On the construction of Tanner graphs

Jesús Martínez Mateo
Universidad Politécnica de Madrid
Outline

• Introduction
  – Low-density parity-check (LDPC) codes
• LDPC decoding
  – Belief propagation based algorithms
• Ensembles of LDPC codes
• LDPC code construction
  – PEG algorithm
  – Cycles, stopping and trapping sets
• Simulation results
• Summary
Low-density parity check codes

INTRODUCTION
Low-density parity-check (LDPC) codes

Introduced by Robert G. Gallager in 1963, but neglected for years. Rediscovered in 90s by MacKay & Neal, and quickly showed that irregular LDPC codes easily outperform the best turbo codes.
Linear codes (encoding)

- Error correction using **parity-checks**

  Multiple constraints (parity-check equations) often written in matrix form, $H$, **parity-check matrix**, a valid codeword $x$ then satisfies

\[
Hx^t = 0
\]

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
\end{pmatrix} = \begin{pmatrix}
0 \\
0 \\
0 \\
\end{pmatrix}
\]

\[
x_3 = x_1 \\
x_4 = x_2 \\
x_5 = x_1 + x_2
\]
Linear codes (decoding I)

• **Maximum likelihood decoder**
  
  Knowing the binary received string, $y$, the best decoder will choose the codeword closest in *Hamming distance* to $y$ (or randomly one of them)

  – *Optimal*, but **too computational expensive**

  The received string has to be compared to every other codeword in the code

---

**Classical block codes**

✓ usually short, and
✓ algebraically designed
Linea codes (decoding II)

• Alternatives:
  
  – Iterative decoding
    Using a graphical representation of the parity-check matrix.
    e.g. message passing algorithms
    Operate by passing messages along the edges of the Tanner graph.
Tanner or bipartite graphs

- The graph consists of two sets of nodes commonly referred to as:
  - Variable, bit or symbol nodes (for the codewords)
  - Check or parity-check nodes (for the parity-check equations)

\[ H = \begin{pmatrix}
1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix} \]
Cycles, Lollipops and Girth

- A **cycle** in a Tanner graph is a sequence of connected nodes that start and end at the same node, and contain other vertices no more than once.
  - The length of a cycle is the number of edges it contains.

- The **girth** is the size of the smallest cycle in the graph.

---

Junio 2013
Message-passing algorithms
Belief propagation decoding
Sum-product algorithm

LDPC DECODING
Message passing algorithms

- Operate by passing messages along the edges of the Tanner graph
  - the decoding performance depends on the number of edges
- Also known as iterative decoding algorithms:
  - messages from symbol (check) nodes to check (symbol) nodes are exchanged iteratively until a result is achieved
- Obey the extrinsic information principle: only extrinsic information is passed along
  i.e. the outgoing message on an edge $e$ is a function of all the incoming messages except the message received on $e$ and the received value in the case of messages from symbol to check nodes
Belief propagation decoding

• Different algorithms are considered depending on the information exchanged in the passed message:
  – Bit flipping and belief propagation decoding are well known message passing algorithms

• **Belief propagation** decoding:
  – Messages are **probabilities** which represent the level of belief on a codeword bit value
  – Variants: **sum-product** and min-sum (Viterbi) algorithms

Sum-product algorithm: Initialization

Compute the prior probabilities:

\[ p_n^0 = \Pr(x_n = 0) \]
\[ p_n^1 = \Pr(x_n = 1) = 1 - p_n^0 \]

Initialize messages from symbols:

\[ q_{m,n}^0 = p_n^0 \]
\[ q_{m,n}^1 = p_n^1 \]

Sum-product algorithm: Horizontal step (1)

\[ r_{m,n}^0 = \sum \text{Pr}(z_n | x_n = 0) \prod q_{m,n'}^{x_n'} \]

\[ r_{m,n}^1 = \sum \text{Pr}(z_n | x_n = 1) \prod q_{m,n'}^{x_n'} \]

Sum-product algorithm: Vertical step (2)

\begin{align*}
q_n^0 &= \alpha_n \prod_{m \in \mathcal{M}(n)} r_{m,n}^0 \\
q_n^1 &= \alpha_n \prod_{m \in \mathcal{M}(n)} r_{m,n}^1
\end{align*}

Pseudo posterior probabilities:

\begin{align*}
q_{m,n}^0 &= \alpha_{m,n} p_n^0 \prod_{m' \in \mathcal{M}(n) \setminus n} r_{m',n}^0 \\
q_{m,n}^1 &= \alpha_{m,n} p_n^1 \prod_{m' \in \mathcal{M}(n) \setminus n} r_{m',n}^1
\end{align*}

ENSEMBLES OF LDPC CODES

Generating polynomials
Symbol and check node degree distributions
Ensembles of LDPC codes

- Usually used the ensemble of all possible codes with certain parameters (e.g., the degree distribution of symbol and check nodes) instead of a particular parity-check matrix

- Generating polynomials

  Two polynomials $\lambda(x)$ and $\rho(x)$ representing symbol and check node degree distributions, respectively

  \[
  \lambda(x) = \sum_i \lambda_i x^{i-1}
  \]

  \[
  \rho(x) = \sum_j \rho_j x^{j-1}
  \]

  s.t. $\sum_i \lambda_i = 1$, $\sum_j \rho_j = 1$

  i.e. $\lambda_i$ ($\rho_i$) denote the fraction of edges connected to $i$-degree ($j$-degree) symbol (check) nodes
Density and differential evolution

• The asymptotic performance (capacity) of a family or ensemble of LDPC codes can be determined using the density evolution [1]
  – The algorithm analyzes the convergence of a particular degree distribution for the cycle-free case, i.e. assuming there is no cycles in the graph what never happens with finite-length codes

• Two common variants:
  – Gaussian approximation
  – Discretized density evolution [2]

• Good families or ensembles of LDPC codes can be determined using the differential evolution [3]

Original LDPC code construction proposed by Gallager
MacKay and Neal construction
Progressive edge-growth (PEG) algorithm
Original LDPC code construction proposed by Gallager

• Valid only for the construction of regular LDPC codes
• Rows are divided into a number of sets
  – First set: rows contain a number of consecutive 1’s (ordered from left to right)
  – Next: rows are chosen randomly from a column permutation of the first set

\[
H = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
MacKay and Neal construction

- Columns are filled from left to right
  - Number of 1’s chosen per column (edges) according to a degree distribution
  - Rows are chosen randomly from those that are not full, i.e. the algorithm looks for a regular check node degree distribution
- Valid for regular and irregular LDPC codes
- The algorithm can be easily adapted to avoid 4-cycles

\[ H = \begin{pmatrix}
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix} \]
Progressive edge-growth (PEG) algorithm

• A Tanner graph is constructed connecting symbol and check nodes in an edge-by-edge manner
  – Symbols are processed sequentially
  – New check nodes are connected to the current symbol till the number of edges (symbol degree) is reached

• The algorithm consists of two basic procedures:
  – a local graph expansion (used to detect and avoid short cycles)
  – and, a check node selection procedure

The PEG algorithm construct codes having a large girth (i.e. avoiding short cycles)
for $j = 1$ to $n$ do
    for $k = 1$ to $\text{deg}(x_j)$ do
        if $k = 1$ then
            $E_{x_j}^1 \leftarrow (z_i, x_j)$, where $E_{x_j}^1$ is the first edge incident to $x_j$, and $z_i$ is a check node such that it has the lowest check-node degree under the current graph setting $E_{x_1} \cup E_{x_2} \cup \cdots \cup E_{x_{j-1}}$
        else
            expand a subgraph from symbol node $x_j$ up to depth $\ell$ under the current graph setting, such that $\mathcal{N}_{x_j}^\ell = \mathcal{N}_{x_j}^{\ell+1}$, or $\overline{\mathcal{N}}_{x_j}^{\ell+1} = \emptyset$
            $E_{x_j}^k \leftarrow (z_i, x_j)$, where $E_{x_j}^k$ is the $k$th edge incident to $x_j$ and $z_i$ is a check node from the set $\overline{\mathcal{N}}_{x_j}^\ell$ having the lowest check-node degree.
        end if
    end for
end for

**PEG algorithm (II)**

- **Input parameters:**
  - Parity-check matrix dimension (i.e. num. symbols & check nodes)
  - Symbol node degree distribution

Note that the check node degree distribution is not considered in the original PEG algorithm.

- **Given the generating polynomial** $\lambda(x)$ and the codeword length $n$, we calculate the number of edges per symbol node, $\text{deg}(x_i)$
  - While $\lambda_i$ denote the fraction of edges connected to $i$-degree symbol nodes
  - $\lambda^*_i$ denote the fraction of symbols with degree $i$

\[
\lambda^*_i = \frac{\lambda_i}{i} \frac{1}{\sum_i \lambda_i} \leq 1
\]
PEG algorithm (III)

• According to the first condition:
  – Every symbol node is firstly connected to the current graph.

In particular, 2-degree symbol nodes are connected in zigzag.

Note that 0-degree check nodes (i.e. those check nodes that are not currently connected to any symbol node) are not considered in the current graph.

PEG algorithm (IV)

• Check node selection:
  – **Lowest check node degree** criterium over the set of candidate checks

• Candidate check nodes:
  – unvisited (non expanded) check nodes (including those that are not in the current graph) no cycle is produced
  – the set of expanded check nodes with highest depth, otherwise it produces the longest (possible) cycle

\[
\mathcal{N}_{x_1}^1 = \{z_1, z_2\} \\
\mathcal{N}_{x_1}^2 = \{z_1, z_2, z_3, z_4, z_5\} \\
\overline{\mathcal{N}}_{x_1}^2 = \{z_6\}
\]

PEG algorithm
(proposed optimizations)

- Two proposed optimizations in the original PEG algorithm:
  - **Nongreedy** version:
    for long-block codes or low-rate codes (in which the minimum distance is -in principle- large), it may be favourable to limit the maximum depth $l$
  - **Look-ahead** enhanced version:
    when several choices exist for placing the $k$th edge, we look one step ahead and choose the one (check node) having the maximum possible depth $l$ in the expanded subgraph

Improved PEG algorithm

- Both degree distributions, for symbol and check nodes, are considered when changing the edge-selection criterion
  - Instead of the node with the lowest check degree we select the one with **highest free check node degree**, i.e. the difference between the number of currently assigned edges and the total number of edges to be assigned

Note that 2-degree symbol nodes are no longer connected in zigzag

Performance, Frame/Bit error rate

SIMULATION RESULTS
Error model for noisy channels

• **Discrete channel**
  A system consisting of
  • An input and an output alphabets
  • And a probability transition matrix $p(x|y)$

  – Channel capacity
    $$C = \max_{p(x)} I(X;Y)$$

• **Binary symmetric channel**
  with crossover probability $\varepsilon$
    $$C_{BSC}(\varepsilon) = 1 - H(\varepsilon)$$

Performance of LDPC codes (I)

Frame and bit error rate

Error rate vs. ε

- Sum-product algorithm
- Maximum 200 decoding iterations
- N=10^x

Waterfall Region
Error Floor Region

Frame error rate
Bit error rate
Performance of LDPC codes (II)
FER vs. codeword length

![Graph showing the Performance of LDPC codes (II) with FER vs. codeword length. The graph includes lines for different codeword lengths, with labels for each curve: $N=2\times10^3$, $N=10^4$, $N=2\times10^4$, $N=10^5$, and $N=2\times10^5$. The x-axis represents error rate ($\varepsilon$) ranging from 0.07 to 0.1, and the y-axis represents FER ranging from $10^{-6}$ to 1. The graph also notes that the maximum number of decoding iterations is 20.]

Junio 2013
Performance of LDPC codes (III)

Improved PEG
Performance of LDPC codes (III)

Improved PEG [Legend]

1) **Original PEG algorithm**

2) **Improved PEG algorithm proposed by Richter**

3) **Modified (2):** a check node in the current graph is selected when adding the first edge to a symbol node

4) **Improved PEG algorithm proposed here**

5) **Mixed version,** the lowest check node degree criterion is used to connect the first edge to a symbol node (not only to 2-degree symbol nodes as proposed here) and the highest free check node degree criterion for the remaining edges
Summary and suggested bibliography

CONCLUDING REMARKS
Summary

• Modern coding techniques:
  – Also based on linear codes
  – Iterative decoding using message passing algorithms:
    • Messages along the edges of a code graph
    • Local calculations (‘divide and conquer’ strategy)

• Ensembles of codes:
  – Designed using density and differential evolution

• Instances of codes:
  – Constructed using a progressive edge-growth algorithm
Suggested bibliography

Books and notebooks

Articles in international journals and conferences

Design of LDPC codes: density and differential evolution

Construction of LDPC codes: PEG algorithm
Thank you!

Questions?