CSE599s: Extremal Combinatorics December 5, 2011

Lecture 19 Expander Graphs (continued)

Lecturer: Anup Rao

Last time we introduced expander graphs, and saw how to obtain error correcting codes from them. This time we shall continue our tour of expander graphs and their applications.

Loosely speaking, an *expander graph* is an undirected graph on n vertices where every vertex has degree d (a small constant), 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.

One can also define expanders using the spectrum of the adjacency matrix. Suppose the adjacency matrix is A:

$$A_{i,j} = \begin{cases} 1 & \text{if } \{i,j\} \text{ is an edge,} \\ 0 & \text{otherwise.} \end{cases}$$

The normalized adjacency matrix is $B = A/d$. B has a natural interpretation: it is the transition matrix for the stochastic process of taking a random step on the graph. In other words, if x is a column vector that corresponds to a probability distribution on the vertices (so that $\sum_i x_i = 1$ and $x_i \geq 0$), then Bx is the vector that is the distribution obtained by first sampling a vertex according to x and then picking a uniformly random neighbor of that vertex. Similarly, B^k is the matrix that corresponds to taking k random steps in the graph. Here are some other properties of B:

- B has exactly n real eigenvalues (possibly repeated): $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$.
- The corresponding eigenvectors form an orthonormal basis.
- $\lambda_1 = 1$, and the first eigenvector is the vector that takes value $(1/\sqrt{n})$ everywhere.
- $\lambda_1 > \lambda_2$ if and only if the graph is connected.
- $\lambda_1 = -\lambda_n$ if and only if the graph is bipartite.

In the last lecture, we were mostly concerned with the *vertex expansion* of the graph. Today we define the edge expansion of the graph:

$$h(G) = \min_{S, |S| \leq n/2} \frac{\text{\# edges coming out of S}}{|S|}$$

Then one can show that edge expansion is closely tied to the value of λ_2:

Theorem 1.

$$d\left(\frac{1-\lambda_2}{2}\right) \leq h(G) \leq d\sqrt{2(1-\lambda_2)}$$

Thus, a good expander will have a constant eigenvalue gap $(1-\lambda_2) = \Omega(1)$.

1 Some Useful Properties of Expanders

1.1 Random Walks on Expanders Converge to Uniform

Suppose we start with an arbitrary distribution p on vertices. Then we can write p in the basis of eigenvectors of B as $p = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$. Since p is a probability distribution ($\sum_i p_i = 1$), we must have that $\alpha_1 v_1$ is simply the vector u for the uniform distribution. Thus, we get that $p - u = \sum_{i=2}^n \alpha_i v_i$, and $B(p-u) = Bp - u = \sum_{i=2}^n \lambda_i \alpha_i v_i$. Thus:

$$\left\|B^k p - u\right\| \le \left\|\sum_{i=2}^n \lambda_i^k \alpha_i v_i\right\| = \sqrt{\sum_{i=2}^n \lambda_i^{2k} \alpha_i^2} \le \lambda_2^k \sqrt{\sum_{i=2}^n \alpha_i^2} \le \lambda_2^k \sqrt{n},$$

so the distribution rapidly converges to the uniform distribution, in $O(\log n)$ steps.

Another very useful property of expanders is the following theorem (whose proof we do not discuss here). Let $\lambda = \max\{|\lambda_2|, |\lambda_n|\}$.

Theorem 2. *Let $f_1, f_2, \ldots, f_t : [n] \to [0,1]$ be a sequence of functions defined on the vertex set of an expander graph, each with mean $\mathbb{E}[f(v)] = \mu$. Let $X_1, \ldots, X_t$ be the vertices of a random walk starting at a uniformly random vertex in the graph. Then*

$$\Pr\left[\left|\sum_{i=1}^t f_i(X_i) - \mu t\right| \ge \epsilon t\right] < 2e^{-\frac{\epsilon^2(1-\lambda)t}{4}}.$$

1.2 Expander Mixing Lemma

The expander mixing lemma says that in a good expander graph, the number of edges between any two sets S, T is about what you would expect for a random graph of that edge density. Let $\lambda = \max\{|\lambda_2|, |\lambda_n|\}$, then we have

Lemma 3 (Expander Mixing Lemma)**.** *Let $E(S,T)$ denote the number of edges from the set S to the set T. Then*

$$\left|E(S,T) - \frac{d|S||T|}{n}\right| \le \lambda d \sqrt{|S||T|}$$

Proof Recall that the eigenvalues of A are $d \ge d\lambda_2 \ge \ldots \ge d\lambda_n$. Let 1_S, 1_T be the (column) indicator vectors for S, T. Then $E(S,T) = 1_S^t \cdot A \cdot 1_T$. We can write each of them in the orthonormal basis of the eigenvectors of A as $1_S = \sum_i \alpha_i v_i$ and $1_T = \sum_i \beta_i v_i$. Then $E(S,T) = d\sum_i \lambda_i \alpha_i \beta_i$. Note that $\alpha_1 = |S|/\sqrt{n}$ and $\beta_1 = |T|/\sqrt{n}$. Thus, the top term is $d|S||T|/n$, and we get

$$\left|E(S,T) - \frac{d|S||T|}{n}\right| \le \lambda d \sum_{i=2}^n |\alpha_i \beta_i| \le \lambda d \|1_S\| \|1_T\| \le \lambda d \sqrt{|S||T|},$$

where the last inequality follows from the Cauchy-Schwartz inequality. ■

In fact, one can prove a converse (but we do not go into the proof here):

Lemma 4. *Suppose*

$$\left|E(S,T) - \frac{d|S||T|}{n}\right| \le \rho \sqrt{|S||T|},$$

then $\lambda \le O(\rho(1 + \log(d/\rho)))$, and the bound is tight.

2 Some Explicit Constructions

To give an idea of exactly how explicit expander constructions can be, we give a couple of them here:

2.1 Margulis's Construction

The vertex set is $\mathbb{Z}_m \times \mathbb{Z}_m$, where $\mathbb{Z}_m$ denotes the integers mod m. The edges of a vertex (x, y) are $(x, x+y), (x+y, y), (x-y, y), (x, y-x), (x+y+1, y), (x-y+1, y), (x, y+x+1), (x, y-x+1)$.

2.2 A Degree 3 Expander

The vertex set is $\mathbb{Z}_p$ for a prime number p. x is connected to $(x+1), (x-1)$ and the multiplicative inverse x^{-1}, with the inverse of 0 being 0.

3 Randomness Efficient Error Reduction

Recall that a randomized algorithm A that computes a function f is in randomized polynomial time with one sided error if for every input x, the algorithm uses n uniformly random bits r to compute a value $A(x, r)$ such that $\Pr_r[A(x, r) = f(x)] \geq 2/3$. We are interested in increasing the probability that the algorithm is correct. One idea is to use the Chernoff bound. We can simply run the algorithm t times on the input x, using completely independent randomness $r_1, \ldots, r_t$, and take the majority (or plurality) outcome. By the Chernoff bound, the probability that the majority is incorrect is at most $2^{-\Omega(t)}$. However, this requires us to use nt random bits.

A better idea is to use Theorem 2: We sample a random walk of length n on the vertex set of an expander graph defined on 2^n vertices. Each vertex gives us a random string r_i, and as before, we output the majority outcome of running the algorithm using each of these random strings. By Theorem 2, the probability of computing the wrong value is at most $\epsilon = 2^{-\Omega(t)}$. However, now we only need to invest $n + O(t)$ random bits in order to achieve this low probability of failure!

Indeed, we can get rid of the additional randomness entirely if $t = O(\log n)$. To do this, let us call a vertex v *bad* if at least $\sqrt{\epsilon}$ of the random walks of length t that begin at v lead to the algorithm computing the wrong value. Then at most $\sqrt{\epsilon}$ of all vertices can be *bad*, since the total fraction of bad random walks is at most ϵ. Thus, we can simply sample a random vertex v, and run over all possible random walks that start at v, and output the majority outcome from all those walks. The number of such walks is at most $2^{O(t)} = \mathsf{poly}(n)$ which is a polynomial in the running time of the original algorithm, so it is not too large. In this way, we obtain an algorithm that uses only n random bits, but whose probability of failure is at most $2^{-\Omega(t)}$.

4 Monotone Formulas for Majority, and Sorting

Question: What is the minimum depth of a monotone boolean circuit that computes the majority of n bits?

One can give a non-monotone circuit of depth $O(\log n)$ that computes the majority by computing the number of 1's in the output. However that circuit will use negations.

One way to compute the majority of n bits is to give a *comparator* circuit that sorts n numbers. A comparator circuit is a circuit where all gates take two inputs and have two outputs. The outputs are the inputs in sorted order. Ajtai, Komlos and Szemeredi found a comparator circuit of depth $O(\log n)$ for sorting n numbers. Note that when the inputs are bits, a comparator gate can be implemented using an OR gate and an AND gate. The $\lceil n/2 \rceil$ bit of the output is the majority of the inputs.

A basic primitive in their construction is a constant depth circuit called an ϵ-halver that can take n inputs and output n numbers such that (except for ϵn errors) the first $n/2$ of the numbers are the lowest $n/2$, and the second $n/2$ are the highest. If we could do this exactly, then we could recursively repeat the construction (by sorting the left half and the right half separately) to obtain a circuit of $O(\log n)$ depth that sorted all the numbers. Now think of the complete bipartite graph with $n/2$ vertices on each side as a union of $n/2$ perfect matchings. We can obtain a comparator network that is a 0-halver of depth $n/2$, by running the comparisons that correspond to each matching in each step. If we do this, we can see that a number that is larger than the median can never end up on the right. Ajtai, Komlos and Szemeredi make progress by using a bipartite expander that is the union of a constant number of matchings to get a constant depth circuit. The expander has the property that for every set of size ϵn on the left and every set of size ϵn on the right, there is an edge between the two sets. This implies that the number of elements that end up in the wrong half is at most $2\epsilon n$.

To actually get this idea to work is quite involved. It remains open to give a simple comparator circuit.

4.1 Valiant's Randomized Circuit Construction for Majority

Now we give a randomized construction of a $O(\log n)$ depth monotone circuit for majority. Note that only the construction is randomized! At the end we will show how to generate a circuit of depth $O(\log n)$ that computes the majority of any input with probability 1.

The construction is based on some simple observations about the majority of independent random bits:

Claim 5. *Suppose X, Y, Z are independent identically distributed bits, such that each one is equal to $b \in \{0, 1\}$ with probability $1/2 + \epsilon$, where here $\epsilon < 1/4$. Then the majority of these bits is equal to b with probability at least $1/2 + (5/4)\epsilon$.*

Proof The probability that the majority is 1 is exactly

$$\begin{aligned}
&3(1/2+\epsilon)^2(1/2-\epsilon) + (1/2+\epsilon)^3 \\
&= 3(1/4+\epsilon+\epsilon^2)(1/2-\epsilon) + (1/8 + 3\epsilon/4 + 3\epsilon^2/2 + \epsilon^3) \\
&= 1/2 + \epsilon(3/2 - 3/4 + 3/4) + \epsilon^2(3/2 - 3 + 3/2) + \epsilon^3(1-3) \\
&\geq 1/2 + 3\epsilon/2 - 2\epsilon^3 \geq 1/2 + (3/2 - 1/8)\epsilon \geq 1/2 + 5\epsilon/4
\end{aligned}$$

■

On the other hand, we have

Claim 6. *Suppose X, Y, Z are independent identically distributed bits, such that each one is equal to $b \in \{0, 1\}$ with probability $1 - \epsilon$, where here $\epsilon < 1/100$. Then the majority of these bits is equal to b with probability at least $1 - \epsilon^{3/2}$.*

Proof The probability that the majority is not b is at most $3\epsilon^2 + \epsilon^3 < \epsilon^{3/2}$ for $\epsilon < 1/100$. ■

Finally, by the Chernoff bound, if we have bits $X_1, \ldots, X_c$ that are each b with probability $1/2 + 1/5$, then for a large enough constant c, we have that the majority is equal to b with probability at least $1 - 1/1000$.

The final randomized construction is as follows. We build a tree of majority gates of depth $d_1 + d_2$. Every gate at depth d_2 computes the majority of c inputs. Every other gate computes the majority of 3 inputs. At the bottom, we feed in completely random inputs into the tree: in other words, the input to each majority gate at depth 1 is a uniformly random input variable. Each majority gate can eventually be replaced with a monotone circuit of constant size (and depth).

We set $d_2 = \Theta(\log n)$ to be large enough so that $(5/4)^{d_2} \geq n$, and d_1 to be large enough so that $(1/100)^{(3/2)^{d_1}} \leq (1/2)^{2n}$. Then for any fixed input, each input gate gets independent bits that are biased towards the majority with bias $1/n$. Thus, after depth d_1, the bias is greater than $1/5$. The majority gates in the d_2 layer then ensure that their outputs are equal to the majority with probability $1 - 1/1000$. The final layer of d_1 gates then ensures that the output of the circuit is equal to the majority except with probability 2^{-2n}. By the union bound, the probability that the circuit is incorrect on any one input is at most 2^{-n}, so there is some fixing of the randomness used that makes the circuit correct on all inputs!