Today

- Linear time encodable and decodable codes.
- Shannon capacity with linear time algorithms.

Recall basic codes from last lecture

- Picture of graph.
- Codewords are assignment to left vertices s.t. right vertices have even parity.
- Linear time decodable!
- Linear time encodable? No!
- Can we use similar principle to design linear time encodable codes? No! (Generator matrix has to be dense).

Spielman Codes

- Basic idea: Use sparse generator and fix what needs to be fixed, recursively.
- Given bipartite graph $G$ with $k$ left nodes and $k/2$ right nodes, think of this as generating a $[3k/2, k, ?]$ code in an obvious way.
- Message = assignment to left.
- Right vertices = get parity of neighbors = check bits.
- Codeword = assignment to all vertices.

Low-Density Generator Codes?

- Let left degree = $c$.
- Clearly code has distance $\leq c + 1$.
- So - not an error-correcting code!
- Spielman: Salvages an error-reduction property.
- Insight: To protect message, need to protect check bits very carefully, but don’t need to protect message bits all that carefully.
- Gives some glimmer of hope. Very careful recursion extracts this.
• Rest of lecture: Formalize insight. Describe recursion.

**Insight: Error-reducing codes**

Lemma: If \( G \) is an expander, and \((x, y)\) is \((a, b)\)-close to \((m, c)\), then FLIP algorithm leads to \((x', y) = (c'', b, b)\)-close to \((m, c)\) provided \(a \leq \ldots\) and \(b \leq \ldots\).

(Will fill in \ldots after proof!)

**Notation:**

- \((m, c)\) - Message \(m\) with check \(c\).
- \((x, y) = (a, b)\)-close to \(u, v\) if \(\Delta(x, u) \leq a\) and \(\Delta(y, v) \leq b\).
- FLIP algorithm = similar to yesterday
  
  If \(\exists u \in L\) with more unsat. ngbrs than sat, flip \(u\).

---

**Analysis of FLIP**

- Set \(\gamma = 7/8c\) and \(c \geq 8\) to get \(s > b/2\) implies \(\exists\) unhappy message bit.

- Clearly runs in linear time.

- Termination conditions:
  - Can be some other codeword (distance not large).
  - Can be non-codeword (if check bits awry).
  - But can’t be far from correct one, if check bits not too far.

- Initial \# unsat. constraints \(\leq c \cdot a + b\).

- \(\Delta(x', m) \leq a + c \cdot a + b\) (at all times).

- If \(\Delta(x', m) = s\) then \# unsat. constraints \(\geq (2\gamma - c)s - b\).
Recursion: 1st Idea

- Construct $C_k$ (for $k$ message bits) as follows.

- Set up error-reducer code $R_k$ ($k$ message bits and $k/2$ check bits.

- Protect check bits with $C_{k/2}$.

- Works? No! (May need to correct $\epsilon k$ errors in check bits, but it corrects only $\epsilon k/2$ errors.

- So need to reduce total number of errors everywhere. How? Use another error-reducing code!

---

Actual recursion

- $C_k$: Encode message using $R_k$ first. Then encode check bits of first step using $C_{k/2}$.

  Finally encode all check bits so far using $R_{2k}$. Get total of $3k$ check bits.

- Encoding: takes linear time (verify!).

- Decoding: takes linear time (verify!).

---

Using to get to Shannon capacity

- Observation 1: Can get lin. time encodable and decodable codes correcting $\epsilon$ fraction errors with rate $1 - f(\epsilon)$ where $f(\epsilon) \to 0$ as $\epsilon \to 0$.

- Observation 2: If encode message first using Spielman code of rate $1 - f(\epsilon)$ and then chop into blocks of constant size and encode each block using constant size, near capacity codes, then rate is near optimal, and error-correction is near optimal and all takes linear time.