CS 880: Advanced Complexity Theory	2/13/06
Lecture 8: Iterative Construction	
Instructor: Dieter van Melkebeek	Scribe: Siddharth Barman, Anand Sinha

This lecture provides details for explicit construction of expander graphs using Zig Zag products. We also introduce the idea of using expander graphs for deterministic amplification of algorithms with one sided error.

1 Iterative Construction

Last time we proved the following

Theorem 1. *$\lambda_2(G ⓩ H) \leq g(\lambda_2(G), \lambda_2(H))$ where $g(x, y) \leq x + y + y^2$ and $g(x, y) < 1$ if $x < 1$ and $y < 1$.*

Zig-zag products allow construction of larger and larger expander graphs with constant degree and without much deterioration in spectral gap. To take care of the deterioration in spectral gap we used squaring.

Idea 1: Combine ⓩ with squaring, so we defined $G_{i+1} = (G_i)^2 ⓩ H$,
substituting values in the equation gives us $N(H) = d(H)^4$ and the other parameters for the graph family as:

- $G_1 = H^2$
- $N(G_i) = (N(H))^i$
- $d(G_i) = (d(H))^2$
- $\lambda_2(G_{i+1}) \leq (\lambda_2(G_i))^2 + \lambda_2(H) + (\lambda_2(H))^2 = \lambda_2(H) + O((\lambda_2(H))^2)$
  in particular, $\lambda_2(G_{i+1}) \leq 2/5$ if $\lambda_2(H) \leq 1/5$.

Eventually we want an explicitly defined family of expander graphs. The above construction is good in terms of parameters but not in terms of explicitness. To take the idea home we introduce certain new features to it.

Definition 1. *(Rotation Map) Given vertex v and edge index i we have $ROT(v, i) = (w, j)$ (see Figure 1).*

The critical step in explicitly defining the zig zag product using,$G_{i+1} = (G_i)^2 ⓩ H$ is to determine the rotation map of G_{i+1}.

This is achieved recursively using the rotation map of G_i. Note that in $G ⓩ H$ each vertex can be labeled using a pair (v, i) where v is a vertex in G and i a vertex in H. Each edge can be labeled by a pair (p, q) as well. Here p and q are edges of H taken in two different clouds of the zig zag step.

We get $ROT_{G ⓩ H}((v, i), (p, q)) = ((w, j), (r, s))$ from $ROT_H(i, p) = (i', s)$, $ROT_G(v, i') = (w, j')$ and $ROT_H(j', q) = (j, r)$

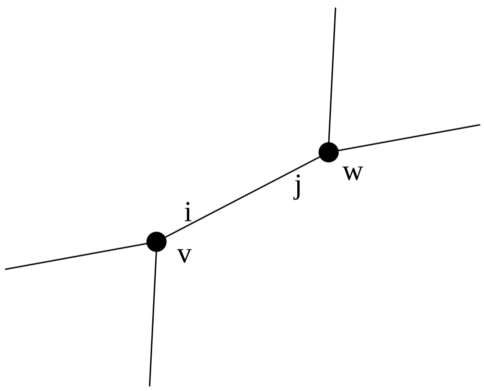

Figure 1: Rotation Map

Thus for a direct zig-zag product we make two oracle calls to the rotation map of H and one to G. For the rotation map of G^2 we have $ROT_{G^2}(v; i, k) = (w; l, j)$ from $ROT_G(v, i) = (z, j)$ and $ROT_G(z, k) = (w, l)$.

The recursion tree for queries in G_{i+1} is as shown in Figure 2: the height of the tree is $i = log_{N(H)}(N)$, where N is the number of nodes in G_{i+1}

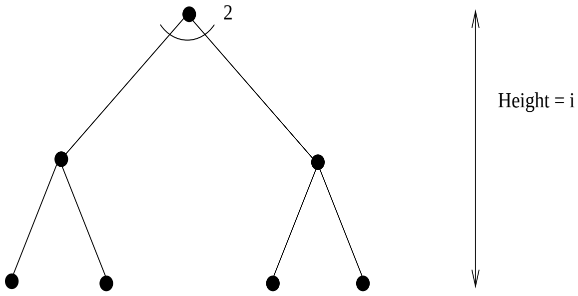

Figure 2: Recursion Tree

The total amount of work is roughly proportional to the number of vertices in the tree, so we get total work as 2^i $polylog(N)$, which is $N^{log_{N(H)}(2)}$ $polylog(N)$. Thus the amount of work is polynomial in the number of vertices which is not good when N is exponential.

To do better we use a recurrence relation that increases the number of vertices as well. We use the tensor product of two graphs. Note that the random walk on the tensor product is the same as taking simultaneous random walk on individual graphs.

Idea 2: Add tensor product in recurrence relation.

For tensor product $G_1 \otimes G_2$ of graphs G_1 and G_2 we have,

- $N(G_1 \otimes G_2) = N(G_1)N(G_2)$
- $d(G_1 \otimes G_2) = d(G_1)d(G_2)$
- $\lambda_2(G_1 \otimes G_2) = max\{\lambda_2(G_1), \lambda_2(G_2)\}$, i.e. the spectral gap does not deteriorate.

Since we still need to improve the spectral gap, we retain the squaring operator. So our modified recurrence relation becomes,

$$G_{i+1} = \left(G_{\lceil \frac{i+1}{2} \rceil} \otimes G_{\lfloor \frac{i+1}{2} \rfloor}\right)^2 ⓩ H.$$

Here we have $N(H) = (d(H))^8$, $N(G_{i+1}) = (N(H))^{2i-1}$ and $d(G_i) = (d(H))^2$. For the base case we use $G_1 = H^2$ and again we have $\lambda_2(G_i) = \lambda_2(H) + O((\lambda_2(H))^2)$. For this modified iterative construction the height of the recurrence tree is $log\ i$ (see Figure 3). Hence the amount of work required here for explicitly stating the graph is $4^{log\ i}\ polylog(N) = i^2\ polylog(N)$, which is polylog in N. The space complexity is also $O(log\ N)$.

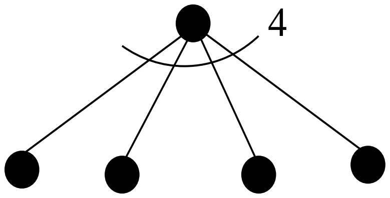

Figure 3: Modified Recursion Tree

2 A family of small degree expanders

The iterative constructions given above have a degree equal to $d(H)^2$. We also want that the spectral gap for this graph is quite half. For instance if $\lambda_2(H) \geq \frac{1}{\phi}$, where ϕ is the golden ratio, then the iterative construction does not guarantee any spectral gap (to be more precise, we would like to say that at least our proof does not work). To get a graph with a high spectral gap means that we are forced to have high degree for the expander family (remember the upper bound on the spectral gap as a function of the degree of the graph).

However, we can use a small trick to reduce the degree of the family. To do this we can just take the zig-zag product of every member of the family with any graph of $d(G ⓩ H)$ many vertices which is connected and is not bipartite. This would guarantee that it has non zero spectral gap. The simplest graph is a cycle. This would make the degree of the resulting family to be equal to 4. There is a subtle point over here. Note that an even cycle is bipartite. This can be dealt with by starting with a graph H of odd degree. The existence of such graphs can be proved by a probabilistic argument. We leave the details as an exercise. Another approach to circumvent this difficulty would be to add an edge between pairs of vertices. At most one vertex might be left unpaired. However, the number of vertices of H is a power of its degree. Both are odd or even together. Note that adding edges does not reduce the expansion of the graph. Hence, it remains an expander.

Next we will see that even the replacement product is good enough for the above task. Hence we can reduce the degree of the family to three. We can not expect any further improvement as we proved that a lesser degree graph does not expand sufficiently.

3 Replacement Product

In the first lecture we stated the following theorem

Theorem 2. *For every pair of graphs, say G, H, if $G \textcircled{r} H$ is well defined, then $\lambda_2(G \textcircled{r} H) \leq f(\lambda_2(G), \lambda_2(H), d(H))$. Furthermore, if both $\lambda_2(G)$ and $\lambda_2(H)$ are strictly less than one, then so is f*

Proof. To prove this we first need to make an observation. Note that $(G \textcircled{r} H)^3$ has all the edges that are present in $G \textcircled{z} H$. This already gives us an intuitive explanation of the second part of the theorem. If G and H are both expanders (read have a nonzero spectral gap), then so has $G \textcircled{z} H$. But this means that $G \textcircled{z} H$ is an expander; and hence adding some edges to it won't reduce the expansion. As such, $(G \textcircled{r} H)^3$ is an expander as well and has non zero spectral gap. The rest follows as the eigenvalues of $G \textcircled{r} H$ and $(G \textcircled{r} H)^3$ are related by a cubic relation.

To formalize this intuition into a rigorous proof, we consider the normalized adjacency matrix of $G \textcircled{r} H$, say $A(R)$. Note that $A(R)$ is also the transition probability matrix for the Markov process of taking random walks on $G \textcircled{r} H$. We already said that there are two kinds of edges in $G \textcircled{r} H$, those which remain in the same cloud (i.e edges from H) and the ones between different copies of H. The normalized adjacency matrix corresponding to the first part is $I \otimes A(H)$. The second matrix is the permutation matrix corresponding to the zag steps in the zig-zag product of G and H. For any vertex the probability that it takes a step which results in the same cloud as it originally was is $p = d(H)/(d(H)+1)$. Hence we have the following relation,

$$A(R) = p(I \otimes A(H)) + (1-p)P.$$

It follows that,

$$A(R)^3 = p(1-p)p(I \otimes A(H))P(I \otimes A(H)) + (1 - p(1-p)p)Q.$$

Here Q is a product of matrices each of which has the spectral radius equal to 1. Note that the first matrix in the above sum is the normalized adjacency matrix of $G \textcircled{z} H$. Hence we have that,

$$\lambda_2(A(R)^3) \leq p(1-p)p\lambda_2(G \textcircled{z} H) + (1 - p(1-p)p) < 1$$

hence $\lambda_2(A(R)) < 1$.

From the preceding discussions, it also follows that $f(x, y, d(H)) = (qg(x, y) + 1 - q)^{1/3}$ where $g(x, y) = x + y + y^2$ and $q = p(1-p)p$. Note that unlike the zig-zag product, the bound obtained here depends also on the degree of the graph H (as p is depends on $d(H)$). $\square$

4 Deterministic Amplification

In the rest of the lecture we will see one use of the expander graphs. The use is motivated by the fast mixing property of random walks in expander graphs. We said that taking a logarithmic step random walk in an expander, starting from any distribution, makes the distribution fairly close to the uniform in the statistical sense. Now, suppose we are simulating an algorithm with one sided error. It is known that running the algorithm independently many times reduces the error exponentially. Suppose the cost of generating "truly random bits" is huge, or for that matter it is hard to get truly random bits. In that case, we would like to reduce the use of randomness as much as possible. Therefore, one needs to generate a large number of random bits starting from a small number of random bits. They might not be independent (actually they cannot be independent), nevertheless, good enough for amplification purposes. One way to do this is to use a small number of random bits to sample a node uniformly in an expander graph. Now we consider all nodes which are at a logarithmic distance from it. It is not difficult to see that running the algorithm on these nodes reduces the error up to any desired polynomial. Here we see the description of a node as a binary sequence and use it to get the random bits needed by our algorithm. Of course, the length of the random walk would depend on the amplification we wish to accomplish. Another small detail is that we would need an expander graph of size exactly equal to that power of two which is equal to the number of random bits needed by the algorithm. Hence we must be able to take one step in a random walk in time polylogarithmic in the size of the graph. The zig-zag family has this property. This is the desired level of explicitness for this use. In some cases, especially when dealing with more restricted models on computation (eg. small depth circuit, space bounded algorithms), we would need even more explicit constructions. Later, hopefully as an assignment problem, we will see that the Gaber-Galil construction is more explicit. Note that we might not have an expander of the desired size. However, we will see that even if we are a constant factor away from the exact size, we can tweak the above idea and make it to work.