CSE599s: Extremal Combinatorics November 29, 2011

Lecture 18 Expander Graphs and Error Correcting Codes

Lecturer: Anup Rao

1 Expanders

Loosely speaking, an *expander graph* is an undirected graph where every vertex has degree d, and yet every (small enough) set S has $\Omega(d|S|)$ neighbors. Recall that the neighbors are usually denoted $\Gamma(S)$. Although it took a significant amount of effort, today we know of several constructions of expander graphs with very good parameters.

It makes sense to talk about bipartite expanders, where we require expansion of sets from the left. Let us start by using the probabilistic method to show the existence of expanders with good parameters.

A bipartite graph on vertex set A, B, $|A| = n, |B| = m$ with left degree d will be called an expander if for every subset $S \subseteq A$, $|S| \leq \frac{n}{100d}$, $|\Gamma(S)| > \frac{3d}{4} \cdot |S|$.

Lemma 1. *Let $m \geq 3n/4$. Then if $d \geq 100$, there exists n_0 such that for every $n \geq n_0$, a random bipartite graph with n vertices on the left, m vertices on the right and left-degree d will satisfy that every subset S of the left vertices of size $\frac{n}{100d}$ will have at least $3d|S|/4$ neighbors with positive probability.*

Proof Take a random graph. For any fixed set S of size s, and set T of size $t = 3ds/4$, the probability that $\Gamma(S) \subseteq T$ is at most $(t/m)^{ds}$. So by union bound, the probability that any such set is not expanding is at most

$$\begin{aligned}
&\sum_{s=1}^{n/100d} \binom{n}{s}\binom{m}{t}(t/m)^{ds} \\
&\leq \sum_{s=1}^{n/100d} (ne/s)^s (me/t)^t (t/m)^{ds} \\
&\leq \sum_{s=1}^{n/100d} (n/s)^s e^{3ds/4+s} (t/m)^{ds/4} \\
&\leq \sum_{s=1}^{n/100d} e^{3ds/4+s} (ds/n)^{ds/4-s} \\
&\leq \sum_{s=1}^{n/100d} e^{3ds/4+s} (1/100)^{ds/4-s} < \sum_{s=1}^{n/100d} (1/4)^s \leq 3/4
\end{aligned}$$

■

In fact, there are polynomial time algorithms that give strong explicit constructions of such graphs — namely there are algorithms that take as input a number from [n] and a number from [d] and output an element of [m] in such a way that the induced graph is an expander.

2 Error Correcting Codes

Recall that an error correcting code is map $E : \{0,1\}^k \to \{0,1\}^n$ with the property that for any $x \neq y$, $E(x)$ and $E(y)$ differ in at least $\delta = \Omega(n)$ coordinates (δ is called the *distance* of the code). Ideally, we would like k to be $\Omega(n)$. Such a code is called a good code. The code is called *linear* if $E : \mathbb{F}_2^k \to \mathbb{F}_2^n$ is a linear function.

In a linear code, the distance between $E(x), E(y)$ is simply the hamming weight (number of non-zero coordinates) of $E(x) + E(y) = E(x + y)$. Thus, the distance of the code is the same as the minimum hamming weight of $E(x)$ over all choices of x.

Next we show how to use expanders to obtain a good code. We describe the image of the code (i.e. the set of *codewords*). The set of codewords will form a linear subspace of $\mathbb{F}_2^n$, which we will specify via some linear constraints. Fix an expander graph as promised by Lemma 1. We shall define an expander graph with $k \geq n - m$, and distance $\delta = \frac{n}{100d}$.

We identify each of the n vertices on the left of the expander with a distinct bit x_i of the code. Every vertex y on the right hand side will correspond to a linear constraint on the bits that correspond to its neighbors. Namely, if $x_{i_1}, \ldots, x_{i_c}$ are the neighbors of the vertex y, we add the constraint $x_{i_1} + x_{i_2} + \cdots + x_{i_c} = 0$. The set of vectors $x \in \mathbb{F}_2^n$ that satisfy all of these constraints form a vector space! Since there are m constraints, the dimension of this space is at least $k \geq n-m$, which can be made as high as $n/4$ with our expander construction. One can easily compute the encoding matrix E that maps k bit vectors to the corresponding codewords. Next we discuss the distance of this code. First we observe that the expander actually guarantees that every small enough set has many unique neighbors:

Lemma 2. *For any $S \subset A$, $|S| \leq n/100d$, there exists a vertex $y \in \Gamma(S)$ that has exactly one neighbor in S.*

Proof Suppose not. Then the number of edges from $\Gamma(S)$ to S is at least $6d|S|/4$, but this is impossible since the number of such edges (counting from the left) is at most $d|S|$. ■

Claim 3. *The distance of the code is at least $\delta = nd/100$.*

Proof Since the code is linear, it suffices to prove that every non-zero codewords must have at least δ non-zero entries. Suppose there is a codeword with hamming weight $< \delta$. Then consider the set of coordinates S that are non-zero. By Lemma 2, there is a vertex on the right that has exactly one neighbor in S. This codeword cannot satisfy the constraint of that vertex, and so cannot be a codeword. ■

Now suppose a sender transmitted a codeword $E(x)$, which was received (after some errors) as $y \in \mathbb{F}_2^n$. We shall show how to recover $E(x)$ from y. First note that if y is obtained from $E(x)$ after more than $\delta/2$ errors to the coordinates, then we cannot hope to recover $E(x)$, since for example, it could be that y is halfway in between two valid codewords. However, we shall show that one can recover $E(x)$ as long as the number of errors is at most $\delta/2$. The algorithm to reconstruct $E(x)$ is extremely simple:

1. Compute all the parities specified by vertices on the right side of the expander. Take an arbitrary input bit x_i such that the majority of x_i's neighboring constraints are not satisfied.

2. Set $x_i = x_i + 1$.

3. Repeat until all constraints are satisfied.

We shall show that the algorithm is well defined, and that it terminates with the correct answer.

Suppose the current vector we are working with is $E(x) + e$, where e is a vector with less than $\delta/2$ 1's. Let S denote the set of non-zero coordinates of e, and let C, U be a partition of the neighbors of S in the expander such that the constraints that correspond to the vertices of C are all satisfied, and the constraints that correspond to the vertices of U are unsatisfied.

First observe that since we always flip a bit whose neighbors are mostly unsatisfied:

Claim 4. *In each step, the size of U must decrease.*

Next we argue that we can always find such a bit to flip. By the property of the expander, we have

$$|U| + |C| > 3d|S|/4 \tag{1}$$

Now count edges from S to $\Gamma(S)$. There are at least $|U|$ edges leaving $|U|$, and at least $2|C|$ edges leaving C. So

$$|U| + 2|C| \leq d|S| \tag{2}$$

Computing $2(1) - (2)$, we get

$$|U| > d|S|/2 \tag{3}$$

Thus, by double counting, there must be some vertex on the left, more than half of whose neighbors are unsatisfied. So the algorithm will continue, as long as there are unsatisfied constraints. So far we have shown that the algorithm is well defined and terminates.

To prove correctness, we claim that if we start with a word that is of the form $x+e$, where e has less than $\delta/2$ ones and x is a codeword, then throughout this process, we remain within distance δ to x. Indeed, the distance changes at most by 1 in each step, and if it ever becomes δ, then by equation (3), $|U| > d\delta/2$. However, when the algorithm begins, we have $|U| \leq d\delta/2$, since U is contained in the neighborhood of non-zero coordinates of e in the beginning. This contradicts the fact that $|U|$ can only decrease.