Simple erasure decoding algorithm

- Want to recover input/source bits (packets) from received output/transmitted bits (packets).
- Decoder needs to know which input bits correspond to each output bit, e.g.
  - Sender & receiver use identical pseudorandom no. generators seeded by synchronized clocks or by key in packet.
- Special case of belief propagation (facts).

1. Find an output node \( t_i \) that is connected to only one input node \( x_j \):
   a) Set \( x_j = t_i \);
   b) Add \( x_j \) to all output nodes \( t_i \) that are connected to \( x_j \);
   c) Remove all edges connected to \( x_j \).
2. Repeat (1) until all \( x_j \) are found.
- Success of decoding algorithm depends on degree distribution

- # encoding & decoding operations roughly proportional to # edges (good for sparse graphs w/ # edges proportional to # nodes)

- Code is an irregular low density generator matrix code
Raptor Codes

Key idea:
• relax condition that the LT code has to recover all its input symbols
• concatenate outer block code with inner LT code

A Raptor Code is specified by parameters \((k, c, \Omega(x))\)

• \(c\) is an \((n, k)\) erasure-correcting block code (pre-code)
• \(\Omega(x)\) is the degree distribution of the LT code:
  \[
  \Omega(x) = \sum \Omega_i x^i
  \]
  where \(\Omega_i\) is the prob that an output node has degree \(i\)
Raptor Code Performance Metrics

- Overhead:
  \[ \frac{\text{# output symbols to recover } k \text{ input symbols reliably}}{k} \]

- Decoding Cost:
  \[ \frac{E(\text{# operations to recover } k \text{ input symbols})}{k} \]

- Encoding Cost:
  \[ \frac{E(\text{# operations for encoding } n \text{ intermediate symbols})}{k} + \text{encoding cost of LT code} \]

- Space Requirement:
  \[ \frac{\text{# intermediate symbols}}{\text{# input symbols}} = \frac{n}{k} = \frac{1}{R} \]
First Examples of Raptor Codes

- LT Codes
  - trivial pre-code
  - sophisticated LT code output distribution
  - logarithmic encoding & decoding cost
  - overhead close to 1
  - space requirement = 1

- Pre-Code-Only (PCO) Raptor Codes
  - sophisticated pre-code
  - trivial LT code output distribution
    \[ \Omega(x) = x \]
  - encoding & decoding cost = that of pre-code
  - overhead = \(-\ln (1-R(1+\varepsilon))/R \approx 1+\varepsilon \) as \(R \to 0\)
  - space requirement = \(1/R\)
RAPTOR CODES WITH GOOD ASYMPTOTIC PERFORMANCE

- Constant encoding & decoding costs
- Space consumption & overhead arbitrarily close to 1
- Possible to satisfy these requirements simultaneously with appropriate pre-code & output distribution.
EDGE DEGREE DISTRIBUTIONS

Consider an LT code with parameters $(n, \Omega(x))$, & the graph associated with $m$ of its output symbols

- $\Omega(x) = \text{output node degree distribution}$
- $a = \Omega'(1) = \text{average output node degree}$
- $w(x) = \text{output edge degree distribution}$
  $$\frac{\Omega'(x)}{\Omega'(1)} = \frac{\Omega'(x)}{a} = \sum_i w_i x^{i-1}$$

- $\Psi(x) = \text{input node degree distribution}$
  $$= (1 - \frac{a(1-x)}{n})^m$$

- $\psi(x) = \text{input edge degree distribution}$
  $$\frac{\Psi'(x)}{\Psi'(1)} = (1 - \frac{a(1-x)}{n})^{m-1} = \sum_i \psi_i x^{i-1}$$
Consider an edge \((v, w)\) chosen uniformly at random from the original graph.

\[ w(p) = \sum_i w_i p^{i-1} \]

- prob that output node \(w\) is released at some stage where \((v, w)\) has not been deleted & a proportion \(p\) of input nodes have been recovered
  (need all other neighbors of \(w\) to be recovered)

\[ \zeta(p) = \sum_i \zeta_i p^{i-1} \]

- prob that input node \(v\) is not recovered at some stage where \((v, w)\) has not been deleted & a proportion \((1-p)\) of output nodes have been released
  (all other neighbors of \(v\) not recovered)
Consider the subgraph $G_z$ induced by the left node $v$ and all neighbors of $v$ within distance $2h$ after deleting $(v, w)$.

- The probability that $G_z$ is not a tree is proportional to $\frac{1}{n}$, $\to 0$ as $n \to \infty$.
- Asymptotic distribution of shape of $G_z$ as $n \to \infty$: random And-Or tree.

- Each OR node evaluates to the 'OR' of its children (0 if no children).
- Each AND node evaluates to the 'AND' of its children.
- Let $y_z$ denote the probability that $v$ evaluates to 0 when each leaf is 1.

\[
y_z = 1 - w(1-y_{z-1})
\]
Want $y_2 \to s$ as $l \to \infty$, which happens if

$$1(1 - w(1-x)) < x \quad (y_2 < y_{x-1})$$

\forall x \in [s, 1]$$
PROOF OF ASYMPTOTIC PERFORMANCE

- show that $\ell(x) \& w(x)$ induced by $\Omega_D(x)$ satisfy
  $\ell(1-w(1-x)) < x$

  → ensures recovery of $(1-S)n$

  intermediate symbols from
  $(1 + \varepsilon/2)n + 1$ output symbols

  → allows decoder for $C_n$ to

  recover input symbols

- overall overhead = $\frac{n(1+\varepsilon/2)}{\ln n} = 1 + \varepsilon$

- encoding & decoding cost of
  LT code proportional to $\Omega_D^\prime(1)$

  = $1 + \frac{H(D)}{1+\mu} = \ln(\frac{1}{\varepsilon}) + \alpha + O(\varepsilon)$

  where $H(D)$ is the harmonic sum up to $D$, &

  $1 < \alpha < 1 + \gamma + \ln(9)$,

  $\gamma = Euler's\\ constant$
AN ASYMPTOTICALLY GOOD RAPTOR CODE

- Pre-code: \((n,k)\) code \(C_n\) of rate 
  \[ R = \frac{k}{n} = \frac{1 + \varepsilon/2}{1 + \varepsilon} \] 
  that can be decoded on a BEC of erasure probability 
  \[ S = \frac{\varepsilon}{4(1 + \varepsilon)} = \frac{1 - R}{2} \] 
  with \(O(n \log(1/\varepsilon))\) encoding & decoding operations 
  e.g. tornado codes, right-regular codes, etc

- LT code: parameters \((n, \Omega_D(x))\),
  where
  \[ \Omega_D(x) = \frac{1}{\mu + 1} \left( \mu x + \frac{x^2}{2} + \frac{x^3}{2 \cdot 3} + \ldots \right) \]
  \[ \ldots + \frac{x^D}{(D-1)D} + \frac{x^{D+1}}{D} \]
  \[ D = \lceil \frac{4(1 + \varepsilon)}{\varepsilon} \rceil \], \(\mu = \frac{\varepsilon}{2} + \left(\frac{\varepsilon}{2}\right)^2\)
Design of finite length raptor codes

- LT code component is obtained by linear programming on a heuristically chosen problem:
  - minimize # edges subject to constraint that the expected # input nodes recovered during each round is at least the square root of some constant factor $c \times (# \text{ unrecovered input nodes})$

- Precode is an LDPC code with optimized degree distribution