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?

Designed by Luby, Mitzenmacher, Shokrollahi et al
Linear codes like RS & random linear codes
The other two give nearly optimal rates
But they are slow:

<table>
<thead>
<tr>
<th>Code</th>
<th>Encoding</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Linear</td>
<td>O(n^2)</td>
<td>O(n^2)</td>
</tr>
<tr>
<td>RS</td>
<td>O(n log n)</td>
<td>O(n^2)</td>
</tr>
<tr>
<td>Tornado</td>
<td>O(n log 1/\varepsilon)</td>
<td>O(n log 1/\varepsilon)</td>
</tr>
</tbody>
</table>

Assuming an (n, (1-p)n, (1-\varepsilon)pn+1)_2 tornado code

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

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)

Expander Graphs (non-bipartite)
**Expander Graphs: Applications**

- **Pseudo-randomness**: implement randomized algorithms with few 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 left.

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.

The eigenvalues of the adjacency matrix of a graph carry information about the expansion of the graph.

**Expander Graphs: Eigenvalues**

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

\[ A x_i = \lambda_i x_i \]

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).

If \( \lambda_1 \) is much smaller than \( \lambda_0 \) then the graph is an expander.

Expansion \( \beta \geq (1/\lambda_1)^2 \)
Expander Graphs: Constructions

Important parameters: size (n), degree (d), expansion (\(\beta\))

**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
- \(A' = A^2 - \frac{1}{d} I\)
- \(\lambda' = \lambda^2 - \frac{1}{d}\)
- \(d' = d^2 - d\)

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 \((b,b')\)
- \(n' = n_1n_2\)
- \(\lambda' = \max (\lambda_1, \lambda_2)\)
- \(d' = d_1d_2\)

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

\[
\begin{align*}
\text{Size} & \uparrow \\
\text{Degree} & \uparrow \\
\text{Expansion} & \downarrow
\end{align*}
\]

\[
\begin{align*}
n_2 &= d_1 \\
d_2 &= \sqrt{d_1}
\end{align*}
\]
**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 ↑
```

**The loss model**

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

For a rate of $1-p$ can correct $(1-\varepsilon)p$ 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.

We will assume $p = .5$

Error Correction can be done with some more effort.

**Tornado codes**

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

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

```
\begin{align*}
\text{degree} &= d \\
\text{degree} &= 2d \\
\text{\# of message bits} &= \text{n}
\end{align*}
```
**Tornado codes: Encoding**

Why is it linear time?

Computes the sum modulo 2 of its neighbors:

\[ m_1, m_2, m_3, \ldots, m_k, c_1, \ldots, c_{pk} \]

---

**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.

\[ m_1 + m_2 + c_1 = m_3 \]

---

**Tornado codes: Decoding**

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

"Unshared neighbors" property:

Consider the set of corrupted message bit and their neighbors. Suppose this set is small.

\[ \Rightarrow \text{at least one message bit has an unshared neighbor.} \]

---

**Tornado codes: Decoding**

Can we always find unshared neighbors?

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

\[
\text{total bits } n = \frac{k}{1-p} \cdot \frac{1 + p + p^2 + \ldots}{k/(1-p)} \\
\text{rate } = \frac{k}{n} = \left(1 - p\right)
\]

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 \(1/\epsilon\) to make the decoding work
Can fix \( kp(1-\epsilon) \) 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 \( \delta \)

Output:
Code \( C(G,C) \) of block length \( dn/2 \), rate \( 2r-1 \), rel. distance \( \frac{\delta^2}{2} \)
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 $= \text{number of edges} = \frac{nd}{2}$

Linear code $C$ has rate $r$
$\Rightarrow$ 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 $= \frac{nd}{2}$
$\Rightarrow$ 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 $= \text{minimum weight of a code word}$

Intuition:
If the weight of a code word is small, then the weight of edges around some vertex is small
$\Rightarrow$ distance of $C$ is small $\Rightarrow$ contradiction

Expander Graphs: Construction

Expander graphs:
Any set of $\alpha n$ nodes must have at most
$m = (\alpha^2 + (\alpha - \alpha^2) \lambda / d) \frac{dn}{2}$ edges
So, a group of $m$ edges must touch at least $\alpha n$ vertices.
One of these vertices touches at most $m/2\alpha n$ edges.
But these should be at least $\delta d$ for the code to be valid.

$\Rightarrow (\alpha + (1-\alpha) \lambda / d) d > \delta d$
$\Rightarrow \alpha > (\delta - \lambda / d)/(1-\lambda / d)$
Minimum distance is at least $\alpha (\alpha + (1-\alpha) \lambda / d) \geq \delta^2$
**Expander Graphs: Properties**

Prob. Dist. \(- \pi\); Uniform dist. \(- u\)

Small \(|\pi-u|\) indicates a large amount of “randomness”

Show that \(|A\pi-u| \cdot \lambda_2|\pi-u|\)

Therefore small \(\lambda_2 \Rightarrow \) fast convergence to uniform

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

---

**Expander Graphs: Properties**

To show that \(|A\pi-u| \cdot \lambda_2|\pi-u|\)

Let \(\pi = u + \pi'\)

\(u\) is the principle eigenvector \(A\pi = u\)

\(\pi'\) is perpendicular to \(u\) \(A\pi' \cdot \lambda_2\pi'\)

So, \(A\pi \cdot u + \lambda_2\pi'\)

Thus, \(|A\pi - u| \cdot \lambda_2|\pi'|\)