Expanders II - Applications

Expander codes

'Good code': Positive constant rate
Positive constant min. distance
~ Efficient to decode/encode

Theorem (Margulis)

For $d \geq 64$, there exists a left d-regular bipartite graph with $|L| = n$ and $|R| = \frac{3}{4}n$, s.t. $|N(S)| \geq 0.8\,d\,|S|$ for all $S \subseteq L: |S| \leq \frac{0.02}{d} n$

- Explicit construction!

TANNER CODE (a type of linear code)

Take $d = 64$ and the Margulis expander.

Our code is defined using the parity check matrix H, where elements of the code (codewords) are all length $|L|$ strings z s.t. $Hz = 0$. (in binary)

Message length $= |L| - |R| = n - \frac{3}{4}n = \frac{1}{4}n$

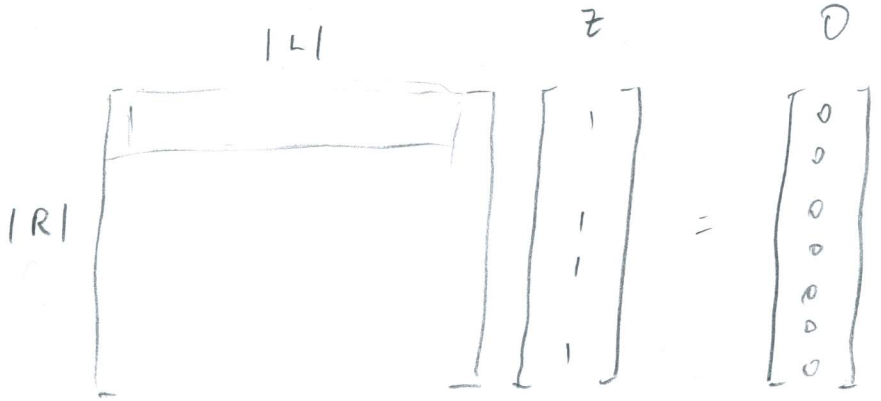
$[n, \frac{1}{4}n, ???]$ code : rate $= \frac{1}{4}$, constant

Claim Distance of the code is $> \frac{0.02}{64} n$

Assume minimum distance $\leq \frac{0.02}{64} n$

$\exists$ nonzero codeword z with Hamming weight $|z| \leq \frac{0.02}{64} n$. Let $S = \{ u \in [n] \mid z_u = 1 \}$ ← vertices

Since z is nonzero, $S \neq \emptyset$, and hence $|S| \leq \frac{0.02}{64} n$.



Each vertex in R must be adjacent to an even number of vertices in S.

Claim ~~$0 < |S| \leq \frac{0.02}{64} n$~~ There exists a $v \in N(S)$ with exactly one neighbor in S.

If all $v \in N(S)$ have ≥ 2 neighbors in S, then since $|N(S)| \geq 0.8 d |S|$:

$$|E(S, N(S))| \geq 2|N(S)| \geq 2 \cdot 0.8 d |S| > 64 |S|$$

But the left partition is 64-regular!

Hence we are done. – z is not a codeword, so distance $> \frac{0.02}{64} n$.

Decoding is efficient! Flip bits of z that decrease the Hamming weight of Hz.

Corrects $< \frac{D}{2}$ errors in polylog n time

ERROR REDUCTION (Miller-Rubin primality)

Algorithm A, probabilistic
Uses n random bits, returns
fails with prob $p \leq 1\%$

YES if YES
NO 99%, YES 1% if NO

Old way of reducing error: repeat d times
• nd random bits, fails with prob $\leq .01^d$

Expanders:
Take the Margulis expander, except with $|L| = |R| = 2^n$.
Vertices on each sides are indexed by n-bit strings.

$|N(S)| \geq 0.8d|S|$ if $|S| \leq \frac{0.02}{d}(2^n)$

Pick random vertex $v \in L$
Compute d neighbors of v and use them all for A!

Uses NO additional random bits, but what's the error...