The Peeling Decoder: Theory and some Applications

Krishna R. Narayanan
Thanks to Henry Pfister, Avinash Vem, Nagaraj Thenkarai Janakiraman, Kannan Ramchandran, Jean-Francois Chamberland

Department of Electrical and Computer Engineering
Texas A&M University
### Message passing algorithms

- Remarkably successful in coding theory
- Used to design capacity-achieving codes/decoders for a variety of channels
- Tools have been developed to analyze their performance
Two main goals

Goal 1
Review some developments in modern coding theory and show how to analyze the performance of a simple peeling decoder for the BEC and $p$-ary symmetric channels.
Two main goals

Goal 1
Review some developments in modern coding theory and show how to analyze the performance of a simple peeling decoder for the BEC and $p$-ary symmetric channels.

Goal 2
Show that the following problems have the same structure as channel coding problems and show how to use the peeling decoder to solve them.

Problems
- Uncoordinated massive multiple access
- Sparse Fourier transform (SFT) computation
- Sparse Walsh-Hadamard transform computation
- Compressed sensing
  - Data stream computing
  - Group testing
  - Compressive phase retrieval
Remembering Sir David MacKay

David Mackay’s rediscovery of LDPC codes and his very interesting book on Information Theory has undoubtedly had a big influence on the field.
Binary erasure channel (BEC) and erasure correction

Channel coding problem

- Transmit a message $\overline{m} = [m_1, \ldots, m_k]^T$ through a binary erasure channel
- Encode the $k$-bit message $\overline{m}$ into a $n$-bit codeword $x$
- Redundancy is measured in terms of rate of the code $R = k/n$
Binary erasure channel (BEC) and erasure correction

Capacity achieving sequence of codes
Binary erasure channel (BEC) and erasure correction

Capacity achieving sequence of codes

- Capacity $C(\epsilon) = 1 - \epsilon$
Binary erasure channel (BEC) and erasure correction

Capacity achieving sequence of codes

- Capacity $C(\epsilon) = 1 - \epsilon$
- A sequence of codes $\{C^n\}$
- Probability of erasure $P^n_e$
- Rate $R^n$
- Capacity achieving if $P^n_e \to 0$ as $n \to \infty$ while $R^n \to C$
Binary erasure channel (BEC) and erasure correction

Encoder
\[ x_1, \ldots, x_n, \quad x_i \in \{0, 1\} \]

BEC(\( \epsilon \)) channel
\[ \begin{array}{ccc}
0 & \xrightarrow{1-\epsilon} & 0 \\
\epsilon & \xrightarrow{\epsilon} & E \\
1 & \xrightarrow{1-\epsilon} & 1 \\
\end{array} \]

Decoder
\[ r_1, \ldots, r_n, \quad r_i \in \{0, 1\} \]

Capacity achieving sequence of codes

- Capacity \( C(\epsilon) = 1 - \epsilon \)
- A sequence of codes \( \{C^n\} \)
- Probability of erasure \( P_e^n \)
- Rate \( R^n \)
- Capacity achieving if \( P_e^n \rightarrow 0 \) as \( n \rightarrow \infty \) while \( R^n \rightarrow C \)
- Find efficient encoders/decoders in terms encoding and decoding complexities

Significance of the erasure channel

- Introduced by Elias in 1954 as a toy example
- Has become the canonical model for coding theorists to gain insight
Binary erasure channel (BEC) and erasure correction

Capacity achieving sequence of codes

- Capacity $C(\epsilon) = 1 - \epsilon$
- A sequence of codes $\{C^n\}$
- Probability of erasure $P^n_e$
- Rate $R^n$
- Capacity achieving if $P^n_e \to 0$ as $n \to \infty$ while $R^n \to C$
- Find efficient encoders/decoders in terms encoding and decoding complexities

Significance of the erasure channel

- Introduced by Elias in 1954 as a toy example
- Has become the canonical model for coding theorists to gain insight
$(n, k)$ Binary linear block codes - basics

**G is a $n \times k$ generator matrix**

\[
\begin{bmatrix}
  g_{1,1} & \cdots & g_{k,l} \\
  \vdots & \ddots & \vdots \\
  g_{n,1} & \cdots & g_{k,l}
\end{bmatrix}
\begin{bmatrix}
  m_1 \\
  \vdots \\
  m_k
\end{bmatrix} =
\begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix}
\]

**Example - (6,3) code**

\[
\begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0 \\
  0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
  1 \\
  1 \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 \\
  1 \\
  0
\end{bmatrix}
\]
Binary linear block codes - basics

\( G \) is a \( n \times k \) generator matrix

\[
\begin{bmatrix}
g_{1,1} & \cdots & g_{k,l} \\
\vdots & \ddots & \vdots \\
g_{n,1} & \cdots & g_{k,l}
\end{bmatrix}
\begin{bmatrix}
m_1 \\
\vdots \\
m_k
\end{bmatrix}
= 
\begin{bmatrix}
x_1 \\
\vdots \\
x_n
\end{bmatrix}
\]

Example - (6,3) code

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
0 \\
1 \\
1 \\
0
\end{bmatrix}
= 
\begin{bmatrix}
1 \\
1 \\
0 \\
1 \\
0 \\
1
\end{bmatrix}
\]

Parity check matrix - \( H \) is a \( (n - k) \times n \) matrix s.t. \( HG = 0 \Rightarrow Hx = 0 \)

\[
H = 
\begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
0
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix}
\]
Tanner graph representation of codes

\[ H = \begin{bmatrix}
  x_1, x_2, x_3, x_4, x_5, x_6 \\
  1 & 0 & 1 & 1 & 0 & 0 \\
  1 & 1 & 0 & 0 & 1 & 0 \\
  0 & 1 & 1 & 0 & 0 & 1 
\end{bmatrix} \]

\[ x_1 \oplus x_3 \oplus x_4 = 0 \]
\[ x_1 \oplus x_2 \oplus x_5 = 0 \]
\[ x_2 \oplus x_3 \oplus x_6 = 0 \]

- Gallager’63, Tanner’81
- Parity check matrix implies that \( Hx = 0 \)
- Code constraints can be specified in terms of a bipartite (Tanner) graph
Peeling decoder for the BEC

\[ H = \begin{bmatrix} x_1, x_2, x_3, x_4, x_5, x_6 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[ x_1 \oplus x_3 \oplus x_4 = 0 \]
\[ x_1 \oplus x_2 \oplus x_5 = 0 \]
\[ x_2 \oplus x_3 \oplus x_6 = 0 \]

Tanner Graph

- Zyablov and Pinsker’74, Luby et al ’95
- Remove edges incident on known variable nodes and adjust check node values
- If there is a check node with a single edge, it can be recovered
Peeling decoder for the BEC

\[ H = \begin{bmatrix} x_1, x_2, x_3, x_4, x_5, x_6 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[ x_1 \oplus x_3 \oplus x_4 = 0 \]
\[ x_1 \oplus x_2 \oplus x_5 = 0 \]
\[ x_2 \oplus x_3 \oplus x_6 = 0 \]

- Zyablov and Pinsker’74, Luby et al ’95
- Remove edges incident on known variable nodes and adjust check node values
- If there is a check node with a single edge, it can be recovered
Peeling decoder for the BEC

\[ H = \begin{bmatrix} x_1, x_2, x_3, x_4, x_5, x_6 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[ x_1 \oplus x_3 \oplus x_4 = 0 \]
\[ x_1 \oplus x_2 \oplus x_5 = 0 \]
\[ x_2 \oplus x_3 \oplus x_6 = 0 \]

- Zyablov and Pinsker’74, Luby et al ’95
- Remove edges incident on known variable nodes and adjust check node values
- If there is a check node with a single edge, it can be recovered
Peeling decoder for the BEC

$H = \begin{bmatrix} x_1, x_2, x_3, x_4, x_5, x_6 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$

$x_1 \oplus x_3 \oplus x_4 = 0$
$x_1 \oplus x_2 \oplus x_5 = 0$
$x_2 \oplus x_3 \oplus x_6 = 0$

Peeling Step 2

• Zyablov and Pinsker’74, Luby et al ’95
• Remove edges incident on known variable nodes and adjust check node values
• If there is a check node with a single edge, it can be recovered
Peeling decoder for the BEC

\[
H = \begin{bmatrix}
  x_1, x_2, x_3, x_4, x_5, x_6 \\
  1 & 0 & 1 & 1 & 0 & 0 \\
  1 & 1 & 0 & 0 & 1 & 0 \\
  0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

\[
x_1 \oplus x_3 \oplus x_4 = 0 \\
x_1 \oplus x_2 \oplus x_5 = 0 \\
x_2 \oplus x_3 \oplus x_6 = 0
\]

Peeling Step 2

- Zyablov and Pinsker’74, Luby et al ’95
- Remove edges incident on known variable nodes and adjust check node values
- If there is a check node with a single edge, it can be recovered
Peeling decoder for the BEC

\[ H = \begin{bmatrix} x_1, x_2, x_3, x_4, x_5, x_6 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[ x_1 \oplus x_3 \oplus x_4 = 0 \]
\[ x_1 \oplus x_2 \oplus x_5 = 0 \]
\[ x_2 \oplus x_3 \oplus x_6 = 0 \]

• Zyablov and Pinsker’74, Luby et al ’95
• Remove edges incident on known variable nodes and adjust check node values
• If there is a check node with a single edge, it can be recovered
Peeling decoder for the BEC

\[ H = \begin{bmatrix}
  x_1, x_2, x_3, x_4, x_5, x_6 \\
  1 & 0 & 1 & 1 & 0 & 0 \\
  1 & 1 & 0 & 0 & 1 & 0 \\
  0 & 1 & 1 & 0 & 0 & 1 
\end{bmatrix} \]

\[ x_1 \oplus x_3 \oplus x_4 = 0 \]
\[ x_1 \oplus x_2 \oplus x_5 = 0 \]
\[ x_2 \oplus x_3 \oplus x_6 = 0 \]

Zyablov and Pinsker’74, Luby et al ’95

- Remove edges incident on known variable nodes and adjust check node values
- If there is a check node with a **single edge**, it can be recovered
Peeling decoder for the BEC

\[ H = \begin{bmatrix}
    x_1, x_2, x_3, x_4, x_5, x_6 \\
    1 & 0 & 1 & 1 & 0 & 0 \\
    1 & 1 & 0 & 0 & 1 & 0 \\
    0 & 1 & 1 & 0 & 0 & 1 
\end{bmatrix} \]

\[ \begin{align*}
    x_1 \oplus x_3 \oplus x_4 &= 0 \\
    x_1 \oplus x_2 \oplus x_5 &= 0 \\
    x_2 \oplus x_3 \oplus x_6 &= 0 
\end{align*} \]

- Zyablov and Pinsker’74, Luby et al ’95
- Remove edges incident on known variable nodes and adjust check node values
- If there is a check node with a single edge, it can be recovered
Message passing decoder for the BEC

Tanner Graph

- Pass messages between variable nodes and check nodes along the edges
- Messages $\in \{\text{value of var node (NE), erasure (E)}\}$
- Var-to-check node message is NE if at least one incoming message is NE
- Check-to-var node message is NE if all other incoming messages are NE
Message passing decoder for the BEC

- Pass messages between variable nodes and check nodes along the edges
- Messages $\in \{\text{value of var node (NE), erasure (E)}\}$
- Var-to-check node message is NE if at least one incoming message is NE
- Check-to-var node message is NE if all other incoming messages are NE
Message passing decoder for the BEC

- Pass messages between variable nodes and check nodes along the edges
- Messages $\in \{\text{value of var node (NE), erasure (E)}\}$
- Var-to-check node message is NE if at least one incoming message is NE
- Check-to-var node message is NE if all other incoming messages are NE
Pass messages between variable nodes and check nodes along the edges
- Messages $\in \{\text{value of var node (NE), erasure (E)}\}$
- Var-to-check node message is NE if at least one incoming message is NE
- Check-to-var node message is NE if all other incoming messages are NE
Message passing decoder for the BEC

- Pass messages between variable nodes and check nodes along the edges
- Messages ∈ \{value of var node (NE), erasure (E)\}
- Var-to-check node message is NE if at least one incoming message is NE
- Check-to-var node message is NE if all other incoming messages are NE
Message passing decoder for the BEC

Pass messages between variable nodes and check nodes along the edges
Messages $\in \{\text{value of var node (NE), erasure (E)}\}$
Var-to-check node message is NE if at least one incoming message is NE
Check-to-var node message is NE if all other incoming messages are NE
Message passing decoder for the BEC

- Pass messages between variable nodes and check nodes along the edges
- Messages ∈ \{value of var node (NE), erasure (E)\}
- Var-to-check node message is NE if at least one incoming message is NE
- Check-to-var node message is NE if all other incoming messages are NE
Message passing decoder for the BEC

Peeling Step 4

- Pass messages between variable nodes and check nodes along the edges
- Messages $\in \{\text{value of var node (NE), erasure (E)}\}$
- Var-to-check node message is NE if at least one incoming message is NE
- Check-to-var node message is NE if all other incoming messages are NE
Peeling decoder is a greedy decoder

\[
H = \begin{bmatrix}
    x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
    1 & 1 & 1 & 1 & 0 & 0 \\
    1 & 0 & 0 & 0 & 1 & 0 \\
    0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

\[
x_1 \oplus x_1 \oplus x_3 \oplus x_4 = 0 \\
x_1 \oplus x_2 \oplus x_5 = 0 \\
x_2 \oplus x_3 \oplus x_6 = 0
\]
Peeling decoder is a greedy decoder

\[ \mathbf{H} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[
x_1 \oplus x_1 \oplus x_3 \oplus x_4 = 0 \\
x_1 \oplus x_2 \oplus x_5 = 0 \\
x_2 \oplus x_3 \oplus x_6 = 0
\]

Linearly independent set of equations

\[
x_1 \oplus x_1 \oplus x_3 = x_4 \\
x_1 \oplus x_2 = x_5 \\
x_2 \oplus x_3 = x_6
\]
Degree distributions

- VN d.d. from node perspective - $L(x) = \sum_i L_i x^i = \frac{3}{6} x + \frac{2}{6} x^2 + \frac{1}{6} x^3$
Degree distributions

- VN d.d. from node perspective - $L(x) = \sum_i L_i x^i = \frac{3}{6} x + \frac{2}{6} x^2 + \frac{1}{6} x^3$
- VN d.d. from edge perspective - $\lambda(x) = \sum_i \lambda_i x^{i-1} = \frac{3}{10} + \frac{4}{10} x + \frac{3}{10} x^2$
Degree distributions

- VN d.d. from node perspective - \( L(x) = \sum_i L_i x^i = \frac{3}{6} x + \frac{2}{6} x^2 + \frac{1}{6} x^3 \)
- VN d.d. from edge perspective - \( \lambda(x) = \sum_i \lambda_i x^{i-1} = \frac{3}{10} + \frac{4}{10} x + \frac{3}{10} x^2 \)
- CN d.d. from node perspective - \( R(x) = \sum_i R_i x^i = \frac{2}{3} x^3 + \frac{1}{3} x^4 \)
Degree distributions

- VN d.d. from node perspective - $L(x) = \sum_i L_i x^i = \frac{3}{6} x + \frac{2}{6} x^2 + \frac{1}{6} x^3$
- VN d.d. from edge perspective - $\lambda(x) = \sum_i \lambda_i x^{i-1} = \frac{3}{10} + \frac{4}{10} x + \frac{3}{10} x^2$
- CN d.d. from node perspective - $R(x) = \sum_i R_i x^i = \frac{2}{3} x^3 + \frac{1}{3} x^4$
- CN d.d. from edge perspective - $\rho(x) = \sum_i \rho_i x^{i-1} = \frac{6}{10} x^2 + \frac{4}{10} x^3$
Degree distributions

- Rate - \( r(\lambda, \rho) = 1 - \frac{l_{\text{avg}}}{r_{\text{avg}}} = 1 - \frac{\int_0^1 \rho(x) \, dx}{\int_0^1 \lambda(x) \, dx} \)

- VN d.d. from node perspective - \( L(x) = \sum_i L_i x^i = \frac{3}{6} x + \frac{2}{6} x^2 + \frac{1}{6} x^3 \)
- VN d.d. from edge perspective - \( \lambda(x) = \sum_i \lambda_i x^{i-1} = \frac{3}{10} + \frac{4}{10} x + \frac{3}{10} x^2 \)
- CN d.d. from node perspective - \( R(x) = \sum_i R_i x^i = \frac{2}{3} x^3 + \frac{1}{3} x^4 \)
- CN d.d. from edge perspective - \( \rho(x) = \sum_i \rho_i x^{i-1} = \frac{6}{10} x^2 + \frac{4}{10} x^3 \)
LDPC code ensemble

$\text{LDPC}(n, \lambda, \rho)$ ensemble

- Ensemble of codes obtained by using different permutations $\pi$
- Assume there is only one edge between every var node and check node
- For every $n$, we get an ensemble of codes with the same $(\lambda, \rho)$
- Low density parity check (LDPC) ensemble if graph is of low density
If we pick a code uniformly at random from the LDPC\((n, \lambda, \rho)\) ensemble and use it over a \(\text{BEC}(\varepsilon)\) with \(l\) iterations of message passing decoding, what will be the probability of erasure \(P_e^n\) in the limit \(l, n \to \infty\)?
If we pick a code uniformly at random from the LDPC\((n, \lambda, \rho)\) ensemble and use it over a BEC\((\epsilon)\) with \(l\) iterations of message passing decoding, what will be the probability of erasure \(P_e^n\) in the limit \(l, n \to \infty\)?

- Analyze the average prob. of erasure over the ensemble
- For almost all realizations \(P_e^n\) concentrates around the average
Analysis of the message passing decoder

- If we pick a code uniformly at random from the LDPC\((n, \lambda, \rho)\) ensemble and use it over a BEC\((\epsilon)\) with \(l\) iterations of message passing decoding, what will be the probability of erasure \(P_{e}^{n}\) in the limit \(l, n \rightarrow \infty\)?
  - Analyze the average prob. of erasure over the ensemble
  - For almost all realizations \(P_{e}^{n}\) concentrates around the average

Relevant literature
- Papers by Luby, Mitzenmacher, Shokrollahi, Spielman, Stemann 97-'02
- Explained in Modern coding theory by Richardson and Urbanke
- Henry Pfister’s course notes on his webpage
Computation graph

Computation graph $C_l(x_1, \lambda, \rho)$ of bit $x_1$ of depth $l$ ($l$-iterations) is the neighborhood graph of node $x_1$ of radius $l$. 
Analysis of the message passing decoder

Computation graph

Computation graph $C_l(x_1, \lambda, \rho)$ of bit $x_1$ of depth $l$ ($l$-iterations) is the neighborhood graph of node $x_1$ of radius $l$. Consider the example $C_{l=1}(\lambda(x) = x, \rho(x) = x^2)$

1. $1 - O(1/n)$
2. $O(1/n)$
3. $O(1/n^2)$
Analysis of the message passing decoder

**Computation graph**

Computation graph $C_l(x_1, \lambda, \rho)$ of bit $x_1$ of depth $l$ ($l$-iterations) is the neighborhood graph of node $x_1$ of radius $l$. Consider the example $C_{l=1}(\lambda(x) = x, \rho(x) = x^2)$.

![Computation graph examples](image)

**Computation tree**

For fixed $(l_{max}, r_{max})$, in the limit of large block lengths a computation graph of depth-$l$ looks like a tree with high probability.
Analysis of the message passing decoder

Computation Tree Ensemble-$\mathcal{T}_l(\lambda, \rho)$

Ensemble of bipartite trees of depth $l$ rooted in a variable node (VN) where

- Root node has $i$ children (CN’s) with probability $L_i$
- Each VN has $i$ children (CN’s) with probability $\lambda_i$
- Each CN has $i$ children (VN’s) with probability $\rho_i$

Example: $C_{l=1}(\lambda(x) = x, \rho(x) = x^2)$
Density evolution

Recall
- $\rho(x) = \sum_i \rho_i x^{i-1}$
- $\sum_i \rho_i = 1$
- $\lambda(x) = \sum_i \lambda_i x^{i-1}$
- $\sum_i \lambda_i = 1$

Recursion
- $x_0 = \epsilon$
- $y_l = 1 - \rho(1 - x_{l-1})$
- $x_l = \epsilon \sum_i \lambda_i y_{l-1}^{i-1} = \epsilon \lambda(y_l)$
- $x_{l-1} = \epsilon \lambda(y_{l-1})$

Depth-1

Depth-$l$
Density evolution

\[ x_l = \epsilon \sum_i \lambda_i y_i^{i-1} = \epsilon \lambda(y_l) \]

Recall
- \( \rho(x) = \sum_i \rho_i x^{i-1} \)
- \( \sum_i \rho_i = 1 \)
- \( \lambda(x) = \sum_i \lambda_i x^{i-1} \)
- \( \sum_i \lambda_i = 1 \)

Recursion
- \( x_0 = \epsilon \)
- \( y_l = 1 - \rho(1 - x_{l-1}) \)
Density evolution

\[ x_l = \epsilon \sum_i \lambda_i y_l^{i-1} = \epsilon \lambda(y_l) \]

Recall
- \( \rho(x) = \sum_i \rho_i x^{i-1} \)
- \( \sum_i \rho_i = 1 \)
- \( \lambda(x) = \sum_i \lambda_i x^{i-1} \)
- \( \sum_i \lambda_i = 1 \)

Recursion
- \( x_0 = \epsilon \)
- \( y_l = 1 - \rho(1 - x_{l-1}) \)
- \( x_l = \epsilon \lambda(y_l) \)
Density evolution

\[ x_l = \epsilon \sum_i \lambda_i y_{l-1}^i = \epsilon \lambda(y_l) \]

\[ y_l = 1 - \rho(1 - x_{l-1}) \]

Recall

- \( \rho(x) = \sum_i \rho_i x^{i-1} \)
- \( \sum_i \rho_i = 1 \)
- \( \lambda(x) = \sum_i \lambda_i x^{i-1} \)
- \( \sum_i \lambda_i = 1 \)

Recursion

\[ x_0 = \epsilon \]
\[ y_0 = 1 - \rho(1 - x_{l-1}) \]
\[ x_l = \epsilon \lambda(y_l) \]
\[ x_{ch} = \epsilon \]
Analysis of the message passing decoder

\[ \lambda(x) = x^2, \rho(x) = \rho_4 x^3 + \rho_5 x^4 \]

\[ x_1 = \epsilon (y_1^{(3)})^2 \]

\[ y_1^{(3)} = 1 - (1 - \epsilon)^3 \]

\[ x_0 = \epsilon \]

\[ P(T) = \rho_4^2 \]
Analysis of the message passing decoder

\[ \lambda(x) = x^2, \rho(x) = \rho_4 x^3 + \rho_5 x^4 \]

\[ y_1^{(3)} = 1 - (1 - \epsilon)^3 \]
\[ y_1^{(4)} = 1 - (1 - \epsilon)^4 \]

\[ P(T) = 2\rho_4\rho_5 \]
Analysis of the message passing decoder

\[ \lambda(x) = x^2, \rho(x) = \rho_4 x^3 + \rho_5 x^4 \]

\[ x_1 = \epsilon (y_1^{(4)})^2 \]

\[ x_0 = \epsilon \]

\[ y_1^{(4)} = 1 - (1 - \epsilon)^4 \]

\[ P(T) = \rho_5^2 \]
Analysis of the message passing decoder

\[ \lambda(x) = x^2, \quad \rho(x) = \rho_4 x^3 + \rho_5 x^4 \]

\[ x_1 = \epsilon (y^{(3)}_1)^2 \]

\[ y^{(3)}_1 = 1 - (1 - \epsilon)^3 \]

\[ x_0 = \epsilon \]

\[ P(T) = \rho_4^2 \]

\[ x_1 = \epsilon y^{(3)}_1 y^{(4)}_1 \]

\[ y^{(3)}_1 = 1 - (1 - \epsilon)^3 \]

\[ y^{(4)}_1 = 1 - (1 - \epsilon)^4 \]

\[ x_0 = \epsilon \]

\[ P(T) = 2 \rho_4 \rho_5 \]

\[ x_1 = \epsilon (y^{(4)}_1)^2 \]

\[ y^{(4)}_1 = 1 - (1 - \epsilon)^4 \]

\[ x_0 = \epsilon \]

\[ P(T) = \rho_5^2 \]

\[
\mathbb{E}_{LDPC}(\lambda, \rho)[x_1] = \sum_{T \in T_1(\lambda, \rho)} P(T) * x_1(T, \epsilon) \\
= \epsilon (\rho_4 y^{(3)}_1 + \rho_5 y^{(4)}_1)^2 \\
= \epsilon (1 - \rho_4 (1 - \epsilon)^3 - \rho_5 (1 - \epsilon)^4)^2 \\
= \epsilon \lambda (1 - \rho (1 - \epsilon))
\]
Threshold

**Convergence condition**

\[ x_l = \epsilon \lambda (1 - \rho (1 - x_{l-1})) = f(\epsilon, x_{l-1}) \]

- \( x_l \) converges to 0 if \( f(\epsilon, x) < x, \; x \in (0, \epsilon] \)
- There is a fixed point if \( f(\epsilon, x) = x, \; \text{for some} \; x \in (0, \epsilon] \)
Threshold

Convergence condition

\[ x_l = \epsilon \lambda (1 - \rho (1 - x_{l-1})) = f(\epsilon, x_{l-1}) \]

\( x_l \) converges to 0 if \( f(\epsilon, x) < x, \ x \in (0, \epsilon] \)

There is a fixed point if \( f(\epsilon, x) = x, \ \text{for some} \ x \in (0, \epsilon] \)
Threshold

Convergence condition

\[ x_l = \epsilon \lambda (1 - \rho (1 - x_{l-1})) = f(\epsilon, x_{l-1}) \]

\( x_l \) converges to 0 if \( f(\epsilon, x) < x \), \( x \in (0, \epsilon] \)

There is a fixed point if \( f(\epsilon, x) = x \), for some \( x \in (0, \epsilon] \)

The threshold \( \epsilon^{BP}(\lambda, \rho) \) is defined as

\[ \epsilon^{BP}(\lambda, \rho) = \sup \{ \epsilon \in [0, 1] : x_l \to 0 \text{ as } l \to \infty \} \]
Node functions

- Var node function: \( v_\epsilon(x) = \epsilon \lambda(x) \)
- Check node function: \( c(x) = 1 - \rho(1 - x) \)
Optimality of EXIT chart matching

- Var node function: $v_\epsilon(x) = \epsilon \lambda(x)$
- Check node function: $c(x) = 1 - \rho(1 - x)$
• Understand what degree distributions \((\lambda(x), \rho(x))\) mean
• Understand what degree distributions \((\lambda(x), \rho(x))\) mean
• Given a \((\lambda, \rho)\) and \(\epsilon\), what will be the \(P_e^n\) as \(l, n \to \infty\)?
Summary

- Understand what degree distributions $(\lambda(x), \rho(x))$ mean
- Given a $(\lambda, \rho)$ and $\epsilon$, what will be the $P^n_e$ as $l, n \to \infty$?
- Can you compute the threshold?
• Understand what degree distributions \((\lambda(x), \rho(x))\) mean
• Given a \((\lambda, \rho)\) and \(\epsilon\), what will be the \(P_e^n\) as \(l, n \rightarrow \infty\) ?
• Can you compute the threshold?
• Is a \((\lambda(x), \rho(x))\) pair optimal?
The changing mobile landscape

• 5G will not only be “4G but faster” but will support new models such as IoT
• Current wireless - a few devices with sustained connectivity
• Future wireless - massive no. of devices requesting sporadic connectivity

R1: today’s systems operating region
R2: high-speed versions of today’s systems
R3: massive access
R4: ultra-reliable communication

<table>
<thead>
<tr>
<th># devices</th>
<th>1 10000 100010010</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: today’s systems</td>
<td>≥99%</td>
</tr>
<tr>
<td>R2: high-speed versions of today’s systems</td>
<td>≥99%</td>
</tr>
<tr>
<td>R3: massive access</td>
<td>≥90-99%</td>
</tr>
<tr>
<td>R4: ultra-reliable communication</td>
<td>≥99.999%</td>
</tr>
</tbody>
</table>
The changing mobile landscape

- Current wireless - a few devices with sustained connectivity
- Future wireless - many uncoordinated devices requesting sporadic connectivity
The changing mobile landscape

- Current wireless - a few devices with sustained connectivity
- Future wireless - many uncoordinated devices requesting sporadic connectivity
A possible MAC frame structure

- Total of $Q$ users out of which $K$ are active
- $Q$ is very large and $K$ is a small fraction of $Q$

Beacon is used to obtain coarse synchronization
Each user transmits a signature sequence
BS estimates the no. of users ($K$) (Chen, Guo ’14, Calderbank)
Picks an $M$ and broadcasts it
System under consideration

- Wireless network with $K$ distributed users (no coordination)
- Each user has one packet of info to transmit to a central receiver
- Total time is split into $M$ slots (packet duration)
  - Some policy used to decide if they transmit in $j$-th slot or not
  - Receiver knows the set of users transmitting in the $j$-th slot

```
Users  Time slots
1      1
2      2
3      3
...    ...
K      K
```

Receiver
Random access paradigm

- \(k\)-th user:
  - Generates a random variable \(D_k \in \{1, \ldots, M\}\)
  - Generating PMF is \(f_D\), i.e., \(Pr(D_k = i) = f_D[i]\)
  - Transmits during \(D_k\) time slots drawn uniformly from \(\{1, \ldots, M\}\)

- In this example, \(D_3 = 3\) and user 3 transmits in slots \(\{1, 3, 5\}\)
Iterative interference cancelation

- If exactly one user transmits per slot, then packet is decoded w.h.p.
- If more than one user transmits per slot, then collision
  - Rx subtracts previously decoded packets from collided packets
  - If Rx can subtract all but one, remaining packet is decoded w.h.p.
  - Otherwise, the received packet is saved for future processing
  - Once all $K$ packets recovered, an ACK terminates the transmission
- Similar to interference cancellation in multi-user detection

<table>
<thead>
<tr>
<th>Users</th>
<th>Time slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1: $x_2 + x_3$ Collision</td>
</tr>
<tr>
<td>2</td>
<td>2: $x_1 + x_4$</td>
</tr>
<tr>
<td>3</td>
<td>3: $x_2 + x_3 + x_4$</td>
</tr>
<tr>
<td>4</td>
<td>4: $x_2$ No collision</td>
</tr>
<tr>
<td></td>
<td>5: $x_3 + x_4$</td>
</tr>
</tbody>
</table>
Performance measure - Efficiency

- Suppose $M$ time slots needed to successfully transmit all $K$ packets

- Then, the efficiency of the system is said to be

  $$\eta = K/M \text{ packets/slot}$$
Graphical representation (Liva 2012)

- Tanner graph representation for the transmission scheme
- Variable nodes ↔ users, Check nodes ↔ received packets
- Message-passing decoder - peeling decoder for the erasure channel

Users | Time slots | Var nodes | Check nodes
---|---|---|---
\(v_1\) | 1 | \(v_1\) | \(+\) | \(x_1\)
\(v_2\) | 2 | \(v_2\) | \(+\) | \(x_2\)
\(v_3\) | 3 | \(v_3\) | \(+\) | \(x_3\)
\(v_4\) | 4 | \(v_4\) | \(+\) | \(x_4\)
| 5 | | |
Graphical representation (Liva 2012)

- Tanner graph representation for the transmission scheme
- Variable nodes ↔ users, Check nodes ↔ received packets
- Message-passing decoder - peeling decoder for the erasure channel

- $L_i (R_i)$ - fraction of left (right) nodes with degree $i$ - notice that $L_i = f_D[i]$
- $\lambda_i (\rho_i)$ - fraction of edges connected to left (right) nodes with deg $i$
Low density generator matrix (LDGM) codes

\[
L(x) = \frac{1}{4} x + \frac{1}{4} x^2 + \frac{1}{2} x^3
\]

\[
\lambda(x) = \frac{1}{9} + \frac{2}{9} x + \frac{6}{9} x^2
\]

\[
R(x) = \frac{1}{5} x + \frac{4}{5} x^2
\]

\[
\rho(x) = \frac{1}{9} + \frac{8}{9} x
\]

Rate \( R = \frac{\int_0^1 \lambda(x) \, dx}{\int_0^1 \rho(x) \, dx} \)

### DE for LDPC

\[
\begin{align*}
x_0 &= \epsilon \\
y_l &= 1 - \rho(1 - x_{l-1}) \\
x_l &= \epsilon \lambda(y_l) \\
x_l &= \epsilon \lambda(1 - \rho(1 - x_{l-1}))
\end{align*}
\]

### DE for LDGM

\[
\begin{align*}
x_0 &= 1 \\
y_l &= 1 - \rho(1 - x_{l-1}) \\
x_l &= \lambda(y_l) \\
x_l &= \lambda(1 - \rho(1 - x_{l-1}))
\end{align*}
\]
Poisson approximation for check node d.d.

Slot transmission probability

User $k$ transmits in slot $m$ with prob. $p = \sum_{i=1}^{\infty} \frac{L_i}{M} = \frac{1_{avg}}{M} = \frac{r_{avg}}{K}$
Poisson approximation for check node d.d.

Slot transmission probability

User $k$ transmits in slot $m$ with prob. $p = \sum_{i=1}^{\infty} L_i \frac{i}{M} = \frac{l_{avg}}{M} = \frac{r_{avg}}{K}$

Optimal multiple access policy

- Poisson approximation for $R(x)$ as $K, M \to \infty$
- Finding optimal $f_D$ - same as finding optimal $\lambda(x)$ for $\rho(x) = e^{-r_{avg}(1-x)}$
Intuition behind the main result (Narayanan, Pfister’12)

Convergence condition: \( \rho(1 - \lambda(y)) > 1 - y \)

\[
\rho(1 - \lambda(y)) = 1 - y \\
e^{-r_{avg}\lambda(y)} = e^{\ln(1-y)}
\]

\[\Rightarrow -r_{avg}\lambda(y) = \ln(1 - y) = -\sum_{i=1}^{\infty} \frac{y^i}{i}\]

\[\Rightarrow r_{avg} \sum_i \lambda_i y^i = \sum_{i=1}^{\infty} \frac{y^i}{i}\]
Intuition behind the main result (Narayanan,Pfister’12)

Convergence condition: \( \rho(1 - \lambda(y)) > 1 - y \)

\[
\rho(1 - \lambda(y)) = 1 - y \\
e^{-r_{avg} \lambda(y)} = e^{\ln(1-y)}
\]

\[
\Rightarrow -r_{avg} \lambda(y) = \ln(1 - y) = -\sum_{i=1}^{\infty} \frac{y^i}{i}
\]

\[
\Rightarrow r_{avg} \sum_i \lambda_i y^i = \sum_{i=1}^{\infty} \frac{y^i}{i}
\]

\[
r_{avg} \lambda_i = \frac{1}{i}
\]

\[
\sum_i \lambda_i = 1 \Rightarrow r_{avg} = \sum_i \frac{1}{i}, \lambda_i = \frac{1/i}{\sum_i 1/i} \Rightarrow L_i = \frac{1}{i(i-1)}, i \geq 2
\]
Graphical interpretation - EXIT chart

Left to Right erasure prob $x_l$
Right to Left erasure prob $y_l$
Soliton max degree = 100
$\eta = 0.5$
$\eta = 0.95$
Main result

- For coordinated transmission, clearly $\eta = 1$,
Main result

- For coordinated transmission, clearly $\eta = 1$,
- ALOHA provides $\eta \approx 0.37$
Main result

- For coordinated transmission, clearly $\eta = 1$,
- ALOHA provides $\eta \approx 0.37$
- But, even for uncoordinated transmission, $\eta \to 1$ as $K \to \infty$
Main result

- For coordinated transmission, clearly $\eta = 1$,
- ALOHA provides $\eta \approx 0.37$
- But, even for uncoordinated transmission, $\eta \to 1$ as $K \to \infty$

Optimal distribution is soliton: $f_D[i] = \frac{1}{i(i-1)}$

<table>
<thead>
<tr>
<th>No. of times</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>…</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraction of users</td>
<td>$\frac{1}{M}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{12}$</td>
<td>…</td>
<td>$\frac{1}{M(M-1)}$</td>
</tr>
</tbody>
</table>
Balls in bins

- $M$ balls thrown into $N$ bins uniformly at random
- If every bin has to be non-empty with prob $1 - \delta$, how large should $M$ be?
Balls in bins

- $M$ balls thrown into $N$ bins uniformly at random
- If every bin has to be non-empty with prob $1 - \delta$, how large should $M$ be?
  \[ N \log \frac{N}{\delta} \]
- For the multiple access problem, an empty bin means a wasted time slot
- Note that for the soliton the average number of edges is indeed $N \log N$
Poisson, soliton pair is optimal for rateless codes

\[ \lambda^{(N)}_{\alpha}(x) = e^{\frac{-\alpha}{1-\epsilon}(1-x)}|_{N} \]

\[ \rho^{(N)}_{\alpha}(x) = -\frac{1}{\alpha} \ln(1-x)|_{N} \]

\[ \epsilon_{\text{Sh}} = 1 - r = 0.472 \]

\[ x = \lambda(1 - (1 - \epsilon)\rho(1 - x)) \]

\[ \lambda(x) = e^{-\frac{\alpha}{1-\epsilon}(1-x)}, \text{ optimal right degree is soliton: } \rho(x) = -\frac{1}{\alpha} \ln(1-x) \]

<table>
<thead>
<tr>
<th>Degree of nodes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>i</th>
<th>...</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraction: (f_{D}[i])</td>
<td>(\frac{1}{K})</td>
<td>(\frac{1}{2})</td>
<td>(\frac{1}{6})</td>
<td>(\frac{1}{12})</td>
<td>...</td>
<td>(\frac{1}{i(i-1)})</td>
<td>...</td>
<td>(\frac{1}{K(K-1)})</td>
</tr>
</tbody>
</table>
Connection with Luby Transform (LT) codes

- For rateless codes $\lambda(x)$ is Poisson and $\rho(x)$ is soliton
- For multiple access $\rho(x)$ is Poisson, optimal $\lambda(x)$ is soliton
- Our result shows that both are optimal pairs
Simulation Results

• Even for $K = 10000$, efficiency close to 0.8 can be obtained
Some open problems

- Fundamental limits on universal multiple access, i.e. $K, \epsilon$ not known
- Uncoordinated multiple access with power constraint and Gaussian noise
  - Power penalty for repeating information $\log n$ times on the average
  - Can we achieve the equal rate point on the MAC region with simple decoding?
Back to theory: from erasures to errors
Finite field with $p$ elements

$p$ is prime

- $\mathbb{F}_p = \{0, 1, 2, \ldots, p - 1\}$
- $a \oplus b = (a + b) \mod p$
- $a \odot b = (ab) \mod p$
- We can $\pm$, $\times$, $\div$, inverses
- $W$ is a (primitive) element such that $1, W, W^2, \ldots, W^{p-1}$ are distinct
Finite field with $p$ elements

$p$ is prime

- $\mathbb{F}_p = \{0, 1, 2, \ldots, p - 1\}$
- $a \oplus b = (a + b) \mod p$
- $a \odot b = (ab) \mod p$
- We can $+, \times, \div$, inverses
- $W$ is a (primitive) element such that $1, W, W^2, \ldots, W^{p-1}$ are distinct

Example $\mathbb{F}_5$

- $W = 2$
- $W^0 = 1, W^1 = 2, W^2 = 4, W^3 = 3$
Finite field with $p$ elements

$p$ is prime
- $\mathbb{F}_p = \{0, 1, 2, \ldots, p - 1\}$
- $a \oplus b = (a + b) \mod p$
- $a \odot b = (ab) \mod p$
- We can $+, \times, \div$, inverses
- $W$ is a (primitive) element such that $1, W, W^2, \ldots, W^{p-1}$ are distinct

Example $\mathbb{F}_5$
- $W = 2$
- $W^0 = 1, W^1 = 2, W^2 = 4, W^3 = 3$

$p$ need not be prime
- Everything can be extended to finite fields with $q = 2^r$ elements
- May be extended to integers - not sure
$p$-symmetric channel and error correction

Error correction coding

- Another simple channel model which has been extensively considered
- Has been the canonical model for algebraic coding theorists
Generalized LDPC code and error channels

- GLDPC introduced by Tanner in 1981
- Each check is a \((\tilde{n}, \tilde{k})\), \(t\)-error correcting code
- If there are \(\leq t\) errors in a check, it can be recovered
- For now, assume no miscorrections
Peeling process is same for erasure and error channels

- Assume 1-error correcting check code and no miscorrections
- One-to-one correspondence between messages passed - DE can be used
- Not optimal for the error channel but it is not bad at high rates
- Spatially coupled versions are optimal at high rates (Jian, Pfister and N)
Erasures to errors - tensoring and peeling

\[ H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[ B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 & W^5 \end{bmatrix} \]

\[ \tilde{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & W^2 & W^3 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & W & 0 & 0 & W^4 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & W & W^2 & 0 & 0 & W^5 \end{bmatrix} \]

- \( W \) is a primitive element in the field
- Each check is a 1-error correcting code
- If there is exactly one error in a check, it can be recovered
**Product code**

- Special case of generalized LDPC code
- Let component code $\mathcal{C}$ be an $(\tilde{n}, \tilde{k}, \tilde{d}_{\text{min}})$ linear code
- Well-known that $\mathcal{P}$ is an $(\tilde{n}^2, \tilde{k}^2, \tilde{d}_{\text{min}}^2)$ linear code

| $X_{0,0}$ | $X_{0,1}$ | $X_{0,2}$ | $X_{0,3}$ | $X_{0,4}$ | $X_{0,5}$ | $X_{0,6}$ |
| $X_{1,0}$ | $X_{1,1}$ | $X_{1,2}$ | $X_{1,3}$ | $X_{1,4}$ | $X_{1,5}$ | $X_{1,6}$ |
| $X_{2,0}$ | $X_{2,1}$ | $X_{2,2}$ | $X_{2,3}$ | $X_{2,4}$ | $X_{2,5}$ | $X_{2,6}$ |
| $X_{3,0}$ | $X_{3,1}$ | $X_{3,2}$ | $X_{3,3}$ | $X_{3,4}$ | $X_{3,5}$ | $X_{3,6}$ |
| $X_{4,0}$ | $X_{4,1}$ | $X_{4,2}$ | $X_{4,3}$ | $X_{4,4}$ | $X_{4,5}$ | $X_{4,6}$ |
| $X_{5,0}$ | $X_{5,1}$ | $X_{5,2}$ | $X_{5,3}$ | $X_{5,4}$ | $X_{5,5}$ | $X_{5,6}$ |
| $X_{6,0}$ | $X_{6,1}$ | $X_{6,2}$ | $X_{6,3}$ | $X_{6,4}$ | $X_{6,5}$ | $X_{6,6}$ |
Product code

- Special case of generalized LDPC code
- Let component code $\mathcal{C}$ be an $(\tilde{n}, \tilde{k}, \tilde{d}_{\text{min}})$ linear code
- Well-known that $\mathcal{P}$ is an $(\tilde{n}^2, \tilde{k}^2, \tilde{d}_{\text{min}}^2)$ linear code
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t = 2$-error-correcting codes, bounded distance decoding

\[ \text{row codes} \quad \text{column codes} \]
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t = 2$-error-correcting codes, bounded distance decoding

Received block

Row decoding

Column decoding

Decoding successful

Or trapped in a stopping set

row codes

column codes
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t = 2$-error-correcting codes, bounded distance decoding
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t = 2$-error-correcting codes, bounded distance decoding
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: \( t = 2 \)-error-correcting codes, bounded distance decoding

![Diagram of peeling decoding](image)

Column decoding
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t = 2$-error-correcting codes, bounded distance decoding
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t = 2$-error-correcting codes, bounded distance decoding
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t=2$-error-correcting codes, bounded distance decoding
Peeling decoding of product codes

- Hard-decision “cascade decoding” by Abramson in 1968
- Identical to a peeling decoder
- Example: $t = 2$-error-correcting codes, bounded distance decoding

Or trapped in a stopping set
What is different about DE?

- Graph is highly structured
- Neighborhood is not tree-like
- Remarkably, randomness in the errors suffices!
### What is different about DE?
- Graph is highly structured
- Neighborhood is not tree-like
- Remarkably, randomness in the errors suffices!

### Assumptions
- Errors are randomly distributed in rows and columns
- \( \# \text{ errors in each row/col } \sim \text{Poisson}(M) \)
What is different about DE?

- Graph is highly structured
- Neighborhood is not tree-like
- Remarkably, randomness in the errors suffices!

Assumptions

- Errors are \textit{randomly distributed} in rows and columns
- \# errors in each row/col $\sim$ Poisson($M$)

Main Idea

- Removal of \textit{corrected vertices} (degree $\leq t$) from row codes $\Leftrightarrow$ removal of random edges from column codes uniformly at random
- \# of errors in row/column changes after each iter
- Track the distribution
Tail of the Poisson distribution

\[ \pi_t(m) = \sum_{j \geq t} e^{-m} \frac{m^j}{j!} \]

Effect of first step of decoding

If the # errors is Poisson with mean \( M \), Mean # of errors after decoding is

\[ m(1) = \sum_{j \geq t+1} j e^{-M} \frac{M^j}{j!} = M \pi_t(M) \]
Evolution of degree distribution ($d = 2$) - first iteration

**Row decoding**
- Before row decoding
  - Distribution: Poisson($M$), Mean: $M$
- After row decoding
  - Distribution: Truncated Poisson($M$)
  - Mean: $M\pi_t(M) = m(1)$

**Column decoding**
- Before column decoding
  - Distribution: Poisson($m(1)$), Mean: $m(1)$
- After column decoding
  - Distribution: Truncated Poisson($m(1)$)
  - Mean: $m(2) = M\pi_t(m(1))$

**After every decoding**
- Distribution is a Truncated Poisson($m(j)$)
- $P[\#\text{errors} = i] = b\frac{m(j)^i}{i!}$
Evolution of the degree distribution - $j$th iteration

Recursion
- $m(0) = M$
- $m(1) = M\pi_t(M)$
- $m(j) = M\pi_t(m(j - 1))$

Reduction in the parameter
- Average no. of errors in each row (column) = $m(j)\pi_t(m(j))$
- Decoding of rows reduces the parameter by $\frac{m(j)\pi_t(m(j))}{m(j-1)\pi_t(m(j-1))} = \frac{M\pi(m(j))}{m(j-1)}$
- New parameter is $m(j + 1) = M\pi(m(j))$

Threshold
In the limit of large $\tilde{n}$ (length in each dimension), a $t$-error correcting product code can correct $\tilde{n}M$ errors when

$$M < \min_m \left\{ \frac{m}{\pi_t(m)} \right\}$$
Thresholds for asymptotically large field size

Threshold = \frac{\text{\# of parity symbols}}{\text{\# of errors}}

<table>
<thead>
<tr>
<th></th>
<th>$d = 2$</th>
<th>$d = 3$</th>
<th>$d = 4$</th>
<th>$d = 5$</th>
<th>$d = 6$</th>
<th>$d = 7$</th>
<th>$d = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = 1$</td>
<td>4.0</td>
<td>2.4436</td>
<td>2.5897</td>
<td>2.8499</td>
<td>3.1393</td>
<td>3.4378</td>
<td>3.7383</td>
</tr>
<tr>
<td>$t = 2$</td>
<td>2.3874</td>
<td>2.5759</td>
<td>2.9993</td>
<td>3.4549</td>
<td>3.9153</td>
<td>4.3736</td>
<td>4.8278</td>
</tr>
<tr>
<td>$t = 3$</td>
<td>2.3304</td>
<td>2.7593</td>
<td>3.3133</td>
<td>3.8817</td>
<td>4.4483</td>
<td>5.0094</td>
<td>5.5641</td>
</tr>
<tr>
<td>$t = 4$</td>
<td>2.3532</td>
<td>2.9125</td>
<td>3.5556</td>
<td>4.2043</td>
<td>4.8468</td>
<td>5.4802</td>
<td>6.1033</td>
</tr>
</tbody>
</table>

Notice that $L, K = O \left( N^{\frac{1-d}{d}} \right)$
Syndrome source coding

- $Hx = 0$
- Receive - $r = x \oplus e$
- $Hr = He = y$
- Recover $x$ and sparse $e$

- $Hs = y$
- Set $r = 0$ (Let a genie add $x$ to $r$)
- $y$ is given to the decoder
- Recover sparse $s$
Problem Statement

\[ x[n] : \text{Time domain signal of length } N \text{ whose spectrum is } K\text{-sparse} \]

\[ x[n] \xrightarrow{\text{DFT}} X[k] \]

\((K\text{-sparse})\)

Compute the locations and values of the \(K\) non-zero coefficients w.h.p
Sparse Fast Fourier Transform (SFFT) Computation

Problem Statement

\[ x[n] : \text{Time domain signal of length } N \text{ whose spectrum is } K\text{-sparse} \]

\[ x[n] \xrightarrow{\text{DFT}} X[k] \]

\((K\text{-sparse})\)

Compute the locations and values of the \(K\) non-zero coefficients w.h.p

Fast Fourier Transform (FFT)

- **Sample complexity**: \(N\) samples
- **Computational complexity**: \(O(N \log N)\)

We want sublinear sample and computational complexity
Sparse Fast Fourier Transform (SFFT) Computation

Problem Statement

\[ x[n] : \text{Time domain signal of length } N \text{ whose spectrum is } K\text{-sparse} \]

\[ x[n] \xrightarrow{\text{DFT}} X[k] \]

\((K\text{-sparse})\)

Compute the locations and values of the \(K\) non-zero coefficients w.h.p

Related work

- Spectral estimation - Prony’s method
- More recently Pawar and Ramchandran’13, Hassanieh, Indyk, Katabi’12
Main Idea - Pawar and Ramchandran 2013

- **Sub-sampling** in time corresponds to **aliasing** in frequency
- Aliased coefficients $\Leftrightarrow$ parity check constraints of GLDPC codes
- CRT guided sub-sampling induces a code good for Peeling decoder
- Problem is identical to syndrome source coding
**SFFT - A Sparse Graph Based Approach**

### Main Idea - Pawar and Ramchandran 2013

- **Sub-sampling** in time corresponds to **aliasing** in frequency
- Aliased coefficients ⇔ parity check constraints of GLDPC codes
- CRT guided sub-sampling induces a code good for Peeling decoder
- Problem is identical to syndrome source coding

### FFAST for Computing the DFT - Pawar and Ramchandran 2013

- **Sampling complexity**: $M = O(K)$ time domain samples
- **Computational complexity**: $O(K \log K)$
Subsampling results in aliasing

- Let \( x[n] \xrightarrow{N-DFT} X[k] \), \( k, n = 0, 1, \ldots, N - 1 \)
- Let \( x_s[n] = x[mL] \), \( m = 0, 1, \ldots, N/L = M \) be a sub-sampled signal
- Let \( x_s[m] \xrightarrow{M-DFT} X_s[l] \) be the DFT of the sub-sampled signal

\[
X_s[l] = M \sum_{p=0}^{L-1} X[l + pM]
\]
Aliasing and Sparse Graph Codes

**$x_s$: Sub-sampled by $f_1 = P_1 = 2$**

\[
\begin{align*}
\text{DFT} &\quad X_s[0] &\quad X_s[1] &\quad X_s[2] \\
\end{align*}
\]

**$z_s$: Sub-sampled by $f_2 = P_2 = 3$**

\[
\begin{align*}
x[0] &\quad x[2] \\
\text{DFT} &\quad Z_s[0] &\quad Z_s[1] \\
x[0] &\quad x[2] \\
\end{align*}
\]

**Factor graph**

\[
\begin{align*}
X[0] &\quad X_s[0] = X[0] + X[3] \\
X[5] &\quad \text{...}
\end{align*}
\]
FFAST Algorithm Example

\[
x = (x[0], x[1], \ldots, x[6])
\]

\[
x_s = (x[0], x[2], x[4])
\]

\[
X_s = (X_s[0], X_s[1], X_s[2])
\]

\[
x_s = (x[1], x[3], x[5])
\]

\[
X_s = (X_s[0], X_s[1], X_s[2])
\]

\[
z_s = (x[0], x[3])
\]

\[
Z_s = (Z_s[0], Z_s[1])
\]

\[
z_s = (x[1], x[4])
\]

\[
\tilde{Z}_s = (\tilde{Z}_s[0], \tilde{Z}_s[1])
\]

\[
w = e^{-j\frac{2\pi}{6}}
\]

\[
X[0]
\]

\[
X[1]
\]

\[
X[2]
\]

\[
X[3]
\]

\[
X[4]
\]

\[
X[5]
\]
Singleton Detection

Let \( i = \frac{-N}{j2\pi} \log\left(\frac{\tilde{X}_s[l]}{X_s[l]}\right) \). If \( 0 \leq i \leq N - 1 \), then checknode \( l \) is a Singleton.

\( Pos(l) = i \) is the only variable node participating and \( X_s[l] \) is its value.

\[ w = e^{-j\frac{2\pi}{6}} \]

\[ X_s[0] = X[0] + X[3] \]
\[ \tilde{X}_s[0] = X[0]w^0 + X[3]w^3 \]

\[ \tilde{X}_s[1] = X[1]w^1 + X[4]w^4 \]


\[ \tilde{Z}_s[0] = X[0]w^0 + X[2]w^2 + X[4]w^4 \]

FFAST Decoder

\[ w = e^{-j\frac{2\pi}{6}} \]

\[
\begin{align*}
X[0] & \quad X_s[0] = X[0] + X[3] \\
& \quad \tilde{X}_s[0] = X[0]w^0 + X[3]w^3 \\
& \quad \tilde{X}_s[1] = X[1]w^1 + X[4]w^4 \\
X[3] & \quad \vdots
\end{align*}
\]

\[
\begin{align*}
& = X[0]w^0 + X[2]w^2 + X[4]w^4 \\
\end{align*}
\]

Peeling decoder

- 1 non-zero value among the neighbors of any right node can be recovered
- Iteratively errors can be corrected and analyzed for random non-zero coeffs
Example 1

Example 1

$$w = e^{-j\frac{2\pi}{6}}$$

<table>
<thead>
<tr>
<th>$X[0]$</th>
<th>$X_s[0] = 9$</th>
<th>$\hat{X}_s[0] = 5w^0 + 4w^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X[1]$</td>
<td>$X_s[1] = 7$</td>
<td>$\hat{X}_s[1] = 7w^4$</td>
</tr>
<tr>
<td>$X[3]$</td>
<td>$Z_s[0] = 12$</td>
<td>$\hat{Z}_s[0] = 5w^0 + 7w^4$</td>
</tr>
<tr>
<td>$X[5]$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 1


$$w = e^{-j \frac{2\pi}{6}}$$

Yes, recoverable!
Reed Solomon component codes

- \((X_s[l_1], \tilde{X}_s[l_1])\) correspond to 2 syndromes of a 1-error correcting RS code
- RS is over the complex field, no miscorrection
Product codes and FFAST ($d = 2$)

- $X$: $K$-sparse spectrum of length $N = P_1 P_2$ ($P_1$ and $P_2$ are co-prime)
- $X'$: $P_1 \times P_2$ matrix formed by rearranging $X$ according to mapping $\mathcal{M}$

$$X_s[l_1] = \sum_{i=0}^{P_2-1} X[l_1 + iP_1], \quad 0 \leq l_1 \leq P_2 - 1$$

$$Z_s[l_2] = \sum_{i=0}^{P_1-1} X[l_2 + iP_2], \quad 0 \leq l_2 \leq P_1 - 1$$

**Mapping**

The mapping from $X(r)$ to $X'(i, j)$ is given by

$$(i, j) = \mathcal{M}(r) \equiv (r \mod P_2, r \mod P_1).$$

**Note:** CRT ensures that $\mathcal{M}$ is bijective
Product codes and FFAST \((d \geq 3)\)

\[ N = P_1 \times P_2 \times \ldots \times P_d \]

\[(i_1, i_2, \ldots, i_d) = \mathcal{M}(r) \equiv (r \mod f_1, r \mod f_2, \ldots, r \mod f_d).\]

**Less-sparse regime**

\[ f_i = N/P_i, \ i = 1, 2, \ldots, d \]

\[ d = 3 \]

**Very-sparse regime**

\[ f_i = P_i, \ i = 1, 2, \ldots, d \]

\[ d = 3 \]
Connections between FFAST and Product Codes

FFAST
- $d$ stages
- $2t$ branches
- Non-zero coefficients
- Recovery of coefficients

$\iff$

Product codes
- $d$-dimensional product code
- $t$-error correcting RS component codes
- Error locations
- Iterative decoding
**Thresholds**

**Theorem 1**

*Less sparse case: In the limit of large $P$, the FFAST algorithm with $d$ branches and $2t$ stages can recover the FFT coefficients w.h.p if $K < \frac{2dt}{c_{d,t}}$.*

$$c_{d,t} = \min_m \{m/\pi^{d-1}(m)\}$$

**Threshold** = \frac{\text{# of measurements}}{\text{recoverable sparsity}} = \frac{2dt}{c_{d,t}}

<table>
<thead>
<tr>
<th>$t$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4.0</td>
<td>2.4436</td>
<td>2.5897</td>
<td>2.8499</td>
<td>3.1393</td>
<td>3.4378</td>
<td>3.7383</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.3874</td>
<td>2.5759</td>
<td>2.9993</td>
<td>3.4549</td>
<td>3.9153</td>
<td>4.3736</td>
<td>4.8278</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.3304</td>
<td>2.7593</td>
<td>3.3133</td>
<td>3.8817</td>
<td>4.4483</td>
<td>5.0094</td>
<td>5.5641</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.3532</td>
<td>2.9125</td>
<td>3.5556</td>
<td>4.2043</td>
<td>4.8468</td>
<td>5.4802</td>
<td>6.1033</td>
<td></td>
</tr>
</tbody>
</table>

Notice that $L, K = O\left(N^{\frac{1-d}{d}}\right)$
Interference-tolerant A/D Converter
Open problems

- If we use MAP decoding, is the subsampling procedure optimal?
- What happens when $N = 2^i$?
- Bursty case? Can we have threshold theorems?
- Using this idea in actual applications
Syndromes and decoding

\[ H_{m \times n} \]

(Parity checks)

\[ c_{n \times 1} \]

(code vector)

\[ e \]

(error vector)

\[ t - \text{sparse} \]

\[ \begin{align*}
    H \times c_{n \times 1} + e &= 0 \\
    y_{m \times 1} &= \text{Syndromes}
\end{align*} \]
Syndromes and decoding

- Syndrome: Linear combination of $h_i$s, i.e., $y = e_i h_i \oplus e_j h_j \oplus e_t h_t$
- Decoding: Find min weight $e$: $y = e_i h_i \oplus e_j h_j \oplus e_t h_t$
Syndromes and decoding

- Syndrome: Linear combination of $h_i$s, i.e., $y = e_i h_i \oplus e_j h_j \oplus e_t h_t$
- Decoding: Find min weight $e$: $y = e_i h_i \oplus e_j h_j \oplus e_t h_t$

Coding theory deals with the construction of $H$ and efficient decoding algorithms, i.e., given a linear combination of the columns of $H$, it develops tools to determine a sparse $e$. 
Syndrome source coding

\[ H_{m \times n} \times (\text{Source vector}) = \text{Syndromes} \]

Compression of a sparse binary source
- Compressed version is the syndrome \( y \)
- Reconstruction is the same as decoding
- Similar to the canonical sparse recovery problem
Review of primitives

- Idea of a check node or a measurement node which is a function of some symbols
- Singleton detection - be able to identify one non-zero symbol
- Peeling - if we know some symbols, be able to remove and adjust measurement
Application 3
Compressed sensing

\[ y_{m \times 1} = A_{m \times n} x_{n \times 1} \]

Classical compressed sensing

- \( x \) is a \( K \)-sparse vector over \( \mathbb{R} \) or \( \mathbb{C} \)
- We ‘compress’ \( x \) by storing only \( y = Ax \)
- Reconstruction - Solve \( \hat{x} = \arg \min ||z||_0 : y = Az \)
- CS - Solve \( \hat{x} = \arg \min ||z||_1 : y = Az \)
Compressed sensing

Classical compressed sensing

- $x$ is a $K$-sparse vector over $\mathbb{R}$ or $\mathbb{C}$
- We ‘compress’ $x$ by storing only $y = Ax$
- Reconstruction - Solve $\hat{x} = \text{arg min} \|z\|_0 : y = Az$
- CS - Solve $\hat{x} = \text{arg min} \|z\|_1 : y = Az$

Coding theoretic approach - syndrome source coding over complex numbers

- Sensing matrix $A \leftrightarrow$ Parity check matrix $H$
Problem - consider a router in a large network

- Count the number of packets from source $i$ to destination $j$, say $x_{ij}$
- Data vector is huge, $n = 2^{64}$
- Heavy hitters - only a few of them are large

Keep only a low dimensional ($m \ll n$) sketch of $x$

- $y_{m \times 1} = Ax \iff$ Syndrome, $x_{i,j} \in \mathbb{Z}^+$
- Reconstruction is same as decoding
Sketch $y$ supports incremental updates to $x$ as the sketching procedure is linear. 

$$x + \Delta_i \rightarrow y + A\Delta_i$$

(adding $i$th column vector of $A$ to existing sketch)
Compressed Sensing (Li, Ramchandran '14)

\[ y_{m \times 1} = A_{m \times n} x_{n \times 1} \]

Sketching matrix \((A_{m \times n})\)

\[ A_{m \times n} = \mathbf{H}_{m/2 \times n} \bigotimes \mathbf{B}_{2 \times n} \]

\((d\text{-left regular Graph}) \quad (\text{Singleton identifier})\)

\[ \mathbf{B} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{n-1} \end{bmatrix} \quad \mathbf{W} = e^{j \frac{2\pi}{n}} \]
Main results for compressed sensing

**Noiseless case**
- Samples: $2K$ versus Info-theoretic limit $K + 1$
- Computations: $O(K)$ versus $O(K^2)$
- If $K = O(n^\delta)$, small price to pay in terms of samples

**Noisy case**
- Sample: $O(k \log^{1.3} n)$ vs limit: $O(k \log(n/k))$ necessary and sufficient
- Computations: $O(k \log^{1.3} n)$
Main results for compressed sensing

Noiseless case
- Samples: $2K$ versus Info-theoretic limit $K + 1$
- Computations: $O(K)$ versus $O(K^2)$
- If $K = O(n^\delta)$, small price to pay in terms of samples

Noisy case
- Sample: $O(k \log^{1.3} n)$ vs limit: $O(k \log(n/k))$ necessary and sufficient
- Computations: $O(k \log^{1.3} n)$

Vem, Thenkarai Janakiraman, N. ITW'16
- Sample: $O(k \log^{1.3} (n/k))$ vs limit: $O(k \log(n/k))$ necessary and sufficient
- Computations: $O(k \log^{1.3} (n/k))$
Group Testing (Lee, Pedarsani, Ramchandran ’15)

- II World War - detect all soldiers with syphilis
- Tests performed on efficiently pooled groups of items
- Least no. of tests \( m \) to identify \( k \) defective items from \( n \) items
Group Testing

Example

Test Results (Observations)

\[ y_i = \bigvee_{j=1}^{N} a_{ij} X_j = < a_i, X > \]

Positive

\[ y_1 = 1 \]

Negative

\[ y_1 = 0 \]

Positive

\[ y_1 = 1 \]

Positive

\[ y_1 = 1 \]
Group Testing

\[
Y_{m \times 1} = A \odot X \quad \text{(Observation vector)}
\]

\[
A_{m \times n} = \begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_m \\
\end{bmatrix}
\quad \text{(Pooling matrix)}
\]

\[
X_{n \times 1} = \begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_n \\
\end{bmatrix}
\quad \text{(K-sparse)}
\]

\[
Y_{m \times 1} = \left[ \begin{array}{c}
< a_1, X > \\
< a_2, X > \\
\vdots \\
< a_m, X > \\
\end{array} \right] = \bigvee_{j=1}^{N} a_{ij} X_j
\]
Singleton detection

\[
\begin{bmatrix}
H_1 \\
H_1
\end{bmatrix} = \begin{bmatrix}
b_1 \\
b_2 \\
b_3 \\
\vdots \\
b_{n-1}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & \cdots & 1 & 1 \\
0 & 0 & 0 & \cdots & 1 & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 1 & \cdots & 1 & 1 \\
0 & 1 & 0 & \cdots & 0 & 1 \\
1 & 1 & 1 & \cdots & 0 & 0 \\
1 & 1 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & 1 & 0 & \cdots & 0 & 0 \\
1 & 0 & 1 & \cdots & 1 & 0
\end{bmatrix}
\]

**Note:** If a checknode is a singleton, with \(i\)th bit-node participating, then the observation vector is the \(i\)th column of \(A\).

- **Singleton** - if the weight of first two observation vectors together is \(L\).
- **Position** of the defective item is - decimal value of the 1st observation vector.
Group Testing

Measurement matrix \((A_{m \times n})\)

\[
A_{m \times n} = \begin{array}{c}
G_{m \times n} \\
\otimes \\
H_{6 \times n}
\end{array}
\]

\((d\text{-left regular Graph}) \quad (\text{Singleton identifier})\)

Let, \(b_i\) denote the \(L\)-bits binary representation of the integer \(i - 1\), \(L = \lceil \log_2 n \rceil\).

\[
H = \begin{bmatrix}
b_1 & b_2 & b_3 & \cdots & b_{n-1} \\
b_1 & b_2 & b_3 & \cdots & b_{n-1} \\
b_{i_1} & b_{i_2} & b_{i_3} & \cdots & b_{i_{n-1}} \\
b_{i_1} & b_{i_2} & b_{i_3} & \cdots & b_{i_{n-1}} \\
b_{j_1} & b_{j_2} & b_{j_3} & \cdots & b_{j_{n-1}} \\
b_{j_1} & b_{j_2} & b_{j_3} & \cdots & b_{j_{n-1}}
\end{bmatrix}
\]

\(s_1 = (i_1, i_2, \cdots, i_{n-1})\) and \(s_2 = (j_1, j_2, \cdots, j_{n-1})\) are permutations

Decoding procedure

- Identify and decodes singletons using weights of the observation vector
- Identify and resolve doubletons by guessing to satisfy the first pair of observation vectors and checking if the guess satisfies the other two pairs of observations
Main results for group testing

Non-adaptive Group Testing (Noiseless and Noisy)

- Recovers $(1 - \epsilon)k$ items with h.p.
- Samples: $m = O(k \log_2 n)$ versus limit: $\Theta(k \log(\frac{n}{k}))$
- Computational complexity: $O(k \log n)$ (order optimal)
Compressive Phase Retrieval

\[ X \in \mathbb{C}^n \] (K-sparse)

\[ A_{m \times n} \rightarrow |AX| \rightarrow \text{Decoder} \rightarrow \hat{X} \]

- Measurement vectors
- Magnitude measurements
- Estimated signal
Compressive Phase Retrieval

\[ y = |AX| \]

\[ \hat{X} \]

\[ X \in \mathbb{C}^n \]
(K-sparse)

\[ y_{m \times 1} \]
(observations)

\[ A_{m \times n} \]
(m ≪ n)

\[ x_{n \times 1} \]
(K-sparse)
Conclusion

• Review of a simple message passing decoder called the peeling decoder
• Density evolution as a tool to analyze its asymptotic performance
• Applications
  • Massive uncoordinated multiple access
  • Sparse Fourier transform computation
  • Compressed sensing type sparse recovery problems
Questions?

Thank you!