15-853: Algorithms in the Real World

- Expander Graphs
- LDPC (Expander) codes
Examples of Codes

**Hamming** codes are binary $(2^r - 1 - 1, 2^r - 1 - r, 3)$ codes.
  Basically $(n, n - \log n, 3)$
**Hadamard** codes are binary $(2^r - 1, r, 2^r - 1)$ codes.
  Basically $(n, \log n, n/2)$

**Reed Solomon** codes are $(n, k, n-k+1)_n$
  Optimal but large alphabet
**Concatenated** codes can get best of both worlds

Next:
Another set of codes, **optimized for fast (de)coding**.
Based on **graphical constructions**.
(\(\alpha, \beta\)) Expander Graphs (non-bipartite)

Properties

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

- **Low degree** – not technically part of the definition, but typically assumed
$(\alpha, \beta)$ 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 technically part of the definition, but typically assumed
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-left-regular** if every vertex on the left has $d$ neighbors on the right.

We consider only d-left-regular constructions. (And call it d-regular with abuse of notation.)
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
- 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

**Theorem:** For every constant $0 < c < 1$, can construct bipartite graphs with

- $n$ nodes on left,
- $cn$ on right,
- $d$-regular (left),

that are $(\alpha, 3d/4)$ expanders, for constants $\alpha$ and $d$ that are functions of $c$ alone.

“Any set containing at most $\alpha$ fraction of the left has $(3d/4)$ times as many neighbors on the right”
From expanders to codes

LDPC CODES
Low Density Parity Check (LDPC) Codes

- Code bits
- Parity check bits
- n
- H
- n-k
Low Density Parity Check (LDPC) Codes

Each row is a vertex on the right and each column is a vertex on the left.

A codeword on the left is valid if each right “parity check” vertex has parity 0.

The graph has \( O(n) \) edges \textbf{(low density)}
Applications in the real world

• 5G cellular technology (3GPP 5G NR, first release Dec. 2017)
• Solid State Drives (NAND Flash memory)
• NASA
  – Proposed for all their space data systems
• 10Gbase-T (IEEE 802.3an, 2006)
  – Standard for 10 Gbits/sec over copper wire
• WiMax (IEEE 802.16e, 2006)
History

Invented by Gallager in 1963 (his PhD thesis)

Generalized by Tanner in 1981 (instead of using parity and binary codes, use other codes for “check” nodes).

Mostly forgotten by community at large until the mid 90s when revisited by Spielman, MacKay and others.
Consider a d-regular LPDC with $(\alpha, 3d/4)$ expansion.

**Theorem**: Distance of code is greater than $\alpha n$.

**Proof.** (by contradiction)

Linear code; distance = min weight of non-0 codeword.

Assume a codeword with weight $w \leq \alpha n$.

Let $W$ be the set of 1 bits in codeword

- $\#edges = wd$
- $\#neighbors$ on right $> 3/4 * wd$
- Max $\#neighbors$ with $>1$ edge from $W$?
  - $\#unique neighbors = wd/2$

So at least one neighbor sees a single 1-bit. Parity check would fail!
Correcting Errors in LPDC codes

We say a vertex is *unsatisfied* if parity ≠ 0

**Algorithm:**
While there are unsatisfied check bits
1. Find a bit on the left for which more than $d/2$ neighbors are unsatisfied
2. Flip that bit

Q: Converges?
Since every step reduces unsatisfied parity by at least 1.
We say a vertex is **unsatisfied** if parity $\neq 0$

**Algorithm:**

While there are unsatisfied check bits

1. Find a bit on the left for which more than $d/2$ neighbors are unsatisfied
2. Flip that bit

Q: Running time?

Runs in linear time (for constant maximum degree on the right).

Why must there be a node with more than $d/2$ unsatisfied neighbors (if we’re not at a codeword)?
Correcting Errors in LPDC codes

**Theorem:** Always exists a node > d/2 unsatisfied neighbors if we’re not at a codeword.

**Proof:** (by contradiction)
Suppose not. (Let d be odd.) Let S be the corrupted bits.
Each such bit has majority of satisfied neighbors
  (sat. neighbors see at least two corrupted bits on left)
  (unsat. neighbors may see only one corrupted bit on left)
Each corrupt bit gives $1 to each unsat nbr, $½ to sat nbr.
  Total money given < 3d/4 |S|.
Each node in N(S) collects $1 at least.
  Total money collected at least |N(S)|.
So |N(S)| < 3d/4 |S|. Contradicts expansion.
Theorem: Assume $(\alpha, 3d/4)$ expansion. If \# of error bits is less than $\alpha n/4$ then simple decoding algorithm converges to closest codeword.

Proof: let:

- $u_i = \#$ of unsatisfied check bits on step $i$
- $r_i = \#$ corrupt code bits on step $i$
- $s_i = \#$ satisfied check bits with corrupt neighbors on step $i$

Q: What do we have to show about $r_i$?
We know that $u_i$ decrements on each step, but what about $r_i$?

Will be continued in the next lecture