15-853:Algorithms in the Real World

Error Correcting Codes III
- Expander graphs
- Tornado codes

Thanks to Shuchi Chawla for the slides

Why Tornado Codes?

Desgined by Luby, Mitzenmacher, Shokrollahi et al

Linear codes like RS & random linear codes

The other two give nearly optimal rates
But they are slow :

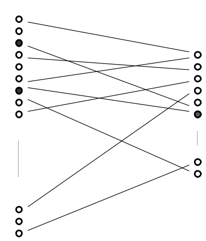
Code	Encoding	Decoding
Random Linear	$O(n^2)$	$O(n^3)$
RS	$O(n \log n)$	$O(n^2)$
Tornado	$O(n \log 1/\varepsilon)$	$O(n \log 1/\varepsilon)$

The idea behind Tornado codes

Easy coding/decoding:
  linear codes with explicit construction

Fast coding/decoding:
  each check bit depends on only a few message bits

Message bits



Check bits

$c_6 = m_3 \oplus m_7$

Think of this as a "regular" Bipartite Graph

Each message bit is used in only a few check bits

=> Low degree bipartite graph

Properties of a good code

There should be "few" check bits

Linear time encoding

- Average degree on the left should be a small constant

Easy error detection/decoding

- Each set of message bits should influence many check bits
- Existence of unshared neighbors

Outline

Expander Graphs

- Applications
- Properties
- Constructions

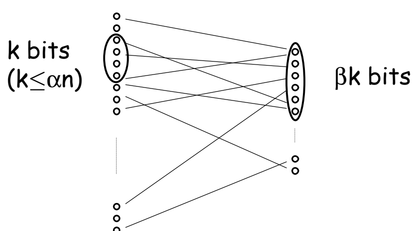
Tornado Codes

- Encoding/Decoding Algorithms
- Brief Analysis

Expander Codes

- Construction
- Brief Analysis

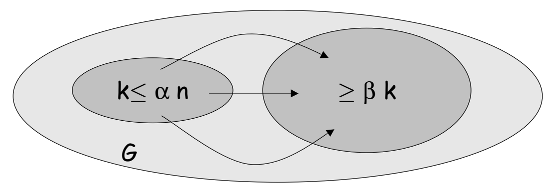
Expander Graphs (bipartite)



Properties

- **Expansion:** every small subset ($k \leq \alpha n$) on left has many ($\geq \beta k$) neighbors on right
- **Low degree** – not strictly part of the definition, but typically assumed

Expander Graphs (non-bipartite)



Properties

- **Expansion:** every small subset ($k \leq \alpha n$) has many ($\geq \beta k$) neighbors
- **Low degree**

Expander Graphs: Applications

Pseudo-randomness: implement randomized algorithms with fewer random bits

Cryptography: strong one-way functions from weak ones.

Hashing: efficient n-wise independent hash functions

Random walks: quickly spreading probability as you walk through a graph

Error Correcting Codes: several constructions

Communication networks: fault tolerance, gossip-based protocols, peer-to-peer networks

d-regular graphs

An undirected graph is **d-regular** if every vertex has d neighbors.

A bipartite graph is **d-regular** if every vertex on the left has d neighbors on the right.

The constructions we will be looking at are all d-regular.

Expander Graphs: Properties

If we start at a node and wander around randomly, in a "short" while, we can reach any part of the graph with "reasonable" probability. (rapid mixing)

Expander graphs do not have small separators.

Expander graphs have certain important properties on the eigenvalues of their adjacency matrix.

Expander Graphs: Eigenvalues

Consider the normalized adjacency matrix A_{ij} for an undirected graph G (all rows sum to 1)

The (x_i,λ_i) satisfying

$A\, x_i = \lambda_i\, x_i$

are the **eigenvectors** and **eigenvalues** of A.

Consider the eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \ldots$

For a d-regular graph, $\lambda_0 = 1$. Why?

The separation of the eigenvalues tell you a lot about the graph (we will revisit this several times).

For expander graphs λ_1 is much smaller than λ_0

Expansion $\beta \approx (1/\lambda_1)^2$

Expander Graphs: Constructions

Important parameters:size (n), degree (d), expansion (β)

Randomized constructions

- A random d-regular graph is an expander with a high probability
- Construct by choosing d random perfect matchings
- Time consuming and cannot be stored compactly

Explicit constructions

- Cayley graphs, Ramanujan graphs etc
- Typical technique – start with a small expander, apply operations to increase its size

Expander Graphs: Constructions

Start with a small expander, and apply operations to make it bigger while preserving expansion

Squaring

- G^2 contains edge (u,w) if G contains edges (u,v) and (v,w) for some node v
- $A' = A^2 - 1/d\ I$
- $\lambda' = \lambda^2 - 1/d$
- $d' \leq d^2 - d$

Size	$\equiv$
Degree	$\uparrow$
Expansion	$\uparrow$

Expander Graphs: Constructions

Start with a small expander, and apply operations to make it bigger while preserving expansion

Tensor Product

- $G = A \times B$ nodes are (a,b) $\forall\, a \in A$ and $b \in B$
- edge between (a,b) and (a',b') if A contains (a,a') and B contains (b,b')
- $n' = n_1 n_2$
- $\lambda' = \max(\lambda_1, \lambda_2)$
- $d' = d_1 d_2$

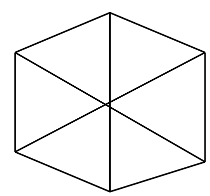
Size	$\uparrow$
Degree	$\uparrow$
Expansion	$\downarrow$

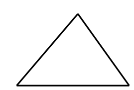
Expander Graphs: Constructions

Start with a small expander, and apply operations to make it bigger while preserving expansion

Zig-Zag product

- "Multiply" a big graph with a small graph



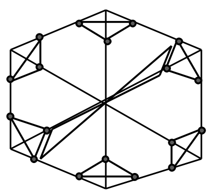


$n_2 = d_1$
$d_2 = \sqrt{d_1}$

Expander Graphs: Constructions

Start with a small expander, and apply operations to make it bigger while preserving expansion

Zig-Zag product
- "Multiply" a big graph with a small graph



Size	↑
Degree	↓
Expansion	↑

Outline

Expander Graphs
- Applications
- Properties
- Constructions

Tornado Codes
- Encoding/Decoding Algorithms
- Brief Analysis

Expander Codes
- Construction
- Brief Analysis

The loss model

Random Erasure Model:
- Each bit is lost independently with some probability μ
- We know the positions of the lost bits

For a **rate** of (1-p) can correct (1-ε)p fraction of the errors.

Seems to imply a

$(n, (1-p)n, (1-\varepsilon)pn+1)_2$

code, but not quite because of random errors assumption (worst case distance might be less).

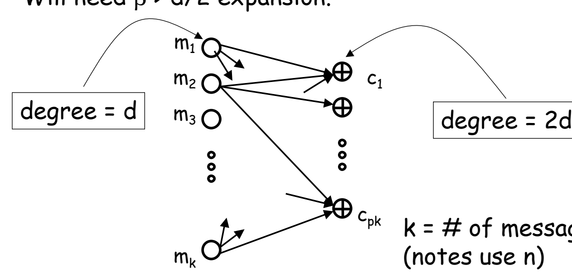
We will assume p = .5.

Error Correction can be done with some more effort

Tornado codes

Will use d-regular bipartite graphs with k nodes on the left and pk on the right (notes assume p = .5)

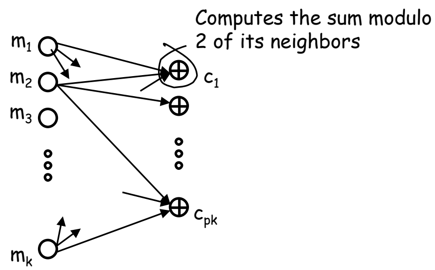
Will need $\beta > d/2$ expansion.



k = # of message bits
(notes use n)

Tornado codes: Encoding

Why is it linear time?

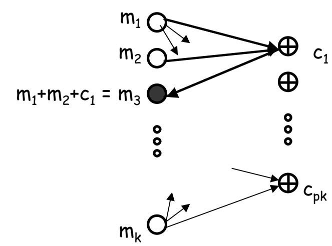


Tornado codes: Decoding

Assume that all the check bits are intact

Find a check bit such that only one of its neighbors is erased (an *unshared neighbor*)

Fix the erased code, and repeat.

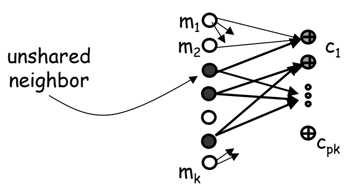


Tornado codes: Decoding

Need to ensure that we can always find such a check bit

"Unshared neighbors" property

- Every small subset ($l \leq \alpha k$) on left has at least ($\geq \delta l$) unshared neighbors on right.
- If $\delta > 0$ then for sufficiently small number of errors ($l < \alpha k$) at least one has an unshared neighbor



Tornado codes: Decoding

Can we always find unshared neighbors?

Expander graphs give us this property if $\beta > d/2$

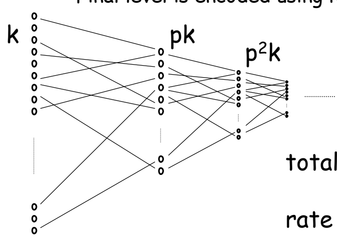
In particular $\delta \geq (2\beta/d) - 1$ (see notes)

Also, [Luby et al] show that if we construct the graph from a specific kind of degree sequence, then we can always find unshared neighbors.

What if check bits are lost?

Cascading

- Use another bipartite graph to construct another level of check bits for the check bits
- Final level is encoded using RS or some other code



$p^l k \approx \sqrt{n}$

total bits $n \approx k(1 + p + p^2 + \ldots)$
$= k/(1-p)$
rate $= k/n = (1-p)$

Cascading

Encoding time

- for the first k stages : $|E| = d \times |V| = O(k)$
- for the last stage: $\sqrt{k} \times \sqrt{k} = O(k)$

Decoding time

- start from the last stage and move left
- again proportional to $|E|$
- also proportional to d, which must be at least $1/\varepsilon$ to make the decoding work

Can fix $kp(1-\varepsilon)$ random erasures

Outline

Expander Graphs
- Applications
- Properties
- Constructions

Tornado Codes
- Encoding/Decoding Algorithms
- Brief Analysis

Expander Codes
- Construction
- Brief Analysis

Expander Codes

Input:

Regular expander G on n nodes, degree d
Code C of block length d, rate r, rel. distance δ

Output:

Code $\mathbb{C}(G,C)$ of block length dn/2, rate 2r-1,
rel. distance $\approx \delta^2$

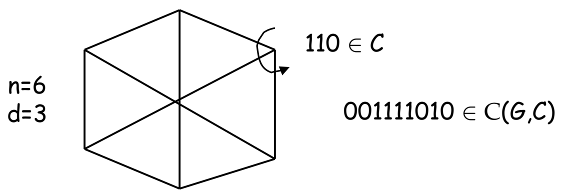
Linear time encoding/decoding

Expander Codes: Construction

We associate each edge in G with a bit of the code

For every vertex, the edges around it form a code word in C

Block length = number of edges = nd/2



Expander Codes: Construction

Linear code C has rate r

=> there are (1-r)d linear constraints on its bits

(these constraints define a linear subspace of dimension rd)

Total number of constraints in the entire graph G = (1-r) nd

Total length of code = nd/2

=> Total number of message bits = nd (r-1/2)

Therefore, rate is 2 (r-1/2) = 2r-1

Expander Codes: Construction

For linear codes, the minimum distance between two code words = minimum weight of a code word

Intuition:

If the weight of a code word is small, then the weight of edges around some vertex is small

=> distance of C is small => contradiction

Expander Graphs: Construction

Expander graphs:

Any set of αn nodes must have at most

$m = (\alpha^2 + (\alpha-\alpha^2)\,\lambda/d)\, dn/2$ edges

So, a group of m edges must touch at least αn vertices

One of these vertices touches at most m/2αn edges

But these should be at least δd for the code to be valid

So, $(\alpha + (1-\alpha)\,\lambda/d)\, d > \delta d$

=> $\alpha > (\delta - \lambda/d)/(1-\lambda/d)$

Minimum distance is atleast $\alpha\,(\alpha + (1-\alpha)\,\lambda/d) \approx \delta^2$

Some extra slides

Expander Graphs: Properties

Prob. Dist. – π ; Uniform dist. – u

Small $|\pi\text{-u}|$ indicates a large amount of "randomness"

Show that $|A\pi\text{-u}| \leq \lambda_2|\pi\text{-u}|$
Therefore small λ_2 => fast convergence to uniform

Expansion $\beta \approx (1/\lambda_2)^2$

Expander Graphs: Properties

To show that $|A\pi\text{-u}| \leq \lambda_2|\pi\text{-u}|$
Let $\pi = u + \pi'$

u is the principle eigenvector $Au = u$
π' is perpendicular to u $A\pi' \leq \lambda_2\pi'$

So, $A\pi \leq u + \lambda_2\pi'$

Thus, $|A\pi - u| \leq \lambda_2|\pi'|$