Lecture 7: Expander Graphs

Topics in Error-Correcting Codes (Fall 2022)
University of Toronto
Swastik Kopparty
Scribe: Mehul Gupta and Sky Li

1 Expander Graphs

We start a new topic discussing expander graphs. We will be able to prove that expander graphs exist, however they are difficult to construct. These objects will provide us with a new way of creating codes.

Let $G = (V, E)$ be a d-regular graph and let $n = |V|$. We think of $d = \Theta(1)$ being constant and n going to ∞.

Definition 1. *Such a G is called an (α, β) expander if for every $S \subset V$ such that $|S| \leq \alpha n$, we have $|\Gamma(S)| \geq \beta|S|$.*

Here $|\Gamma(S)| = \{v \in V : \exists w \in S$ which is adjacent to $v\}$ is the set of neighbors of S. This definition can be paraphrased as "all small sets have big neighborhoods". The interesting cases are when β is at least 1. Note that any (α, β) expander will always have $\beta \leq d$. We also have a special definition for bipartite expander graphs.

Definition 2. *Let $G = (L \cup R, E)$ be a bipartite graph where the partition L is d-regular and R is d'-regular. G is an (α, β)-left expander if for every $S \subset L$ such that $|S| \leq \alpha n$, we have $|\Gamma(S)| \geq \beta|S|$.*

Again, any (α, β)-left expander will always have $\beta \leq d$.

Theorem 3 (Existence of Expander Graphs)**.** *For any constant d, as n goes to ∞ there exists a d-regular graph G with n vertices which is a $(\frac{1}{10d}, d - 10)$ expander.*

Theorem 4 (Existence of Bipartite Expander Graphs)**.** *For any constants d, d', as n goes to ∞ there exists a bipartite graph G where the left partition has n vertices and is d-regular, the right partition has $\frac{dn}{d'}$ vertices and is d'-regular, and the graph is a $(\frac{1}{10d'}, d - 10)$-left expander.*

We will provide a proof of the bipartite case later. First we discuss how this allows us to construct an error correcting code.

2 Constructing ECC's from Expander Graphs

Consider a bipartite graph $G = (L \cup R, E)$ where the partition L is d-regular and R is d'-regular. Fix $d = 100, d' = 1000$. Let $n = |L|$. Then because of regularity, $|R| = \frac{n}{10}$. We construct a linear code $C \subset \mathbb{F}_2^L$ using this graph. A vector $x \in \mathbb{F}_2^L$ is in the code C if and only if for every vertex $w \in R$,

$$\sum_{v \sim w} x_v = 0.$$

That is, the sum of x_v taken over vertices v which are adjacent to w must be 0 for every vertex $w \in R$. So, the codewords in C are the solutions to a system of homogeneous equations. Notice that the parity check matrix of the code is the bipartite adjacency matrix of the graph.

Claim 5. *Rate*$(C) \geq \frac{9}{10}$.

This is because C is the set of solutions to $n/10$ homogeneous equations.

Claim 6. *If $\beta \geq \frac{d}{2} + 1$ and G is an (α, β)-left expander then C has distance at least αn.*

Proof. Since the code is linear, to show the distance is high, it is enough to show that the support of any nonzero codeword is high. Let $x \in C$ be a nonzero codeword. Let $S \subset L$ be the set of coordinates where x is nonzero. So we want to show that $|S| \geq \alpha n$.

Suppose, for a contradiction, that $|S| < \alpha n$. Then by expansion, $|\Gamma(S)| \geq \beta|S| \geq (\frac{d}{2} + 1)|S|$.

Suppose, for a contradiction, that every vertex in $\Gamma(S)$ has at least two neighbors in S. Then the number of edges from $\Gamma(S)$ to S is at least $2|\Gamma(S)| \geq 2(\frac{d}{2} + 1)|S| > d|S|$. But $S \subset L$ is d-regular, so this is a contradiction. Hence some vertex w in $\Gamma(S)$ has exactly one neighbor v' in S.

But then, using the definition of S,

$$\sum_{v \sim w} x_v = x_{v'} \neq 0$$

which contradicts the fact that x is a codeword. Therefore $|S| \geq \alpha n$. □

Using Theorem 4, this gives us a code with distance $n/10000$ and Rate $\geq 9/10$.

3 Existence of Expander Graphs

We will now prove Theorem 4. This is done using the probabilistic method.

Proof. We want to randomly construct a d, d'-regular bipartite graph. We will construct it as follows:

Consider two disjoint sets L, R such that $|L| = |R| = dn = d'n' = t$. We can wlog order L and R by labeling them with elements of $[t]$. Then let $M \subset [t] \times [t]$ be a uniformly random matching of L and R. To construct our final graph G, we want to "fuse" together groups of d vertices in L and groups of d' vertices in R.

That is, we fuse the first d vertices in L into one vertex, and any edges that came out of these vertices now come out of the new single vertex. Each of these edges still has the same vertex in R on its other end. This will potentially result in multiple copies of an edge, so we are forming a multigraph. This process is then repeated for the next d vertices in L (i.e. vertices $d+1$ to $d+d$), and then the next d, and so on, until all t vertices in L have been fused into $t/d = n$ vertices. This same process is then done for R, fusing d' vertices at a time. This gives us a bipartite multigraph G' where the left partition L' is d-regular with n vertices and the right partition R' is d'-regular with n' vertices.

We will show probabilistically that a random expander graph can satisfy the expansion property.

Consider $S \subseteq L', |S| \leq \alpha n$ and $T \subseteq R', |T| < \beta|S|$. Then the event $E_{S,T} =$"all neighbours of S are in T" serves as a counterexample to G being an expander graph. Thus, the probability that G' is not a valid expander graph can be computed by summing over $\mathbb{P}(E_{S,T})$ for all S, T.

Observe that, for a vertex v in the original graph, observe that $\mathbb{P}(v$ matched with something in $T) = \frac{|T|d'}{t}$; more generally, in the pre-collapsed graph, the neighbourhood of the $|S|d$ vertices that were fused into S is a uniformly random subset of $[t]$ of size $d|S|$.

Thus,

$$\begin{aligned}\mathbb{P}(E_{S,T}) &= \mathbb{P}(\text{a uniformly random subset of } [t] \text{ is contained in the } |T|d' \text{ vertices underlying } T) \\ &= \frac{\binom{|T|d'}{|S|d}}{\binom{t}{|S|d}} \leq \frac{\binom{\beta|S|d'}{|S|d}}{\binom{t}{|S|d}} \leq \left(\frac{\beta|S|d'}{t-|S|d}\right)^{|S|d}\end{aligned}$$

Summing over all S, T, we have that

$$\begin{aligned}\mathbb{P}(\text{failure}) = \sum_{S,T \text{ of interest}} \mathbb{P}(E_{S,T}) &\leq \sum_{\substack{S\subset[n] \\ |S|\leq\alpha n}} \binom{n'}{\leq \beta|S|} \frac{\binom{\beta|S|d'}{|S|d}}{\binom{t}{|S|d}} \\ &\leq \sum_{k=1}^{\alpha n} \binom{n}{k}\binom{n'}{\leq \beta k}\left(\frac{\beta|S|d'}{t-|S|d}\right)^{|S|d}\end{aligned}$$

Clearly[1], this goes to 0 as $n \to \infty$.

□

4 Decoding Expander Codes

Note that randomly generating an expander code (by generating a random matching matrix in the previous construction) with high probability will be a correct code; this is notably different from linear codes in general, where there is no efficient (polynomial time) method for generating them.

Even for expander codes generated randomly, we can decode (some of) them efficiently - specifically, we can decode an $(\alpha, \beta)-$encoder code if β is large, i.e. let $\beta = (1-\epsilon)d$.

Formally, given $r \in \{0,1\}^n$ such that $\exists x \in C$ where $\Delta(r,x) \leq \frac{\alpha}{10}n$, we want to find x. Note that an ideal decoder would be able to decode any x values within an $\frac{\alpha}{2}$ distance of a codeword r, but this is the slightly restriction version of the problem that the algorithm we prove will solve.

[1] I actually have no idea how to do this out

4.1 Belief Propagation Algorithm

The decoding algorithm we will use is a specific instance of the more generalized belief propagation method. Recall that in the bipartite graph construction of an encoder code, each vertex in the (wlog) right partition R of the graph corresponds to one constraint of the encoder code, and the left partition L of the graph corresponds to the input string r that we want to decode. We can also see that flipping one vertex $v \in L$ will change the parity of every constraint associated with v and therefore flip whether or not it is satisfied.

If, for all $v \in L$, all constraints that use v are satisfied, then $r \in C$ and we are done. At a high level, we want to look for $v \in L$ such that a majority of its constraints are unsatisfied and flip v. Ideally, every bit flip will decrease the total number of unsatisfied constraints until we reach a codeword x.

Formally, the algorithm can be described as follows:

```
Let $z = r$;
while $\exists i \in [n]$ such that a majority of constraints involving $i$ are unsatisfied do
    Flip $z_i$;
end
```

There are two conditions required for correctness.

First, whenever z is within αn of the "correct" codeword x, there will always be some i such that a majority of constraints involving i are unsatisfied. This guarantees that we can always make progress if z is within αn distance of x.

Claim 7. *$\forall z$ such that $\Delta(z, x) \leq \alpha n, \exists i \in [n]$ such that a majority of constraints involving i are unsatisfied.*

Proof. Let $S = \{i : z_i \neq x_i\}$ be the set of disagreements between z and x. Note that any constraint on the right partition with an odd number of neighbours in S is unsatisfied. In addition $|S| \geq 1$, since otherwise $z = x$ and the algorithm would have terminated.

We will show that $\forall S, |S| \leq \alpha n$, there exists at least $(1 - 2\epsilon)d|S|$ unique neighbours of S.

Let $\Gamma_u(S)$ represent the unique neighbours of S (i.e. neighbours of S that have exactly one neighbour in S).

There are exactly $d|S|$ edges from S to $\Gamma(S)$. By expansion, $|\Gamma(S)| \geq (1-\epsilon)d|S|$. Thus, we see that

$$\begin{aligned}
d|S| &= \Gamma_u(S) + \text{ edges from } \Gamma(S) \setminus \Gamma_u(S) \text{ to } S \\
&\geq |\Gamma_u(S)| + 2(|\Gamma(S) - |\Gamma_u(S)|) \\
\implies \Gamma_u(S) &\geq 2|\Gamma(S)| - d|S| \\
&= d(1-\epsilon)|S|
\end{aligned}$$

If $\epsilon \leq \frac{1}{4}$, then $|\Gamma_u(S)| > \frac{1}{2}d|S|$, so some vertex $v \in S$ has greater than $\frac{1}{2}d$ neighbours in $\Gamma_u(S)$ □

Note, however, that $v \in S$ may not be the vertex chosen by the algorithm to flip; thus, each individual step of the algorithm may possibly move us further away from x. Thus, we need the following claim as well:

Claim: At all steps during the execution of the algorithm, $\Delta(z, x) \leq \alpha n$.

This is sufficient to prove correctness, since we always reduce the number of unsatisfied constraints and we never get too far away from the target codeword.

Proof. Note that initially, there are at most $\frac{\alpha n}{10} d$ unsatisfied constraints. If, at any point, $\Delta(z, x) > \alpha n$, there will be at least $d(1-2\epsilon)\alpha n$ unsatisfied constraints. However, this leads to a contradiction, since $d(1 - 2\epsilon)\alpha n > \frac{\alpha n}{10} d$ (we assume ϵ is small, e.g. $\epsilon \leq \frac{1}{4}$) and every step of the algorithm is guaranteed to reduce the number of unsatisfied constraints. $\square$