Tornado and Luby Transform Codes

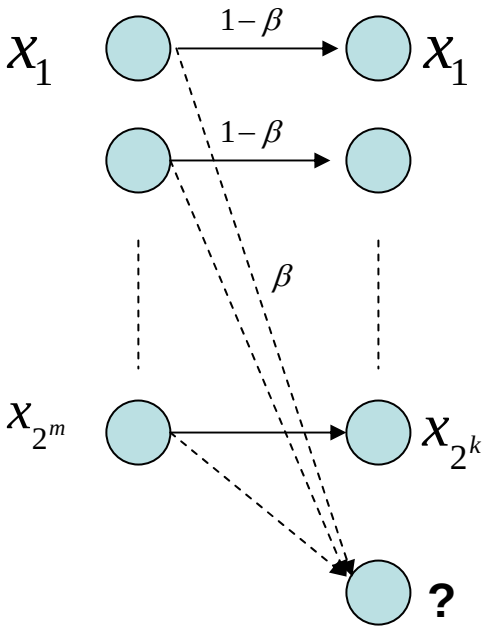
Ashish Khisti

6.454 Presentation

October 22, 2003

Background: Erasure Channel

Elias[1956] studied the Erasure Channel

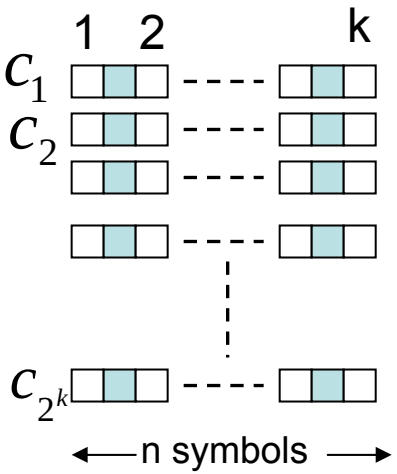


- Capacity of Noiseless Erasure Channel is $m(1-\beta)$
- No Feedback is necessary to achieve capacity
- A random linear code can achieve capacity.
  Encoding: $O(n^2)$ Decoding: $O(n^3)$

Applications

- Communication Links over the Internet
- Storage Media

Classical MDS Codes



Features

- Any set of k co-ordinates is an information set for *(n,k,d)* MDS Code.
- The receiver knows the codeword once it receives any k symbols *and* knows their positions.
- Capacity achieving codes.

Drawbacks

- Reed Solomon Codes (RS) codes require $O(k^2)$ time for decoding.
- Block codes : Need prior knowledge of erasure probability.

Digital Fountain Approach

- A new protocol for *bulk data distribution*
- Scenario: One Server multiple Receivers

Encoding:

Construct encoding symbols on the fly and send them when atleast one receiver is listening.

Decoding:

Collect desired number of symbols from the server and reconstruct the original file

Goals:

Reliable, Efficient, On Demand and Tolerant

Tornado Codes

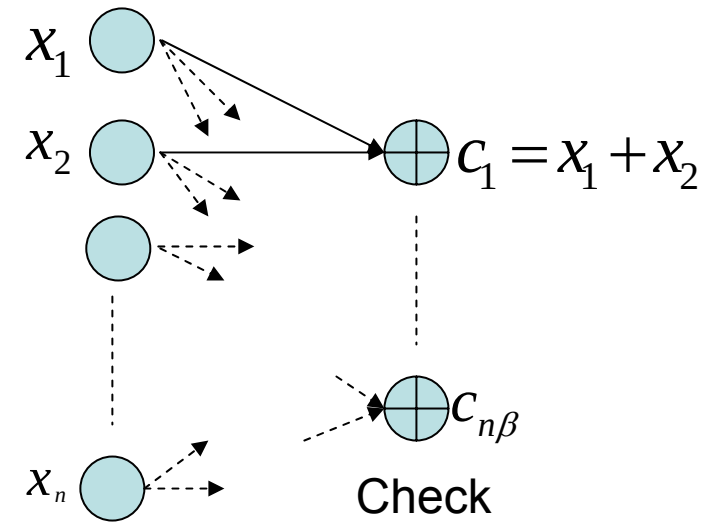
Features

- Correct 1-R(1- ε) errors over BEC.
- Time for encoding and decoding is proportional to $n \log\left(1/\varepsilon\right)$
- Very fast software implementations.

Tradeoffs

- The assumption of independent erasures is critical.
- High latency.
- Low Rate Implementations are less attractive.
- Block Codes – Not suitable for heterogeneous receiver population.

Irregular Bipartite Graph



- Irregular Random Graphs are used for generating check symbols
- $(x_1, x_2, \ldots x_n) \longrightarrow (x_1 \ldots x_n, c_1 \ldots c_{n\beta})$

Degree Sequences

Left Degree Sequence: $(\lambda_1, \lambda_2, \ldots \lambda_n)$

Right Degree Sequence: $(\rho_1, \rho_2 \ldots \rho_m)$

Definition: $\lambda_k(\rho_k)$ is the fraction of edges that are incident on a left(right) node of degree k.

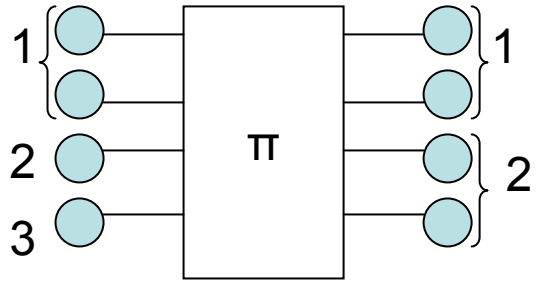
Irregular Graphs: Example

Given:

$(\lambda_1, \lambda_2) = (1/2,1/2)$

$(\rho_1, \rho_2) = (0,1)$

Number of Edges E = 4

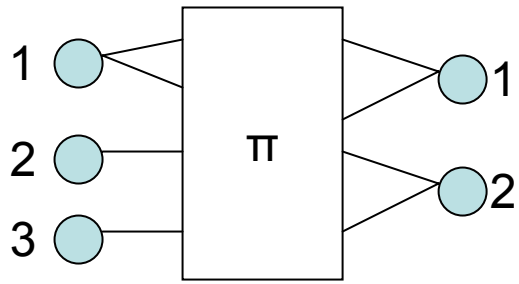


l_i = number of left nodes of degree *i*

r_i = number of right nodes of degree *i*

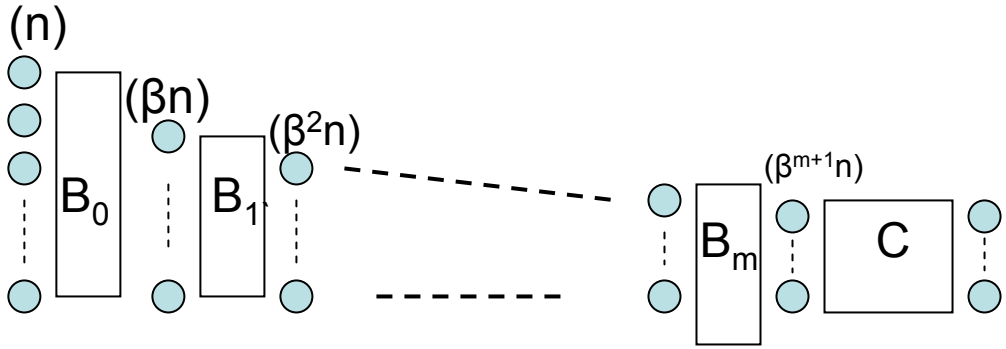
$$l_i = \frac{\lambda_i E}{i} \qquad r_i = \frac{\rho_i E}{i}$$

$$(l_1, l_2) = (2,1) \qquad (r_1, r_2) = (0,2)$$



Random permutation between edges, induces a uniform distribution over the ensemble.

Construction of Tornado Codes



B_i: Irregular Graph

C : Conventional Code

Code $C(B_0,B_1,B_2 \ldots B_m,C)$:

- Each B_i is an irregular bipartite graph with same degree sequences
- C is a conventional rate (1- β) code with $O(n^2)$ complexity.
- *m* is chosen such that $n\beta^m \approx \sqrt{n}$

Length of *C:*

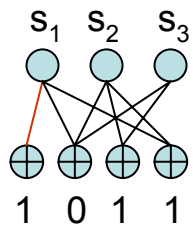
$$\sum_{i=0}^{m+1} n\beta^i + \frac{n\beta^{m+2}}{1-\beta} = \frac{n}{1-\beta}$$

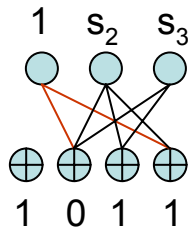
This is a rate (1- β) code with encoding/decoding complexity of O(n).

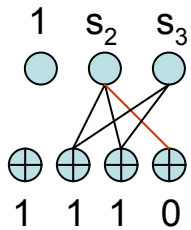
Linear Time Decoding Algorithm

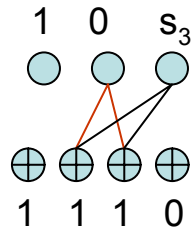
1. Find a check node c_n that is connected to only one source node s_k. If no such c_n stop and declare error.
   (a) set $s_k = c_n$
   (b) find all $c_{n'}$ that are neighbors of s_k and set
   $$c_{n'}=c_{n'} + s_k$$
   (c) Remove all edges connected to s_k

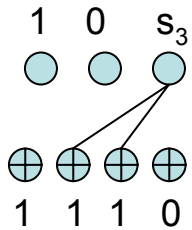
2. Repeat (1) until all source nodes are determined.

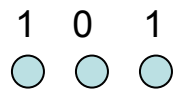
1.


2.


3.


4.


5.


6.


What has to be solved?

So far…

- Identified the structure of encoder as a cascade of irregular bi-partite graphs.
- Suggested a candidate decoding algorithm which has linear complexity.

Goal:

Specify the set of degree sequences $(\lambda_1, \lambda_2, \ldots \lambda_n)$ and $(\rho_1, \rho_2 \ldots \rho_m)$ for which the this simple decoding algorithm succeeds.

Main Contribution of the Paper:

1. Develop mathematical conditions on the degree sequences under which this decoding scheme succeeds
2. Provide explicit degree sequences that achieve the capacity of BEC.

Conditions on Degree Sequences

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

δ: Erasure Probability of the memoriless channel

Necessary Condition: If the decoding algorithm succeeds in recovering all message symbols then

$$\rho(1-\delta\lambda(x)) > 1-x, \forall x \in (0,1]$$

Approach: Compute the expected value of the fraction of edges with degree one on the right and require that it is > 0

Sufficient Condition: The above condition is also sufficient if we impose $\lambda_1 = \lambda_2 = 0$.

Approach: The proof uses tools from statistical mechanics to show that the variance in degree distribution is small.

Capacity Achieving Distribution

Fix an integer D > 0, Let

$$\rho_i = \frac{e^{-\alpha}\alpha^{i-1}}{(i-1)!}, i = 1.2,3... \qquad \lambda_i = \frac{1}{H(D)(i-1)}, i = 2,3...D+1$$

- Average degree of left nodes = $E / \left(\sum_i \frac{\lambda_i}{i} \right) E = \left(\sum_i \frac{\lambda_i}{i} \right)^{-1} \approx \log(D)$
- Average degree of right nodes = $\left(\sum_i \frac{\rho_i}{i} \right)^{-1} = \frac{\alpha e^{\alpha}}{e^{\alpha} - 1} \approx \frac{\log(D)}{\beta}$

Intuition:

- Poisson distribution is natural if all the edges from the left uniformly choose the right nodes.
- This distribution is preserved when edges are successively removed from the graph.
- Heavy tail distribution produces some message nodes of high degrees that get decoded first and remove many edges from the graph.

Capacity Achieving Distribution

- Note that:

$$\rho(x) = e^{\alpha(x-1)} \qquad \lambda(x) = -\log(1-x)_D$$

- For the above choice of ρ(x) and λ(x), it is easy to verify that $\rho(1-\delta\lambda(x)) > 1-x, \forall x \in (0,1]$ whenever,

$$\delta < \beta / 1+1/D$$

- Let D = 1/ε. It follows that ≈ β(1- ε) fraction of erasures can be corrected by this rate 1- β code. The maximum degree ≈ log(D), implies that the number of operations in decoding is proportional to nlog(1/ ε).

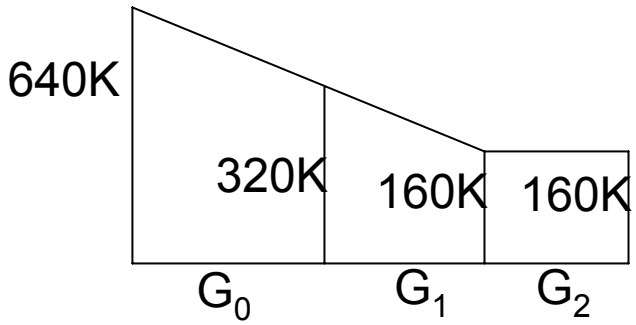
Linear Programming Approach

- Fix ($\lambda_1, \lambda_2, \ldots \lambda_n$) and δ. The objective is to find ($\rho_1, \rho_2 \ldots \rho_m$) for some fixed m.
- Let $x_i = i/N$, for *i*=1,2..N. We have the following constraints:
  - $\rho(1- \delta \lambda(x_i)) > 1-x_i$
  - $\rho_i \geq 0$ and $\rho(1) = 1$

  Minimize $\sum_{i=1}^{N} (\rho(1-\delta\lambda(x_i)+ x_i -1)$
- The solution for $\rho(x)$ is feasible if the inequality holds for all x in (0,1]
- Once a feasible solution has been found the optimal δ is found by a binary search.
- An iterative approach is suggested that uses the dual condition

  $\delta\lambda(1-\rho(y)) < y, \forall y \in (0,1]$

Practical Implementations



Tornado Z Code

- Rate ½ code. Takes 640,000 packets (each 256 byte) as input.
- Only three cascades have been used.
- G_0 and G_1 use heavytail / poisson distribution as noted.
- G_2 cannot use a standard quadratic time code. It's degree distribution is obtained through linear programming.
- On a 200MHz Pentium machine, the decoding operation takes 1.73 seconds.

Issues

- The assumption of independent erasures is critical in design of Tornado codes. So deep interleaving and very long block lengths are necessary.
- High latency is incurred in encoding and decoding operations, since both encoding and decoding must be delayed by at least one block size.
- Heavy memory usage: Decoding each block of Tornado Z requires 32 MB of RAM.
- Since they are block codes, they have to be optimized for a particular rate.
- The number of encoding symbols is fixed when the input block length and rate is fixed.

Luby Transform Codes

Features

- *k* input symbols can be recovered from any set of $k + O(\sqrt{k}\log^2(k/\delta))$ symbols with probability 1-δ.
- Encoding time: O(log(k/ δ)) per encoding symbol.
- Decoding time: O(k log(k/ δ))

- These codes are rate-less: the number of distinct encoding symbols that can be generated is extremely large.
- Encoding symbols are generated on the fly
- The construction does not make use of channel erasure probability and hence can optimally serve heterogeneous receivers

Encoding of LT Codes

Fix a degree distribution $\rho(d)$

To produce each encoding symbol:

- Generate the degree $D \sim \rho(d)$
- For each of the D edges, randomly pick one input symbol node.
- Compute the XOR of all the D neighbors and assign this value to the encoding symbol.

How does the decoder know the neighbors of an encoding symbol it receives?

- This information can be explicitly included as an overhead in each packet.
- Pseudo-randomness can be exploited to duplicate the encoding process at the receiver. The receiver has to be given the seed and/or keys associated with the process.

Decoding LT Codes

- The decoding process is virtually the same as that of Tornado Codes.
- At the start **release** all encoding symbols of degree 1. Their neighbors are now covered and form a **ripple**.
- In each subsequent step, process one message symbol from the ripple is **processed**. It is removed as a neighbor of the encoding symbols.
- If any encoding symbol now has a degree one, it is released. If its neighbor is not already in the ripple, it gets added to the ripple.
- The process ends when the ripple is empty. If some message symbols remain unprocessed, this is a decoding failure.

LT Analysis-1 (ρ(1)=1)

- How many encoding symbols (each of degree 1) will guarantee that all message symbol nodes will be covered with probability > 1- δ ?

Ans: k log(k/ δ).

The probability that a message node is not covered is

$$(1-1/k)^{k\log(k/\delta)} \approx \delta/k$$

By using the union bound estimate, the desired result follows.

- k log(k/ δ) encoding symbols is unacceptable.
- Since all edges are randomly incident on message nodes,

  k log(k/ δ) edges are required to cover all the nodes.

LT Analysis-2

Suppose L input symbols remain unprocessed during a decoding step.

- Any encoding symbol is equally likely to get released independent of all other encoding symbols.
- If an encoding symbols is released, it is equally likely to cover any of the L symbols.

Define: q(i,L) = probability that an encoding symbol of degree i is released, when L input symbols remain unprocessed.

$$q(1,k) = 1$$

$$q(i,L) = \frac{{}^{i}C_{i-2}\,{}^{k-(L+1)}P_{i-2}\,{}^{2}C_{1}\,{}^{L}C_{1}}{{}^{k}P_{i}}, i = 2..k, L = k-i+1..1$$

$$q(i,L) = 0 \quad \text{otherwise}$$

LT Analysis-3

Soliton Distribution:

$$\rho(1) = 1/k$$
$$\rho(i) = 1/i(i-1), i = 2..k$$

r(L) : probability that a an encoding symbol is released

$$r(L) = \sum_i \rho(i) q(i, L) = \frac{1}{k} \sum_i \frac{L \dfrac{(k-1-L)!}{(k-L-1-(i-2))!}}{\dfrac{(k-1)!}{(k-i)!}} = \frac{1}{k}$$

- Thus at each step, we expect one encoding symbol to be released
- The size of the ripple at each step is one.

Properties of Soliton Distribution

- At each step one encoding symbol is released. Only k encoding symbols are needed on average to retrieve k input symbols.
- It expected number of edges in the graph is k log(k).
- The ideal soliton distribution compresses the number of encoding symbols to the minimum possible value, keeping the number of edges in the graph minimum.
- The ideal soliton distribution does not work well in practice, since the expected size of ripple is one. It is extremely sensitive to small variations.

Robust Soliton Distribution

- Maintain the size of Ripple to a larger value
  R ~ $c\sqrt{k}\log(k/\delta)$
- Define the following distribution

$$\tau(i) = \begin{cases} R/ik & \text{for } i=1\ldots k/R-1 \\ R\log(R/\delta)/k & \text{for } i = k/R \\ 0 & \text{for I = k/R+1 ..k} \end{cases}$$

Intuition

- The value of τ(1) is chosen so that R encoding symbols are expected to be released initially. This generates a ripple of size R
- When L encoding symbols are unprocessed, the most probable symbols to be released have degree k/L.

Robust Soliton Distribution (contd.)

- When L = R, we require that all the unprocessed symbols be covered. This is ensured by choosing
  $\tau(k/R) = R \log(R/\delta)/k$.

- The probability any covered symbol gets included in the ripple is (L-R)/L. We need L/(L-R) releases to expect that the size of ripple remains the same. Thus the fraction of encoding symbols of degree i=k/L is proportional to:

$$\frac{L}{L-R}\frac{1}{i(i-1)} = \frac{k}{i(i-1)(k-iR)}$$

$$= \frac{1}{i(i-1)} + \frac{R}{(i-1)(k-iR)} \approx \rho(i) + \tau(i)$$

Robust Soliton Distribution

- The robust soliton distribution is given by μ(i) = (τ(i) + ρ(i))/ς, where ς = $\sum_i$(τ(i) + ρ(i)) .

One can show that:

- Average number of encoding symbols to recover message symbols is $k + O(\sqrt{k}\log^2(k/\delta))$
- Decoding takes time proportional to O(k log(k/δ)) and encoding takes time O(log(k/δ)) per symbol
- The probability that the decoding algorithm fails to recover the message symbols is less than δ .

Conclusions

- Tornado Codes achieve linear time encoding and decoding but cannot solve the heterogeneous user case
- LT Codes can simultaneously serve heterogeneous users, but require O(k logk) time.
- Raptor codes (2000) achieve the best of both worlds and will be discussed next week.