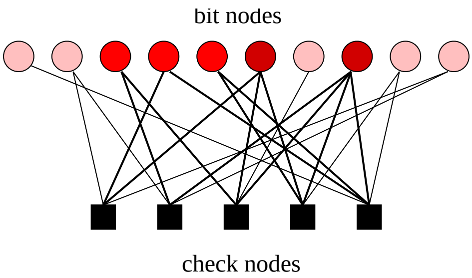
Hierarchical Optimisation Algorithm



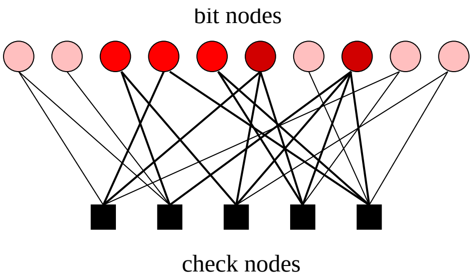
Hierarchical Optimisation Algorithm



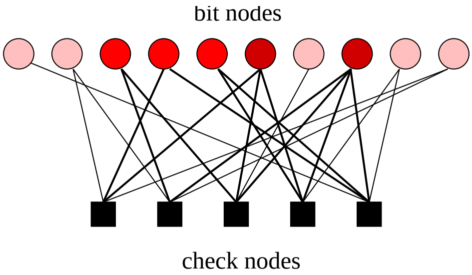
Hierarchical Optimisation Algorithm



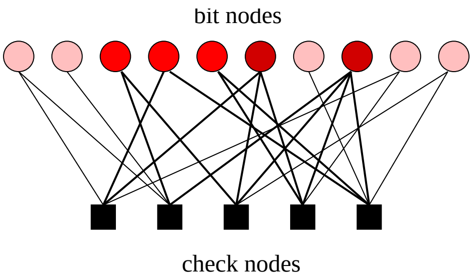
Hierarchical Optimisation Algorithm



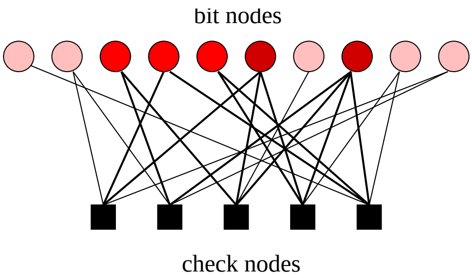
Hierarchical Optimisation Algorithm



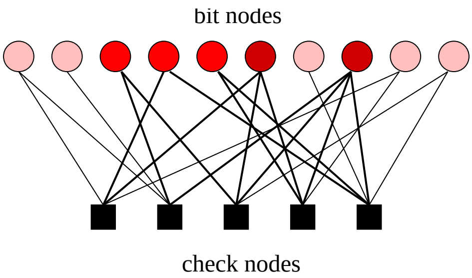
Hierarchical Optimisation Algorithm



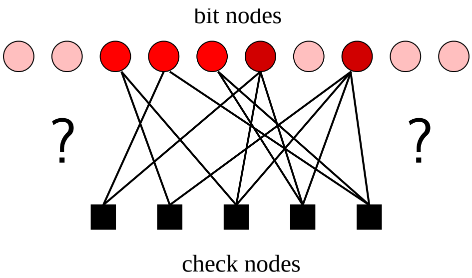
Hierarchical Optimisation Algorithm



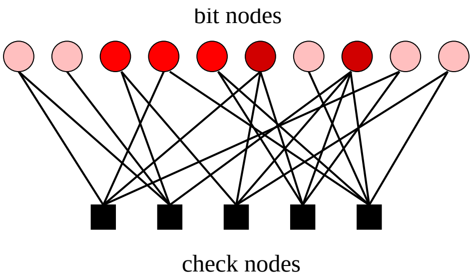
Hierarchical Optimisation Algorithm



Hierarchical Optimisation Algorithm

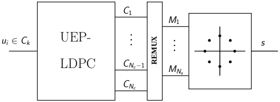


Hierarchical Optimisation Algorithm



Bit-Irregular UEP LDPC Codes for Higher Order Constellations

Motivation



- Heterogeneous bit-error probabilities
- Example 8-PSK:

$$P_{b,d_0} = 1/2 \cdot P_s$$
$$P_{b,d_1} = P_{b,d_2} = 1/4 \cdot P_s$$

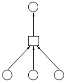
- Conventional LDPC-design assumes homogeneous channel noise variance σ^2 in density evolution
- Noise variance is not any more equal for all bits!

Tasks for improved design

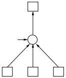
- Find a way to treat bit positions differently
- Find equivalent noise variances for the bit positions

Improved Design

Extended density evolution



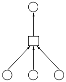
$$I_{cv}^{(l-1)} = 1 - \sum_{t=2}^{d_{c_{max}}} \rho_t \cdot J\left((t-1) \cdot J^{-1}\left(1 - I_{vc}^{(l-1)}\right)\right)$$



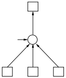
$$I_{vc}^{(l)} = \sum_{i=2}^{d_{v_{max}}} \lambda_i \cdot J\left(\frac{2}{\sigma^2} + (i-1) \cdot J^{-1}(I_{cv}^{(l-1)})\right)$$

Improved Design

Extended density evolution



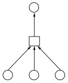
$$I_{cv}^{(l-1)} = 1 - \sum_{t=2}^{d_{cmax}} \rho_t \cdot J\left((t-1)\cdot J^{-1}\left(1 - I_{vc}^{(l-1)}\right)\right)$$



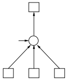
$$I_{vc}^{(l)} = \sum_{k=1}^{N_c} \sum_{i=2}^{d_{vmax}} \lambda_i^{(k)} \cdot J\left(\frac{2}{\sigma^2} + (i-1)\cdot J^{-1}(I_{cv}^{(l-1)})\right)$$

Improved Design

Extended density evolution



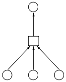
$$I_{cv}^{(l-1)} = 1 - \sum_{t=2}^{d_{c_{max}}} \rho_t \cdot J\left((t-1) \cdot J^{-1}\left(1 - I_{vc}^{(l-1)}\right)\right)$$



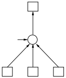
$$I_{vc}^{(l)} = \sum_{j=1}^{N_s} \sum_{k=1}^{N_c} \sum_{i=2}^{d_{v_{max}}} \lambda_i^{(k,j)} \cdot J\left(\frac{2}{\sigma_j^2} + (i-1) \cdot J^{-1}(I_{cv}^{(l-1)})\right)$$

Improved Design

Extended density evolution



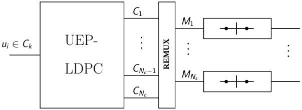
$$I_{cv}^{(l-1)} = 1 - \sum_{t=2}^{d_{c_{max}}} \rho_t \cdot J\left(\left(t-1\right) \cdot J^{-1}\left(1 - I_{vc}^{(l-1)}\right)\right)$$



$$I_{vc}^{(l)} = \sum_{j=1}^{N_s} \sum_{k=1}^{N_c} \sum_{i=2}^{d_{v_{max}}} \lambda_i^{(k,j)} \cdot J\left(\frac{2}{\sigma_j^2} + (i-1) \cdot J^{-1}(I_{cv}^{(l-1)})\right)$$

Equivalent noise variances σ_j^2

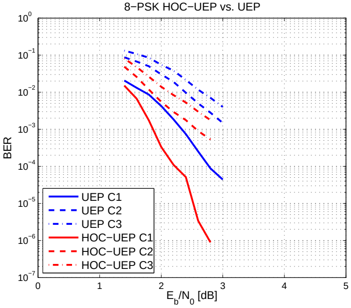
- Gray labelling: $\log_2 M$ equivalent BPSK channels
- Equivalent noise variances based on bit-error probabilities
- $\sigma^2 \overset{\text{8-PSK}}{\equiv} P_s \overset{\text{Gray}}{\longrightarrow} P_{b,d_i} \overset{\text{BPSK}}{\equiv} \sigma_j^2$

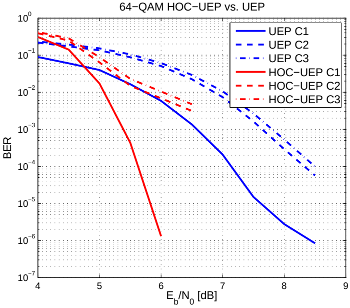


UEP LDPC codes for HOC - Results

We compare

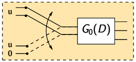
- UEP codes designed for BPSK (UEP) with
- UEP codes designed for 8-PSK/64-QAM (HOC-UEP).
- Both schemes are used with higher order constellations.
- $N_c = 3,\ n = 4096,\ R = 0.5,\ N_{it} = 20$



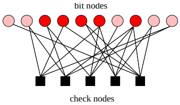


Summary

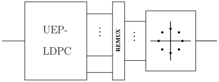
- Pruned UEP convolutional and Turbo codes
  - → Pruning through inserting zeros into the info sequence
  - → Set of BER curves in both directions



- Bit-irregular UEP LDPC codes
  - → Different bit-node profiles for protection classes
  - → Hierarchical optimisation algorithm



- UEP LDPC codes for higher order constellations
  - → Awareness of heterogeneous noise variance
  - → Extended density evolution



Thank you!